

VLT® 5000 / 5000 FLUX



Instruction Manual

Programmable
SyncPos motion controller



Chapter 1	■ Safety	3
<hr/>		
Chapter 2	■ Introduction	5
<hr/>		
Chapter 3	■ Input/Output terminals	9
<hr/>		
Chapter 4	■ Fundamentals of the SyncPos program	19
	Starting the SyncPos motion controller	
	step-by-step	26
	Optimizing the PID-Controller	32
	CAM Control	38
	CAM Box	48
<hr/>		
Chapter 5	■ PC Software interface	49
	■ PC Software interface in MTC10 Mode	81
<hr/>		
Chapter 6	■ Programming with SyncPos	87
	Introduction and basics	89
	■ Software Reference	96
	List of Commands	96
	All commands from ACC to #INCLUDE	99
	■ Parameter Reference	163
	VLT and SyncPos parameter	163
	VLT parameter list	167
	SyncPos parameters	168
<hr/>		
Chapter 7	■ Messages and Error Reference	199
<hr/>		
Chapter 8	■ Program Samples	207
<hr/>		
Chapter 9	■ Appendix	211
	Glossary terms	212
	What's new in Version ... ?	213
	Technical Reference	218
	Index	222

Programmable SyncPos motion controller for VLT[®]5000 and VLT[®]5000Flux

This instruction manual covers two versions of the programmable SyncPos motion controller:

Ordering number	Description	Type code
175Z0833	SyncPos for VLT5000	A10 C0
175Z3029	SyncPos for VLT5000 with conformal coating	A10 C1
175Z3683	SyncPos for VLT5000Flux	A10 C0
175Z3684	SyncPos for VLT5000Flux with conformal coating	A10 C1

The basic functionality of the two versions is the same, the only differences are:

- The Flux version has two additional outputs on the VLT5000Flux control card.
- The Flux version does not yet support absolute encoders.

This instruction manual is valid for the following software versions:

SyncPos PC software version 2.5X
VLT5000/SyncPos software version 3.XX/5.0X
VLT5000Flux/SyncPos software version 5.XX/5.0X

© Danfoss A/S 2003

VLT is a registered Danfoss trademark.

IBM is a registered trademark of International Business Machines, Inc.

Microsoft, MS, MS-DOS, Microsoft NT, Windows and Wingdings are either registered trademarks or trademarks of the Microsoft Corporation in the USA and/or other countries.

Chapter 1

■ Safety 4

Safety



The voltage of the frequency converter is dangerous whenever the equipment is connected to mains. Incorrect installation of the motor or the frequency converter may cause damage to the equipment, serious personal injury or death. Consequently, the instructions in this manual, as well as national and local rules and safety regulations, must be complied with.

■ Safety regulations

1. The VLT frequency converter must be disconnected from mains if repair work is to be carried out.
Check that the mains supply has been disconnected and that the necessary time has passed before removing motor and mains plugs.
2. The [STOP/RESET] key on the control panel of the VLT frequency converter does not disconnect the equipment from mains and is thus not to be used as a safety switch.
3. Correct protective earthing of the equipment must be established, the user must be protected against supply voltage, and the motor must be protected against overload in accordance with applicable national and local regulations.
4. The earth leakage currents are higher than 3.5 mA.
5. Protection against motor overload is not included in the factory setting. If this function is desired, set parameter 128 to data value *ETR trip* or data value *ETR warning*.
Note: The function is initialised at 1.16 x rated motor current and rated motor frequency (see page 92).

For the North American market: The ETR functions provide class 20 motor overload protection in accordance with NEC.

6. Do not remove the plugs for the motor and mains supply while the VLT frequency converter is connected to mains. Check that the mains supply has been disconnected and that the necessary time has passed before removing motor and mains plugs.
7. Please note that the VLT frequency converter has more voltage inputs than L1, L2 and L3, when loadsharing (linking of DC intermediate circuit) and external 24 V DC have been installed.

Check that all voltage inputs have been disconnected and that the necessary time has passed before repair work is commenced.

■ Warning against unintended start

1. The motor can be brought to a stop by means of digital commands, bus commands, references or a local stop, while the frequency converter is connected to mains.
If personal safety considerations make it necessary to ensure that no unintended start occurs, these stop functions are not sufficient.
2. While parameters are being changed, the motor may start. Consequently, the stop key [STOP/RESET] must always be activated, following which data can be modified.
3. A motor that has been stopped may start if faults occur in the electronics of the VLT frequency converter, or if a temporary overload or a fault in the supply mains or the motor connection ceases.



Warning:

Touching the electrical parts may be fatal - even after the equipment has been disconnected from mains.

Also make sure that other voltage inputs have been disconnected, such as external 24 V DC, load-sharing (linkage of DC intermediate circuit), as well as the motor connection for kinetic back-up.

- Using VLT 5001-5006 220 and 500 V units: wait at least 4 minutes
- Using VLT 5008-5500 220 and 500 V units: wait at least 15 minutes
- Using VLT 5001-5005 550-600 V units: wait at least 4 minutes
- Using VLT 5006-5022 550-600 V units: wait at least 15 minutes
- Using VLT 5027-5250 550-600 V units: wait at least 30 minutes

Chapter 2	■ Introduction	6
	■ About this manual: How is it arranged?	7
	■ Conventions	7

■ About this manual: How is it arranged?

Please read these operating instructions in full and, in order to be able to work with the system safely and professionally, particularly observe the hints and cautionary remarks.



Indicates a general warning



Indicates something to be noted by the reader



Indicates a high-voltage warning

■ Chapter Input/output terminals

Information on the technical data of the option card, the allocation of the connecting terminals and the general connection requirements can be found in this chapter.

■ Fundamentals of the SyncPos program

First, let us explain the principle of SyncPos, the macro programming language, to you in brief. Then you will learn about the fundamentals of the SyncPos program – this is particularly important if you are not familiar with SyncPos or Windows.

Starting the SyncPos-Option step-by-step

Then you will start up the VLT with the 'SyncPos Motion Controller step-by-step'. This section offers you a quick general introduction which includes turning on the SyncPos option and familiarizing yourself with it, as well as starting operation of the control unit with the test programs provided and the most important basic settings.

Optimizing the PID-Controller

In this chapter you learn all about the PID control parameters and how to optimize the controller during one or more test runs, for example, in order to achieve better positioning results or shorter cycle times. Detailed instructions on how to proceed can be found in the section 'Ten steps for optimum control'.

CAM control and CAM box

This section explains the basics of CAM control and of CAM box. The editing of curves and the programming of controls can be reconstructed in detail with the included examples.

■ Chapter PC-Software Interface

All menus and functions are described in detail. For example, the Development menu to test the new programs or the Controller menu used to manage the programs and set the parameters and the CAM-Editor.

■ Chapter PC Software Interface in MCT10-Mode

SyncPos behaves different, if started directly by clicking on the application's icon or if started indirectly by opening up a program or configuration file via the Motion Control Tool MCT10. The main differences are related to the fact that all file handling is limited to MCT10 only. These limitations concerning the SyncPos GUI are explained in this chapter, e.g. how to open and save files or how to edit a cnf-curve or set parameters.

■ Chapter Programming with SyncPos, Software and Parameter Reference

You can learn everything about programming and about the commands in these chapters, ranging from the basic construction of the commands to a detailed description of all commands from ACC to #INCLUDE and also of the parameters.

First look in the general summary – here the commands are sorted according to groups, for example, control commands, commands for velocity control or for synchronization – and then look in the alphabetically ordered list where the commands are described more precisely.

The next section describes all parameters – first in alphabetical order in the summary – and then individually with the factory settings and ranges.

■ Chapter Messages and Error Reference

In the chapter for Messages and Error Reference an extra section is dedicated to messages from the VLT. The table is arranged according to error number in increasing order. It contains detailed information on possible errors and troubleshooting. The next section explains the most important messages from the SyncPos user interface.

■ Chapter Program samples

The manual presents some program samples (you will find much more in the online help) which you can use to familiarize yourself with the program or copy directly into your program.

■ Appendix

A short glossary explains used terms.

Get a quick overview on news in Version 2.5x and in the corresponding versions of the SyncPos Motion Controller in the appendix. These are primarily the necessary extensions and a modified GUI to use the SyncPos program in MCT10 mode.

Experienced users will find detailed information in the technical reference material for example the „Array Structure of CAM Profiles“.

Plus, the handbook ends with a detailed index for a direct navigation.

■ Conventions

The information in this manual follows the system and uses the typographical features described below to the greatest extent possible:

Menus and functions

Menus and functions are printed in normal text with capitals, for example: "CONTROLLER" → "PARAMETERS".

Commands and parameters

Commands and parameter names are written in capitals, for example: AXEND and KPROP; Parameters are printed in bold faced italics, for example: ***Proportional factor***.

Variables

Variables are written in small letters, when they are being quoted they are emphasized with italics, for example: timeout or *timeout*.

Keys

The names of keys and function keys are emphasized in brackets, for example the control key [CNTL] key (or just [CNTL]), the [ESC] key or the [F1] key.

Cross references

Cross references to other parts of the text are underlined in the manual and are also marked in color in the online help, for example EXECUTE.

Chapter 3

■ **Input/Output terminals**

VLT control card terminals	10
Option card terminals	10
Supply voltages	12
Encoder monitor	12
Option card layout, Dip switch	13
Technical data	14
Connection examples	15

There are two interfaces to the SyncPos option:

- 36 terminals on the option card
- 24 terminals on the VLT control card

■ VLT control card terminals

The terminals on the control card can be allocated for synchronizing and positioning functions if the following parameter settings are made:

Digital inputs 16, 17, 18, 19, 27, 29, 32 and 33:

It is always possible to read the status of the digital inputs from the SyncPos application program with the IN command.

If Parameter 300–303 and 305–307 are set to "No operation" (default setting) then the inputs are ignored by the control card but they can still be used as inputs to the application program.

Parameter 304 can not be set to "No operation" which means that input 27 always has a stop function. There is one way to avoid this: Leave parameter 304 at "Coast inverse" and select "Serial port" in parameter 502.

Analogue inputs 53, 54 and 60:

It is always possible to read the value on the analogue inputs from the SyncPos application program with the INAD command.

If parameters 308, 311 and 314 are set to "No operation" then the inputs are ignored by the control card but they can still be used as inputs to the option card.

VLT5000: Digital/analogue outputs 42 and 45:

Outputs 42 and 45 can be controlled from the SyncPos application program with the OUTDA command, when parameters 319 and 321 are set to one of the following four settings:

- ```
OPTION DIGITAL [90] digital output
OPTION 0 ... 20 mA [91] analogue output
 (default setting)
OPTION 4 ... 20 mA [92] analogue output
OPTION 0 ... 32000P [93] pulse output
```

#### VLT5000Flux: Digital/pulse outputs 26 and 46

The outputs 26 and 46 on the VLT5000Flux control card can be controlled from the SyncPos application program with the OUTDA command when parameters 341 and 355 are set to one of the following settings:

- ```
OPTION DIGITAL [90] digital output
OPTION 0...50000P [91] pulse output
```

VLT5000Flux: Analogue outputs 42 and 45

The outputs 42 and 45 on the VLT5000Flux control card can be controlled from the SyncPos application program with the OUTDA command when parameters 319 + 321 are set to one of the following settings:

- ```
OPTION 0 ... 20 mA [90] analogue output
 0...20mA
OPTION 4 ... 20 mA [91] analogue output
 4...20mA
```

#### Relay outputs 01 and 04:

The relay outputs can be controlled from the SyncPos application program with the OUT command when parameters 323 and 326 are set to the default setting

```
CTRL WORD BIT 11/12
```

#### Technical data

Technical data on the control card terminals can be found in the VLT 5000 design guide.

### ■ Option card terminals

There are two encoder interfaces which are covering the following functions:

- Feedback encoder input
- Master encoder input / virtual master output

There are 8 digital inputs, 8 digital output and terminals for 5 V and 24 V supply. The functions and technical data of the terminals are described in the following.

#### Terminal description

There are 4 terminal blocks, 2 with 10 poles and 2 with 8 poles. (See figure below)

**MK3A** Digital Inputs

|    |    |    |    |    |    |    |    |      |     |
|----|----|----|----|----|----|----|----|------|-----|
|    |    |    |    |    |    |    |    |      |     |
| I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | 24 V | COM |

**MK3C** Digital Outputs

|    |    |    |    |    |    |    |    |      |     |
|----|----|----|----|----|----|----|----|------|-----|
|    |    |    |    |    |    |    |    |      |     |
| O1 | O2 | O3 | O4 | O5 | O6 | O7 | O8 | 24 V | COM |

**MK3B** Master / Virtual Master

|     |     |    |    |    |    |    |    |  |  |
|-----|-----|----|----|----|----|----|----|--|--|
|     |     |    |    |    |    |    |    |  |  |
| 5 V | COM | A1 | A1 | B1 | B1 | Z1 | Z1 |  |  |

**MK3D** Feedback

|     |     |    |    |    |    |    |    |  |  |
|-----|-----|----|----|----|----|----|----|--|--|
|     |     |    |    |    |    |    |    |  |  |
| 5 V | COM | A2 | A2 | B2 | B2 | Z2 | Z2 |  |  |

### Digital inputs

MK3A is a 10pole terminal block with 8 digital inputs (I1–I8) and 2 terminals for 24 V supply (See *Supply voltages on page 12*).

The digital inputs are used by the SyncPos application program and are therefore free programmable. All the inputs can also be assigned functions by the following parameters:

|                   |                              |
|-------------------|------------------------------|
| I_BREAK (105)     | interrupt a running program  |
| I_CONTINUE (106)  | restart interrupted program  |
| I_ERRCLR (107)    | reset fault                  |
| I_NEGLIMITSW (47) | negative limit switch        |
| I_POSLIMITSW (46) | positive limit switch        |
| I_PRGCHOICE (104) | selection of SyncPos program |
| I_PRGSTART (103)  | start of SyncPos program     |
| I_REFSWITCH (45)  | home switch                  |

Only two of the inputs are assigned a specific function by parameter:

- **I5** is used as marker input for the Master when “External marker signal” is selected in parameter SYNCMTYPM (60).
- **I6** is used as marker input for the Slave when “External marker signal” is selected in parameter SYNCMTYPS (61).

### Digital outputs

MK3C is a 10pole terminal block with 8 digital outputs and 2 terminals for 24 V supply (See *Supply voltages on page 12*).

The digital outputs are controlled by the SyncPos application program and are thus free programmable. All the outputs can also be assigned functions by the following parameters:

|               |                                |
|---------------|--------------------------------|
| O_AXMOVE (64) | axe is moving                  |
| O_BRAKE (48)  | activation of mechanical brake |
| O_ERROR (108) | fault indication               |

### Encoder interface 1

MK3B is a 8pole terminal block with 6 terminals for the encoder signals and 2 terminals with 5 V supply.

(See *Supply voltages on page 12*).

Encoder interface 1 can be used for one of the following 2 functions:

- Master encoder input (incremental or absolute) for synchronizing.
- Virtual master encoder output (incremental).

The table below shows the function of each terminal in the 3 possible modes. (See *fig. 1*)



#### NB!

When using the virtual master function termination must be switched off (sw 1.3) in all options except on the first and the last station connected in the network. See also connection example on page 16 and 17.

### Encoder interface 2

MK3D is a 8pole terminal block with 6 terminals for the encoder signals and 2 terminals with 5 V supply

(See *Supply voltages on page 12*).

Encoder interface 2 can be used for one of the following 2 functions:

- Slave encoder input (incremental or absolute) for synchronizing.
- Feedback encoder input (incremental or absolute) for positioning.

The table below shows the function of each terminal in the 2 possible modes. (See *fig. 2*)

| Terminal          | A1      | $\overline{A1}$      | B1      | $\overline{B1}$      | Z1       | $\overline{Z1}$    |
|-------------------|---------|----------------------|---------|----------------------|----------|--------------------|
| Incremental input | A in    | $\overline{A}$ in    | B in    | $\overline{B}$ in    | Z in     | $\overline{Z}$ in  |
| Absolute input    | Clk out | $\overline{Clk}$ out | Data in | $\overline{Data}$ in | Not used | Not used           |
| Virtual master    | A out   | $\overline{A}$ out   | B out   | $\overline{B}$ out   | Z out    | $\overline{Z}$ out |

Fig. 1

| Terminal          | A2      | $\overline{A2}$      | B2      | $\overline{B2}$      | Z2       | $\overline{Z2}$   |
|-------------------|---------|----------------------|---------|----------------------|----------|-------------------|
| Incremental input | A in    | $\overline{A}$ in    | B in    | $\overline{B}$ in    | Z in     | $\overline{Z}$ in |
| Absolute input    | Clk out | $\overline{Clk}$ out | Data in | $\overline{Data}$ in | Not used | Not used          |

Fig. 2

### ■ Supply voltages:

The option card is supplied by the internal 24 V DC supply of VLT 5000, but as the available power is limited it can be necessary to use an external 24 V DC supply.

The 24 V DC supply of VLT 5000 can supply a total of 420 mA including the load on the control card (terminal 12, 13 and output 42 and 45).

The 5 V output on the option card is generated from the 24 V supply. The maximum power on the 5 V side is  $5\text{ V} * 280\text{ mA} = 1.4\text{ W}$ , this corresponds to app. 60 mA on the 24 V side.

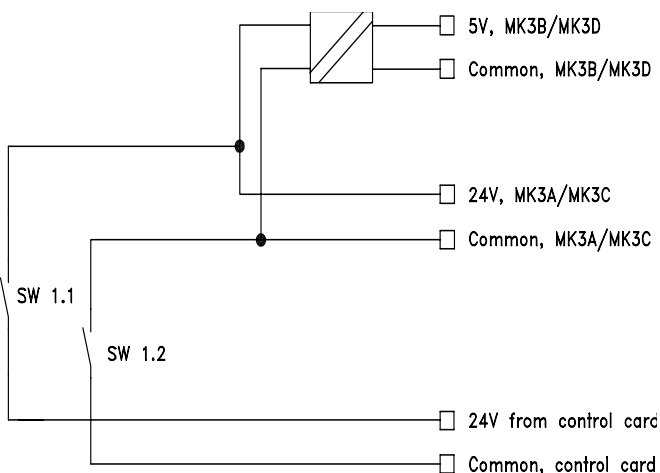
When an external 24 V DC voltage source is used the internal 24 V supply from the control card must be disconnected, this is done by opening switch 1.1 and 1.2

Each digital input on the option card takes 8 mA.

Each digital output on the option card can supply up to 0.7 A (external 24V-supply) depending on the load.

The load from the 24 V supply (internal or external) can be calculated as follows:

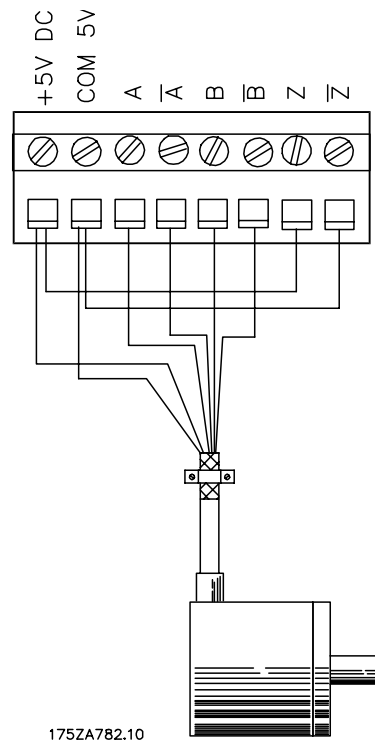
$$\begin{aligned}
 &8\text{ mA} * \text{number of digital inputs} \\
 &+ \\
 &\text{Load on digital outputs} \\
 &\quad (\text{mk3 C, O1 - O8}) \\
 &+ \\
 &\text{load on 5 V supply} \\
 &\quad (\text{mk3 B/D, 5 V/com}) \\
 &+ \\
 &\text{Load on control card} \\
 &\quad (24\text{ V supply, terminal 12/13 and} \\
 &\quad \text{outputs, terminal 42/45})
 \end{aligned}$$



DANFOSS  
175HA461.11

### ■ Encoder monitor

Both encoder interfaces are equipped with a monitoring circuit that can detect open circuit as well as short circuit of each encoder channel. Each encoder channel has a LED showing the status: Green light means ok, no light means fault. Zero channel monitoring can be switched off by means of switch 1.4, this is necessary when using incremental encoders without Zero channel or absolute encoders. Switch 1.4 disables monitoring of both master and slave Zero channel. If disabling of only one of the two Zero channels is required (e.g. when using incremental master encoder and absolute slave encoder) the unused Zero channel input must be connected to 5V/common as shown below.



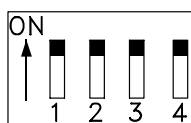
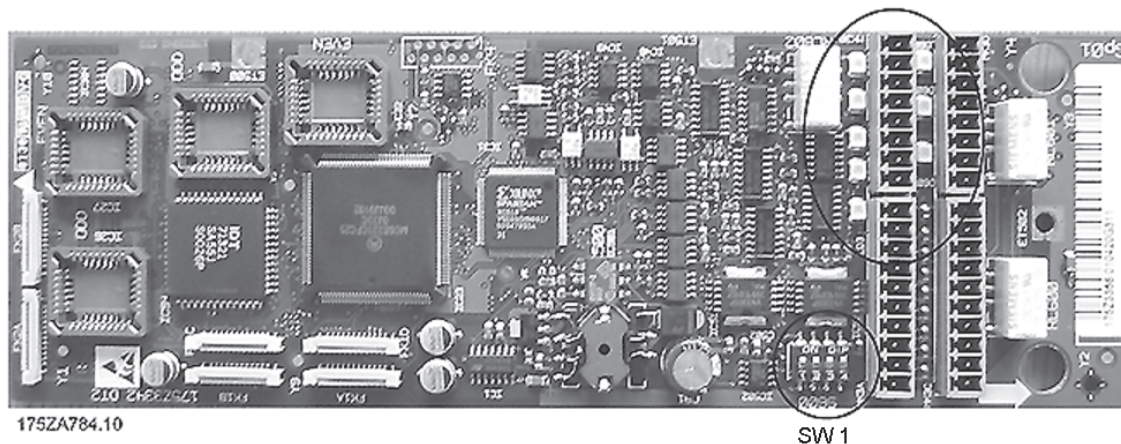
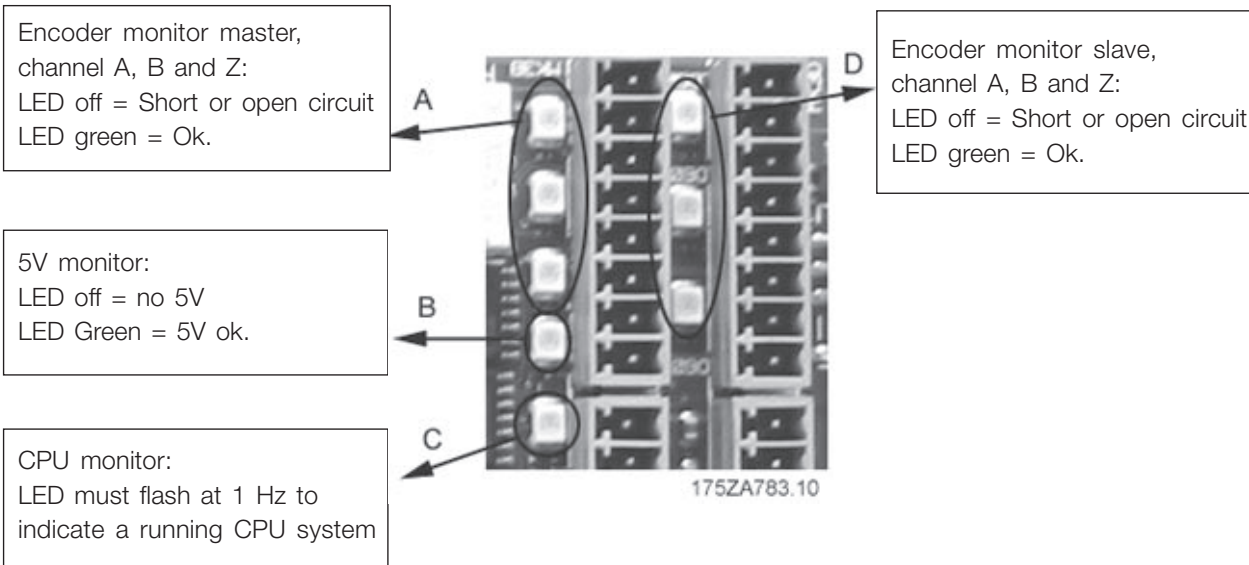
Physical placing of the LED's and the switches can be seen in the chapter "Option card layout".

An encoder fault will only result in an "Option error" calling the error handler (ON ERROR) if encoder monitoring is activated via parameter MENCODER (master) and ENCODER (slave).

Note: Monitoring of the master encoder is disabled when switch 1.3 is "OFF".

■ **Option card layout:**

Option card layout showing the position of connectors and dip switch.



DANFOSS  
175ZA068.10

- SW 1.1: Connect(ON)/disconnect(OFF) 24 V from control card (see description of supply voltages).
- SW 1.2: Connect(ON)/disconnect(OFF) 24 V common from control card.
- SW 1.3: Connect(ON)/disconnect(OFF) termination resistor for master encoder (see description of virtual master function).  
Note: When OFF the master encoder monitor is disabled.
- SW 1.4: Switch Z-channel encoder monitor ON/OFF for both master and slave.

Default setting of switch 1.1. - 1.4 is ON.



**NB!**

When using the virtual master function termination must be switched off (sw 1.3) in all options except on the first and the last station connected in the network. See also connection example on page 16 and 17.

**■ Technical data**
Terminals:

Type..... Plugs with screw connections  
 Maximum cable size ..... 1.3 mm<sup>2</sup> (AWG 16)

Digital inputs, MK3A:

Number of inputs which are used by SyncPos program ..... 8  
 Terminal designations ..... I1 – I8  
 Voltage level ..... 0 – 24 V DC (PNP positive logic)  
 Voltage threshold logical “0” ..... 5 V DC  
 Voltage threshold logical “1” ..... 10 V DC  
 Maximum voltage ..... 28 V DC  
 Input impedance ..... 4 kΩ  
 Min. pulse duration (ON INT) ..... 1 msec

*Galvanic isolation: All digital inputs are galvanically isolated by means of optocouplers, but with the same common as the digital outputs.*

Digital outputs, MK3C:

Number of outputs which are used by SyncPos program ..... 8  
 Terminal designations ..... O1 – O8  
 Voltage level ..... 0 – 24 V DC  
 Maximum load ..... 0.7A (with external power supply)  
 Update rate ..... 1 msec

*Galvanic isolation: All digital outputs are galvanically isolated by means of optocouplers, but with the same common as the digital inputs.*

External 24 V DC supply:

(see VLT 5000 manual)

Encoder input 1, MK3B (master):

Terminal designations ..... A1,  $\overline{A1}$ , B1,  $\overline{B1}$ , Z1,  $\overline{Z1}$ .

*Incremental:*

Signal level ..... 5 V differential  
 Signal type ..... Linedriver, RS 422  
 Input impedance ..... 120 Ω (Dip switch 1.3 = ON)  
 ..... > 24 kΩ (Dip switch 1.3 = OFF)  
 Maximum frequency ..... 220 kHz (at 50 % duty cycle)  
 Phase displacement between A and B ..... 90° ±30°

*Absolute:*

Signal level ..... 5 V differential  
 Signal type ..... SSI  
 Data coding ..... Gray code  
 Data length ..... 25 bit  
 Parity ..... none  
 Clock frequency ..... 105 or 260 kHz  
 Protocol ..... Gray  
 Maximum positions per revolution ..... 8192  
 Maximum number of revolutions ..... 4096

Encoder input 2, MK3D (slave):

|                                          |                                                                 |
|------------------------------------------|-----------------------------------------------------------------|
| Terminal designations .....              | A2, $\overline{A2}$ , B2, $\overline{B2}$ , Z2, $\overline{Z2}$ |
| <i>Incremental:</i>                      |                                                                 |
| Signal level .....                       | 5 V differential                                                |
| Signal type .....                        | Linedriver, RS422                                               |
| Input impedance .....                    | 120 $\Omega$                                                    |
| Maximum frequency .....                  | 220 kHz (at 50 % duty cycle)                                    |
| Phase displacement between A and B ..... | 90° $\pm$ 30°                                                   |
| <i>Absolute:</i>                         |                                                                 |
| Signal level .....                       | 5 V differential                                                |
| Signal type .....                        | SSI                                                             |
| Protocol .....                           | Gray code                                                       |
| Data length .....                        | 25 bit                                                          |
| Parity .....                             | none                                                            |
| Clock frequency .....                    | 105 or 260 kHz                                                  |
| Maximum positions per revolution .....   | 8192                                                            |
| Maximum number of revolutions .....      | 4096                                                            |

Encoder cable:

Cable type .. Twisted pair and screened. **Note:** Please observe the prescriptions of the encoder supplier  
 Cable length..... Observe the prescriptions of the encoder supplier.  
 Absolute encoder is tested ok up to 150 meter cable at 105 kHz clock and 100 m at 262 kHz clock.  
*(Tested with TR electronic encoder type CE-65 M 8192\*4096 and appropriate cable prescribed by TR electronic.)*  
 Maximum allowed time delay between clock and data signal measured at the controller terminals .....

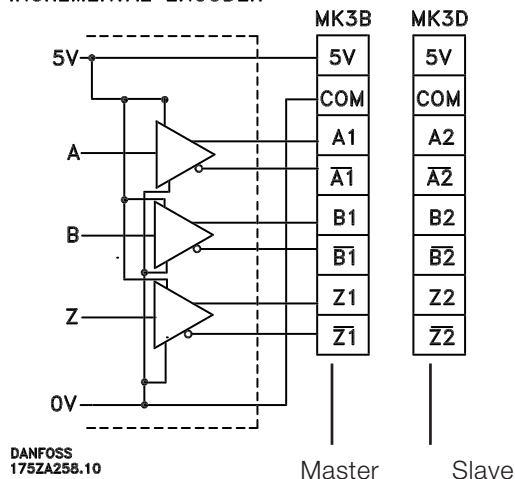
|       |                               |
|-------|-------------------------------|
| ..... | 105 kHz clock = 9 $\mu$ sec   |
| ..... | 262 kHz clock = 3.5 $\mu$ sec |

Encoder output, MK3B:

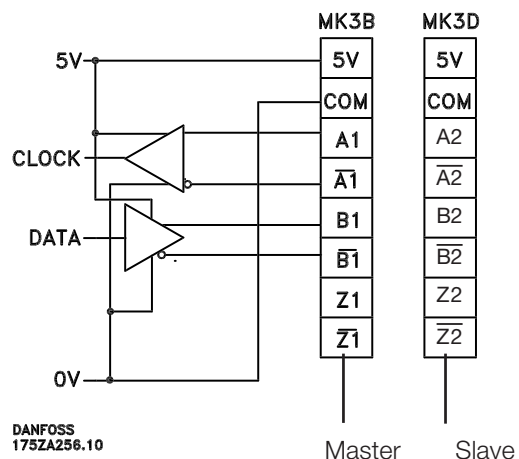
|                                |                                                                 |
|--------------------------------|-----------------------------------------------------------------|
| Terminal designations .....    | A1, $\overline{A1}$ , B1, $\overline{B1}$ , Z1, $\overline{Z1}$ |
| Signal type .....              | Linedriver, RS485                                               |
| Maximum frequency .....        | 150 kHz                                                         |
| Minimum frequency .....        | 150 Hz                                                          |
| Maximum number of slaves ..... | 31 (more when using repeaters)                                  |
| Maximum cable length .....     | 400 m                                                           |

■ Connection examples

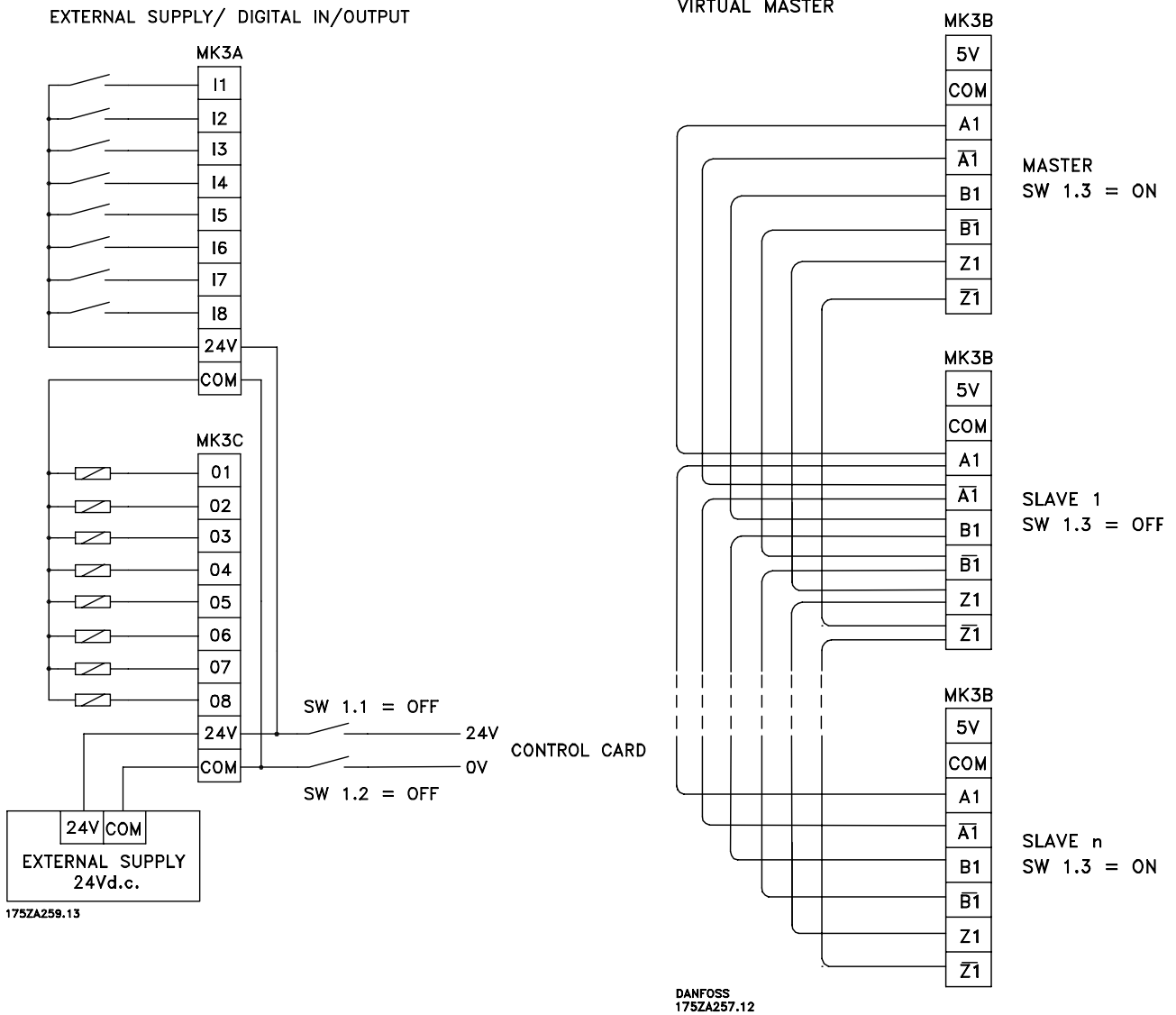
INCREMENTAL ENCODER



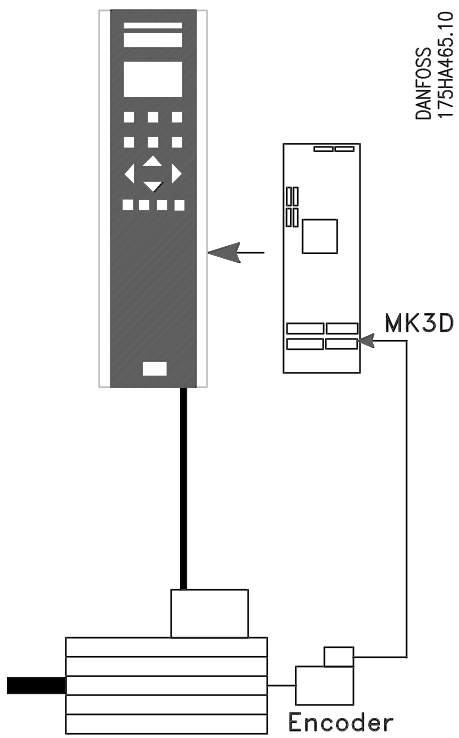
ABSOLUTE ENCODER



Input/Output terminals

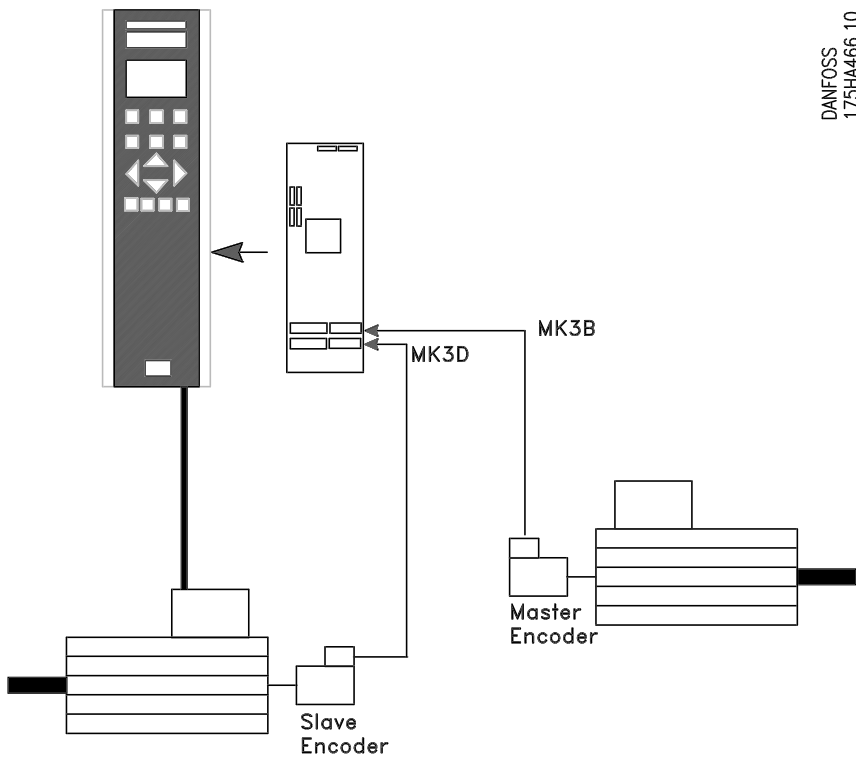


The terminating resistors on both end of the bus have to be switch on with dip switch SW 1.3.

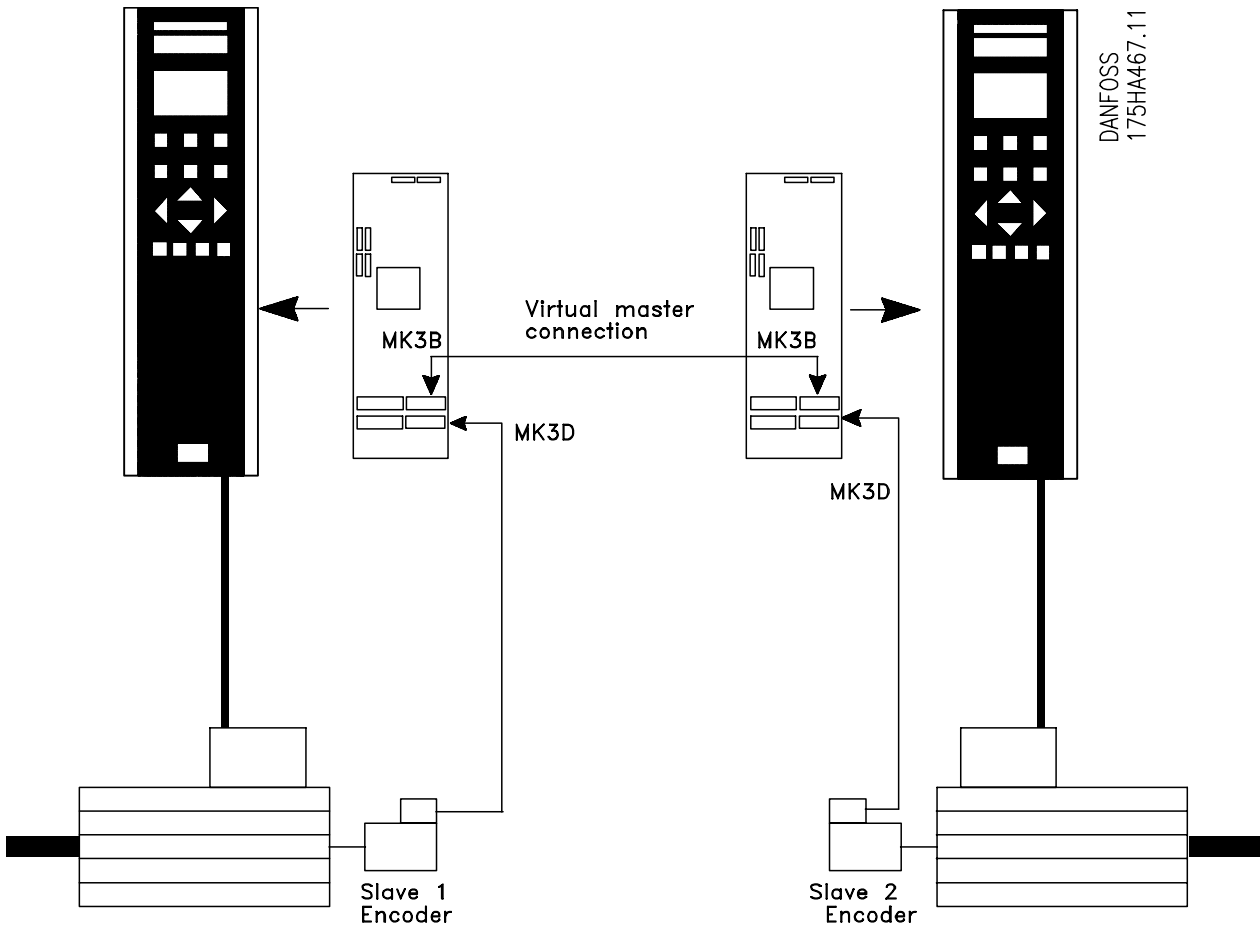


left: encoder connection for positioning applications

below: encoder connection for master-/slave synchronization



Input/Output terminals



Encoder connections for the synchronizing with virtual master.

**Chapter 4**

|                                                     |    |
|-----------------------------------------------------|----|
| <b>■ Fundamentals of the SyncPos program</b>        |    |
| How SyncPos functions .....                         | 21 |
| Requirements .....                                  | 22 |
| The SyncPos Window .....                            | 22 |
| Using the mouse .....                               | 24 |
| Keyboard .....                                      | 24 |
| List of shortcuts .....                             | 25 |
| List of function keys.....                          | 25 |
| <b>■ Starting the SyncPos motion controller</b>     |    |
| <b>step-by-step</b>                                 |    |
| Safety tips .....                                   | 26 |
| Installing SyncPos .....                            | 26 |
| Starting SyncPos .....                              | 26 |
| Setting of VLT parameters .....                     | 27 |
| Setting up communication .....                      | 27 |
| Setting of SyncPos option parameters .....          | 28 |
| Checking encoder connection and                     |    |
| direction of rotation .....                         | 30 |
| Execute the test run program .....                  | 31 |
| <b>■ Optimizing the PID controller</b>              |    |
| How the control process works .....                 | 32 |
| Significance and influence of the control           |    |
| parameters .....                                    | 32 |
| Optimizing your controller settings step-by-step .. | 34 |
| What to do if ... ..                                | 36 |
| Examples of control optimization .....              | 36 |
| <b>■ CAM Control</b>                                |    |
| How a CAM Control with SyncPos operates .....       | 38 |
| Quick Start Tutorial for impatient users .....      | 39 |
| Example: Stamping of boxes with use-by date ...     | 39 |
| Example: Printing of cardboard boxes                |    |
| with marker correction .....                        | 42 |
| If the sensor distance is larger than one           |    |
| master cycle length .....                           | 43 |
| Problematic situations in the determination         |    |
| of the marker distance .....                        | 44 |
| Example: Slave synchronization with marker .....    | 45 |
| <b>■ CAM Box</b>                                    |    |
| How a CAM Box with SyncPos operates .....           | 48 |
| Example of a CAM Box .....                          | 48 |

SyncPos is a programming language specially designed for synchronizing and positioning control which is easy to learn, the commands of which are based on frequently used control terminology. This macro-language makes it possible to realize complex functions with simple commands even without knowledge of the hardware processes and thus create control programs and general control programs in a very short time.

The SyncPos software contains all commands and menus necessary for the configuration, programming, optimization and, finally, for the transfer of commands to the SyncPos motion controller.

**■ How SyncPos functions**

Let us explain the basic principle of SyncPos in brief:

Determining parameters

Factory settings are stored in the program for all parameters. These factory settings are active upon delivery and can be re-activated at any time by performing a reset (see page 42).

You can adjust all the parameters for your controller. These user parameters are permanently saved in the EEPROM and are valid for all programs.

Before you start programming it is necessary to determine basic parameters of the VLT connected, such as **Maximum velocity** VELMAX (1) and **Shortest ramp** RAMPMIN (31), set the PID filter values and define the **User factor** with POSFACT\_Z (23) and POSFACT\_N (26).

Within a program you can temporarily alter the parameters with the command SET. After running the program these values are once again replaced with the user parameters which have been saved.

Programming with the SyncPos macro-language

In the "EDIT" menu you can create and comment on the programs just like in a text program. All commands are described in detail in the [Chapter Software Reference](#).

Each command consists of a COMMAND WORD + **parameter** (if necessary), whereas the parameter can be a variable, constant or an array.

Comments are written between /\* ... \*/ or after // for example:

```
POSA 3000
```

```
/* axis absolute to actual zero point move to position 3000 */
```

```
// axis absolute to actual zero point move to position 3000
```

It is particularly easy to write your program by using the "COMMAND LIST". Once you have selected the command, all the necessary input fields are immediately opened. After entering the values the syntax is automatically formed and you can insert the entire command in your program.

With teach-in programming you simply move the axis to the desired position and store the position which has been reached. In this manner you can quickly program the most complicated adjustments and sequences of movements.

Running and testing programs

In the "DEVELOPMENT" menu you can test new programs. The program is loaded into the VLT and started only after the "EXECUTE" function has been activated.

Naturally, you can run the program being tested in "SINGLE STEPS" or start the program at a certain point and have it executed step-by-step.

Before every run a new program is automatically checked to ensure that the commands are correct. Or you can start the "SYNTAX CHECK" without running the program.

Saving programs in the VLT

Every time you Execute a program or start the syntax check, this program is temporarily stored in the RAM in an area that is always overwritten with every subsequent test.

Once you have finished writing a program for the SyncPos option, the temporary program can be saved permanently in EEPROM. It will then be assigned a number or name and can be tagged with "AUTOSTART" so it automatically will be started after turning on the VLT. This way you can drive the VLT offline.

All programs can also be started using the program number via the inputs, for example from a PLC. For this the inputs must be set accordingly with "CONTROLLER" → "PARAMETERS" → "GLOBAL".

Optimizing the controller with the control parameters

The position control unit integrated in the SyncPos motion controller automatically calculates a theoretical set course during each sequence of movements and tries to control the VLT or the motor so that the best possible convergence to the set course is achieved. By means of the control parameters you can directly influence to what degree and how quickly a deviation from the theoretical set course is counteracted.

These PID filter parameters can also, theoretically, be determined if you have comprehensive knowledge of the entire drive, including the load connected. However, the experimental method with the functions in the "TESTRUN" menu is considerably faster and simpler.

After every "TESTRUN" it is also possible to evaluate the control parameters on the basis of four graphics: they show the actual and set curves for the velocity, the acceleration, the position and the power curve. Thus, you can successively adjust the PID filter parameters and optimize the controller.

We wish you the best in your work with SyncPos! For questions regarding programming or operating the controller, please contact your supplier.

### ■ Requirements

SyncPos is for use on standard PC's with operating systems Windows 95 resp. NT 3.5. You should be familiar with the basic functions and terminology of the Microsoft Windows interface, for example the **Task bar** and the **Explorer**. You can find further information in the corresponding Windows manuals if necessary.

### ■ The SyncPos Window

The following figure serves to explain the most important elements of the SyncPos Windows.

Each window represents a SyncPos program which can be connected with a VLT. Thus, you can open at least the same number of edit windows as the number of controllers resp. VLTs you have selected.

#### Symbol bar

Click on the icons in the symbol bar to quickly select a function.



From left to right: New file, Open file, Save file, Cut, Copy, Paste, Info and CAM-Editor.

Title bar shows the names of the SyncPos File, the number and name of the VLT and the error number if applicable.

Open the menus in the Menu bar to select the SyncPos functions.

Menus with the functions which you can mark and select with the mouse.

Click on the icon in the Symbol bar to quickly select a function.

Edit window

The blinking cursor shows where the text to be entered will appear.

Dialog field

Communications window for messages from the controller and the compiler.

Use the Scroll Bar to scroll the file up or down

or left from left to right.

The status bar shows the line number and position where the cursor is located, information about the function keys and whether the [NUM lock key] or the [Shift lock key] have been pressed and are active.

If you want to change the size of the Edit window or the Communications window, move the cursor to the lower edge of the scrollbar and – as soon as the cursor has changed its shape – click and pull the window in the desired direction.

## Menus

A check means that the function has been activated, for example in the Window menu it shows which files are open.

Disabled functions are not available until you trigger a preceding action, for example, when you have copied something to the notepad and want to insert it someplace else or "PREPARE SINGLE-STEP" to trigger each "SINGLE-STEP".

You can also select most menu commands using the keyboard. Press [ALT] and the underlined letter in the menu name and then the underlined letter in the command name, e.g. [ALT] + [C] + [P] for "CONTROLLER" → "PROGRAMS".

## Dialog fields

Once you have selected a function, a dialog field often appears in which you can determine certain options. If an option is disabled, then it is not available for the current process.

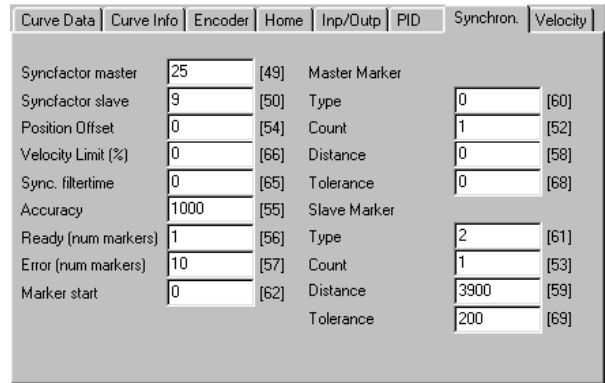
## Index Cards

In the Windows user interface, dialog fields are provided on several levels together with the index cards for the selection of functions or the input of data. For example, the two index cards → "FIX POINTS" and → "START STOP POINTS" in CAM-Editor.



Click on the name of the index card and the corresponding level will be brought to the front.

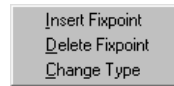
In the case of the index cards "CURVE DATA", "CURVE INFO" and "PARAMETER", click on the scroll bars to scroll to the right and show additional index cards:



This will automatically change the division of the CAM-Editor. If you click on "WINDOW" → "STANDARD", then on → "RECALC" or any other field in CAM-Editor, the standard window will be displayed again immediately.

## Pop-up Menu

Pop-up menus are provided at certain program locations if you click on the right mouse button. For example: the context menu in the SyncPos edit window or a selection menu in CAM-Editor for inserting or deleting fixpoints. The pop-up menus are closed automatically when the selected function is being executed or if you click on any other location on the screen with the left mouse button.

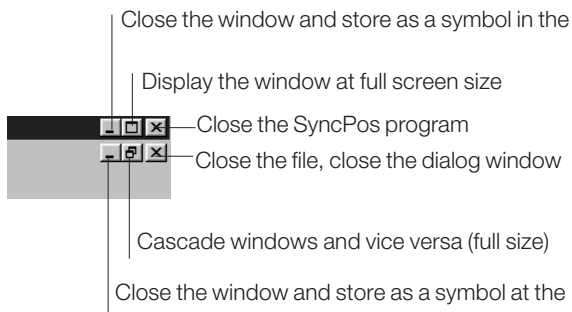


## Edit window

In this window you can write your program with the assistance of the functions in the "EDIT" menu just like in a text editor. Different colors are used to distinguish between comments, program sections, operators, numbers etc. You can alter the colors using "SETTINGS" → "COLORS EDITOR".

## Changing window size and window icons

If you want to change the size of the SyncPos windows, simply move the cursor to an outside edge of the window and pull - once the cursor has changed its shape - the window while pressing the mouse button until the window is the size you desire. The icons on the upper right in every window do the following ...



### ■ Using the mouse

If you have a mouse with several keys, the left mouse button is generally the "primary" key (provided you haven't changed the configuration). "Click" means that you press the mouse key for a short moment and then immediately release the button without moving the mouse. If nothing to the contrary is stated then always click with the left, or primary, key. "Pull" means that you point to the element and hold the mouse button depressed while moving the mouse.

### ■ Keyboard

#### [ESC] key

In addition to the standard functions of the [ESC] key, you can also use this key at any time to abort a program running under SyncPos.



#### **NB!**

A rotating drive will slow down with the maximum allowed deceleration!

#### Arrows

Using the arrows, [↓ key], [↑ key], [← key] and [→ key], you can move the insert marks in a file.

#### Direction keys

With the direction keys [HOME] and [END] you can move the cursor to the beginning or end of a line and with [Page ↑] and [Page ↓] to the beginning or end of a of a screen page. Some direction keys can also be combined with each other, thus, for example with [CNTL] + [HOME] the cursor will move to the beginning of the file.

#### Number pad

If you have an expanded keyboard available, you can also input numbers via the number pad, once you have pressed the [Num lock key] before inputting numbers.

#### Short cuts

Keys are often used as shortcuts with other keys either as a key combination or as a key sequence. For a key combination you have to hold the first key depressed while you press the second key, e.g. [SHIFT] + [INSERT], in order to insert the contents of the notepad. For key sequences you can press the keys one after the other, e.g. [ALT] + [E] in order to open the "EDIT" menu.

#### Function keys

Frequently used functions are allocated to the function keys, e.g. with [F9] you can very effectively control the step-by-step execution of a program: each time you press [F9], one line of the program is run. Or with [F1] you can access the on-line help.

**■ List of shortcuts**
**Cut, Copy and Paste ...**

|                                                        |                                      |                 |
|--------------------------------------------------------|--------------------------------------|-----------------|
| Copy the marked text to the notepad                    | [CNTL] + [INSERT]                    | or [CNTL] + [C] |
| Cut and save the text marked in the notepad            | [CNTL] + [DEL]<br>or [SHIFT] + [DEL] | or [CNTL] + [X] |
| Paste the contents of the notepad                      | [SHIFT] + [INSERT]                   | or [CNTL] + [V] |
| Line open above                                        | [CNTL] + [SHIFT] + [N]               |                 |
| Delete the (remaining) word to the right of the cursor | [CNTL] + [DEL]                       |                 |
| Delete the (remaining) word to the left of the cursor  | [CNTL] + [BACKSPACE]                 |                 |
| Line by line deletion                                  | [CNTL] + [Y]                         |                 |

**Cursor positioning**

|                                          |                  |                        |
|------------------------------------------|------------------|------------------------|
| Go to document end                       | [CNTL] + [END]   |                        |
| Go to document start                     | [CNTL] + [HOME]  |                        |
| Go to line n                             | [CNTL] + [G]     |                        |
| Shift scrollbars up or down line by line | [CNTL] + [↑-KEY] | resp. [CNTL] + [↓-KEY] |
| Shift scrollbars to the left or right    | [CNTL] + [PAGE↑] | resp. [CNTL] + [PAGE↓] |

**Expand the text marked ...**

|                                                                                                                    |                                                |               |
|--------------------------------------------------------------------------------------------------------------------|------------------------------------------------|---------------|
| ...by one character to the right resp. to the left                                                                 | [SHIFT] + [→ KEY]                              | resp. [← KEY] |
| ...to the end of the word                                                                                          | [CNTL] + [SHIFT] + [→ KEY]                     |               |
| ...to the beginning of the word                                                                                    | [CNTL] + [SHIFT] + [← KEY]                     |               |
| ...to the end of the line                                                                                          | [SHIFT] + [END]                                |               |
| ...to the beginning of the line                                                                                    | [SHIFT] + [HOME]                               |               |
| ...down one line resp. up one line                                                                                 | [SHIFT] + [↓KEY]                               | resp. [↑KEY]  |
| ...to the beginning of the file                                                                                    | [CNTL] + [SHIFT] + [HOME] or [SHIFT] + [PAGE↑] |               |
| ...to the end of the file                                                                                          | [CNTL] + [SHIFT] + [END] or [SHIFT] + [PAGE↓]  |               |
| Mark the next or the previous command in the menu (when the menu is displayed).                                    | [↑ KEY] resp. [↓KEY]                           |               |
| Mark the menu on the right or left side, or changes from the main menu to the sub-menu if a sub-menu is displayed. | [← KEY] resp. [→ KEY]                          |               |

**Further editing tools**

|                                                                                                 |                                     |
|-------------------------------------------------------------------------------------------------|-------------------------------------|
| Undo the last action (File Save deletes the Undo memory.)                                       | [ALT] + [BACKSPACE] or [CNTL] + [Z] |
| In CAM-Editor: Undo the input up to the previous "RECALC". (File Save deletes the Undo memory.) | [ALT] + [BACKSPACE]                 |
| In CAM-Editor: "REDO": The action of the Undo command is cancelled.                             | [ALT] + [SHIFT] + [BACKSPACE]       |

**■ List of function keys**

|                                                                            |                       |
|----------------------------------------------------------------------------|-----------------------|
| Abort program                                                              | [ESC]                 |
| Access online help                                                         | [F1]                  |
| Switches the mouse cursor to context-sensitive help                        | [SHIFT] + [F1]        |
| In "FIND" mode: jumps back and forth between the sites found.              | [F2]                  |
| In "FIND" mode: jumps from one site found to the next one.                 | [F3]                  |
| Start "DEVELOPMENT" → "EXECUTE"                                            | [F5]                  |
| Select line                                                                | [CNTL] + [ALT] + [F8] |
| In the "SINGLE-STEP" mode starts one program line each time it is pressed. |                       |
| or in CAM-Editor → "RECALCULATION"                                         | [F9]                  |
| Calls up the "COMMAND LIST"                                                | [F12]                 |

## ■ Starting the SyncPos Motion Controller step-by-step

This section offers you a quick general introduction which includes turning on and familiarizing yourself with the program, how to set up a VLT with SyncPos motion controller using the test programs provided and the most important basic settings.



### NB!

O.ERR 13 will show up right after power up if the VLT is not ready. The VLT is in the "Not ready" state when:

- it has an alarm (trip),
- it is in local mode (parameter 002 = local),
- local LCP stip is activated (display flashing),
- there is no signal on input 27 (coast).

O.ERR 13 can only be reset using the ERRCLR command or with "BREAK" [ESC] in the PC software and only when the VLT is in the "ready" state, which means none of the above may be true.

The VLT monitoring function can be switched off by selecting [2] in parameter 700.

Please follow the step-by-step guideline:

1. SyncPos installation and starting
2. Setting VLT parameters
3. Setting up communication
4. Setting of SyncPos parameters
5. Check encoder connection and direction of rotation
6. Execute test run program
7. Optimizing the PID controller

## ■ Safety tips



The controller and the motor must be able to be switched off at any time with an EMERGENCY STOP button.



The motor must be able to turn completely freely so that a sudden jolt can not cause damage.

Furthermore, it is absolutely essential to be familiar with and observe the safety tips in the hardware manual.

## ■ Installing SyncPos



Insert disk 1 in the disk drive, in the task bar click on "START" and "RUN..." and enter the name of the drive and install, for example **a: \install**.

Follow the directions given during the installation program and list the hard drive on which SyncPos should be installed. SyncPos and the program samples included will be installed in the directory listed.

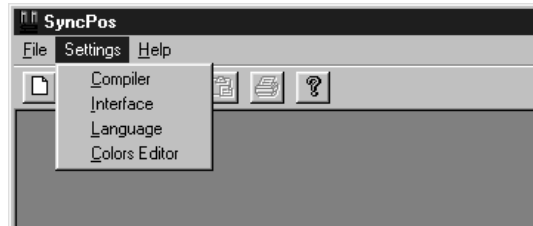
## ■ Starting SyncPos

Turn on the VLT, however, make sure the motor is not connected or it is not connected to a power source.

In the task bar, click on "START" → "PROGRAMS" → SyncPos.

### Changing the dialog language

Now the SyncPos window has been opened – English is the standard dialog language.



If you desire another language, click on "SETTINGS" → "LANGUAGE" (before you open a file) and select for example "GERMAN" in the dialog field which will subsequently appear. "EXIT PROGRAM" and start SyncPos again.

### Master Reset

If you press the [CANCEL] key on the VLT during the power-up, the SyncPos option card will not start a program even if the corresponding start conditions are fulfilled (auto recognition / Start input).

The SyncPos option card instead remains in the idle mode and waits for new commands. Error 19 User Interruption is triggered at the same time.

### Exit SyncPos

You can only abort or end a program with [ESC]. In order to do this the file which is linked with the controller resp. VLT must be open or re-opened.

**Setting of VLT parameters**

During booting the VLT parameters are set to the factory settings. Only the motor parameters remain to be set:

Perform AMA (see VLT5000/VLT5000Flux manual) or manual optimizing of the VLT to the connected motor.

Adjust maximum output frequency in par. 202 (Flux: output speed high limit) and maximum reference in par. 205 according to maximum velocity of the encoder. Note that the maximum output frequency must be higher than the frequency corresponding to maximum allowed velocity of the shaft because of the slip of the motor.

Terminal 27 must be connected to 24 V or parameter 502 must be set to "serial port".

Function of inputs and output must be selected in parameter group 3xx according to the required functions. Note that the default values are different from a drive without option.

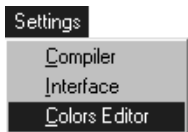
Please note that Dead Time Compensation in parameter 780 (not Flux) is set to OFF. This parameter is to prevent oscillation at standstill.

**Setting up communication**

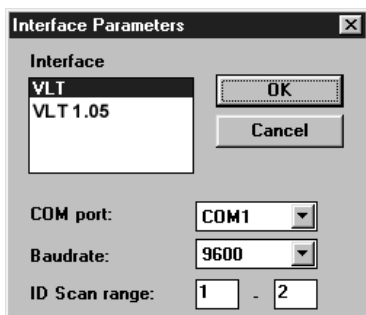
Before you begin...

Check whether the baud rate has been set in the VLT (par. 501); for a serial connection the baud rate is permanently set at the factory. In this case set up communication in your PC as follows:

Open an existing file or create a new one. Click "SETTINGS" → "INTERFACE".



The VLT and the baud rate are pre-set. Click on OK.



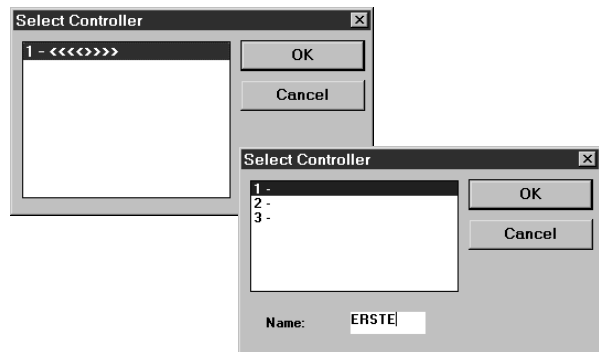
**NB!**

The baud rate in the VLT (Par 501) and in the program must always agree.

"SELECT CONTROLLER"



Click on "DEVELOPMENT" → "SELECT CONTROLLER", in the subsequent dialog field mark the VLT that you want to put into operation and click on "OK". For each VLT you have connected the address set on parameter 500 will automatically appear in the dialog field.



In addition to the addresses, you can also enter names for each controller in the menu "CONTROLLER" → "PARAMETER" → "NAME".

RS485 connection

You need the RS232 standard interface in the PC or an additional RS232 interface card and an external converter for a RS485 connection.

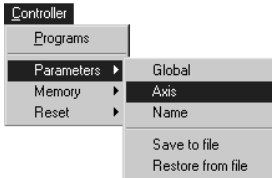
Ending communication Setup

A successful connection is reported in the communications window; the number and name of the controller is listed in the title bar of the actual file in addition to the file name.

### ■ Setting of SyncPos parameters

The following parameters must always be checked and if necessary adjusted. Depending on the requirements of the application it might be necessary to adjust other parameters as well.

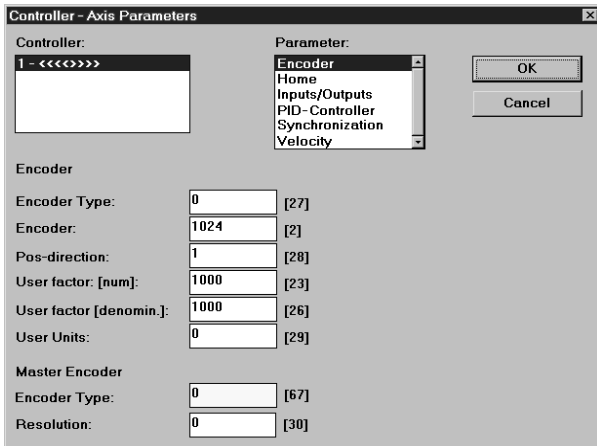
For the other parameters you can use the factory settings at first and then optimize the controller as needed at a later point in time with a "TESTRUN".



Click on "CONTROLLER" → "PARAMETER" → "AXIS" and select the VLT, of which you are currently adjusting the settings.

In the field **Parameter** mark the parameter group that you wish to define, for example **Encoder**, and enter the values in the corresponding fields.

Click on "OK" in order to load the new parameter values in the VLT and simultaneously save them.



For a detailed description of all global parameters and axis parameters please refer to the section [Parameter Reference](#) in the chapter Software Reference; for information concerning the use of dialog fields refer to Programming with SyncPos in the section menu "CONTROLLER" → "PARAMETERS" → "AXIS".

### ■ Setting of SyncPos parameters: Encoder

#### ENCODERTYPE (27)

Define the used type of encoder:

0 = incremental

1 = absolute encoder, standard ca. 262 kHz

2 = absolute Encoder, ca. 105 kHz

3 = absolute encoder without overflow (linear) but with error correction, approx. 262 kHz

4 = absolute encoder without overflow (linear) but with error correction, approx. 105 kHz

100 ... 104 = like 0 ... 4, however, the monitoring of the encoder will then be activated. If the encoder leads are interrupted, error 92 will be issued.

#### ENCODER (2)

Resolution of the encoder in pulses per revolution.

The following 2 parameters are only relevant for synchronizing applications:

#### MENCODERTYPE (67)

Define the used master encoder type:

0 = incremental

1 = absolute encoder, standard ca. 262 kHz

2 = absolute encoder, ca. 105 kHz

6 = The master position is not read by the encoder; instead, it is set with the system variable SYSVAR[4105].

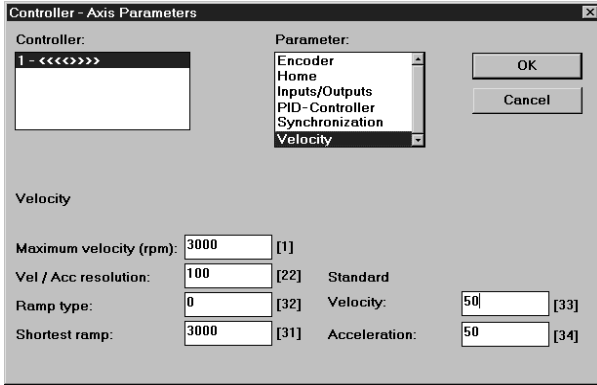
100 ... 102 = like 0 ... 2, however, the monitoring of the encoder will then be activated. If the encoder leads are interrupted, option error 92 will be issued

#### MENCODER (30)

Resolution of the master encoder in pulses per revolution.

■ **Setting of SyncPos parameters: Velocity**

This parameters you will find in the parameter group **Velocity** in the menu "CONTROLLER" → "PARAMETER" → "AXIS":



VELMAX (1)

**Maximum velocity** of the shaft where the encoder is mounted in RPM.



**NB!**

For synchronizing the setting must be at least the same as the maximum velocity of the master in order to be able to synchronize. For position synchronizing it must even be higher so that the slave can catch up lost position related to the master. All velocity commands (VEL, CVEL) are related to this value.

RAMPMIN (31)

The **Shortest ramp** is the time from 0 to maximum velocity and from maximum velocity to 0. All acceleration and deceleration commands (ACC, DEC) are related to this value.

■ **Setting of SyncPos option parameters: Home**

Homing is not necessary in standard synchronization applications and applications using an absolute encoder.

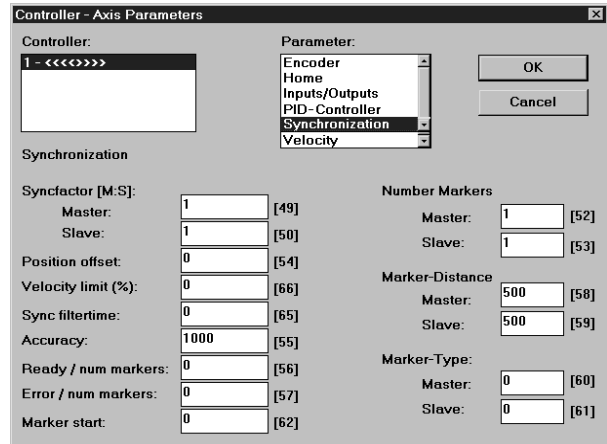
When using an incremental encoder the controller must be run to home after being switched on. During this process the reference switch defines the position at which 0 is located and how the VLT functions during a home run: input depends on the application.

HOME\_VEL (7)

**Home speed** is entered in % relative to the **maximum speed** of the drive. You can find these values in the description of the motor.

■ **Setting of SyncPos parameters: Synchronization**

The following parameters are only relevant for synchronizing applications: Open in the menu "CONTROLLER" → "PARAMETERS" → "AXIS" the parameter group **Synchronization**.



SYNCFACTM (49) and SYNCFAC (50)

The **Synchronizing factors master** and **slave** SYNCFACTM and SYNCFAC must be set according to the gear ratio between master and slave encoder.

Example: Both encoders has 1024 ppr, master is running 305 RPM and slave must run 1220 RPM.

SYNCFACTM = 305 and  
SYNCFAC = 1220.

Alternative: SYNCFACTM = 1  
SYNCFAC = 4.

The following parameters are only relevant when using synchronizing with marker correction (SYNCM).

Open the parameter group **Synchronization** in the menu "CONTROLLER" → "PARAMETERS" → "AXIS":

SYNCMARKM (52) and SYNCMARKS (53)

Number of master marker pulses and number of slave marker pulses.

SYNCMARKM and SYNCMARKS must be set according to the ratio between the number of marker signals from master and slave. A ratio of 1:1 means that each slave marker will be aligned with each master marker. A ratio of 2:1 means that each slave marker will be aligned with each second master marker.

### SYNCPULSM (58) and SYNCPULSS (59)

When using the encoder zero pulse as marker signal the distance between 2 markers is the resolution (qc) of the encoder.

When external marker signals are used, the marker distance can be measured by means of the program sample "Marker count" (see chapter 7) if it is unknown.

### SYNCMTPM (60) and SYNCMTPS (61)

Master marker type and Slave marker type.

Master marker signal: Input 5.

Slave marker signal: Input 6

Marker signal type must be selected for master and slave:

- 0 = index pulse (positive flank)
- 1 = index pulse (negative flank)
- 2 = external marker (positive flank)
- 3 = external marker (negative flank)

### ■ Checking encoder connection and direction of rotation

If you have not yet done so, now is the time to connect and test the encoder.



#### NB!

Remember to turn off the power before connecting the encoder.

Check the encoder connections by means of the encoder test program. In the menu bar click on "FILE" and "OPEN" the file **Enc-S.m**, which is the first test program for starting operation.

```

SyncPos - [Enc-S.m]
File Edit Development Controller Testrun Cam-Editor Settings Windows Help
/* Test program for testing encoder connection */
/* Encoder must be connected.
Load this program and run the motor forward in local mode
(parameter 002 = Local).
If the position is counting positive the encoder connection is ok.
If the position is counting negative, A and B track of the encoder
or the two motor phases must be swapped.
*/
/* Start of program */
MOTOR OFF /* Motor coast */
start:
PRINT "Position: ",apos /* read actual position */
WAIT 750 /* wait 750 ms */
GOTO start

```

In the "DEVELOPMENT" menu click on "EXECUTE" in order to start the test program. Run the drive forward for example in local mode (Parameter 002 = "Local") then the position must count positive. If the position is counting negative you must swap A and B channels from the encoder or two motor phases.

The position 0 is registered in the communications window.

If you turn the motor by hand (the motor should not be connected!), you can test whether the encoder functions: the position is continuously registered in the communications window. For a full rotation you should receive 4 times the value of the resolution of the encoder, that means 2000 if the Encoder counts per revolution is 500.

### Checking encoder for master synchronizing applications

If there is a master synchronizing application change the test program: replace the command APOS by MAPOS in "Enc-S.m" and run the master forward then the master position must count positive as well. If the position is counting negative you must swap A and B channel from the master encoder.

### Checking direction of rotation

For this the VLT parameter 002 must also be set and the motor is to be driven with a local set value (Par. 003).

By turning the axis it is possible to make sure that the direction of rotation is correct. When rotating to the right, as viewed from the front, looking towards the end of the axis, the impulse generator must count up.

Otherwise the encoder tracks A and B as well as A and B must be exchanged. Often it is easier to swap two motor-phases though.

Or you simply use the parameter **Positive Direction** POSDRCT (28) to invert the evaluation of the encoder information.

### Ending the encoder check

End the test of the encoder with the [ESC] key and close the test program with "FILE" → "CLOSE". A successful test of the encoder is a requirement for further starting up of operations.

### If the encoder doesn't work

This could be a result of incorrect cable installation. Measure the signals coming from the encoder and compare them to the values listed in the specifications. Check whether the connection was made according to the application.

### ■ Execute the test run program

Now connect the motor to the VLT, make sure that the motor can turn completely freely.



#### NB!

The motor must be provided with an [EMERGENCY STOP] button.

Click on "FILE" and "OPEN" the file **Move-S.m**.

```

SyncPos - [Move-S.m]
File Edit Development Controller Testrun Cam-Editor Settings Windows Help
** MOVETST.M - Positioning Test **
/* For this test, the encoder and the motor should both be fully
 connected to the controller unit.
 As with any early test, the motor should be installed so that sudden
 drastic movements or free running can't damage anything or harm
 anybody.
*/
DEF ORIGIN /* define current position as '0' */
ACC 10 /* set acceleration to 10% of the maximum */
VEL 10 /* set velocity to 10% of the maximum */
start: /* our little loop starts here */
 POSA 500 /* move axis to position '500' */
 WAITT 500 /* wait for 0.5s to allow for movement to complete */
 PRINT "Position: ",apos /* print current position */
 POSA 0 /* move axis to position '0' */
 WAITT 500 /* wait for 0.5s ... */
GOTO start /* do it again */

```

Click on "DEVELOPMENT" and start the test program with "EXECUTE" or [F5].

The test is successful if the motor runs slowly back and forth and position 500 is registered.

End the test with [ESC] and "CLOSE" the "FILE".

If the motor sets off uncontrolled or vibrates heavily



#### NB!

Turn off the motor immediately with the EMERGENCY STOP button if it vibrates heavily or sets off uncontrolled.

If the motor sets off uncontrolled, but the encoder test previously was successful, then decrease the **Proportional factor** (parameter 702). (See Optimizing the PID controller)

If the motor doesn't move

If the motor doesn't move at all, then the proportional factor of the PID filter is probably too low or the VLT has not been enabled.

Check the VLT enable (terminal 27 = 24 V) and check that the VLT wasn't stopped via the LCP (flashing display). Then increase the **Proportional factor** (parameter 702). (See Optimizing the PID controller)

If the motor vibrates heavily ...

... then you have to optimize the PID controller and to adjust the other parameters of the controller: reduce either the **Proportional factor** KPROP (11) or increase the **Derivative factor** KDER (12).

If a "tolerated position error is exceeded" is reported

If the drive stops due to a "tolerated position error is exceeded" message it is possible to determine whether the drive was rotating in the wrong direction by comparing the curves of the set and actual values.

Check the connections of the motor or encoder. If the connections are correct, then it is necessary to increase the **Tolerated position error** POSERR (15). (See Optimizing the PID controller)

## ■ Optimizing the PID controller

### ■ How the control process works

The "TESTRUN" explained in the chapter PC Software Interface can be used as a tool to optimize the SyncPos controller settings, thereby optimizing your system performance. To do that you only need to know a few things about the control scheme of SyncPos:

The SyncPos position controller has two parts:

1. The *Setpoint Generator* interprets the various positioning commands in SyncPos and generates a series of setpoint positions that eventually ends with the desired position. Normally, all positioning processes have a trapeze-shaped velocity curve. That means that after a phase of constant acceleration there is a phase with constant velocity and finally a phase with constant deceleration, which ends at the desired target position.
2. The *PID controller* receives the setpoint positions from the *Setpoint generator* and in turn calculates the speed reference needed for the motor to follow the current setpoint position. By setting the PID control parameters you can directly influence to what degree and how quickly a deviation from a theoretical set path (as specified by the setpoint series) should be counteracted.

The following behavior indicate that the control parameters are not optimally adjusted:

- Drive vibrates
- Drive is loud
- Frequent occurrence of position errors
- Poor control accuracy



**NB!** The control parameters are load-dependent. Thus the drive must be optimized under the actual conditions of use.

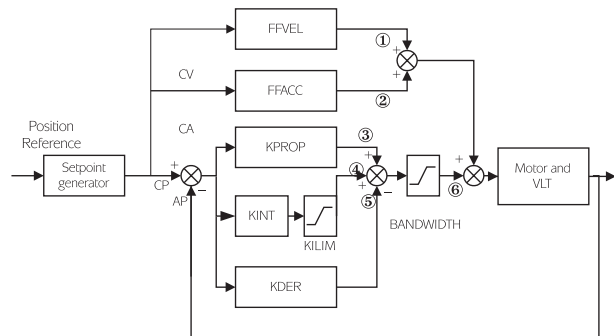
In exceptional cases it may be necessary to determine various sets of control parameters while working with heavily changing load conditions and to re-program them in subsequent application programs depending on the motion process.

### ■ Significance and influence of the control parameters

The PID control unit of the SyncPos option transfers the necessary output frequency via an internal speed reference to the VLT. This set value is periodically re-calculated with an interval of one millisecond (interval is programmable by the TIMER parameter). The SyncPos option is by default set with soft 'fit for all' controller parameters.

The PID filter works according to the following formula:

- ① = FFVEL \* (Setpoint velocity)
- ② = FFACC \* (Setpoint acceleration)
- ③ = KPROP \* (Position deviation)
- ④ = KINT \* (Sum of all previous position deviations) (limited by KILIM)
- ⑤ = KDER \* (Velocity of position deviation)
- = ③ + ④ + ⑤
- (limited by BANDWIDTH)



AV is the actual velocity

AP is the actual position (calculated from encoder feedback) in qc (Quad Counts).

CP is the current Setpoint position in qc.

CV is the Setpoint velocity in qc/ms.

(Position deviation) is calculated by CP-AP.



**NB!** In SYNCV mode the PID controller is working with speed deviation instead of position deviation. Speed deviation is calculated by CV-AV.

The controller in the SyncPos card utilizes two control strategies at the same time:

1. An open-loop feed-forward control. Since the asynchronous motor inherently has a good open loop performance the feed-forward control is a very important part of the controller in most applications. Benefits from using feed-forwards control is a very fast and accurate response to changes in the *setpoint reference*.

2. A closed-loop PID control. The PID controller monitors the difference between the actual position and the setpoint position. Based on this information it calculates a control signal to minimize the position deviance. Thus the SyncPos option is able to compensate for changes in load or friction. The PID controller is also necessary to compensate for any position deviance caused by inaccurate setting of the open-loop feed-forward controller.

In short: The feed-forward control is used to handle changes in the setpoint reference (especially important in synchronization applications), while the PID control is used to handle changes in load conditions or inaccuracies of the feed-forward control.

### Proportional factor KPROP (11)

The **Proportional factor** is multiplied with the position deviance and the result is added to the control signal (the internal speed-reference to the VLT). Since the calculated control signal is proportional to the position deviance (or error) this kind of control is called proportional control. The behavior of the proportional control is similar to that of a spring – the further the spring is extended the stronger the counter-force it produces.

#### Influence of the **Proportional factor**:

- |                 |                                                                                     |
|-----------------|-------------------------------------------------------------------------------------|
| KPROP too small | large position deviation due to non-compensatable load and frictional moment;       |
| KPROP larger    | quicker reaction, smaller steady-state deviation, larger overshoot, lesser damping; |
| KPROP too great | heavy vibrations, instability.                                                      |

### Derivative factor KDER (12)

The **Derivative factor** is multiplied with the derivative of the position deviance (the 'velocity' of the position deviance) and the result is added to the control signal. The behavior of the derivative control is similar to that of an absorber – the faster the absorber is extended the stronger the counter-force it produces. Thus using the **Derivative factor** increases damping in your system.

#### Influence of the **Derivative factor**:

- |                |                                                                                                                                                |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| KDER small     | no effect;                                                                                                                                     |
| KDER larger    | better dampening, lesser overshoot; if KPROP is increased simultaneously: faster reaction to control deviation at the same level of vibration; |
| KDER too large | heavy vibrations, instability.                                                                                                                 |

### Integral factor KINT (13)

The sum of all error is calculated every time the control signal is updated. The **Integral factor** is then multiplied with the sum of all positioning errors and added to the overall control signal. Thus in case that steady-state position errors occurs in your application, make sure you use the integral part of the controller. Steady-state errors will be levelled out as the summed error increases over time until the control signal eventually matches the load.

It is possible to set a limit for the control signal generated by the integral part of the controller (anti-windup).

#### Influence of the **Integral factor**:

- |                 |                                                                                  |
|-----------------|----------------------------------------------------------------------------------|
| KINT very small | steady-state position deviance is very slowly regulated to zero;                 |
| KINT larger     | faster regulation towards zero steady-state position deviance, larger overshoot; |
| KINT too large  | heavy vibrations, instability.                                                   |

### Integration limit KILIM (21)

The **Integration limit** sets a limit for the control signal generated by the integral part of the controller. This helps to prevent the so called 'wind-up' problems which typically occurs in applications where the overall control signal (the internal speed-reference) is maxed out for long periods of time.

This feature is also very helpful in applications where the power to the motor is turned off and on while the option card is controlling the VLT. Cutting of power to the motor (by setting terminal 27 low) while little positioning deviance is present in the controller, could result in an enormous control signal being generated once the power is turned back on.

### Velocity Feed-forward FFVEL (36)

The **Velocity feed-forward** factor is a scaling factor that is multiplied with the derivative of the setpoint position (the velocity of the setpoint). The result of this operation is added to the overall control signal. This feature is especially useful in applications where there is a good correlation between the control signal (the VLT speed reference), and the speed of the motor. This is indeed the case with most applications using an asynchronous motor.



#### NB!

The scaling of the FFVEL parameter is dependent on the correct setting of the maximum reference (VLT parameter #205) as well as the SyncPos parameters VELMAX (1) and ENCODER (2).

Acceleration Feed-forward FFACC (37)

The **Acceleration feed-forward factor** is multiplied with the 2nd derivative of the setpoint position (the acceleration of the setpoint) and the result is added to the control signal. This feature should be used to compensate for the torque used to accelerate/decelerate the system inertia.


**NB!**

Scaling of the FFACC factor is depending on the setting of the **Shortest ramp time** (31).

You should therefore increase the FFACC accordingly when decreasing SyncPos parameter

**Shortest ramp time** (31) and vice versa.

Sampling time for the entire control algorithm  
TIMER (14)

For particularly slow systems you can slow down the entire control system by entering multiples of 1 ms as the sampling time. However, it is important to remember that such a change influences all the control parameters!

Therefore, normally you should not deviate from the value of 1 ms.

BANDWIDTH

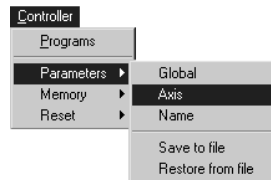
A **Bandwidth** of 1000 means that the set value is being executed 100%, thus **Derivative**, **Proportional** and **Integral factors** are effective as defined. But if you are operating a system which could be jeopardized by vibrations, for example, a crane with a heavy load then you can limit the bandwidth in which the PID controller should function. BANDWIDTH (35) of 300 makes a limitation of 30% possible. The build-up of a vibration is thus prevented since the control is only moved to with 30% of the calculated set value.

However, then it is necessary also to use the feed-forward part of the controller in order to achieve the corresponding control.

**■ Optimizing your controller settings  
step-by-step**

For best results use the functions in the **testrun** menu for this purpose; these help you to evaluate and optimize the PID controller on the basis of graphs of the set and actual curves.

However, we recommend only changing one value at a time and subsequently determining the improvement with a testrun.



Click on "CONTROLLER" → "PARAMETER" → "AXIS" and select the VLT, of which you are currently adjusting the settings.

Setting controller behavior

Before you adjust the controller parameters it is important to determine which controller behavior should be achieved.


**NB!**

The drive elements must never be operated outside of the technical specifications. Thus the maximum acceleration is determined by the "weakest" drive element.

- "Stiff" axis: the fastest reaction possible is mainly influenced by the **Proportional factor**. You can judge the results on the basis of the velocity graph.
- Damping of vibrations are mainly influenced by the **Derivative factor**. The results can best be assessed in the velocity graph.
- Temporary (static) deviations in position are mainly reduced by the **Integral factor** and can best be judged on the basis of the positioning graph.

**Ten steps for optimum control**


---

The following procedure will optimize your controller settings in most applications:

1. Make sure that you have specified correct values for VLT parameter #205, as well as the SyncPos parameters VELMAX (1), ENCODER (2) and RAMPMIN (31). If you change these settings again at a later point you may need to optimize the controller again.
2. Set the POSERR (15) parameter to a very high value e.g. 1000000 to avoid getting the option error no. 8 during the following tests.


**NB!**

To avoid damaging the system, make sure, that you set the POSERR within the limits of the system, because the position error monitoring is not active with extremely high values.

3. Optimize the **Velocity feed-forward** control:
  - Step 1) Execute a "TESTRUN" with KPROP=0, KDER=0, KINT=0, FFACC=0 and FFVEL=100.
  - Step 2) View the velocity profiles: If the actual velocity profile is *lower* than the setpoint velocity profile, return to the TESTRUN dialog, click "REPEAT" and increase FFVEL. Of course if the actual velocity profile is *higher* than the specified setpoint velocity you should decrease FFVEL.
  - Step 3) Run successive TESTRUNs until the two velocity profiles shown in the TESTRUN graph have the same maximum value.
  - Step 4) FFVEL is now optimized, save the current value.
4. In systems with large inertia and/or rapid changes in the reference velocity it is a good idea to use and optimize the **Acceleration feed-forward** control (make sure the inertial load is connected when optimizing this parameter):
  - Step 1) Execute a "TESTRUN" with KPROP=0, KDER=0, KINT=0, FFACC=0 and FFVEL at the optimized value found above. Use the highest possible acceleration setting. If RAMPMIN (31) is adjusted properly an acceleration value of 100 and a deceleration value of 100 should be sufficient. Start out with a low setting of FFACC approx. 10.

## Step 2)

View the velocity profiles: If during acceleration the actual velocity is constantly lower than the reference velocity profile, then click "REPEAT" and set a higher value of FFACC. Then execute "TESTRUN" again.

## Step 3)

Run successive TESTRUNs until the two velocity profiles shown in the TESTRUN graph have similar ramp-up and ramp-down curves.

## Step 4)

FFACC is now optimized, save the current value.

5. Next step is finding the maximum stable value of the **Proportional factor** in the PID controller:

## Step 1)

Execute a "TESTRUN" with KPROP=0, KDER=0, KINT=0. Set FFVEL and FFACC at the optimized values found above.

## Step 2)

View the velocity profile. If the velocity profile isn't oscillating then click "REPEAT" and increase KPROP.

## Step 3)

Run successive TESTRUNs until the actual velocity profiles is oscillating mildly.

## Step 4)

Decrease this "mildly" unstable KPROP value to about 70 %. Save this new value.

6. In order to dampen the oscillations created by the KPROP-part of the controller, the **Derivative factor** should now be optimized.

## Step 1)

Execute a "TESTRUN" with KINT=0 and KDER=200. Set FFVEL, FFACC and KPROP at the optimized values found above.

## Step 2)

Run successive TESTRUNs with increasing values of the KDER factor. At first the oscillations will gradually reduce. Stop increasing KDER when the oscillations begin to increase.

## Step 3)

Save the last value of KDER.

7. In any system that requires a zero steady-state error, the integration part of the controller must be used. Setting this parameter though is a trade-off between achieving zero steady-state error fast (which is good) and increasing overshoot and oscillations in the system (which is bad).

8. If you are using the integration part of the PID controller, remember to reduce the KILIM as much as possible (without losing the KINT-effect of course) in order to reduce oscillations and overshoot as much as possible.
9. Reduce the BANDWIDTH as much as possible. With a properly optimized open-loop control BANDWIDTH could be reduced to as little as 6 or 12 %.
10. Set the POSERR (15) parameter back to normal e.g. 20000.

"TEST PARAMETER" → "SAVE"

Once you have concluded the "TESTRUN" → "SAVE" the new parameters as the user parameters. Thus, these parameters are saved in the controller and in the future will be used for all programs.

### ■ What to do if...

... there is a tendency towards instability

In the event of a strong tendency towards instability reduce the proportional and derivative factors again, or reset the integral factor.

... stationary precision is required

If stationary precision is required then you must increase the integral factor.

... the tolerated position error is exceeded

If the test run is constantly interrupted with the message "position error" set the parameter for the **Tolerated position error** POSERR (15) – within the tolerable limits– as large as possible.

If the position error occur during the acceleration phase that suggests that the set acceleration cannot be achieved under the existing load conditions. Increase the **Tolerated position error** POSERR (15) or determine a maximum acceleration suitable for the entire system.

If position error do not occur until after the acceleration phase and they can be delayed but not eliminated by increasing the **Tolerated position error**, this suggests that the maximum velocity (rpm) chosen is too high. Determine a maximum velocity suitable for the entire system.

.. the maximum acceleration is not achieved

In general, the technical data for a drive are only valid for a freely rotating axle end. If the drive is carrying a load the maximum acceleration is reduced.

The theoretical maximum acceleration will also not be achieved if, for example, the PID controller output is too small or the VLT/motor is not sized correctly and therefore does not provide enough energy for peak consumption during acceleration.

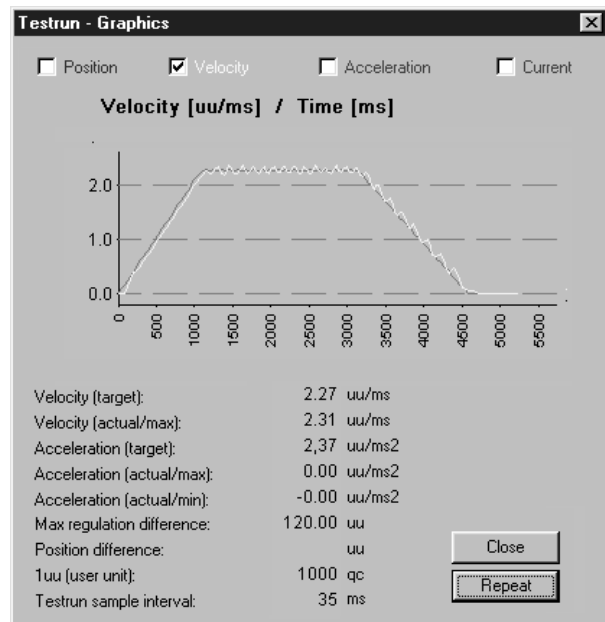
### ■ Examples of control optimization

Dampening of vibrations

| Testrun - Parameters    |      |
|-------------------------|------|
| Distance:               | 7500 |
| Velocity:               | 85   |
| Acceleration:           | 80   |
| Deceleration:           | 60   |
| Sampling Number:        | 150  |
| Sampling Interval (ms): | 35   |
| Proportional:           | 9000 |
| Derivative:             | 5    |
| Integral:               | 0    |
| Integration Limit:      | 100  |
| Sample interval:        | 1    |
| Bandwidth:              | 1000 |
| FF Velocity:            | 5    |
| FF Acceleration:        | 0    |

Buttons: Load, Save, Close, Start

The drive vibrates:

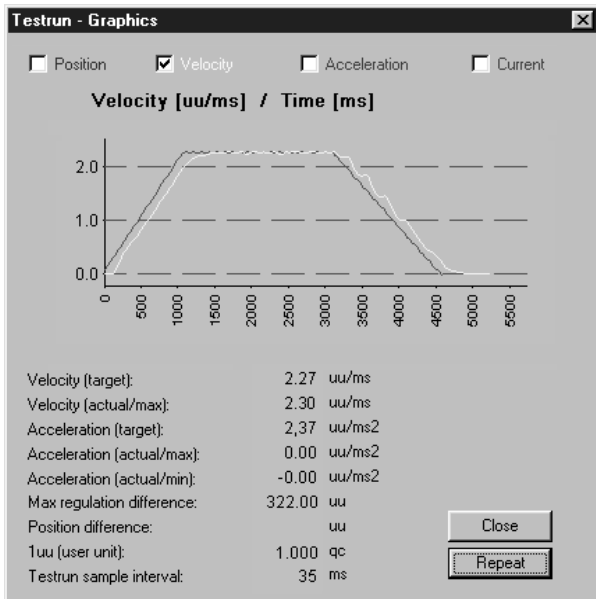


By increasing the **Derivative factor** to 5000 and reducing the **Proportional factor** to 3000 vibration during running and braking is lessened.

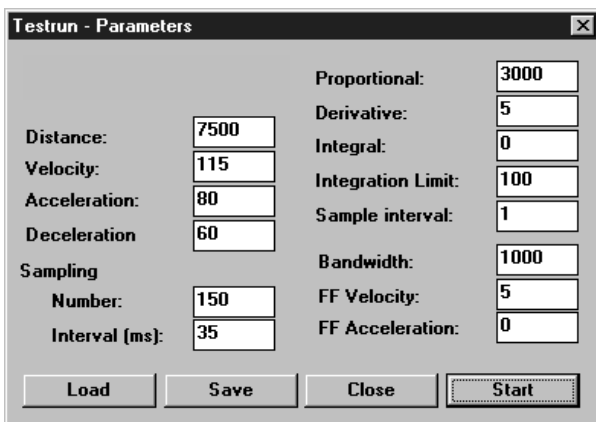
| Testrun - Parameters    |      |
|-------------------------|------|
| Distance:               | 7500 |
| Velocity:               | 85   |
| Acceleration:           | 80   |
| Deceleration:           | 60   |
| Sampling Number:        | 150  |
| Sampling Interval (ms): | 35   |
| Proportional:           | 3000 |
| Derivative:             | 5000 |
| Integral:               | 0    |
| Integration Limit:      | 100  |
| Sample interval:        | 1    |
| Bandwidth:              | 1000 |
| FF Velocity:            | 5000 |
| FF Acceleration:        | 0    |

Buttons: Load, Save, Close, Start

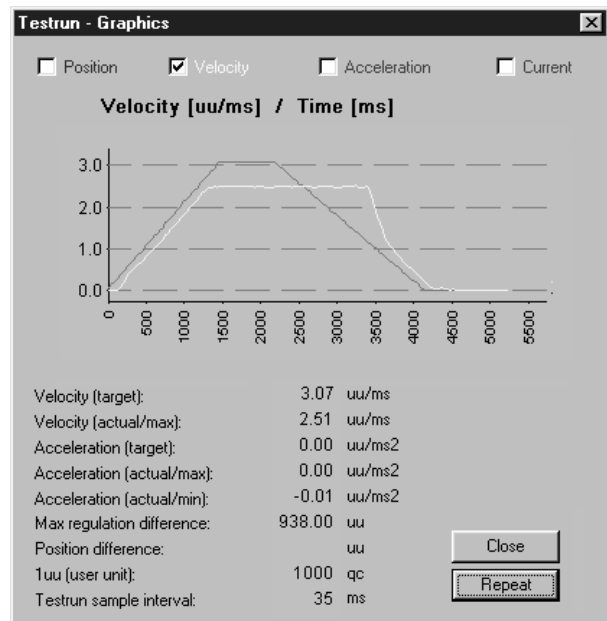
However, the drive now reacts somewhat slowly to changes in velocity, which leads to a deviation from the set curve at the beginning of the acceleration phase:



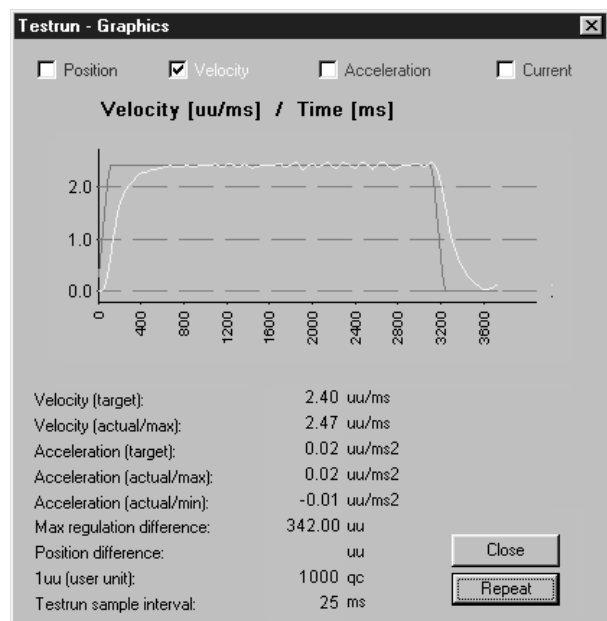
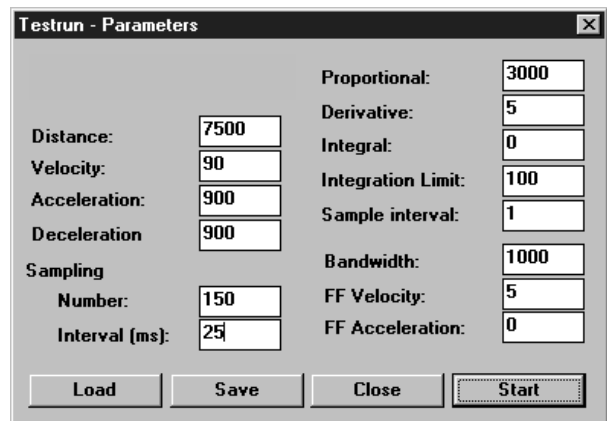
Determining the maximum velocity or maximum speed



The drive does not achieve maximum speed and thus starts to brake too late, since it is trying to catch up with the position reference.



By reducing the velocity to 90% of the maximum velocity and increasing acceleration and the deceleration to 900% the drive can achieve the required 2.47 UU/ms.



### ■ CAM Control

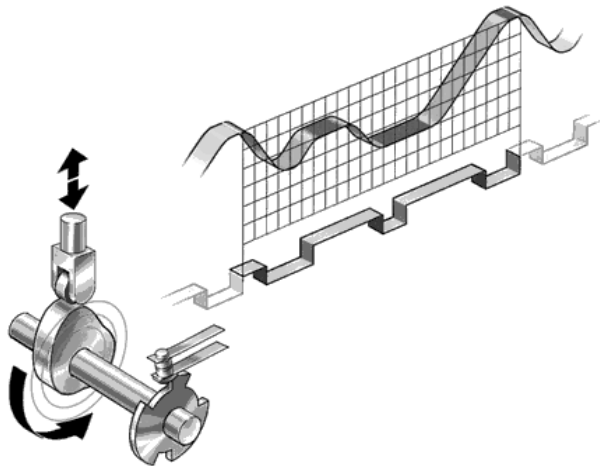
#### ■ How CAM Control with SyncPos operates

In order to realize CAM control, you need – depending on the application – at least one curve which describes the slave position in relation to the master position, as well as the engaging and disengaging behavior. Of course, many additional parameters are required for a CAM control which, together with the fixpoints of the curve, produce a curve profile.

The synchronization in CAM-Mode (SYNCC command) can also be performed with marker correction (SYNCCMM and SYNCCMS). This would be required, for example, if the products are transported irregularly on a conveyor belt, or if adding errors need to be compensated for.

#### Diagram of the principle

The mechanical cam disc and the mechanical camshaft are shown on the left, the curves for the electronic cam control and the electronic cam box are shown on the right:



In order to create the curve profile, use the → "CAM-EDITOR", into which you first load the VLT parameters that have already been set. Then you set the fixpoints of the curve and define the parameters required for your application. You can enter all values in physical or user-defined units under a Windows interface. You can constantly control the curve profile graphically; in this way, you can check velocity and acceleration of the slave axis.

#### Interpolation

The CAM-Editor calculates the curve from fixpoints with the help of a spline interpolation. This is optimized to a minimum torque. In order to prevent speed leaps in the case of repeated curve cycles, the velocity at the beginning and the end is equated. You can choose between two types of curve for this calculation. In either type, the interpolation takes account of the gradient of the curve at the beginning and the end. In the case of curve type 0, the gradient at the beginning and at the end is averaged; in the case of curve type 1, the gradient at the beginning of the curve is also used for the end of the curve.

#### Tangent Points for straight sections

For areas where the velocity must be constant and the acceleration = 0, you need to use tangent points. A straight line will be drawn instead of a spline between these points.

#### Accuracy

The fixpoints are used directly as interpolation points provided that this is permitted by the interval distance. The CAM-Editor performs a linear interpolation between the interpolation points. If a fixpoint is not hit by the selected interval distance, then the corresponding slave reference value does not exist in the interpolation table. You can avoid such deviations if you have activated →  "SNAP ON GRID".

#### Internal realization as array

The curve profiles are realized internally as arrays which you can call up by means of a DIM instruction and the SETCURVE command.

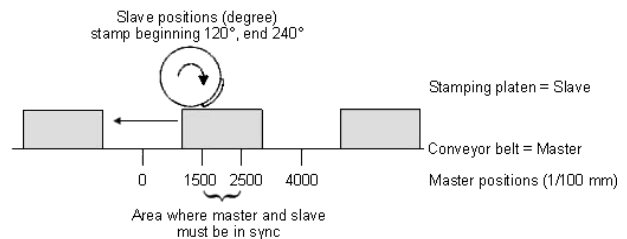
## ■ Quick Start Tutorial for impatient users

1. Start → "CAM-EDITOR" and load VLT parameters as cnf file into the CAM-Editor: "FILE" → "LOAD CNF".
2. Enter gearing factors or set user units MU and UU.
3. Enter at least three → "FIX POINTS" for master and slave in the index card with the same name.
4. Enter → "START STOP POINTS" for the engaging and disengaging.
5. Define the → "NUMBER OF INTERVALS" for a master cycle length in the index card "CURVE DATA".  
The → "INTERVAL TIME" in the index card "CURVE INFO" should not be less than 20 ms.
6. Define the → "SLAVE STOP POSITION" in the index card "CURVE INFO".
7. Enter the number of → "CYCLES/MIN MASTER" in the index card "CURVE INFO".
8. Check the velocity and acceleration of the slave with the help of the diagram.
9. Save curve as cnf file → "SAVE CNF AS" and load into the VLT.

## ■ Example: Stamping of boxes with use-by date

The following example explains step by step how to edit the curve for this application of the CAM control and then how to incorporate it into your control program.

A platen is supposed to stamp an inscription with a length of 10 cm on cardboard boxes. The stamp corresponds to a platen section of 120 degrees. 60 cardboard boxes are transported on the conveyor belt per minute. The cardboard boxes are always transported on the conveyor belt at exactly the same distance from each other (e.g. by means of a mechanical set pattern). During the printing process, the stamping platen and the cardboard box must run in sync:



### How to edit the curve step by step

1. Set the VLT with the required parameters and save these user parameters with "PARAMETERS" → "SAVE TO FILE" with the extension „cnf“.
2. Start the → "CAM-EDITOR" and load this cnf file with File → "LOAD CNF".
3. Determine the gearing factor of the master in MU units.  
The input should be possible in 1/10 mm resolution.  
The drive is connected with the conveyor belt by means of a gearing of 25:11; i.e. the motor makes 25, the drive pulley 11 revolutions.  
Gearing factor = 25/11  
Incremental encoder directly on the master drive; encoder resolution = 4096.  
The drive pulley has 20 teeth/revolution, 2 teeth correspond to 10 mm, thus 1 revolution corresponds to = 100 mm conveyor belt feed or 1000/10 mm.  
Thus, the scaling factor is 1000.

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} \text{ qc} = 1 \text{ MU}$$

$$\frac{25/11 * 4096 * 4}{1000} qc = 1 \text{ MU}$$

$$\frac{25 * 4096 * 4}{1000 * 11} qc = \frac{2048}{55} qc = 1 \text{ MU} = \frac{\text{SYNCFAC TM (49)}}{\text{SYNCFAC TS (50)}}$$

Enter these values in the index card → "SYNCHRONIZATION" (the selected units should always be whole numbers):

Syncfactor Master [49] = 2048  
 Syncfactor Slave [50] = 55

- Enter the gearing factor of the slave in UU units:  
 Gearing factor = 5/1  
 Encoder resolution (incremental encoder) = 500  
 One revolution of the platen is 360 degrees.  
 We are going to work with a resolution of 1/10 degrees. This means that we are dividing one revolution of the platen into 3600 work units:

Scaling factor = 3600

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} qc = 1 \text{ UU}$$

$$\frac{5/100 * 500 * 4}{3600} qc = 1 \text{ UU}$$

$$\frac{5 * 500 * 4}{3600} qc = \frac{25}{9} qc = 1 \text{ UU} = \frac{\text{POSFACT\_Z (23)}}{\text{POSFACT\_N (26)}}$$

Enter these whole number values in the index card → "ENCODER":

User factor numerator [23] = 25  
 User factor denominator [26] = 9

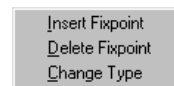
- Determine a whole number factor for the intervals in the index card → "CURVE DATA" so that the fixpoints are on the interpolation points.  
 A complete cycle length of the master is 400 mm; this corresponds to 4000 MU.  
 The → "NUMBER OF INTERVALS" = 40 results in a reasonable interval time of 25 ms.

- Define → "FIX POINTS" for the conveyor belt (master) and the platen (slave) and the point type „1“ for curve points. Click on the → "RECALC" button in order to depict the curve. The function → "SNAP ON GRID" should be activated.

| Fix Points |        | Start Stop Points |      |
|------------|--------|-------------------|------|
| Punkt      | Master | Slave             | Type |
| 1          | 0      | 0                 | 1    |
| 2          | 1500   | 1200              | 1    |
| 3          | 2500   | 2400              | 1    |
| 4          | 4000   | 3600              | 1    |
| 5          |        |                   | 1    |
| 6          |        |                   | 1    |

- Master and slave must run synchronously with the same velocity between the position 1500 and 2500. This requires a straight line which is determined by two tangent points. Enter „2“ for tangent point in the column → "TYPE" for the two positions in the index card → "FIX POINTS".

Alternatively, you can move the cursor to the fixpoint 2500, click on the right mouse button and select → "CHANGE TYPE" in the subsequent pop-up menu. Since two tangent points are always required, the previous one (left) will also be changed at the same time.

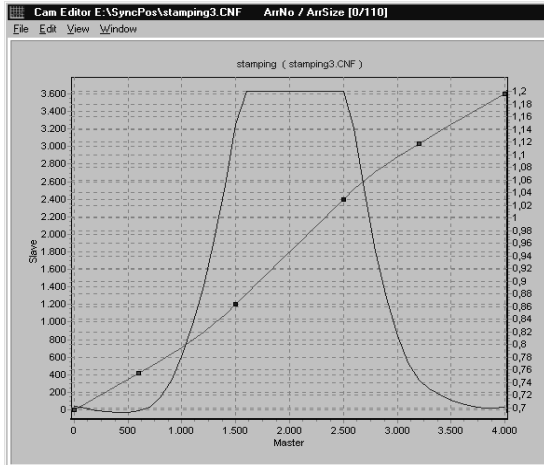


- The curve will not be shown because the calculation of a spline always requires three points. In this case, you will therefore need an additional fixpoint before and after the tangent (straight section).

**You can enter points in the table:** Click on the row of the second fixpoints with the right mouse button and select → "INSERT FIXPOINT" in the pop-up menu that follows. The point will be inserted between this point and the previous one (left).

**Alternatively, you can enter points in the diagram:** Move the cursor to the (now) fifth point until the hand icon appears. Click on the right mouse button and select → "INSERT FIXPOINT" in the pop-up menu.

Activate the diagram of the →  "VELOCITY" and move the two newly inserted fixpoints with the help of the mouse until you get a uniform velocity:



9. Enter the → "CYCLES/MIN MASTER" = 60 in the index card → "CURVE INFO". This is the (maximum) number of cardboard boxes that are processed per minute.
10. Verify whether the acceleration of the slave is within the limit. For this purpose, you must activate the illustration of the →  "ACCELERATION" and of the →  "ACC. LIMIT".
11. In order to load the curve into your control system, you must first save the file as a cnf file "FILE" → "SAVE AS ..." . You will see the name of the curve and the number of array elements in the title bar. You will need the latter for the DIM instruction during programming.

### How to load the curve into the VLT

You can load the cnf file with the modified parameters and the – automatically generated – curve arrays into the VLT by means of "PARAMETERS" → "RESTORE FROM FILE".

How to integrate the curve into the control program  
Since the curve is stored internally as an array, the DIM instruction must appear first in your program:

DIM stamping[92]

// Number of elements from the title bar  
// of the CAM-Editor

HOME

// Slave axis performs a home run  
// (switch for zero position on top)  
// Afterwards, the slave will be in the  
// zero position (0 degrees)  
// (Omitted if an absolute encoder is used)

SETCURVE stamping

// Set stamping curve  
// Assumption: A box is positioned at the  
// processing point with its front edge  
// and the master stands idle

DEFMCPOS 1000 // 1000 corresponds to this  
// position (front edge of box)

POSA CURVEPOS

// Bring slave to the curve position that  
// corresponds to the master position

SYNCC 0

// Change into and remain in CAM-Mode

SYNCCSTART 0

// Engage platen immediately with the set  
// maximum velocity  
// This does not cause any movement since  
// the master is idle and in the correct position  
// Now the master can be started

start:

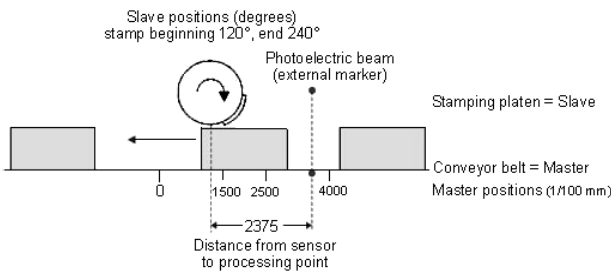
// Empty main loop so that program will  
// not be terminated  
// Additional processing could take place here  
goto start

### Example: Printing of cardboard boxes with Marker Correction

If the boxes are not always transported at the exact same distance from each other, markers will be required that can recognize a box and correct the synchronization.

The following section describes how you can adapt the curve of the previous example to this application.

Again, a platen is supposed to stamp an inscription with a length of 10 cm on cardboard boxes. A maximum of 60 boxes are transported on the conveyor belt per minute. During the printing operation, both the stamping platen and the box must run in sync.



#### How to edit a curve for Marker Synchronization

- Steps 1 to 9 are the same as in the previous example.
- Define the point pairs for the engaging and disengaging in the index card → "START STOP POINTS". We make the assumption that engaging takes place at the beginning of the box and disengaging is to take place until the end of the box.

| Fix Points |        | Start Stop Points |  |
|------------|--------|-------------------|--|
| Punkt      | Master |                   |  |
| 1 a        | 1000   |                   |  |
| 1 b        | 1500   |                   |  |
| 2 a        | 2500   |                   |  |
| 2 b        | 3000   |                   |  |
| 3 a        |        |                   |  |
| 3 b        |        |                   |  |

- In the index card → "CURVE DATA", determine the position in which the platen should stop if no other Slave Stop Position is defined in the program.  
The platen always needs to return to the position 0 degrees:  
→ "SLAVE STOP POSITION" = 0

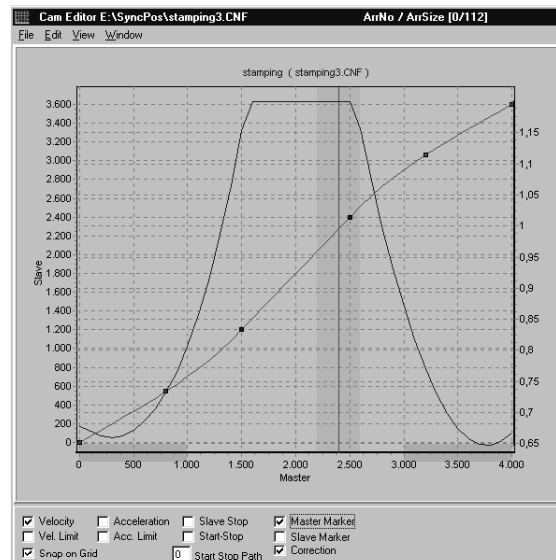
- The photoelectric beam (external marker) has a distance of 237.5 mm from the processing point (= stamp touches the cardboard box) and detects the beginning of the box (corresponding to master position 1000). The marker distance is therefore 2375. Enter this value in the index card → "SYNCHRONIZATION" and define the permitted tolerance for the appearance of the markers and the external marker type = 2 for the master:  
Master Marker

Distance      SYNCMPULSM (58) = 2375  
Tolerance     SYNCMWINM (68) = 200  
Typ            SYNCTYPM (60) = 2

Enter the master position in the index card → "CURVE DATA":

Master-Marker-Position = 1000

- Take a look at the curve profile in order to determine when the correction of the synchronization may begin at the earliest and when it must be finished. The vertical green line indicates the master position where the marker is recognized, the light green area shows the tolerance window for the appearance of the master marker.



At the earliest, the correction may begin when the printing of a box has been completed, since any change of velocity during the printing process would damage the box. The correction must be finished completely when the next box reaches the processing point. In this example, the master positions at the end and beginning of a box are quite suitable:

Correction Start = 3000  
Correction End = 1000

Enter the values in the index card → "CURVE DATA"; the depiction of the area is shaded in blue in the curve profile.

15. Verify whether the velocity and acceleration of the slave remain within the limit. For this purpose, you must activate the illustration of the →  "VELOCITY" and of the →  "VEL.-LIMIT" and then the illustration of the →  "ACCELERATION" and of the →  "ACC.-LIMIT".

16. "FILE" → "SAVE as CNF"

### How to load the curve into the VLT

You can load the cnf file with the modified parameters and the – automatically generated – curve arrays into the VLT by means of "PARAMETERS" → "RESTORE FROM FILE".

### How to integrate the curve "marker synchronization" into the control program

Since the curve is stored internally as an array, the DIM instruction must appear first in your program:

```

DIM marker[112] // See number of elements in
 // the title bar of CAM-Editor
HOME
 // Slave axis performs a home run (switch for
 // zero position on top); afterwards, the slave
 // will be in the zero position (0 degrees)
 // (Omitted if an absolute encoder is used)
SETCURVE marker
 // Set stamp curve with marker
dist = GET SYNCMPULSM // Distance to sensor
DEFMCPPOS (1000-dist)
 // This is the location that corresponds
 // to the sensor signal
SET SYNCMSTART 2000
 // Counting of the master pulse does not begin
 // until the next edge comes from the sensor
SYNCCMM 0
 // Synchronize in CAM-Mode until motor stop
SYNCCSTART 1
 // Engage platen with start point pair 1

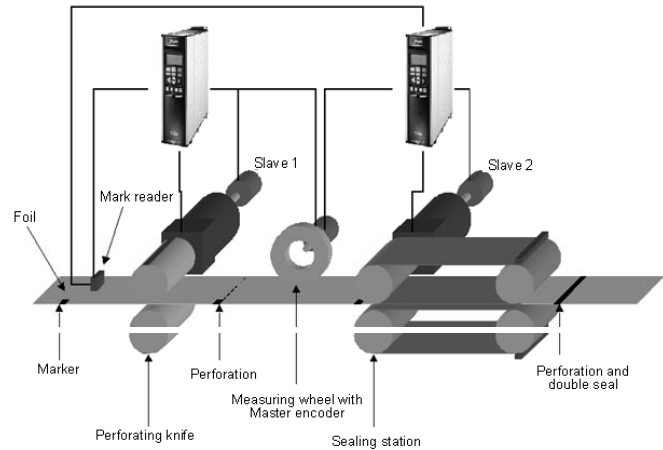
 // Synchronous operation

WAITI 4 ON
 // Wait for input signal when conveyor belt
 // is being switched off
SYNCCSTOP 2 0
 // Disengage platen with stop point pair 1
 // and stop at position 0 degrees

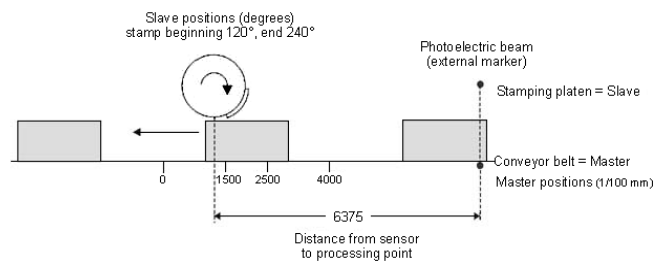
```

### ■ If the sensor distance is larger than one master cycle length

In many applications, the marker cannot be placed within one master cycle length, for example in the case of the following equipment for the production of plastic bags:



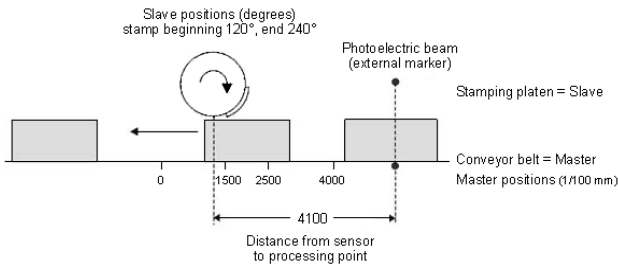
Since no markers can be installed between the slaves here, there is only a marker reader in this application; the welding station is much farther away than one master cycle length. Since the distance of the sensor is larger than one master cycle length, a buffer is provided for the marker deviation. When the marker appears, the value is entered in the buffer and read out upon the appearance of the next marker:



In order to assess in what area corrections may be made, you should subtract the master cycle length as often as necessary until the value is < 1 master cycle length. This is the maximum permitted distance for making corrections. In this example, it is  $6375 - 4000 = 2375$  and thus the same correction range as in the previous example.

### ■ Problematic situations in the determination of the marker distance

If the marker is mounted so close to the processing point that there is no time left to correct the synchronization after the marker has been detected, then you can remedy the problem only through mechanical modification of the marker. On the other hand, the same effect might also occur if the marker distance is larger than the master cycle length and if an insufficient distance likewise remains after the subtraction of this value, for example:



The value is entered into the buffer when the marker appears. The buffer is read out only when the next marker is recognized. This means that the marker is only "recognized" at the master position 900 and that there is little time left in our example to correct the error. It is the same effect as if the sensor had been mounted at the value (distance – master cycle length) or (4100 – 4000), respectively, i.e. only 10 mm in front of the processing point.

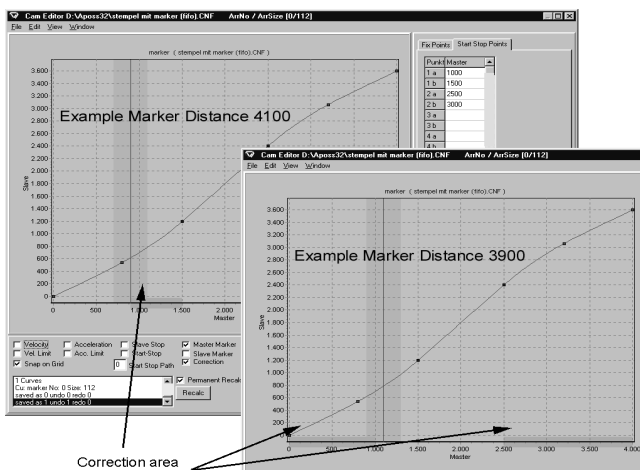
Alternatively, the sensor could be installed further away, for example at a distance of 7900; this has the same effect as if the sensor had been installed at distance – master cycle length (7900 – 4000), i.e. 3900 in front of the processing point. This allows enough time to correct the synchronization. If this cannot be done mechanically, then the values must be manipulated somewhat in order to avoid the solution with the buffer. Please proceed as follows:

Subtract a value x from the actual distance so that the distance becomes < than the master cycle length, for example 4100 – 200 = 3900. You also subtract the value x from the master position, i.e. 1000 – 200 = 800.

Enter both values in the index cards → "SYNCHRONIZATION" and → "CURVE DATA":

Master Marker Distance  
 SYNCMPULSM(58) = 3900  
 Master Marker Position = 800

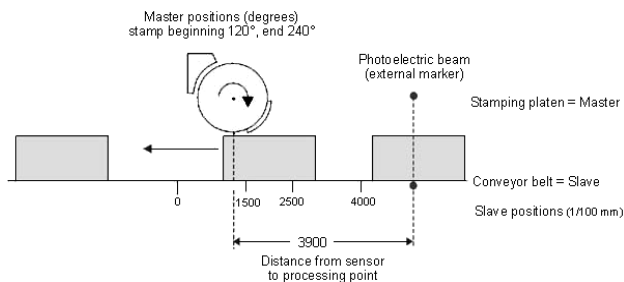
Since no buffer is generated now, it would be possible to correct from 2500 to 800, for example.



Thus, it would be better to install the sensor in such a way that the distance to the processing point is either smaller or substantially larger than one master cycle length; here, for example, at a distance of 3900. Then it is possible to correct from 2500 to 1000.

### ■ Example: Slave Synchronization with Marker

In the following example, the conveyor belt is the slave and the stamp platen the master, since the take-up and delivery of the dye must be continuous for a uniform printing process. A maximum of 20 cardboard boxes are transported on the conveyor belt per minute. The distance of the boxes is not larger than one master cycle length. Stamp platen and box must run in sync during the printing operation:



In contrast to the synchronization with master marker correction, here the slave position is being corrected instead of the curve.

#### How to edit a curve for the Slave Synchronization

1. Set the VLT with the required parameters and save these user parameters with "PARAMETERS" → "SAVE TO FILE" with the extension „cnf“.
2. Start the → "CAM-EDITOR" and load this cnf file with "FILE" → "LOAD CNF".
3. Determine the gearing factor of the master in MU units.  
 Gearing factor = 5/1  
 Encoder resolution (Increm. encoder) = 500  
 One revolution of the platen is 360 degrees.  
 We are going to work with a resolution of 1/10 degrees. Therefore we are dividing one revolution of the platen into 3600 work units:  
 Scaling factor = 3600

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} \quad q_c = 1 \text{ MU}$$

Enter these whole number values in the index card → "SYNCHRONIZATION":

- Syncfactor Master [49] = 25
- Syncfactor Slave [50] = 9

4. Enter the gearing factor of the slave in user untis [UU]:

The input should be possible in 1/10 mm resolution.

The drive is connected with the conveyor belt by means of a gearing of 25:11; i.e. the motor makes 25, the drive pulley 11 revolutions.  
 Gearing factor = 25/11

Incremental encoder directly on the master drive; encoder resolution = 4096

The drive pulley has 20 teeth/revolution, 2 teeth correspond to 10 mm, thus 1 revolution corresponds to = 100 mm conveyor belt feed or 1000/10 mm.

Thus, the scaling factor is 1000

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} \quad q_c = 1 \text{ UU}$$

Enter these values in the index card → "ENCODER":

- User factor numerator [23] = 2048
- User factor denominator [26] = 55

5. Determine a whole number factor for the intervals in the index card → "CURVE DATA" so that the fixpoints are on the interpolation points.

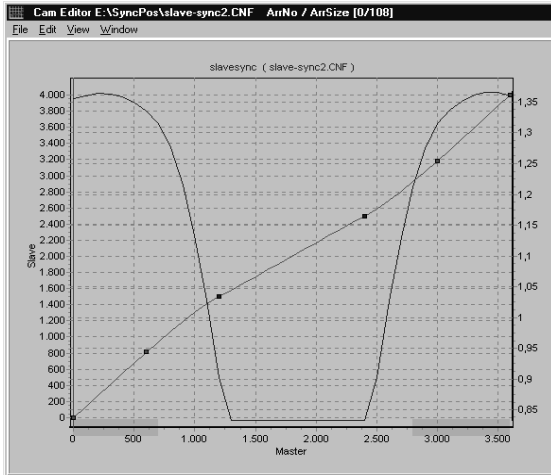
A complete cycle length of the master is 360 degrees; this corresponds to 3600 MU. The → "NUMBER OF INTERVALS" = 36 produces a reasonable interval time of 27.7ms.

6. Define → "FIX POINTS" for the platen (slave) and the conveyor belt (master) and the point type „1“ for curve points. The function → "SNAP ON GRID" should be activated.

| Points | Master | Slave | Type |
|--------|--------|-------|------|
| 1      | 0      | 0     | 1    |
| 2      | 1200   | 1500  | 1    |
| 3      | 2400   | 2500  | 1    |
| 4      | 3600   | 4000  | 1    |

7. Master and slave must run synchronously with the same velocity between the master position 1200 to 2400. For this, you need to have a straight line which is determined with two tangent points. Thus, for these two fixpoints you must define the → "TYPE" = 2 for tangent point and insert two fixpoints so that the curve can be calculated.

Activate the diagram of the →  "VELOCITY" and move the two newly inserted fixpoints with the help of the mouse until you get a uniform velocity progression:



8. Enter the → "CYCLES/MIN MASTER" = 20 in the index card → "CURVE INFO". This is the (maximum) number of cardboard boxes that can be processed per minute.
9. Verify whether the acceleration of the slave is within the limit. For this purpose, you must activate the illustration of the →  "ACCELERATION" and of the →  "ACC. LIMIT".
10. Define in the index card → "START STOP POINTS" with some safe distance in order to start the synchronization at the beginning. Engaging should take place between 20 and 100 degrees because it must be completed at 120 degrees.

| Points | Master |
|--------|--------|
| 1a     | 200    |
| 1b     | 1000   |

11. In the index card → "CURVE DATA", define the position where the conveyor belt should stop if no other Slave Stop Position is being defined in the program:  
The conveyor belt should always stop in position 0:  
→ "SLAVE STOP POSITION" = 0

12. The photoelectric beam (external marker) has a distance of 390 mm from the processing point (= stamp touches the box) and detects the beginning of the box (corresponds to slave position 1000). Thus, the marker distance is 3900. Enter this value in the index card → "SYNCHRONIZATION" and define the permitted tolerance for the appearance of the markers and the external marker type = 2 for the slave:

Slave Marker

Distance SYNCMPULSM (59) = 3900

Tolerance SYNCMWINM (69) = 200

Type SYNCTYPM (61) = 2

Enter the values in the index card → "CURVE DATA":

Slave-Marker-Position = 1000

14. Take a look at the curve profile in order to determine when the correction of the synchronization may begin at the earliest and when it must be finished. The green horizontal line indicates the master position where the marker is recognized, the light green area shows the tolerance window for the appearance of the master marker.  
At the earliest, the correction may begin when the printing of a cardboard box has been completed, since any change of velocity during the printing process would damage the printing stamp and/or the box. Also, the correction must have been completed in its entirety when the next box arrives at the processing point. In this example, the slave positions at the end and beginning of a box are quite suitable.

Enter the values in the index card → "CURVE DATA":

Correction Start = 2800

Correction End = 750

15. Verify whether the velocity and acceleration of the slave remain within the limit. For this purpose, you must activate the illustration of the →  "VELOCITY" and of the →  "VEL.-LIMIT" and then the illustration of the →  "ACCELERATION" and of the →  "ACC.-LIMIT".
16. "FILE" → "SAVE as CNF"

How to load the curve into the VLT

You can load the cnf file with the modified parameters and the – automatically generated – curve arrays into the VLT by means of "PARAMETERS" → "RESTORE FROM FILE".

How to integrate the curve "slave synchronization" into the control program

In order to determine the master position, a switch on the master is required that indicates the zero position. In order to put the slave into the correct position, it will be moved forward to the photoelectric beam. This corresponds to the beginning of the box = 1000. Then the slave will be moved further by 2900 (= marker distance 3900–1000); thus, the slave is exactly in front of the processing point with the beginning of the cardboard box 1000, i.e. at slave position 0.

```

DIM slavesync[108]
 // see number of elements in the title bar
 // of the CAM-Editor window
HOME
 // Slave axis performs a home run
 // (switch for zero position on top)
 // Afterwards, the slave will be in the
 // zero position (0 degrees)
 // (Omitted if an absolute encoder is used)
DEFMCPOS 0
 // Curve starts at master position 0
SET SYNCMSTART 2000
 // Counting of master pulse does not begin
 // until the next edge comes from the sensor
SETCURVE slavesync
 // Set curve for the slave synchronization

 // Go to start
CSTART
CVEL 10
 // Go forward slowly until photoelectric
 // beam appears
oldi = IPOS
 // oldi = last marker position of the slave
WHILE (oldi == IPOS) DO
 // Wait until box is detected
ENDWHILE
POSA (IPOS + 2900)
 // Move box forward by 2900
SYNCCMS 0
 // Synchronize in CAM-Mode
SYNCCSTART 1
 // Engage with start-stop point pair

```

### ■ CAM Box

#### ■ How a CAM box with SyncPos operates

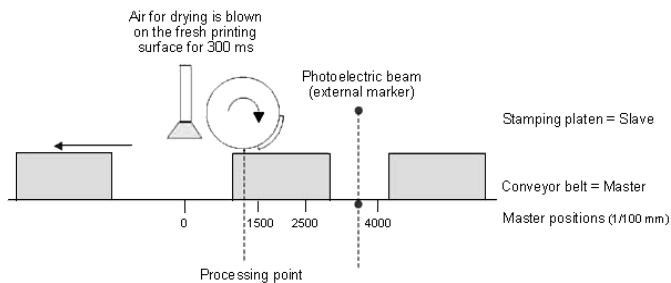
The mechanical camshaft is also reproduced by one (or more) curves. In order to realize a CAM box, it must be possible to engage and disengage the slave at specific master positions over and over again.

This is possible with SyncPos with the interrupt command ON MAPOS .. GOSUB and ON APOS .. GOSUB. It is possible to call up a subprogram whenever a defined master position has been passed (both in the positive or negative direction, to be precise).

It is possible to realize many applications that are typical for the packaging industry in connection with a curve profile in which several have been defined for engaging and disengaging.

#### ■ Example of a CAM Box

After a cardboard box has been printed, the fresh print is to be dried immediately in the air stream:



ON MCPOS 2500 GOSUB drier

```
// Call up a subprogram when the master
// position 2500 is passed in
// positive direction
```

SUBMAINPROG

SUBPROG drier

```
OUT 1 1 // Turn on drier
DELAY 300 // Dry for 300 ms
OUT 1 0 // Turn off drier
```

RETURN

ENDPROG

**Chapter 5**
**■ PC Software Interface**
**■ "FILE" menu**

|                                     |    |
|-------------------------------------|----|
| "NEW" .....                         | 51 |
| "OPEN" .....                        | 51 |
| "CLOSE" .....                       | 51 |
| "SAVE" and "SAVE AS" .....          | 52 |
| Printing and printer settings ..... | 52 |
| "EXIT PROGRAM" .....                | 52 |
| Delete File .....                   | 52 |

**■ "EDIT" menu**

|                            |    |
|----------------------------|----|
| "UNDO" .....               | 53 |
| "CUT" .....                | 53 |
| "COPY" and "PASTE" .....   | 53 |
| "FIND" and "REPLACE" ..... | 53 |

**■ "DEVELOPMENT" menu**

|                                               |    |
|-----------------------------------------------|----|
| "EXECUTE" .....                               | 54 |
| Run programs in several VLTs .....            | 55 |
| "BREAK" .....                                 | 55 |
| "CONTINUE" program .....                      | 55 |
| Preparing "SINGLE-STEP" .....                 | 55 |
| Set Breakpoints .....                         | 55 |
| Change and read variables online .....        | 56 |
| "GO" (Debug) and "SINGLESTEP" .....           | 56 |
| "END DEBUG" .....                             | 56 |
| "SHOW WATCH" .....                            | 56 |
| "SYNTAX CHECK" .....                          | 57 |
| "BREAK ALL" .....                             | 57 |
| "SELECT CONTROLLER" .....                     | 57 |
| "COMMAND LIST" .....                          | 57 |
| "POSITION DRIVE" (Teach-in-programming) ..... | 58 |

**■ "CONTROLLER" menu**

|                                                     |    |
|-----------------------------------------------------|----|
| "PROGRAMS" .....                                    | 58 |
| "AUTOSTART" .....                                   | 59 |
| "CONTROLLER" → "PARAMETERS" .....                   | 60 |
| Global parameters .....                             | 60 |
| Axis parameters .....                               | 61 |
| Set or change axis parameters online .....          | 61 |
| How to change parameters of a configuration file .. | 62 |
| "PARAMETERS" → "NAME" .....                         | 63 |
| "PARAMETERS" → "SAVE TO FILE" .....                 | 63 |
| "PARAMETERS" → "RESTORE FROM FILE" .....            | 63 |
| "MEMORY" → ... .....                                | 63 |
| "RESET" → ... .....                                 | 64 |

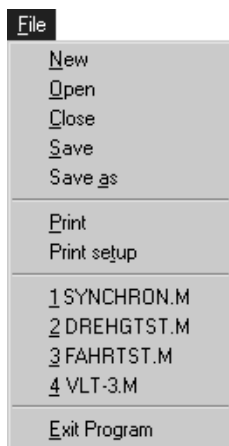
|                                                               |           |
|---------------------------------------------------------------|-----------|
| ■ <b>"TESTRUN" menu</b>                                       |           |
| Setting parameters for a "TESTRUN" .....                      | 65        |
| "EXECUTE TESTRUN" .....                                       | 66        |
| "DISPLAY RECORDING" .....                                     | 67        |
| Evaluation motion figures .....                               | 67        |
| ■ <b>"CAM-EDITOR" menu</b>                                    |           |
| CAM-Editor Window .....                                       | 68        |
| "FILE" menu CNF .....                                         | 70        |
| "EDIT" curve menu .....                                       | 71        |
| "VIEW" parameter menu .....                                   | 71        |
| "WINDOW" CAM menu .....                                       | 72        |
| CAM Profile .....                                             | 72        |
| CAM-Editor Messages .....                                     | 73        |
| Index Card "FIX POINTS" .....                                 | 73        |
| Index Card "START STOP POINTS" .....                          | 74        |
| Index Cards "CURVE DATA", "CURVE INFO"<br>and Parameter ..... | 75        |
| Index Card "CURVE DATA" .....                                 | 75        |
| Index Card "CURVE INFO" .....                                 | 76        |
| Index Cards Encoder, Home, Inputs/Outputs<br>and PID .....    | 76        |
| Index Card Synchronization .....                              | 77        |
| Index Card Velocity .....                                     | 77        |
| ■ <b>"SETTINGS" menu</b>                                      |           |
| "COMPILER" .....                                              | 77        |
| "INTERFACE" .....                                             | 78        |
| "COLORS EDITOR" .....                                         | 78        |
| "LANGUAGE" .....                                              | 78        |
| ■ <b>"WINDOW" menu</b>                                        |           |
| "CASCADE" .....                                               | 79        |
| "TILE VERTICALLY" .....                                       | 79        |
| "TILE HORIZONTALLY" .....                                     | 79        |
| ■ <b>"HELP" menu .....</b>                                    | <b>79</b> |
| "INDEX" .....                                                 | 79        |
| "ABOUT PROGRAM" .....                                         | 79        |

### ■ PC Software Interface

Following chapter describes the individual menus of the SyncPos user interface. If you are familiar with the Windows interface then, naturally, you can skip the familiar menus and immediately move to the SyncPos-specific information.

The basic information about program layout, command structure, interrupt, elements of the programming language, arithmetic and User Unit can be found in the introduction to the software reference. All commands and parameters are described in detail and then alphabetically ordered in the chapters Software Reference and Parameter Reference.

### ■ "FILE" menu




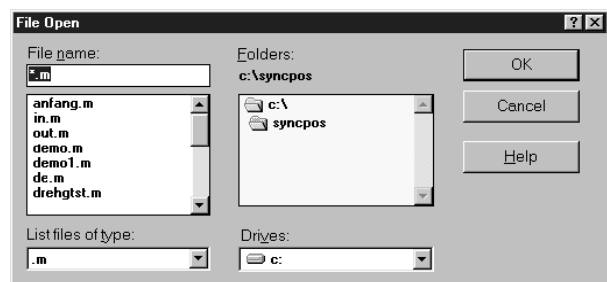
The "FILE" menu contains all the commands necessary to create, open, save and print a program. All commands can be used with a mouse click or with the key combination [ALT] and the underlined letter, for example [ALT] + [F] + [S], if you want to "FILE" → "SAVE".

### ■ "NEW"

In the menu bar click on "FILE" and then on "NEW" or click on the  icon in order to write a new program. The edit window is opened with the name "Unnamed Program1" and you can begin to write your program.

### ■ "OPEN"

Click on "FILE" and "OPEN" or on the  icon in order to open a program file. In the menu "FILE" → "OPEN" select the file you want with a double-click or by clicking on "OK". If the file you are looking for is still in the list of files last used (max. 9 files) then click on it and open the file in this manner. Since all SyncPos program files require the extension .m, this file type must also be selected. If this is not the case, select the file type **.m** or **All files (\*.\*)** from the **List files of type**.



If you can't find the file in the left field, click on a different directory in the **Folders** field or click on the drive which contains the file in the field **Drives**. Continue to double-click on the sub-directories until you can mark the file you are looking for in the left field. Then click on "OK".

### Looking for lost files

If you have "lost" a file in the system, you can search for all \*.m-files with the **Explorer**.

### ■ "CLOSE"

#### "FILE" → "CLOSE"

Click on "FILE" → "CLOSE" if you want to close the program file which has been loaded or written, but don't want to end the SyncPos program.




### NB!

If you have not yet saved the newly written file or the changes in the old file, answer the question which follows with "YES", otherwise the new file will not be saved.

**■ "SAVE" and "SAVE AS"**
**"FILE" → "SAVE"**


---

Click on "FILE" → "SAVE" or on the  icon in order to save a new or changed program on the computer's hard drive from time to time or before closing the file. If you have not yet saved a new program file the corresponding window is automatically opened under "FILE" → "SAVE AS".

Click on "FILE" → "SAVE AS" in order to save a new or altered program under a new name. All program files have the extension **.m**.



You should always save your files on the PC. This is because although the program is also automatically saved for every execute and for each syntax check in the SyncPos-Option, at this point the program has already been compiled and thus can no longer be altered in the computer. For reasons of compatibility between Windows '95 and earlier versions of Windows the input of more than 8 characters for the file names is not allowed. You can save as many program files as will fit in the free memory on the hard drive of the computer.

**■ Printing and printer settings**
**"FILE" → "PRINT"**


---

To print the file click on "FILE" → "PRINT", choose the area to be printed and the desired number of copies and start the printer with "OK".


The dialog field that appears is somewhat different depending on your operating system – Windows 95 or Windows NT. For further information please refer to the Windows help.

Click on "PRINT SETUP..." (or "PROPERTIES"), if you want to set other factors such as paper format or alignment (horizontal or vertical). These settings can also be reached by clicking on "PRINTER SETTINGS".

Click on → "PRINTER SETTINGS", if you want to set the paper size, the output size (scaling) or other options. You can find more information on the possibilities for Window's "PRINTER SETUP" in the Windows help.

**■ "EXIT PROGRAM"**
**Ending SyncPos**


---

The SyncPos program can be ended by clicking on "EXIT PROGRAM" or on the  icon. If you have not yet saved a new file or changes to an old file, then you will have the chance to do this.



However, "EXIT PROGRAM" does not end a program running in the VLT. You can only abort or end a program with [ESC]. In order to do this the file which is linked with the controller resp. VLT must be open or re-opened.



However, if the VLT stops at "FILE" → "EXIT PROGRAM" then this can be due to the fact that the VLT is sending PRINT commands which can no longer be displayed by the communications window.


**■ Delete file**

You can delete one or more files – as usual in Windows – in the **Explorer**.

### ■ "EDIT" menu



The "EDIT" menu offers the necessary editing help for programming. Most of these commands can also be reached via certain keys and key combinations, as is usual in Windows.

When you are writing your program use the tabulators  and the different colors to visually structure your program. The tab increments are permanently set.

#### Marking text

Point to the position where the marking should begin and pull the mouse cursor over the text. If only one word should be marked, simply double-click on the corresponding word.

You can expand marked text in all directions using the shortcuts listed; for example, with [SHIFT] + [→] by one character to the right, with [CNTL] + [SHIFT] + [→] key to the end of the word, with [SHIFT] + [END] to the end of the line, etc. The various possibilities can be found in the [List of shortcuts](#).

#### Deleting text

Single characters can be deleted either with the [DEL] key (which deletes character-by-character to the right of the cursor) or with the [BACKSPACE] key (which deletes character-by-character to the left of the cursor). If you are deleting words or lines which have been marked, both keys have the same effect.

#### Line number

Within your program you can use the line numbers for orientation purposes. For example, the syntax check not only places the cursor in the corresponding line, but also names the line number containing the incorrect command.

The current line number can be found in the status bar, for example 13:1. This means that the cursor is located in line 13 at position 1.

#### Context menu

Click on the right mouse key and then the "EDIT" menu is available as a context menu (only for Windows 95).


Context menus appear in the window at the location where you are currently working.

Close the context menu by clicking outside of the context menu without having selected a command. Or press [ESC].

### ■ "UNDO"

Click on "EDIT" and "UNDO", if you want to undo the last command. By clicking on "UNDO" again, you can restore the command which was undone. This can also be accomplished with the key combination [CNTL] + [Z] or [ALT] + [BACKSPACE].

### ■ "CUT"

With "CUT" it is possible to copy and delete text. Mark the desired text fragment and click on "CUT" or on the  icon. The text will be temporarily saved for inserting. With the next copy command the text stored in the temporary memory is automatically overwritten.

The key combination [CNTL] + [DEL] also serves the same function.



 icon.

The key combination [CNTL] + [INSERT] serves the same purpose.

### ■ "FIND" and "REPLACE"

Find and replace is realized in accordance with the Windows conventions and supplemented by some useful functions.

Click on "EDIT" → "FIND" or press [CNTL] + [F] and enter the term searched into the following dialog field. Use [F3] to jump from one site found to the next one.

### Mark all

Click on → "MARK ALL" instead of → "FIND" and all sites found are immediately marked with a blue triangle at the left margin. You can then jump back and forth with [F2] between the sites found.

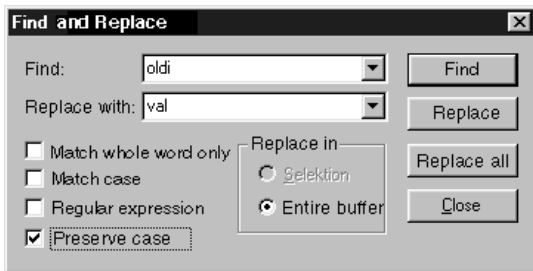
### Regular Expression

The regular expression implementation in search and replace functionality handles the following syntax:

- Wildcards: ? (for any character), + (for one or more of something), \* (for zero or more of something).
- Sets of characters: Characters enclosed in square brackets will be treated as an option set. Character ranges may be specified with a - (e.g. [a-c]).
- Logical OR: Subexpressions may be ORed together with the | pipe symbol.
- Parenthesized subexpressions: A regular expression may be enclosed within parentheses and will be treated as a unit.
- Escape characters: Sequences such as \t, etc. will be substituted for an equivalent single character. \ represents the backslash.

### ■ Replace

Use "EDIT" → "REPLACE" or [CNTL] + R to replace the found passage(s).



### ■ "DEVELOPMENT" menu

| Development        |     |
|--------------------|-----|
| Execute            | F5  |
| Break              | ESC |
| Continue           |     |
| Prepare Singlestep |     |
| Go                 |     |
| Singlestep         | F9  |
| End Debug          |     |
| Show Watch         |     |
| Syntax Check       |     |
| Break all          |     |
| Select Controller  |     |
| Command List       | F12 |

With the functions in the "DEVELOPMENT" menu you can run, abort, continue or run the programs step-by-step, which is particularly useful when searching for errors. A debug mode and online status information as well as the possibility to change the variables during program execution make programming easier. You should always subject newly written programs to a "SYNTAX CHECK" before running them for the first time.

However, before you begin it is always necessary to select a controller resp. a VLT.

Several helpful functions can be found in the "COMMAND LIST". First of all, this lists a general overview of all SyncPos commands, which can be selected and transferred to the edit window immediately. Secondly, you can work here with the teach-in programming.

### ■ "EXECUTE"

Click on "DEVELOPMENT" → "EXECUTE" or press [F5] and the program that is opened and displayed in the editor will be started.

In doing so the program is loaded into the VLT but first it is compiled, or translated into the internal system language. The size of a program is reduced considerably during compilation and thus the memory of the SyncPos Motion Controller is used very efficiently.

At the same time the program is loaded into the temporary sector of the RAM; this is overwritten with each subsequent execute command. Thus, when programming you have a quick and uncomplicated process to test the programs.



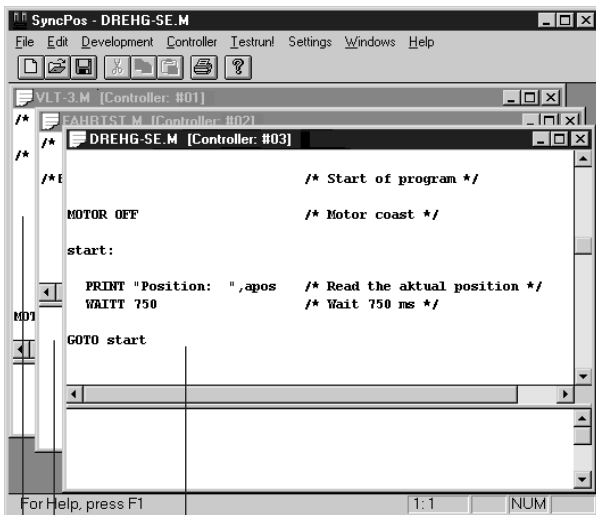
### NB!

However, it is neither possible to move a compiled program back to the computer, nor to edit the source file on the PC again. Thus, in general, you should also save all programs on the hard drive of the computer.

Run programs in several VLTs

If you want to load the program into several VLTs, link the program with the corresponding VLT and click on → "EXECUTE".

If you want to load a different program in each VLT, open a different edit window for each VLT, then open the desired program file and connect it to the VLT with → "SELECT CONTROLLER". Then start each program, one after the other, with → "EXECUTE".



VLT 1  
VLT 2  
VLT 3  
Controller, resp. VLT 3

### ■ "BREAK"

Click on "DEVELOPMENT" → "BREAK" or press [ESC] in order to immediately abort the program. When doing so, it is possible that active motion processes could be ended prematurely.



**NB!**

Braking is done with the maximum deceleration permitted.

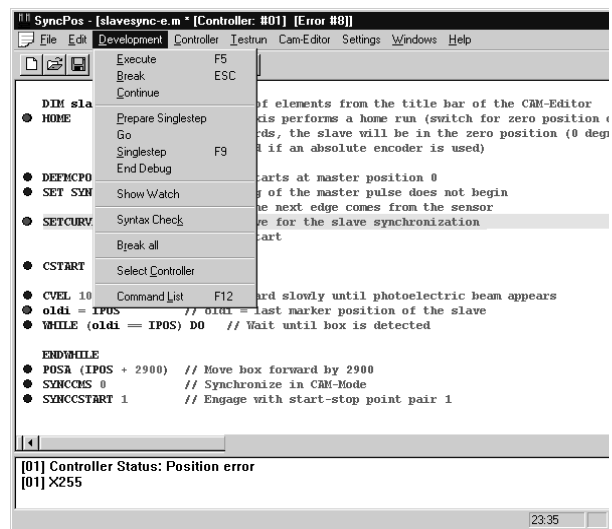
### ■ "CONTINUE" program

Click on "DEVELOPMENT" → "CONTINUE", in order to continue a program which was just aborted. In doing so any motion processes which were interrupted will be completed.

If a program with an error message was aborted, you can → "CONTINUE" it again with this function once you have removed the error and/or erased the error message.

### ■ Preparing "SINGLESTEP"

Single-step processing (Tracing) is particularly suitable for testing newly developed programs and can be helpful when searching for errors. Click on "DEVELOPMENT" → "PREPARE SINGLE-STEP" and the program opened is prepared for the debug mode: It is compiled and a debug file is produced, the program is loaded into the VLT and all executable program lines are marked by blue dots. Now, also the respective menu items are activated.



### Set Breakpoints

By double-clicking you can set a breakpoint before every program line marked with a blue dot. This will be highlighted in red.

The program execution will then stop before this program line – being highlighted in yellow – is executed.

By further double-clicking the red breakpoints are changed again into blue markings for the program lines which shall be skipped while tracing, i.e. no break in debug mode.



**NB!**

Depending on the speed of the program execution and communication the number of breakpoints should be limited to a reasonable amount. A maximum of 10 breakpoints are allowed.



**NB!**

ON PERIOD functions should be deactivated during debugging, since the internal timer doesn't stop with singlestep. The program tries to recover the ON PERIOD features later on and this could be problematic.

### Compatibility

The red breakpoints replace the former #DEBUG on/off commands, which is no longer executed from firmware version

VLT5000/SyncPos Software Version 3.xx/4.2x

VLT5000/FluxSyncPos Software Version 5.xx/4.2x onwards.

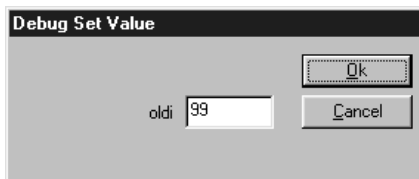
Existing #DEBUG commands may not necessarily be removed from the program as they will be ignored.

If a VLT with an earlier version is used, the program execution is started with "DEVELOPMENT" → "PREPARE SINGLESTEP" and gradually processed with → "SINGLESTEP" or [F9]. Other debug functions such as "GO" or "SHOW WATCH" cannot be selected.

### Change variables online

In the debug mode you can change the variables during program execution. Please observe that such a change should also be considered for the program.

Click on the variable with the right mouse button and set the desired value in the following field:



### Read variables

In the debug mode you can read the current value of the variables after the program execution.

Click on the variable with the left mouse button and the value is displayed until you move the mouse cursor again.

```

● CVEL 10 // 1.
● o1di = IPOS // 0.
● WHILE [1] (o1di = IPOS) DO
 ENDWHILE

```

### ■ "GO" (Debug) and "SINGLESTEP"

The program execution stops at the first breakpoint and waits for your input:

Click on "DEVELOPMENT" → "SINGLESTEP" or press F9 in order to execute the next program line.

Or click on "DEVELOPMENT" → "EXECUTE" or press [F5] in order to process the program until the next breakpoint.

By pressing [F9] the program will then stop before the next program line, by pressing [F5] before each breakpoint.

### Interrupt program execution in the Debug Mode

Click on "DEVELOPMENT" → "BREAK" or press [ESC] in order to immediately abort the program execution. When doing so, it is possible that active motion processes could be ended prematurely.



### NB!

Braking is done with the maximum deceleration permitted.

Then, the cursor is in the program line which should be executed next. You can continue with "DEVELOPMENT" → "EXECUTE" [F5] or → "SINGLESTEP" [F9].

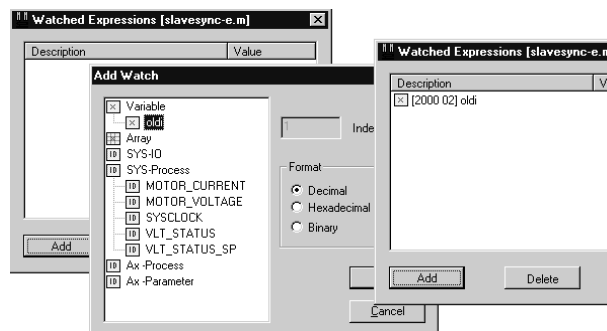
### ■ "END DEBUG"

With "DEVELOPMENT" → "END DEBUG" the program execution is ended immediately and you exit the debug mode. The marking of the program lines is removed, but the breakpoints are still displayed so that they can be used again with the next debugging. That means that if you insert program lines, the breakpoints wander along with it.

### ■ "SHOW WATCH"

This function enables online monitoring of the variables, arrays, system and axis processing data (according to the SYSVAR indices) and axis parameters.

Click on "DEVELOPMENT" → "SHOW WATCH" and in the following dialog window on → "ADD". In the next dialog window you can choose between the variables, arrays and parameters.



By double-clicking on the desired type, e.g. variable, you can choose between all variables used in the program. Mark the expression to be monitored and select the format in which it should be displayed (decimal, hexadecimal, binary). Then click on OK.

### "ADD WATCH"

You can add more expressions for monitoring → "ADD" and, of course, delete again → "DELETE". You can monitor a maximum of 10 expressions simultaneously.



#### NB!

The monitoring window is updated constantly. Thus, depending on the speed of the program execution and communication the number of the monitored expressions should be limited to a reasonable amount.



#### NB!

The array watching is limited to the first 250 elements.

### Change Watch window

If you want to change the size of the monitoring window (watched expressions), move the cursor to the lower edge of the dialog field and – as soon as the cursor has changed its shape – click and pull the window in the desired direction.

### Close Watch window

Click on "DEVELOPMENT" → "CLOSE WATCH" or on the close symbol in the dialog window. If you open it again later, the previously selected expressions are monitored online and displayed.

### ■ "SYNTAX CHECK"

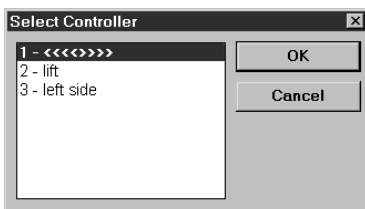
Check a newly written program before you start it for the very first time; click on "DEVELOPMENT" → "SYNTAX CHECK" and the program will be aborted as soon as a faulty command is found. The line number and an error description are outputted to the communications window. The cursor is automatically placed at the exact position of the syntax error and the program stops at this point..

### ■ "BREAK ALL"

If you run the programs in several VLTs click on "DEVELOPMENT" → "BREAK ALL", to abort the programs running.

### ■ "SELECT CONTROLLER"

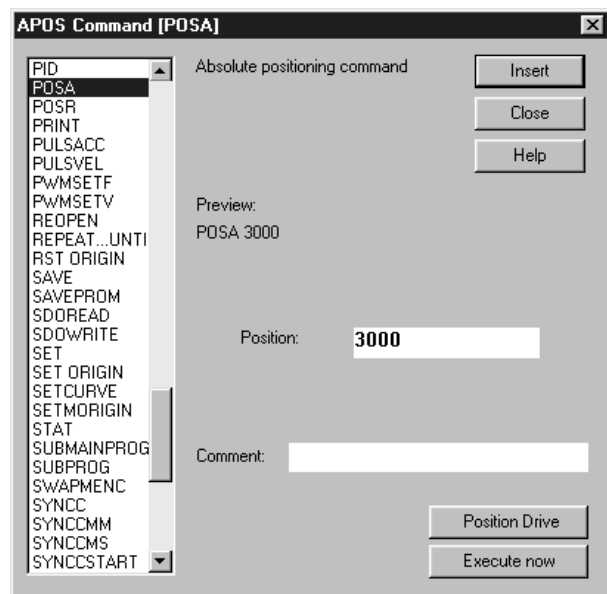
If you have configured more than one VLT, then use "DEVELOPMENT" → "SELECT CONTROLLER" to select the VLT that you want to use to load and start the programs. For this, simply mark the number of the VLT and click on "OK".



### ■ "COMMAND LIST"

The Command List not only contains all commands in alphabetical order, but also displays the necessary input fields for every command and automatically constructs the correct syntax for each command, which you can simply "INSERT" into your program. In addition, it is also possible to program your VLT with teach-in programming.

Move the cursor in the edit window to the position where you want to insert one or more new commands, click on "DEVELOPMENT" → "COMMAND LIST" and select the necessary command in the dialog field, e.g. POSA.



### "HELP"

You can find detailed information on all commands in this dialog field. Simply click on "HELP" or press [F1] and you get information about the marked command.

Mark up a command and the preview shows you the exact syntax of the command. Now you can select from three alternatives which you can mix randomly to program the VLT.

### "INSERT"

Either you can insert the command in your program immediately: click on "INSERT".

### "EXECUTE NOW"

Or click on "EXECUTE NOW" and test this command before inserting it in your program.

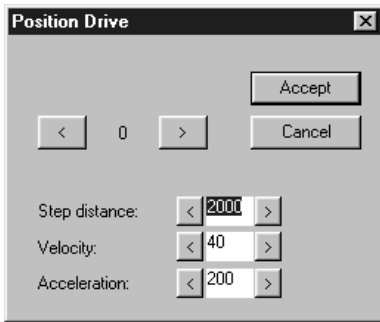


#### NB!

Enabled drives start up.

### "POSITION DRIVE"

Or you can use the teach-in function and click on "POSITION DRIVE": in the dialog field the actual position of the axis is displayed. Click on the forwards > or on the backwards < symbol and move the drive to the position desired. This can be done either step-by-step, with individual mouse clicks, or with continuous movement by holding the mouse button depressed.



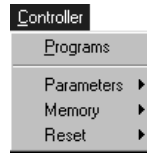
Once the drive has reached the desired position, click on "ACCEPT" and the value is entered in the dialog field for the axis.



#### NB!

In general, the values entered during programming are not tested whether they are within the permissible range. Due to the multitude of possible applications and various motor sizes this is not possible nor is it desirable.

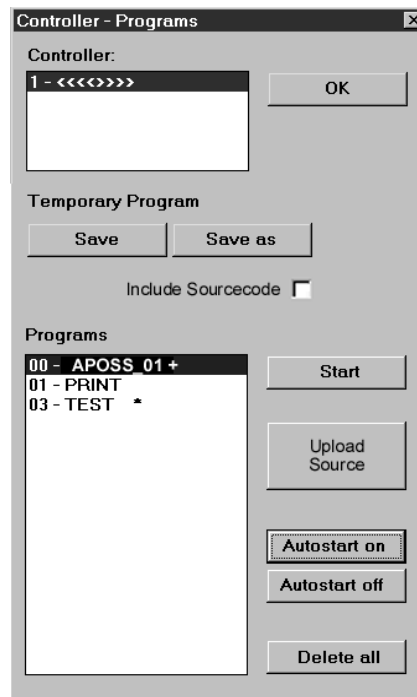
### ■ "CONTROLLER" menu



With the functions in the "CONTROLLER" menu you can manage your programs: You can save or delete the programs in the option EEPROM and can mark a program for autostart. In addition, you can also set all parameters here and make various resets and assign a name to every VLT connected.

### ■ "PROGRAMS"

Click on "CONTROLLER" → "PROGRAMS" and in the dialog field you will see all the VLTs in the first field. The controller marked is the VLT which is currently linked to the program displayed in the edit window. Naturally, you can also mark and edit another VLT.



### "PROGRAMS" → "SAVE"

Whenever you run a program, that means load a program into the controller, it is loaded into a temporary sector in the RAM, which is overwritten with each subsequent run. Here you can permanently "SAVE" the last **temporary program** executed.



#### NB!

Don't forget to always save and archive the program file on the hard drive of the computer since you can no longer edit the compiled source file in the VLT.

Click on "SAVE" and enter a name in the subsequent dialog field or confirm the file name suggested. The program number will be assigned automatically.

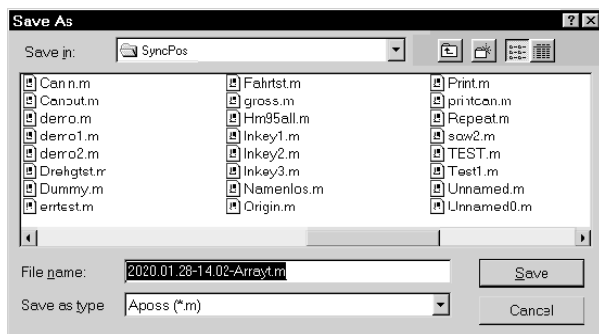
Click on "SAVE AS" and you can also assign the program number (0 to 127) yourself, in addition to the program name.

Using this program number any program can also be started over the inputs, for example from a PLC. For this all inputs are to be set accordingly with "CONTROLLER" → "PARAMETERS" → "GLOBAL".

### "SAVE" → "INCLUDE SOURCECODE"

When the check box is activated, the source coding is protected in VLT, in addition to directly executable and compiled program files. This can be re-accessed when necessary and saved in a file on a PC.

Click on "SAVE AS" and enter a name in the subsequent dialog field or confirm the file name. The suggested filename contains the date and time to be safe from overwriting actual files in case of uploading.



The source coding is saved in Flash EPROM. If insufficient space is available for the source coding in Flash EPROM a message is displayed, and other program files must be erased before saving the new program file.

All programs saved using source coding are marked with a '+' sign.

### "PROGRAMS" → "START"

In this dialog window you can select a program and start it directly.

### "UPLOAD SOURCE"

All programs marked with '+' can be read out of the control in source coding format and can be filed on your PC for subsequent use.

Select the desired program and click on "UPLOAD SOURCE". You can edit or duplicate the file for other VLT's.

### "AUTOSTART ON / OFF"

With Autostart you can mark a program that, in the future, is to be started immediately after the VLT is turned on. Select the desired program and click on "AUTOSTART ON". The program selected will then be marked with a '\*'.

If you want to remove an autostart command once it has been assigned, click on "AUTOSTART OFF" or simply mark another program.

If you want to have more than one program run with autostart, use the parameter PRGPAR (102). This allows you to determine which program should be started after the conclusion of the program run in autostart.

If nothing else has been determined in PRGPAR (102) or L\_PRGSTART (103), the program marked with autostart will always be started.

### A pre-set "AUTOSTART" has the following effect:

If no error is registered after a cold start (exceptions: tolerated position error is exceeded, end switch error and SW end switch error) the corresponding autostart program will be started.

If the autostart program is aborted by the user (SyncPos) it will not start again unless a new cold start is made. In this case no programs are started due to inputs or PRGPAR (102).

If the autostart program is aborted due to an error (since no ON ERROR routine was defined) or ended normally, then the program subsequently checks whether a start is planned through inputs or if the parameter PRGPAR (102) is set. If so, then the corresponding program is executed or the program waits for the start input. If not, the autostart program starts over again from the beginning. It follows that:

One-time execution of the autostart program  
If, in principle, you plan to start the programs via the parameter PRGPAR (102) or via the inputs, then the autostart program is only executed once (for example for HOME functions).

## Repeated execution of the autostart program

In all other instances the autostart program is started repeatedly.

Thus, it is also possible to simply start a program over again with an EXIT command. This is useful if, in an error situation (ON ERROR), you do not wish to continue with RETURN, but rather, for example, wish to force a repeated home run. However, it is important to make sure that an error has not occurred (except for a tolerated position error is exceeded, end switch errors and SW end switch errors), since otherwise the autostart program will not be started again.

## Linking autostart programs

Naturally, starting via the parameter PRGPAR (102) can also be used for linking purposes: After a program has been started the program number defined by PRGPAR (102) can be converted. Thus, it is possible to determine which program is to be executed next.



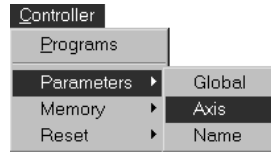
### NB!

If no autostart program has been defined, then it is not possible to start a program via PRGPAR (102); this always requires a terminated autostart program.

## "DELETE ALL"

Click on "DELETE ALL" if you want to delete all the programs in the VLT. Make sure beforehand that you have saved the programs on the PC or in the archive for safety reasons.

## ■ "CONTROLLER" → "PARAMETERS"



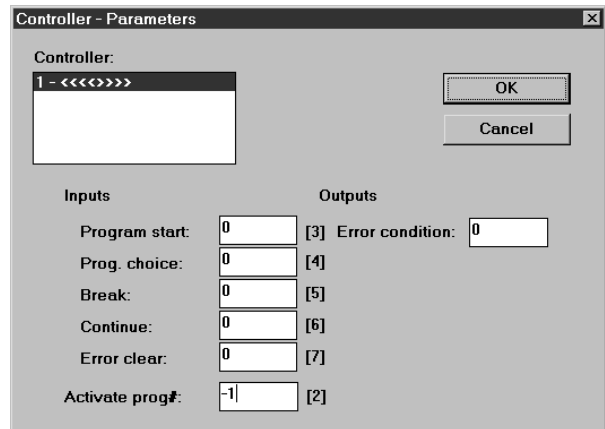
The parameters in the "CONTROLLER" menu are divided into two groups: the global parameters and the axis parameters. Both the general overview and the details on all parameters and parameter names with the respective factory settings can be found in the chapter Software Reference, section [Parameters](#).

Or simply press [F1] when the mouse cursor is in one of the input fields and you get information about the corresponding parameter.

## ■ Global parameters

The global parameters include the functions of the inputs and outputs (GLI group) and the standard parameters (GLS group). The corresponding internal parameter number is listed after the name of the parameter. With these numbers, which are listed consecutively in the next chapter, [Parameter Reference](#), you can find detailed information on the global parameters.

Mark the VLT you wish to edit. You can change each pre-set value individually. Click on "OK" to load the changes in the VLT as the user parameters.



With "RESET" → "PARAMETERS" in the "CONTROLLER" menu can reset all the factory settings, however in doing so all parameters, including the axis parameters, are reset to the factory settings.

**Axis parameters**

The axis parameters are always valid for all the programs belonging to one controller. The units of the parameter values, the factory settings and all other information concerning the parameters can be found in the Software Reference in the section Parameters. This information is arranged according to the internal parameter numbers (in increasing order).

There are two possibilities to set or change the parameters:

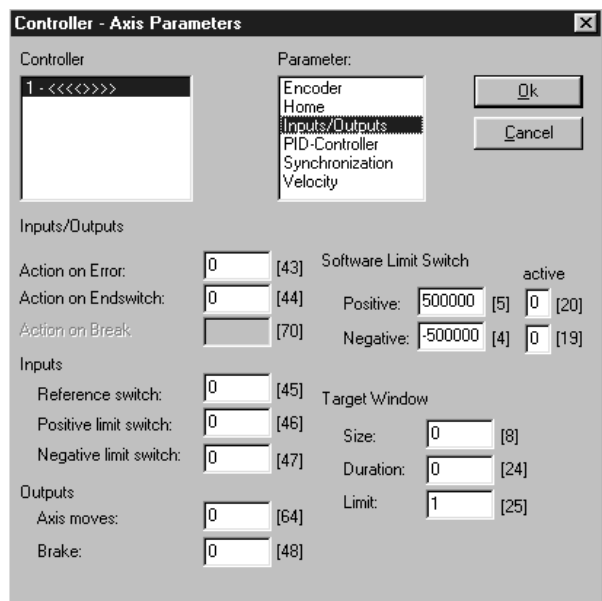
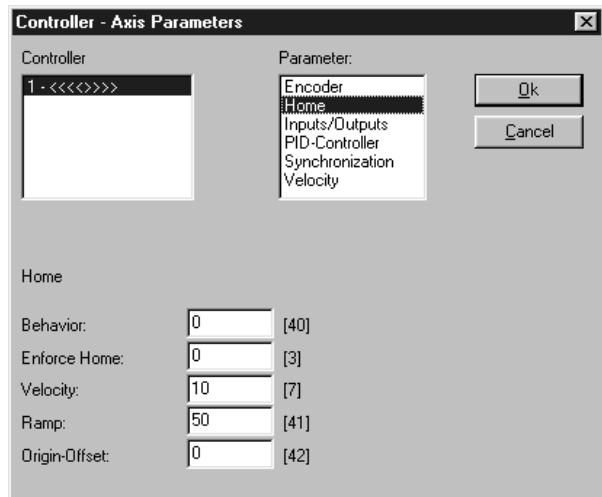
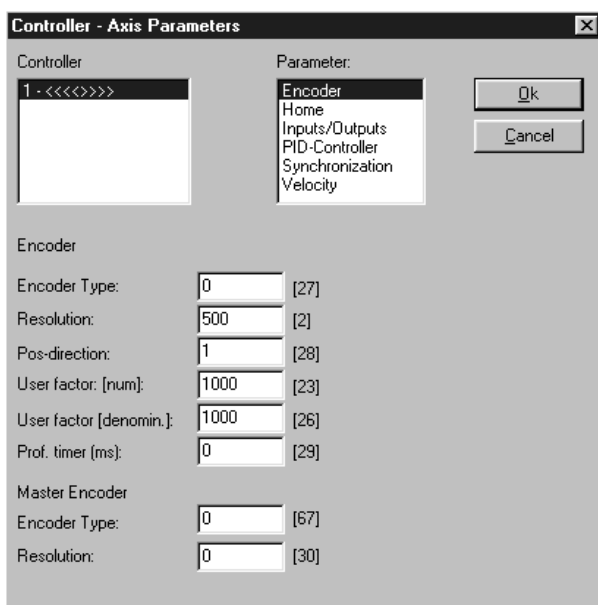
Set or change Axis parameters online

Click on "CONTROLLER" → "PARAMETERS" → "AXIS" and mark the controller the parameters of which you want to view or change in the subsequent dialog window. Also select the type in the field "PARAMETERS":

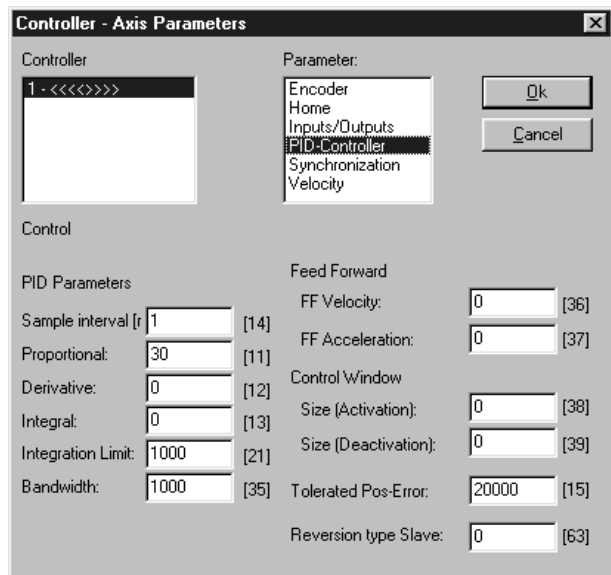
|                  |     |
|------------------|-----|
| Encoder          | AXE |
| Home             | AXH |
| Inputs / Outputs | AXI |
| PID-Controller   | AXR |
| Synchronization  | AXS |
| Velocity         | AXV |

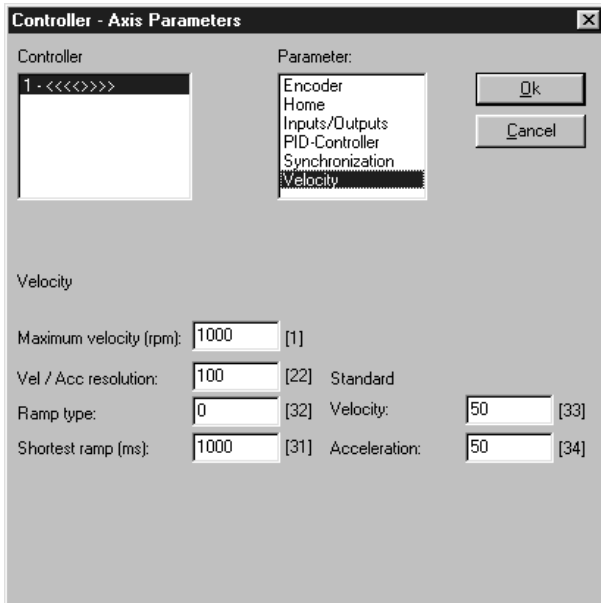
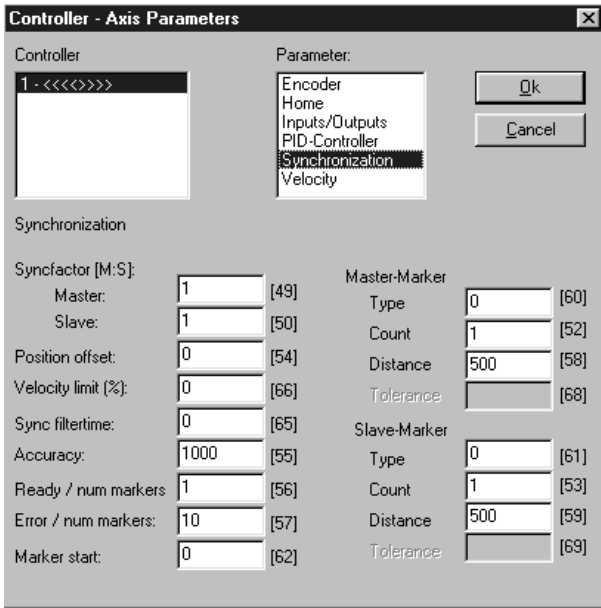
You can change each parameter and re-load it in the VLT by clicking on "OK". But you can also immediately select another VLT, change the parameters and then load all the changes into the VLT simultaneously with "OK".

Axis parameters with their factory settings



In this parameter group the inputs and outputs are assigned fixed functions – depending on the application. If necessary it is also possible to define a software end switch here.



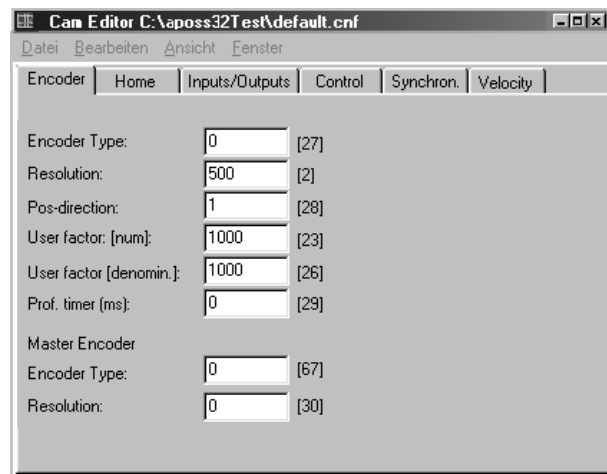


How to change the parameters of a configuration file  
 Beside the possibility to change online the axis and global parameter, you can change all parameter settings of a saved configuration file, too. For this case open the → "CAM EDITOR" and change the parameter in the corresponding index cards.



**NB!**

These changes only relate to the cnf-file, but not to the parameter in the controller. To accept the changed settings of the cnf-file into the controller, you have to load the cnf-file into the controller: "CONTROLLER" → "PARAMETER" → "RESTORE FROM FILE".



### Reset Axis Parameter

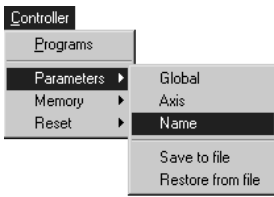
If you want the standard settings for all axis parameters, simply click on "CONTROLLER" → "RESET" → "PARAMETERS".



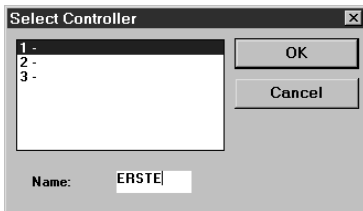
**NB!**

The global parameters will also be reset to the factory settings if you do this.

■ "PARAMETERS" → "NAME"

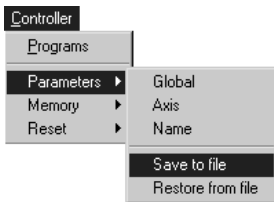


You can also enter a name for each VLT in addition to the number or change an existing name with this function. Click on "CONTROLLER" → "PARAMETERS" → "NAME" and select the VLT that you want to start in the subsequent dialog field:

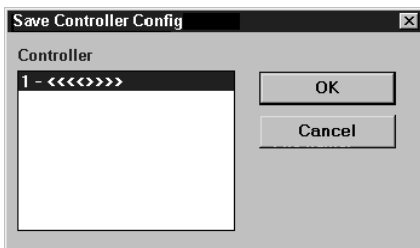


In the field "NAME" enter a name for the VLT (no longer than 8 characters) or overwrite the existing name and click on "OK".

■ "PARAMETERS" → "SAVE TO FILE"

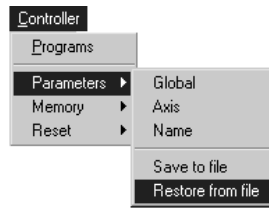


With "PARAMETERS" → "SAVE TO FILE" you save the user parameters including the arrays in a file with the extension ".cnf". This way you can quickly load the parameters in another VLT or re-load them in the VLT at any subsequent time, for example after deleting the EEPROM.

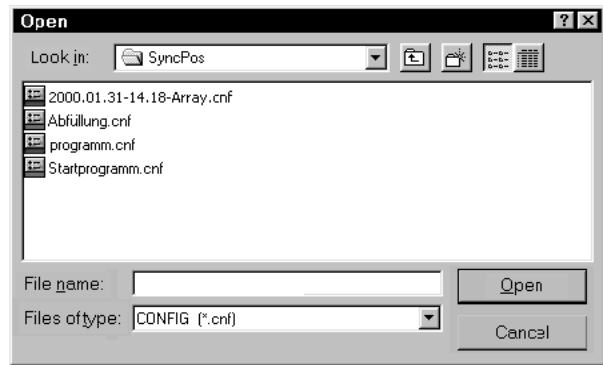
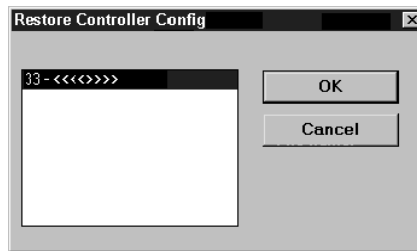


Click on "SAVE AS" and enter a name in the subsequent dialog field or confirm the file name. The suggested filename contains the date and time to be safe from overwriting actual parameters in case of restoring from file.

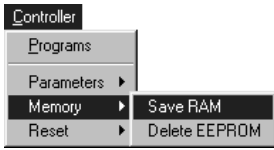
■ "PARAMETERS" → "RESTORE FROM FILE"



Click on "PARAMETERS" → "RESTORE FROM FILE" and select the VLT in which the data should be loaded.



Then select the file and click on "OPEN". The user parameters saved, including the arrays, are immediately loaded in the VLT.

**"MEMORY" → ...**



---

**"MEMORY" → "SAVE RAM"**

The function save RAM is usually not needed since the programs and parameters are automatically saved.

With → "SAVE RAM" you can also save the current array values in the EEPROM. "SAVE RAM" corresponds to the command SAVEPROM, since all programs, parameters and arrays are saved.

---

**"MEMORY" → "DELETE EEPROM"**

Delete the EEPROM in the SyncPos option if you either want to undo the array definition or want to reset all parameters to the factory settings.

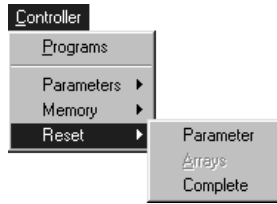


**NB!** When you delete the EEPROM all parameters are reset to the factory settings. However, this is only done after the VLT has been turned off.



**NB!** Remember the following when you delete the EEPROM:

- Check whether you have saved all the necessary programs on the computer so that you can load these into the VLT again once the EEPROM has been deleted.
- Check whether you have saved the parameters for all the VLTs connected in a file on the computer.
- Click on "MEMORY" → "DELETE EEPROM".
- Re-load the user parameters and the necessary programs in the VLT resp. VLTs.

**"RESET" → ...**



---

**"RESET" → "PARAMETERS"**

With "RESET" → "PARAMETERS" all global parameters and all axis parameters in the SyncPos motion controller are reset to the factory settings.

---

**"RESET" → "ARRAYS"**

With "RESET" → "ARRAYS" you can delete all arrays in the RAM without – as was formerly the case – deleting the parameters etc. This new command has the same effect as the menu command DELETE ARRAYS.



**NB!** If you then carry out SAVE ARRAYS, the arrays in the EEPROM are also overwritten!

---

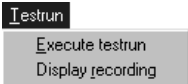
**"RESET" → "COMPLETE"**

With "RESET" → "COMPLETE" not only are all parameters, but also the programs and arrays are erased and the SyncPos option is reset to the basic factory setting ...



**NB!** ... and this happens immediately – not only after the controller has been turned on and off as is the case when you delete EEPROM.

### ■ "TESTRUN" menu



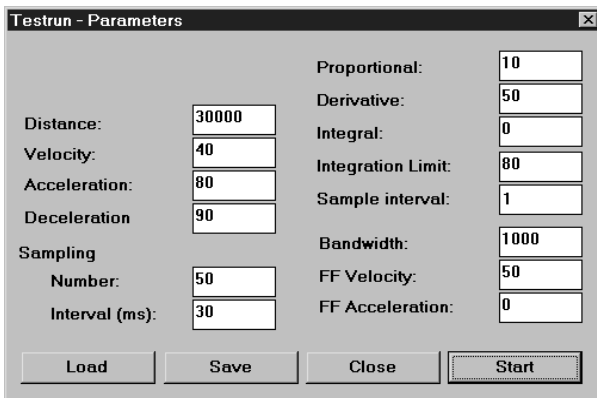
The "TESTRUN" menu offers the functions "EXECUTE TESTRUN" from the entry of the test run parameters up to the graphic representation of the test run results.

If you have used TESTSETP to define a test run with different parameters, you can also graphically display these results after execution (TESTSTART) with "TESTRUN" → "DISPLAY RECORDING".

In order to correctly set the controller parameters it is important to choose the correct test parameters.

### ■ Setting parameters for a test run

Click on "TESTRUN" → "EXECUTE TESTRUN" and enter the test run parameters in the dialog field:



If possible only change one parameter per measurement and check the effect this has.

### Path of motion

Set the path of motion in user units and utilize the entire duration of the measurement as well as possible:

The number of measurement points multiplied by the time difference between two measurements result in the entire duration of measurement and thus determines the graph. For an optimal evaluation of the figures the path of motion should be selected so that the end position is reached after approximately 80% of the entire recording time. Thus vibrations at the target position are easily recognizable.

#### Example:

50 measurements in intervals of 30 ms = 1.5 sec recording time.

### Velocity, acceleration and deceleration

The test run parameters of velocity, acceleration and deceleration are entered in percent of the respective maximum value.

Complete the measurements with the values most often needed for the controller for velocity, acceleration and deceleration.

In order to be able to evaluate the vibration behavior of the final velocity you should try to achieve a trapeze-shaped velocity course. For this it may be necessary to increase the sampling interval or reduce the final velocity.



#### NB!

Before you start to optimize the control behavior, check whether the maximum velocity and the maximum acceleration have been achieved.

### Number of readings

The number of samples and the sample interval determine the entire duration of the measurement. 50 to 100 measurement points are sufficient for an optimal graph.

The number of maximum possible measurement points is limited by the internal memory of the SyncPos option and also by any programs that are stored there. If the memory is not sufficient for the desired number of samples then it is necessary to delete the programs stored in the SyncPos motion controller with "CONTROLLER" → "PROGRAMS" → "DELETE ALL".

### Data sampling interval

Select a sampling interval which is suitable for the system and for the frequency converter, for example 20 to 30 ms.

For dynamic applications the sampling interval can be decreased to 1 ms. The shortest possible sampling interval should be set for servo-motors.

Naturally, to record slower or very long motion processes the time difference in milliseconds can be increased, however the maximum is 255 milliseconds.



#### NB!

The sampling interval mentioned here is the internal between measurements, not the controller sample interval.

### ■ "EXECUTE TESTRUN"

Before starting a testrun: Safety notes

Move the drive to the starting position, it is important to do this before you open the test window.



**NB!**

While optimizing the controller with the function "TESTRUN" the drive is automatically returned to the starting position after reaching the target position.

If run in reverse is not allowed, then the REVERS (63) parameter must be set to "no reversing = 2". Click on "CONTROLLER" → "PARAMETERS" → "AXIS" and change the setting in the parameter group **PID Controller**.



**NB!**

Make sure that any and all brakes are released and that there are not any obstacles in the positioning path.



**NB!**

Improperly adjusted control parameters can cause the motor and the mechanism to be damaged. For this reason never optimize controls without having an EMERGENCY STOP button.

Starting the Testrun

Click on "TESTRUN" → "EXECUTE TESTRUN" and the dialog field with the test run parameters is opened. The test run parameters last saved and the current axis parameters are already entered here:

| Testrun - Parameters                                                                                                                              |                                    |
|---------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| Distance:                                                                                                                                         | <input type="text" value="30000"/> |
| Velocity:                                                                                                                                         | <input type="text" value="40"/>    |
| Acceleration:                                                                                                                                     | <input type="text" value="80"/>    |
| Deceleration:                                                                                                                                     | <input type="text" value="90"/>    |
| Sampling                                                                                                                                          |                                    |
| Number:                                                                                                                                           | <input type="text" value="50"/>    |
| Interval (ms):                                                                                                                                    | <input type="text" value="30"/>    |
| Proportional:                                                                                                                                     | <input type="text" value="10"/>    |
| Derivative:                                                                                                                                       | <input type="text" value="50"/>    |
| Integral:                                                                                                                                         | <input type="text" value="0"/>     |
| Integration Limit:                                                                                                                                | <input type="text" value="80"/>    |
| Sample interval:                                                                                                                                  | <input type="text" value="1"/>     |
| Bandwidth:                                                                                                                                        | <input type="text" value="1000"/>  |
| FF Velocity:                                                                                                                                      | <input type="text" value="50"/>    |
| FF Acceleration:                                                                                                                                  | <input type="text" value="0"/>     |
| <input type="button" value="Load"/> <input type="button" value="Save"/> <input type="button" value="Close"/> <input type="button" value="Start"/> |                                    |

Begin the test series with 'stable' control parameters: If the standard set control parameters cause the drive to vibrate heavily in the starting position, select a low **Proportional** and **Derivative factor** (ca. 20) and set the integral factor to zero. Then, starting from these values, optimize the controller.

If it is at all possible you should optimize the controller first with the motor and the drive until the readings are no longer in the critical range. Then connect the mechanical load and complete the fine optimization.

Click on "START": The test run is executed, the current position values, etc. are saved and, at the end of the test run, transferred to the computer for evaluation.



**NB!**

Observe the behavior and temperature of the motor: if there are strong vibrations or an excessive increase in the motor temperature the motion process must be aborted prematurely with the EMERGENCY STOP button and different control parameters must be selected.

After making an EMERGENCY STOP you must move the drive back to the starting position before making another test run. Reduce the proportional factor, and if necessary also the derivative factor, before starting the next test run or measurement.

After the measurement is complete all measurement data is automatically transferred to the computer and, during this process, the drive returns to the starting position at a reduced velocity. The figures are plotted automatically.

The dialog field for the test run parameters is opened with "REPEAT".

Saving and loading test run parameters

Click on "SAVE", to save the test run parameters in the VLT. If, during additional test runs, you receive poorer control results then you can "LOAD" the parameters previously used again.

■ "DISPLAY RECORDING"

If you have used TESTSETP to define a test run with different parameters, you can also graphically display these results after execution (TESTSTART) with "TESTRUN" → "DISPLAY RECORDING".

Insofar as this is possible with the parameters you have selected, i.e. that the result can actually be displayed with the four graphics.

These four diagrams or seven curves are used as follows:

- (1) The actual position curve shows the values of index w1 (see TESTSETP),
- (2) the set position curve the values of index w2,
- (3) and the current curve the values of index w3.
- (4) The actual speed curve shows the difference of the recorded data to the values of w1; in the case that position data is being recorded this means the change in the position in ms = speed.
- (5) The set speed curve shows the difference to the values of w2; in the case that position data is being recorded this means the change in the position in ms = speed.
- (6) The actual acceleration curve shows the difference to the values of the actual speed (see 4); in the case that position data is being recorded this means the change in the speed in ms = acceleration.
- (7) The set acceleration curve shows the difference to the values of the set speed (see 5); in the case that position data is being recorded this means the change in the speed in ms = acceleration.

■ Evaluating motion figures

Check the maximum position, the maximum velocity, the number of "overshoots" and the duration of the building-up process.

For each graph the most important test run parameters, the corresponding maximum values and actual values are displayed:

- velocity in user units/ms,
- acceleration in user units/ms<sup>2</sup>,
- the maximum regulation difference which occurred during the process are shown,
- the actual position difference at the target,
- user units in qc,
- data sample interval in ms.

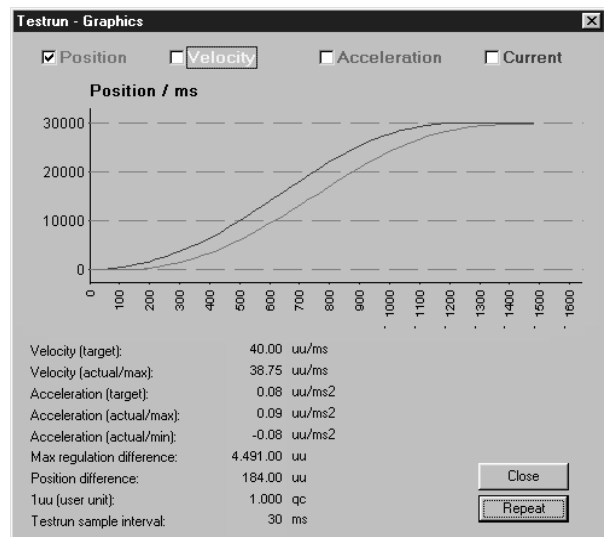
Click on the corresponding input field to view one of the other graphs, for example for velocity. You can also view two or even all four graphs at once.

However, then no units are shown on the x-axis.

Repeat

If you click on "REPEAT" then the dialog field for the test run parameters is opened once again: You can change the parameters and start a new test run.

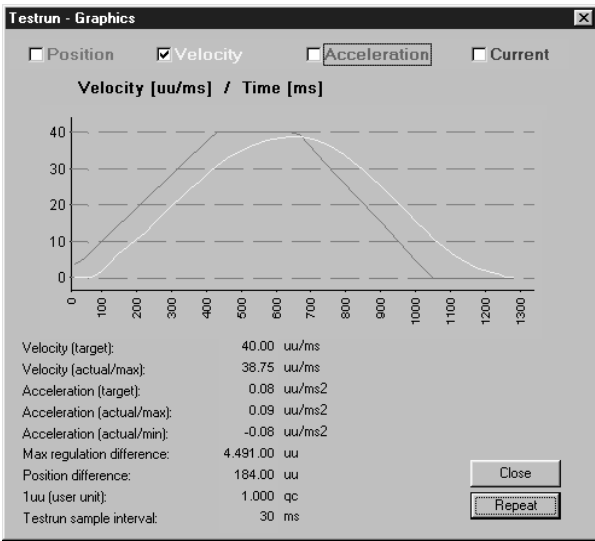
"TESTRUN" positioning graph



The positioning graph shows the set positions (dark or brown curve) and the positions actually achieved (light or red curve).

"TESTRUN" velocity graph

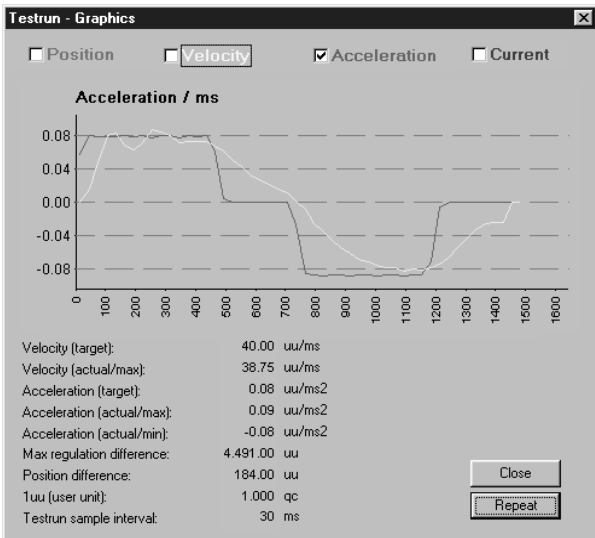
The yellow (light) curve shows the achieved velocity path, the brown (dark) curve shows the desired trapeze-shaped set curve.



In special instances the trapeze-shaped velocity curve can degenerate to a triangle shape. This effect occurs when the positioning distance is too short to achieve the maximum velocity at the desired acceleration.

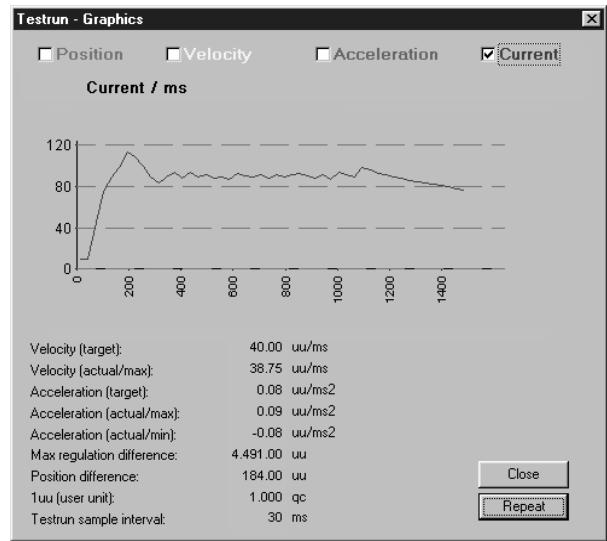
"TESTRUN" acceleration graph

The light green curve shows the actual path of acceleration, the dark curve the desired trapeze-shaped set curve during acceleration and deceleration.



"TESTRUN" current graph

The blue line shows the actual motor current.



## ■ CAM-Editor menu

The curve profiles for any cam controls are realized with the CAM-Editor. The individual curves are defined by fixpoints, parameters for the engage and disengage motion, as well as parameters for the synchronization with the marker. There are index cards in the CAM-Editor for this input and for other curve data. The curves and parameters are illustrated in the diagram; in addition, you can also enter and manipulate the fixpoints interactively.

While you are working with the CAM-Editor, the SyncPos window remains open. You can switch back and forth as you like, for example to check parameters in the control or to take over the number of array elements into the DIM instruction. One or more curves can be generated for a cam control. All curves required for an application are saved in the cnf file and loaded into the control. Click on → "CAM-EDITOR" to start the editor. The CAM-Editor window will be opened and ...

### Before you begin to edit a curve

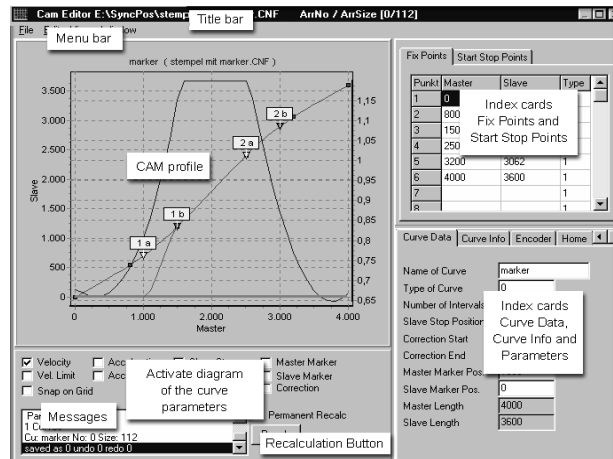
... you should load the parameters of the control as a cnf file into the curve editor because then the parameters will already be entered properly into the corresponding index card. You can get this file with "CONTROLLER" → "PARAMETERS SAVE TO FILE". During this operation, existing arrays, if any, will also be output.

If you do not load a cnf file, the VLT will be set to the factory settings.

## ■ CAM-Editor Window

The CAM-Editor window is divided into four sectors:

- Curve profile diagram,
- Checkbox for the activation of the diagram of parameters and calculation modi and Recalc-Button
- Index cards: Table of the "FIX POINTS" and "START STOP POINTS",
- Index cards: "CURVE DATA", "CURVE INFO" and all parameters according to the dialog fields of the SyncPos program: Encoder, Home, Inputs/Outputs, PID-Controller, Synchronization and Velocity. Scroll to the left or to the right in order to view all index cards.



The entire CAM-Editor window can be enlarged or reduced as desired. When this happens, the two sectors on the right for the alphanumeric input remain the same size while the diagram is enlarged or reduced and the lower left sector is even clipped in the case of a substantial enlargement. If you enter data in the sectors of index cards that are initially not displayed, for example in Synchronization, then the entire card will be displayed temporarily and the diagram will be reduced in size. Click on "WINDOW" → "STANDARD", on → "RECALC" or on another field in the CAM-Editor, and the standard window will be displayed again immediately.

However, you can also modify the size of the individual sectors by moving on the horizontal or vertical borders of the sectors with the mouse cursor and by clicking – as soon as the cursor changes its form – and by dragging it in the desired direction.

### CAM-Editor Title Bar

The title bar displays the name of the cnf file and the complete path.

As soon as a curve has been saved → "SAVE CNF as .." in a cnf file for the first time, you will see on the right side of the title bar the number (the location) of the array in the cnf file and the quantity of the array elements which you need for the DIM instruction in the program.

Arr.Nr/ArrSize [0/350]

### CAM-Editor Menu Bar

The menu bar offers the menus for loading and saving of files, for different settings, for calculation and view options.

**■ "FILE" menu CNF**


The menu File contains the commands for creating, opening and saving of a cnf file with the curves and parameters; it also contains functions for exporting, importing and printing of the cnf files. You can also go to all commands via mouse click or with the key combination Alt and the underlined letter, for example [ALT] + [D] + [S], if you want to "FILE" → "SAVE".

---

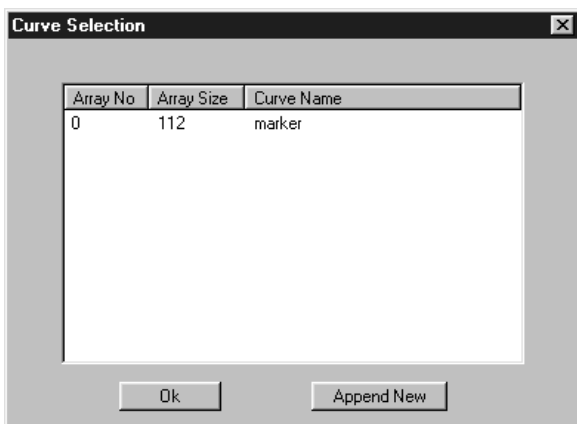
**"NEW CNF"**

With "FILE" → "NEW", an „empty“ new cnf file is provided. This means that the VLT factory settings are entered in the parameters here.

---

**"LOAD CNF"**

You can open an existing cnf file with "FILE" → "LOAD CNF": Either the control parameters previously saved with "PARAMETERS" → "SAVE TO FILE", if you want to edit new curves; or a cnf file that already contains one or more curves for the control parameters which you want to edit. Select the desired file in the following dialog field by double-clicking or by clicking on OK. If the selected cnf file contains more than one curve, another dialog field follows where you select the curve to be edited:



Click on → "APPEND NEW", if you do not want to edit any of these curves, but want to edit another one.

If the selected cnf file does not contain a curve, an empty file will be opened.

---

**"SAVE CNF" and "SAVE CNF AS .."**

Click on "FILE" → "SAVE CNF" in order to save a new or modified cnf file from time to time or at the latest before closing the file.

If you have never saved a new cnf file, the corresponding dialog window "FILE" → "SAVE CNF AS" will be offered automatically.

Click on "FILE" → "SAVE CNF AS .." in order to save a new or modified cnf file under a new name. All of these files are automatically provided with the extension .cnf.

---

**"EXPORT"**

If you want to export the data to another format, for example in order to print or to further process these data, click on → "EXPORT". The data will be saved in the ASCII format with the extension „.dat“.

---

**"IMPORT"**

You can only import dat files which you have created with → "EXPORT".

---

**"PRINT" and "PRINT BW"**

Depending on the connected printer, you can print the curve diagram. In black & white or color. If your printer is unable to convert the color diagram into black & white, you should use → "PRINT BW" for black & white.

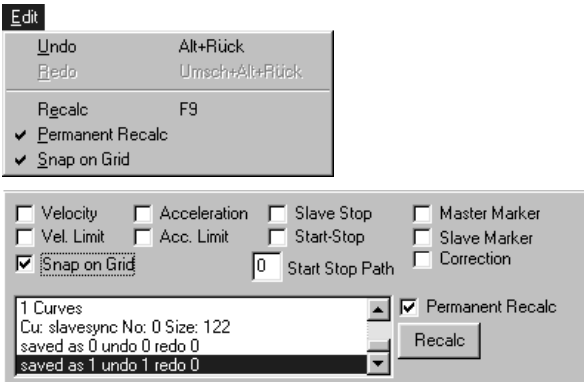
---

**"EXIT"**

You can close the CAM-Editor with → "EXIT" or by clicking on the exit icon.

### ■ "EDIT" curve menu

In the menu "EDIT", you will find functions that are required for the calculation of the curve; provided that they are frequently needed, they can also be started directly via a button in the editor or be selected in checkboxes.



### "UNDO" and "REDO"

How often and in what work steps Undo has an effect is determined by "RECALC". Every → "UNDO" restores the situation up to the previous Recalc. If you → "SAVE" the file, the Undo memory will be deleted.

Select "EDIT" → "UNDO" or use the key combination [ALT] + [BACKSPACE].

"EDIT" → "REDO" or the key combination [ALT] + [SHIFT] + [BACKSPACE] restores the work steps prior to the Undo command.

### "RECALCULATION"

In many actions – for example in all interactive actions – the curve is recalculated and displayed immediately. When you fill out the fields such as the fixpoints or parameters, you determine the moment yourself with this function.

Select "EDIT" → "RECALC" or click on the Recalc-Button or press [F9] in order to activate the Recalculation and the new curve display. This is always required if you add new fixpoints, for example, or if you change parameters or Curve Data.

During every recalculation of the curve, a consistency check is performed, too. This can lead to a series of error messages and notes which you must confirm. For example, if end and start points are not connected or if there are not enough fixpoints for a curve section (at least three).

### "PERMANENT RECALCULATION"

In all interactive actions and many other actions the curve is recalculated and displayed immediately. Turn off → "PERMANENT RECALC", if the interactive redrawing of the curve during the modification of the points is disruptive. You can then move the point with the help of the mouse first, before the curve is redrawn.

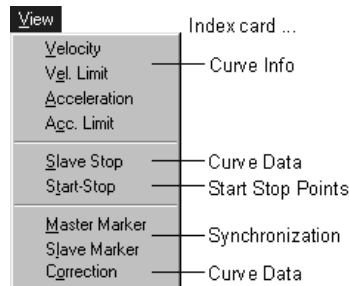
### "SNAP ON GRID"

If the fixpoints are to be set automatically on the interpolation grid – this is recommended – you should activate → "SNAP ON GRID" align.

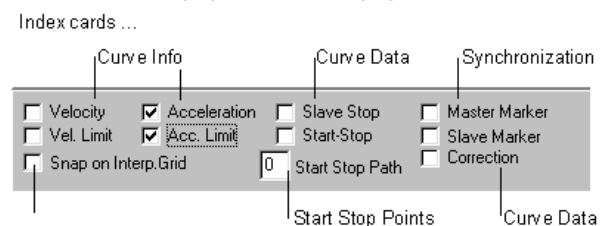
The function will not be executed if the master length is not divisible in whole numbers by the number of intervals.

### ■ "VIEW" menu

In addition to the fixpoints, you can graphically display the following curve information and parameters. For example, if you have reduced the size of the CAM-Editor window to such an extent that the checkboxes are no longer displayed, you can turn the graphic illustration of the parameters in the View menu on and off. The only item you cannot activate here is the diagram of the engage and disengage curves, since the curve and direction are determined by one input. The ü checks indicate the currently activated views.



The menu selection and the checkboxes are divided into groups which correspond to the index cards. Alternatively, you can activate the graphic display of the parameters by clicking on the checkboxes or by entering the number of the point pair in the case of the engage and disengage curves.



Familiarize yourself with the meaning and graphic illustration of the parameters in the corresponding index cards.

### ■ "WINDOW" CAM menu



Since many index cards temporarily change the size of the CAM window due to their size, you can display the standard window again by clicking on → "STANDARD". You can achieve the same result with → "RECALC" or by clicking on another input field outside of the "large" index card.

### ■ CAM Profile

You can view the graphic illustration of the curve, parameters and other information in the curve profile; you are also able to interactively modify and insert fixpoints with the mouse and to enlarge the illustration.

A blue header shows the curve name and the file name of the displayed curve.

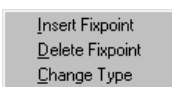
#### Inserting or deleting fixpoints with the mouse

Before you can interactively edit the fixpoints with the mouse, you must have defined at least two fixpoints with master and slave coordinates in the table. However, the curve is only displayed if at least three points have been defined and if the curve can be calculated.

You can insert points directly on the curve or at any position you like in the coordinate field.

#### Inserting points on the curve between already existing points:

Move the mouse cursor to a fixpoint until the hand icon appears, then click on the right mouse button. Select the desired action in the following pop-up menu:



An additional fixpoint is inserted between the selected and the previous point (to the left) with → "INSERT FIXPOINTS", it is also being added immediately in the table. You cannot insert an additional point in front of the first one with the mouse.

You can delete the selected point with → "DELETE FIXPOINT"; it will also be deleted immediately in the table. You cannot delete the first and last fixpoint.

#### How to set any points into the coordinate field

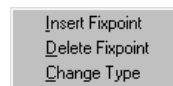
Click on the desired position with the right mouse button and select → "INSERT FIXPOINTS" in the pop-up menu.

#### How to move Fixpoints interactively with the mouse

Move the mouse cursor to a fixpoint until the hand icon appears, click on the left mouse button and drag the point to the desired new position. The curve will be redrawn simultaneously if "PERMANENT RECALC" is activated. If this creates a disruption on your computer, you can deactivate Permanent Recalc; the curve will then be recalculated and displayed as soon as you release the mouse button. The fixpoints should be placed on interval borders. Otherwise, there can be no assurance that the curve really goes through the fixpoints. Thus, you should always activate → "SNAP ON GRID" if possible, provided that a meaningful → "NUMBER OF INTERVALS" has already been defined.

#### How to change the Point Type in the diagram

There are curve points (type 1, shown in green) and tangent points (type 2, shown in blue). Move the mouse cursor to a fixpoint until the hand icon appears and click on the right mouse button. In the following pop-up menu, select the desired action:



The selected and the previous point (to the left) are changed with → "CHANGE TYPE", either from curve points to tangent points or vice versa. The changes are also immediately executed in the table (index card fixpoints).

#### How to zoom CAM profiles

In order to enlarge the diagram, click and hold the left mouse button and drag a rectangle downwards and to the right. The diagram will be enlarged in accordance with the excerpt. You can repeat this procedure several times until you obtain an extremely detailed reproduction.

In order to reduce the diagram, click and hold the left mouse button and drag a rectangle upwards and to the left. The diagram will be displayed in the standard size again.

#### How to scroll CAM profiles

Click and hold the right mouse button and move the mouse in the desired direction. In order to display the diagram in the standard position again, click and hold the left mouse button and drag a rectangle upwards and to the left. Any possible enlargement will also be reset in the course of this action.

### What to do ... if the curve is not displayed

If only points instead of a curve are displayed in the diagram window, then the curve could not be calculated. For example, you may have

- entered too few points (at least three points are required for the illustration),
- selected a larger number of intervals as master length,
- set tangent points incorrectly. (Rule of thumb: Two tangent points must always alternate with at least two curve points.)

### ■ CAM-Editor Messages

The display indicates input and syntax errors. You can scroll back and forth through the messages.



### ■ Index card "FIX POINTS"

On principle, you must enter the fixpoints for the master and the slave in ascending order in the curve table. Do not select user units that are too small in order for the curve to get a reasonable resolution despite the whole number input. It is recommended to use at least four-figure units.

| Punkt | Master | Slave | Type |
|-------|--------|-------|------|
| 1     | 0      | 0     | 1    |
| 2     | 1500   | 1200  | 1    |
| 3     | 2500   | 2400  | 1    |
| 4     | 4000   | 3600  | 1    |
| 5     |        |       | 1    |
| 6     |        |       | 1    |

Click on → "RECALC" to display the curve. A curve will only be displayed when you have entered at least three points each for the master and slave, respectively, and entered the point type „1“.

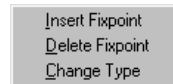
As soon as you have defined at least the start and end point of the curve, you can also set the additional fixpoints interactively with the help of the mouse.

The fixpoints should lie on interval borders. Thus, you should always activate → "SNAP ON GRID", if possible.

### How to insert or delete Fixpoints in the table

You can always delete the last point or insert an additional point in the table. You can insert or delete points in the table with the help of the pop-up menu.

Using the right mouse button, click on the point you want to delete, or in front of which you want to insert a fixpoint. Select → "INSERT FIXPOINT" in the subsequent pop-up menu. The point will be inserted between this point and the previous one.



Alternatively, you can insert or delete the desired points interactively with the help of the mouse in the diagram.

You can change the set points in the table by overwriting in any way desired.

### "TYPE": Curve and Tangent Points

Type 1 = curve point

Type 2 = tangent point for straight sections

The curve will be calculated as spline interpolation between curve points. For sectors where the velocity must be constant and the acceleration = 0, you should use the tangent points. A straight line instead of a spline will be placed between these points.

The calculation of a spline always requires three points, i.e. at least two fixpoints and one tangent (straight section) consisting of two tangent points. You can determine the point type either in the table by means of numerical input or by changing the existing type with the help of the pop-up menu.

Using the right mouse button, click on the point from which the tangent (straight section) should lead to the previous one. In the subsequent pop-up menu, select → "CHANGE TYPE". The type will be changed to tangent point = 2 for both fixpoints. You can execute the same action in the diagram: Move the cursor to the point until the hand icon appears. Click on the right mouse button and select → "CHANGE TYPE". Here, both points are changed immediately, too.

### ■ Index Card "START STOP POINTS"

In this table, you define point pairs for engaging and disengaging the slave during the synchronization. You need one point pair to determine the master position where the synchronization should start and where the engaging should take place. You can determine with an additional point pair from what point on the disengaging should be made and where the synchronization should be stopped.

| Fix Points |        | Start Stop Points |  |
|------------|--------|-------------------|--|
| Punkt      | Master |                   |  |
| 1 a        | 1000   |                   |  |
| 1 b        | 1500   |                   |  |
| 2 a        | 2500   |                   |  |
| 2 b        | 3000   |                   |  |
| 3 a        |        |                   |  |
| 3 b        |        |                   |  |

You can define several point pairs (a maximum of 25), for example for multiple starts and stops in a cycle in order to take account of different situations during the start, for example. With the commands `SYNCCSTART pnum` and `SYNCCSTOP pnum slavepos`, you can determine in your program which point pair is to be used.

If the A and B points are identical, the slave will be engaged with the set maximum velocity, i.e. without curve, as soon as the master has reached this point. If no start stop points have been defined, the slave will be engaged with the set maximum velocity in the case of `SYNCCSTART`.

These points are always entered in ascending order, too. This is because the run direction will be taken into account automatically by the program: When moving forward, the synchronization begins at point A and is finished up to point B. When moving backward, it begins at point B and is finished up to point A.

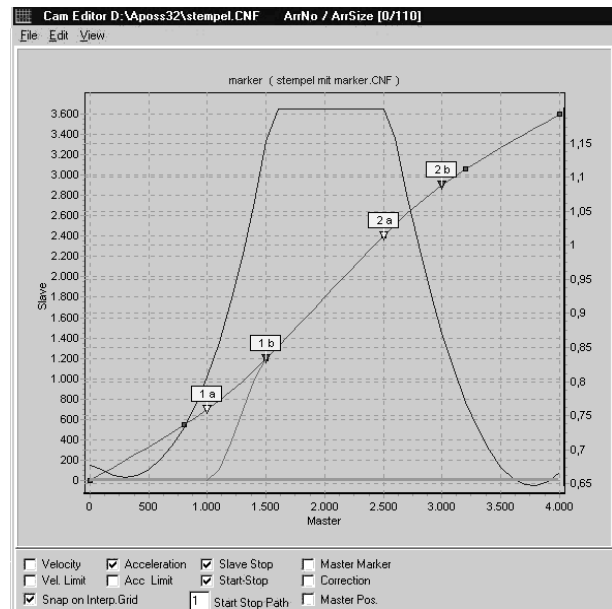
If the program is closed without the explicit command `SYNCCSTOP pnum slavepos`, the second point pair will always be used for disengaging.

How to graphically display Start Stop Points and progression

You can visually display the start stop points and the engage and disengage curves in the curve profile. Activate →  "START STOP". Yellow flags indicate the point pairs for the engaging and disengaging in the synchronization.

In order to display the engage and disengage curves, enter the number of the point pair whose progression you want to see in → "START STOP PATH". Without an algebraic sign, engaging is shown when the master moves forward. Numbers with a negative sign show disengaging when the master moves forward.

When the master moves backward, the algebraic signs apply to the reverse situation, i.e. plus for disengaging and minus for engaging.



### ■ Index Cards "CURVE DATA", "CURVE INFO" and Parameter

Before you edit a curve, you should always first load the parameters of your control into the CAM-Editor.

You can save the user parameters including the arrays into a file with the extension „cnf“ with "PARAMETERS" → "SAVE TO FILE". → "LOAD" this file into the CAM-Editor.

If you do not load any parameters, you will find that the VLT factory settings have been entered.

If you change the parameters in the course of creating the curve, they will also be saved in the cnf file and loaded into the control with Parameter → "RESTORE FROM FILE" and entered in the corresponding dialog fields of the axis parameters.

### ■ Index Card "CURVE DATA"

You can determine important key data of your curve in the index card → "CURVE DATA":

| Curve Data          | Curve Info | Encoder | Home |
|---------------------|------------|---------|------|
| Name of Curve       | marker     |         |      |
| Type of Curve       | 0          |         |      |
| Number of Intervals | 40         |         |      |
| Slave Stop Position | 0          |         |      |
| Correction Start    | 3000       |         |      |
| Correction End      | 1000       |         |      |
| Master Marker Pos.  | 1000       |         |      |
| Slave Marker Pos.   | 0          |         |      |
| Master Length       | 4000       |         |      |
| Slave Length        | 3600       |         |      |

#### Name of Curve

If you edit several curves, you can give meaningful names to the curves for your own information here. If a cnf file contains several curves, these names are offered for selection in "FILE" → "LOAD CNF".

#### Type of Curve

In order to prevent velocity leaps in the case of repeated curve cycles, you can choose between two curve types. In either case, the interpolation takes account of the gradient of the curve at the beginning and end.

Select the curve type:

- 0 = The gradient of the curve at the beginning and end is averaged.
- 1 = The gradient at the beginning of the curve is also used for the end of the curve.

#### Number of Intervals

You can define the number of straight sections that are used for the entire curve with the number of intervals.

Do not select intervals that are too small (this would only lead to an unnecessary overhead), and use a whole number factor of the master length, if possible. For example, use an interval of 30 or 60 in the case of a master length of 3000. You can then put on the fixpoints without error when →  "SNAP ON GRID" is turned on.

#### Slave Stop Position

Determine the position where the slave should run to and stop if no SYNCSTOP command with the variable *slavepos* was set in the program. This position will also be used if SYNC starts with a specific number of cycles and does not use a SYNCSTOP command.

A grey line indicates this position in the curve profile. Activate →  "SLAVE STOP" for this purpose.

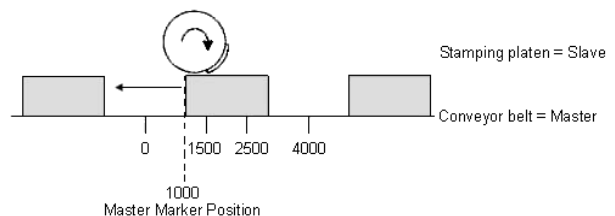
#### Correction Start / Correction End

Enter the master positions where the master correction is supposed to begin and where it is supposed to end. Be careful to leave enough time to correct the synchronization before the processing point is reached.

The correction area is shown in blue in the curve profile. Activate →  "CORRECTION" for this purpose.

#### Master Marker Position / Slave Marker Position

Enter the master position (or the slave position in the case of a slave synchronization with marker) for which the marker has been set, here for example the beginning of a cardboard box.



The position of the curve where the marker is detected is calculated from the master marker position and the marker distance. This position is shown as a green line in the curve profile and allows you to fix the correction area. Activate →  "MASTER MARKER" or "SLAVE MARKER" for this purpose.

### Master Length / Slave Length

Information about the cycle length of the master resp. slave set in the table of fixpoints.



#### NB!

Slave length must be positive, this can be ensured by defining positive direction in parameter POSDRCT (28).

### ■ Index Card "CURVE INFO"

In this index card, you can determine the number of → "CYCLES/MIN MASTER" in the input field. In the other fields, you can find curve information calculated from the parameters and the curve application.

|                     |         |
|---------------------|---------|
| Cycles / min Master | 60      |
| Max actual Velocity | 1,200   |
| Max Slave Velocity  | 2,250   |
| Max actual Accel.   | 0,001   |
| Max Slave Accel.    | 0,001   |
| Interval Size       | 100,000 |
| Interval Time (ms)  | 25,000  |

### Cycles/min Master

Enter the number of cycles of the master per minute. In most cases, this will be the (maximum) number of products that are processed per minute.

### Max. Actual Velocity

This value indicates the maximum velocity of the slave in this curve application in units. You can graphically display the entire velocity progression in the curve profile. Activate →  "VELOCITY". The blue curve shows the velocity progression of the slave in this curve application. On the right axis of the diagram, you can read the values in units that describe the change of the slave in relation to the master:  $\Delta$  UU/MU.

### Max. Slave Velocity and Velocity Limit

This is the maximum velocity which the slave can reach, depending on the velocity of the master. The faster the master runs or the more cycles/min. it runs, the lower the velocity limit becomes. If the velocity progression exceeds this limit, the slave will not be able to follow the master. You can also graphically display the velocity limit in the curve profile. Activate →  "VEL. LIMIT".

### Max. Actual Acceleration

The value indicates the maximum acceleration of the slave in this curve application in units that describe the change of the slave in relation to the master:  $\Delta$  UU/MU.

You can graphically display the entire progression of the acceleration in the curve profile. Activate →  "ACCELERATION". The yellow line shows the acceleration progression of the slave in this curve application. Normally, the acceleration is relatively small in comparison with the velocity. Thus, it is recommended to deactivate the display of the velocity and the velocity limit in order to see a good representation of the acceleration in the diagram.

### Max. Slave Acceleration and Acceleration Limit

This is the maximum acceleration which the slave can reach, depending on the acceleration of the master. The faster the master runs or the more cycles/min. it runs, the lower the acceleration limit becomes. If the acceleration progression exceeds this limit, the slave will not be able to follow the master.

You can also graphically display the acceleration limit in the curve profile. Activate →  "ACC. LIMIT".

### Interval Size

The interval size is derived from the number of intervals per master cycle length.

### Interval Time (ms)

The time in (ms) for an interval is also derived from the number of intervals per master cycle length. It should not be smaller than 30 ms. (30 to 100 ms are suitable values.) Thus, you should make changes in "CURVE DATA" → "NUMBER OF INTERVALS" in order to get a reasonable value.

### ■ Index Cards Encoder, Home, Inputs/Outputs and PID

For further information regarding content, units, input areas and factory settings of these parameters, consult the [Parameter Reference](#) section or select the input field of the parameter and press [F1].

## Index Card Synchronization

For further information regarding content, units, input areas and factory settings of these parameters, consult the [Parameter Reference](#) section or select the input field of the parameter and press [F1].

| Curve Data          | Curve Info | Encoder | Home          | Inp/Outp | PID  | Synchron. | Velocity |
|---------------------|------------|---------|---------------|----------|------|-----------|----------|
| Syncfactor master   | 25         | [49]    | Master Marker |          |      |           |          |
| Syncfactor slave    | 9          | [50]    | Type          | 0        | [60] |           |          |
| Position Offset     | 0          | [54]    | Count         | 1        | [52] |           |          |
| Velocity Limit (%)  | 0          | [66]    | Distance      | 0        | [58] |           |          |
| Sync. filtertime    | 0          | [65]    | Tolerance     | 0        | [68] |           |          |
| Accuracy            | 1000       | [55]    | Slave Marker  |          |      |           |          |
| Ready (num markers) | 1          | [56]    | Type          | 2        | [61] |           |          |
| Error (num markers) | 10         | [57]    | Count         | 1        | [53] |           |          |
| Marker start        | 0          | [62]    | Distance      | 3900     | [59] |           |          |
|                     |            |         | Tolerance     | 200      | [69] |           |          |

For a CAM Control you need following parameters:

### Syncfactor Master and Slave

The two parameters SYNCFACTM [49] and SYNCFACTS [50] are used to determine the MU units in the cam control.

### Marker Distance

Enter the distance of the sensor to the processing point here; in the case of master markers in SYNCPULSM [58] and in the case of slave markers in SYNCPULSM [59].

The position of the curve where the marker is detected is calculated from the master marker position and the marker distance. This position is shown as a green line in the curve profile and allows you to fix the correction area.

Activate →☒  "MASTER MARKER" or "SLAVE MARKER" for this purpose.

### Tolerance

Tolerance window for the appearance of the master markers (Marker monitoring) SYNCMWINM [68] or the slave markers SYNCMWINS [69]. The tolerance window is shown as a green area in the curve profile. Activate →☒  "MASTER MARKER" or "SLAVE MARKER" for this purpose.

## Index card Velocity

For further information regarding content, units, input areas and factory settings of these parameters, consult the [Parameter Reference](#) section or select the input field of the parameter and press [F1].

In addition, the maximum velocity and acceleration reached in the current application are calculated here in qc/scan time. The display in the curve profile occurs in units. Activate →☒  "VELOCITY" or "ACCELERATION" for this purpose.

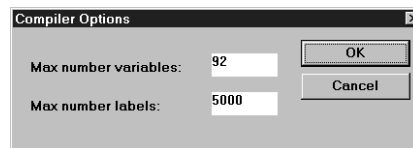
## "SETTINGS" menu



This menu offers compiler options and interface settings. You can set the colors in the editor as you wish. If you would like to change the language of the user interface, please see below.

## "COMPILER"

The default values for the compiler options are set accordingly for most applications. Thus, they do not require too much memory and, at the same time, they allow the necessary input to be made.



### Maximum number of variables

The number of variables has a direct effect on the amount of memory available in the option. It is important to remember that an array also occupies the space of a variable.

If you need more than 92 variables (incl. arrays) then increase this number.

### Maximum number of labels

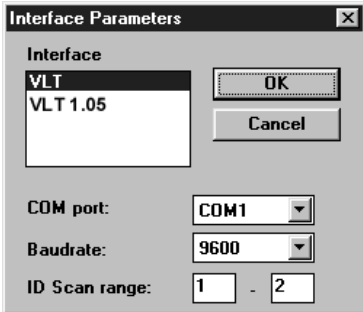
The maximum number of labels determines how much memory is available for internal hyperlinks. Internal hyperlinks are automatically created for all branches of the program (GOTO, IF, LOOP, REPEAT, WHILE, GOSUB) during compiling. The recommended range is between 100 and 500 internal labels.

Increase the maximum values permitted if the number of labels is not sufficient for text input.

■ "INTERFACE"

You have already determined the interface settings while starting the VLT.

If you have to change one of the settings, for example the baud rate, click on "SETTINGS" → "INTERFACE" and enter the change.



**NB!**

However, it is important to remember that a change in the interface settings can cause the communication between the PC and the VLT to be lost.

Scan Range

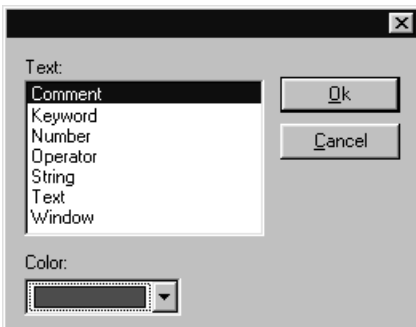
Define the range for scanning the connected VLTs. If you add one or more VLTs in the network, you have to extend the range.

RS485 connection

You need the RS232 standard interface in the PC or an additional RS232 interface card and an external converter for a RS485 connection.

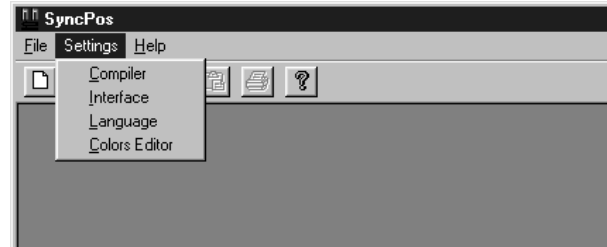
■ "COLORS EDITOR"

In order to provide greater clarity, different colors can be assigned to the various program sections such as comment, key word, number etc. To do this, open the "COLORS EDITOR" in menu "SETTINGS." Select the type, e.g. Comment, and select the desired color. Click OK to store the new settings.



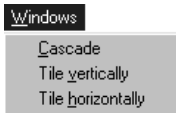
■ "LANGUAGE"

If you want to access the "SETTINGS" menu to change the language, then all dialog windows and edit windows must be closed. To close these windows click on the close icon in the upper right-hand corner of the open edit or dialog window.



If you desire another language, click on "SETTINGS" → "LANGUAGE" and choose from English and German in the subsequent dialog field. "EXIT PROGRAM" and start SyncPos again.

### ■ "WINDOW" menu



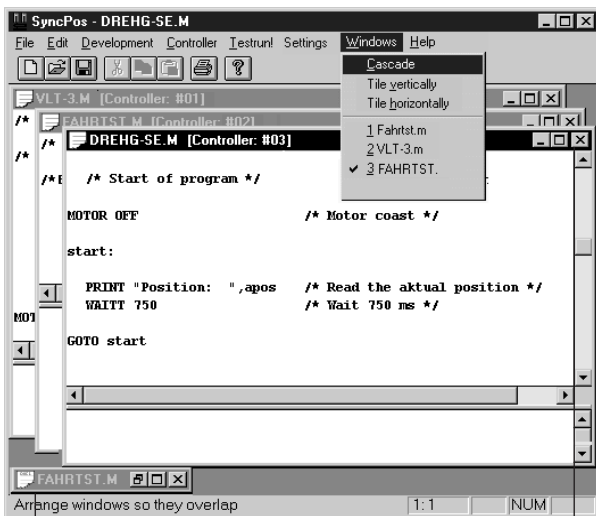
You can open several edit windows resp. files and each edit window can be linked with a controller. In order to display a number of editing windows choose between → "CASCADE", "VERTICAL" (alongside each other) or "HORIZONTAL" (on top of each other) in the "WINDOWS" menu. Click on the full screen symbol, if you do not want to cascade the windows or display them next to each other.

In general, if you chose to display files as symbols they are displayed at the bottom left of the SyncPos window.

### ■ "CASCADE"

All opened files are listed in the "WINDOW" menu. Click on the file that should be on top when the windows are cascaded.

The files are shown slightly staggered on top of each other:



File symbol

Full size symbol

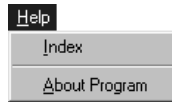
### ■ "TILE VERTICALLY"

If you want to look at a number of programs at the same time you can choose between different window representations. You can, for example, split the SyncPos window and display the files alongside each other.

### ■ "TILE HORIZONTALLY"

The SyncPos window is divided up and the files shown on top of each other.

### ■ "HELP" menu



The representation and functionality of the online-help varies, depending on the operating system used.

### ■ "INDEX"

Click on "HELP" → "INDEX". For example, if you are looking for details on the ACC command, select in the table of contents the topic "All SyncPos Commands from ACC to #INCLUDE" and click on ACC.

#### Text retrieval

For text retrieval first click on "SEARCH", so that the word list is created. Then enter the search term "acc", mark an equivalent term and then select the desired topic from the list displayed.

#### Search via Index

Or enter "acc" in the index and select the corresponding index entry.

#### Context-sensitive help

The "COMMAND LIST", the parameter dialog fields in the menu "CONTROLLER" as well as the "CAM-Editor" offers a direct access to the online help. Mark a command in the "COMMAND LIST" resp. select one of the input fields of a parameter and press [F1]. You will get the corresponding information.

#### Cross-references

Cross-references to other texts are marked in green. Click on the marked cross-reference and the desired text is displayed. Click next on "BACK" in the menu bar to read again the previous section.

Click on the words marked with green dotted line and the corresponding popup shows images, graphics and explanation from the glossary, for example MLONG. Click on beside of the popup or press the Esc-key to close the popup.

#### Printing or copying help texts

You can print the help text or insert it in your edit window with "EDIT" → "COPY". For example, this can be used to insert complete program strings from program samples.

### ■ "ABOUT PROGRAM"

Here you can find the version numbers of the SyncPos program, the program library and the compiler.



**Chapter 5 a**

- **PC Software Interface in MTC10 Mode**
- **Operation Mode: Offline or Online ..... 83**
- **Opening SyncPos in MCT10 Mode ..... 83**
- **Closing SyncPos in MCT10 Mode ..... 83**
  - Manual closing ..... 83
  - Automatic closing ..... 83
- **File Handling ..... 83**
- **"FILE" menu in MCT10 Mode ..... 84**
  - "FILE" → "NEW" ..... 84
  - "FILE" → "OPEN" ..... 84
  - "FILE" → "SAVE AS" ..... 84
  - "FILE" → "PRINT" and → "PRINT SETUP" ..... 84
  - "FILE" → "LAST FILE" ..... 84
- **"DEVELOPMENT" menu in MCT10 Mode ..... 84**
  - Online Mode ..... 84
  - Offline Mode ..... 84
  - "DEVELOPMENT" → "SYNTAX CHECK" ..... 84
  - "DEVELOPMENT" → "COMMAND LIST" ..... 84
- **CAM Editor in MCT10 Mode ..... 85**
- **CAM-Editor window without curve data ..... 85**
  - "FILE" menu (CAM-Editor) in MCT10 Mode ..... 85
  - "EDIT" menu (CAM-Editor) in MCT Mode ..... 86
  - "VIEW" menu (CAM-Editor) in MCT Mode ..... 86
- **"SETTINGS" menu in MCT10 mode ..... 86**

SyncPos behaves different, if started directly by clicking on the application's icon or if started indirectly by opening up a program or configuration file via the Motion Control Tool MCT10. The main differences are related to the fact that all file handling is limited to MCT10 only. These limitations concerning the SyncPos GUI are explained in this chapter, e.g. how to open and save files or how to edit a cnf-curve or set parameters.

The following chapter describes the differences in the SyncPos User Interface, when running in MCT10 Mode, for example the file handling or editing a CAM curve. In addition the user interface is different depending of the mode of operation: Offline or Online Mode.

**■ Operation Mode: Offline or Online**

SyncPos is called up by MCT10 in Offline or Online mode depending, if MCT10 has a connection established to the drive. The selection of Offline or Online operating mode as well as the interface setting is done by MCT10 at start-up of SyncPos and can not be changed while SyncPos is running.

**■ Opening SyncPos in MCT10 Mode**

In MCT10 mode you cannot start up SyncPos as usual with "START" → "PROGRAMS" → "SYNCPoS", but it is opened automatically by selecting a program file (\*.m) or a configuration file (\*.cnf) via the MCT10.

MCT10 starts SyncPos, transfers the selected file name to it and opens this file. It is possible to open up multiple instances of SyncPos via the MCT10 file selection.

MCT10 also configures the language and interface settings of SyncPos.

**■ Closing SyncPos in MCT10 Mode**
Manual Closing

SyncPos can be closed by the menu item "FILE" → "EXIT PROGRAM" or the Exit button.

Automatic Closing

SyncPos is automatically closed in case of exit the program editor as well as when a possible opened configuration file in the CAM Editor is closed by the user. The reason for this is, that any file handling has to be done via MCT10 and there is no possibility to open up a new or existing file in SyncPos, if running in MCT10 mode. So, it isn't necessary to keep SyncPos active, if all files are closed.

**■ File Handling**

MCT10 is responsible for the handling of SyncPos program files (\*.m) as well as SyncPos configuration files (\*.cnf). This means that files can only be created, opened up, renamed or printed via MCT10. There is no file selection possible in SyncPos running in MCT10 mode. MCT10 commands SyncPos to open up the file selected by the user via the MCT10. Due to the fact that file handling is strictly restricted to MCT10, there are some menu items and functions of SyncPos disabled.

**■ The SyncPos Window in MCT10 Mode**
MCT10 Operation Mode: Online

The CAM-Editor menu in the title bar and most of the icons in the symbol bar are disabled:



The "CAM EDITOR" can not be opened up by SyncPos running in MCT10 mode, but it is always opened up automatically, if a configuration file (\*.cnf) is selected for editing via MCT10 or if the program contains a cnf-file.

The icons "NEW FILE", "FILE OPEN", "PRINT" and the "CAM-EDITOR" are disabled, too. See the following description, how to create a new file, open and save files.

MCT10 Operation Mode: Offline

In operation mode offline all functions, which require access to the drive, can not be used. There are some more menu items disabled:


"CONTROLLER" menu

There is no connection to the drive possible. It can not be configured or commanded via SyncPos.

"TESTRUN" menu

There is no connection to the drive possible. So, there is no possibility for a testrun.

"CAM-EDITOR" menu

The CAM editor can not be opened up by SyncPos running in MCT10 mode, but it is always opened up automatically, if a configuration file (\*.cnf) is selected for editing via MCT10 or if the program contains a cnf-file.

**■ "FILE" Menu in MCT10 Mode**

Due to the fact that all file handling is done by MCT10, the following items of the File menu are disabled:


**"FILE" → "NEW"**


---

New files have to be created via MCT10.

**"FILE" → "OPEN"**


---

Select the file via the MCT10. SyncPos and with that the file is opened automatically.

**"FILE" → "SAVE AS"**


---

Please use the features of the MCT10, to rename a program file (\*.m) or copy it thereby. But you can → "SAVE" the actual program or changes with the same file name in SyncPos.

**"FILE" → "PRINT" and → "PRINT SETUP"**


---

You can print program files (\*.m) via the MCT10. The print setup has to be done via the MCT10 of course.

**"FILE" → "LAST FILE"**


---

There is no last file list, because SyncPos can not open up any file, if running in MCT10 mode.

**■ "DEVELOPMENT" Menu in MCT10 Mode**
**Online Mode**


---

In Online Mode there is no difference to the SyncPos user interface; you can use all functions.

**Offline Mode**


---

In Offline Mode all functions, which require access to the drive, can not be used. Most of the menu items are disabled:



But following two items you can use in MTC10 offline mode:

**"DEVELOPMENT" → "SYNTAX CHECK"**


---

This can be used to check the syntax of the editor's program file.

**"DEVELOPMENT" → "COMMAND LIST"**


---

The Command List offers a limited functionality. It can just be used to insert commands in the editor's program file or to get context-sensitive help of a command.

But it is not possible to execute any of the commands directly or to move the drive to position.

## ■ CAM Editor in MCT10 Mode

You can open configurations files (\*.cnf) only via the MCT10 such as it is with program files. MCT10 commands SyncPos to open up the file selected by the user via the MCT10.

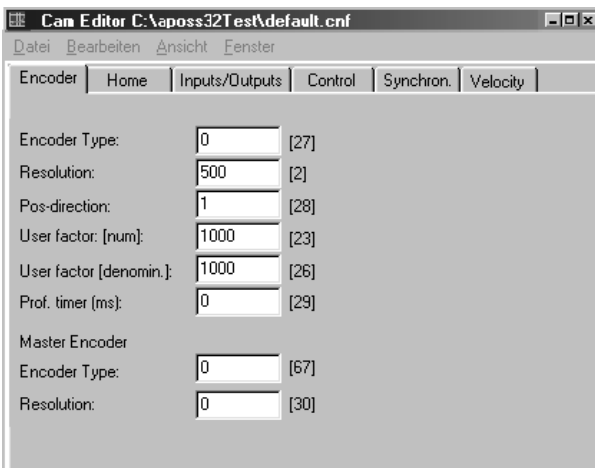
If there is a configuration file (\*.cnf) including curve data, the CAM-Editor is open up automatically and you can edit the parameters (which the file contains) and edit interactive a possible present curve profile using all CAM-Editor features.

If there is a configuration without curve data, a modified reduced CAM-Editor including the parameter index cards is opened and it is possible to → "INSERT CAM-PROFILE" further if necessary.

If there is no configuration file with the start-up of SyncPos, the CAM-Editor is not opened and there is no possibility to do that by way of addition or to open a file for the CAM-Editor.

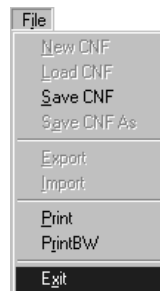
## ■ CAM-Editor window without curve data

Even when the configuration file doesn't contain curve data, you can change the parameters which are stored in the file using the reduced CAM-Editor. In this case the modified CAM-Editor shows instead of the four sectors just the parameter index cards.



## "FILE" menu (CAM-Editor) in MCT10 Mode

Due to the fact that all file handling is done by MCT10, many menu items are disabled permanent or specific:



## Disabled menu items

"FILE" → "NEW CNF"

Create new configuration files via MCT10.

"FILE" → "LOAD CNF"

The file selection and opening has to be done via the MCT10. The file will be load automatically when SyncPos is opened.

If the automatically opened configuration file doesn't contain a curve, you can insert one with "EDIT" → "INSERT CAM-PROFILE".

"FILE" → "SAVE CNF AS"

Please use the MTC10 features to rename a configuration file (\*.cnf) or copy it thereby. But you can save the actual file resp. the changes under the same file name using → "SAVE CNF".

"FILE" → "EXPORT" and → "IMPORT"

It is not possible to export or import any curve data. This has to be done via MCT10.

## Specific menu items

"FILE" → "PRINT" and → "PRINT SW"

If there are curves within a configuration file, you can print the graphics, because there is no possibility to print diagrams with the MCT10. If there are no curves within a configuration file, the menu item is disabled.

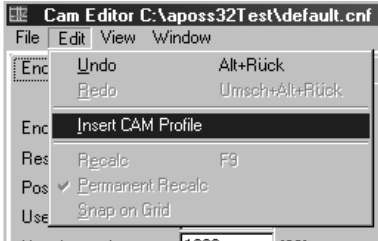
"FILE" → "EXIT"

You cannot open again a configuration file, if you had closed them manual with "FILE" → "EXIT" or with clicking on the **Exit** button, because they have to be selected via the MCT10. If the program file isn't opened anymore, SyncPos will be closed also.

**"EDIT" menu (CAM-Editor) in MCT Mode**


---

A CAM profile can be inserted via the new item "EDIT" → "INSERT CAM PROFILE" in configuration files, which doesn't contain a curve.

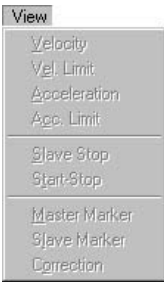


A blank curve profile is created and the "CAM-EDITOR" is opened with all four sectors to define the curve points and the curve data and to edit interactive the curve.

**"VIEW" menu (CAM-Editor) in MCT Mode**


---

As long as there is no curve data present in the configuration file, all items are disabled:


**■ "SETTINGS" Menu in MCT10 mode**

The menu is reduced to the "SETTINGS" → "COMPILER" and → "COLORS EDITOR".


**Interface Settings and Language Selection**


---

All the interface settings and the language selection are given to SyncPos during start-up by MCT10. The interface settings and the language selection too, can not be modified by the user via SyncPos in MCT10 mode.

**Chapter 6**

|                                                    |     |
|----------------------------------------------------|-----|
| <b>■ Programming with SyncPos</b>                  |     |
| Fundamental Program Layout .....                   | 89  |
| Fundamental Command Structure .....                | 89  |
| Interrupt .....                                    | 90  |
| Elements of the Programming Language .....         | 92  |
| Arithmetic .....                                   | 93  |
| User unit .....                                    | 95  |
| <b>■ Software Reference</b>                        |     |
| List of Commands .....                             | 96  |
| All Commands from ACC to #INCLUDE.....             | 99  |
| <b>■ Parameter Reference</b>                       |     |
| VLT parameter and SyncPos parameter .....          | 163 |
| VLT parameters .....                               | 166 |
| VLT parameter list .....                           | 167 |
| All SyncPos parameters in Alphabetical Order ..... | 168 |
| All SyncPos parameters in Detail .....             | 170 |

The following chapters describe how to program using SyncPos. Beginners should read the basic explanations on the programming language SyncPos, i.e. program layout, command structure, interrupt, elements of the programming language, arithmetic and user unit. Experienced users should inform themselves about the SyncPos-specific basic principles, e.g. user unit or parameters.

All commands are described in the Software Reference, first in a general overview and then alphabetically ordered, in detail and complemented with short examples. You can reconstruct as many as 50 programs with assistance of the information in the example programs in the Online-Help.

And in chapter Parameter Reference all the parameters are described, first in general and then in detail.

**■ Programming with SyncPos**
**■ Fundamental Program Layout**

- Definitions: Arrays, Interrupts, User parameters
- Initializing: Setting parameters, flags and variables
- Main program loop
 

```
main:
-
GOTO main
```
- Sub program area
 

```
SUBMAINPROG
SUBPROG name
-
RETURN
ENDPROG
```

Definitions

|                 |                                                                                                               |
|-----------------|---------------------------------------------------------------------------------------------------------------|
| Array           | DIM send[12], receive[12]                                                                                     |
| Interrupt       | ON ERROR GOSUB errhandle<br>ON INT -1 GOSUB stopprog<br>ON PERIOD 500 GOSUB calc<br>ON TIME 10000 GOSUB break |
| User parameters | LINKXPAR SYNCACCURACY 710<br>"ACCURACY [qc]" 0 100000 0<br><br>LINKGPAR 133 716 "Offset [qc]" 0<br>100000 0   |

Initializing

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| Parameters        | SET POSERR 100000000<br>SET 133 10000<br>SETVLT 205 50 |
| Flags/variables   | offset = 0<br>sync_flag = 0                            |
| System parameters | VEL 100<br>ACC 100<br>DEC 100                          |

Main program

```
main:
IF (IN 3 == 1) THEN
/* Go into synchronizing mode, if input 3 = 1 */
GOSUB syncprog
ELSE
GOSUB speedprog
/* If input 3 not = 1, run in speed mode */
GOTO main
```

Sub programs

```
SUBMAINPROG:
SUBPROG syncprog
IF (sync_flag ==0) THEN
/* synchronize, if not already synchronized */
SYNCP
sync_flag = 1
ENDIF
RETURN
SUBPROG errhandle
WAITI 18 on
/* waiting for digital input 18, clear the error */
sync_flag = 0
ERRCLR
RETURN
ENDPROG
```

Sequential command processing

In general a command is processed to the end before a new command is begun. This means that for position commands the program waits until the target position has been reached.  
Exception: If NOWAIT has been set to ON.

**■ Fundamental Command Structure**

All instructions consist of: COMMAND WORD + possible **Parameter**. A variable can also be used as parameter instead of an absolute number.

Example

```
POSA 10000
or
pos = 10000
POSA pos
```

Command run times

If the command execution times are critical in an application it is possible to measure the run times of a command sequence under the different operating conditions using the command TIME.

Tips for Increasing Program Readability

- Use of capital and small initial letters (i.e. all commands capital letters, all variables small).
- Placement of spacing between command parts.
- Place comments in your program. The comments are between `/* ... */` or after `//...`  
`/* Begin COMMENT End */`  
`// Begin COMMENT End`  
Inadmissible is nesting comments (`/* ... /*...*/`)
- Use of line identification within the loop.

Input values

As in other programming languages the values inputted are not tested. Thus, it is the programmer's responsibility to ensure that extreme values do not lead to problems. When searching for such potential problems use the debug mode.

■ **Interrupt**

In general there are three types of interrupts:

- ON INT                                   Interrupt at the edges of an input
- ON PERIOD / ON TIME                Interrupt after a certain period of time
- ON COMBIT / ON STATBIT          Interrupt when Bit n is set
- ON PARAM                              Interrupt when a parameter n is changed

General processing of interrupt procedures

After every internal SyncPos command a query is made whether an interrupt event has occurred. It is important to remember that with every internal SyncPos command the compiler creates a command in SyncPos machine code.

Thus, for example, a simple command such as:

```
POSA (target + 1000)
```

is broken down into the following SyncPos machine code:

```
MOVE target to register 101
MOVE immediate 1000 to register 102
ADDREG register 102 plus register 101 to register 101
POSA axis to register 101
```

Furthermore, for commands which take longer (such as DELAY or WAITX) the program constantly checks whether an interrupt event has occurred. If this is the case, the command is interrupted and continued once the interrupt has been processed.



**NB!**

Do not use WAITT in connection with interrupts since the waiting process starts again after the interruption.

Use of variables within interrupt procedures

The example above with the "SyncPos machine code" clearly shows that it is necessary to use the utmost care when assigning variables within interrupt procedures.

If, for example, in the main program the following assignment is made:

$$target = target + value - 1000$$

this is broken down into a series of SyncPos machine code commands and the intermediate results are stored in temporary registers. Only at the end of the sequence is the result stored in *target*.

If during the execution of this command an interrupt is triggered and in the corresponding procedure the following command is executed:

$$target = 0$$

then, in this instance, problems will arise. This is because after processing the interrupt procedure the program jumps back to the main program and then the intermediate result which still exists is stored in *target*: Thus, the 0 in *target* is overwritten once again.

ON PERIOD

In contrast, for ON PERIOD functions the time when the next call instruction should take place is calculated at the start of such a function, thus

$$START\_TIME = TIME + PERIOD.$$

As soon as this time has been reached the function is executed and subsequently the next start time is calculated with the following formula

$$START\_TIME = START\_TIME + PERIOD.$$

This ensures that the call intervals are really the same since the execution time does not influence the calculation. But this means that the user must make sure that the period of time is actually longer than the execution time as otherwise a "jam" is created. That means that actually only the ON PERIOD function is executed.

### Response times

The existence of an interrupt is checked in a special function which is also used as a watch dog control. For this reason this is generally called up in any procedure which could last somewhat longer and in all loops, etc.

This procedure checks every 1 ms whether such an event exists and, if necessary, sets a corresponding flag. At the latest this flag is detected and evaluated after the current APOS machine code has been processed.

The response time is the maximum run time of the machine code or 1 ms, whichever is greater. For the VLT it is approx. 1 ms.

One exception is the time interrupt (ON TIME / ON PERIOD). This checks whether the time has elapsed every 20 ms. Thus, it is not logical to define an ON PERIOD with less than 20 ms.



#### NB!

Furthermore, in general, it is important to make sure that interrupt functions do not last too long. Particularly for ON PERIOD functions it is important to ensure that the function does not last longer than the period since otherwise a „jam“ of function procedure calls will be created.

### Priorities

If two interrupt events should occur simultaneously then the processing is prioritized as follows:

If two interrupt events should occur simultaneously then the processing is prioritized as follows:

ON INT comes before

ON APOS, ON MAPOS, ON MCPOS before

ON COMBIT before

ON STATBIT before

ON PARAM before

ON TIME / ON PERIOD

but the other events are not lost.

Within the individual types of interrupts the following is true:

### ON INT / ON COMBIT / ON STATBIT

If two (input) interrupts occur simultaneously, then the one with the lower number is executed first, however the other is not lost. After the interrupt procedure is completed the other is called up accordingly.

If the same input resp. interrupt occurs again while the procedure is being executed this is noted again and subsequently executed.

Thus, an interrupt can only be lost if it occurs twice during the execution of an interrupt procedure.

### ON TIME / ON PERIOD

As described above the execution time for every temporal function is stored in an internal structure.

For simultaneous execution times the procedure that is first on the list will be executed first. The priority is thus determined by the sequence of the ON PERIOD commands.

### ON PARAM

If several of these interrupts occur simultaneously, they are processed according to the sequence of the ON PARAM commands in the program.

### Interrupt nesting

It is not possible for one interrupt to be suspended by another. Accordingly, while one interrupt is being processed a second interrupt cannot be processed. The only exception is the ON ERROR function, which is also possible during the processing of interrupts.

However, an ON ERROR function cannot be suspended by an interrupt.

### NOWAIT in interrupts

In general, during an interrupt NOWAIT is set to ON, that means that the program does not wait for the completion of POSA commands.

This is necessary since otherwise a POSA command cannot be suspended by an interrupt procedure, since this would immediately wait for the arrival of the axis. Thus, if you wish to wait for the arrival of an axis during an interrupt procedure, this must be done explicitly with WAITAX.

**■ Elements of the Programming Language**
Constants

Constants can be used anywhere where parameters or values are expected. Constants are usually entered in integral numbers, for example:

value = 5000

Constants ...

- ... are integer number values between -2 to +2 billion,
- ... are valid within the entire program (they are global),
- ... can be entered as a decimal, hexadecimal (0x + hexadecimal number), octal (0 + octal number) or in ASCII (between apostrophes), for example:

value = 5000           = decimal 5000

value = 0x7F           = decimal 127

value = 0100           = decimal 64

value = 'A'            = decimal 65

Hexadecimal and ASCII entries, in particular, avoid many conversions and make the program more readable, for example:

key = 'A'

The advantage of constants is, that they don't need own storage capacity.

Variables ...

- ... can only be used for intermediate data storage of inquiry and calculation results.
- ... occur via the allocation of a value.
- ... must not be defined separately.
- ... are valid within the whole program, (i.e. they are global)
- ... contain integer number values between -2 to +2 billion.
- ... can be used within commands, instead of constant values.
- ... must be allocated a value before use in a command.

Variable Identification Names

- ... can be of any length
- ... can consist of letters, numerals and underlines
- ... must not contain any country-specifics, such as "ä", "é"
- ... must begin with a letter
- ... can be written in small or capital letters (no difference!)
- ... may not be identical to a command name

Special Variables

ERRNO = A system variable, which contains the relevant error number

Arrays

Writing programs with dialog requires user input or positions to be stored for a longer period of time, for example, even after the VLT has been turned off. Usually such input consists of several values which are best stored in fields or arrays.

Arrays are stored in the memory area of the user program and are defined globally, that means they are independent of the current program. The user can determine how many arrays are defined and how large the individual arrays should be. This determination is made with the DIM statement and is then fixed and cannot be changed (except by erasing memory). Each program that is intended to use arrays must contain a corresponding DIM statement which corresponds to the original definition. Otherwise an error will be indicated.

DIM Statement

The DIM statement has to be the first statement in the program and must appear before the subroutine area.

The DIM statement specifies the arrays to be subsequently used. If no arrays have been previously created then they will be created now. If arrays had been previously defined then it is important that the information corresponds with the original definition.

Example

```
DIM target1[20], target2[20], target3[20], plant_offset[50]
DIM parameter[10]
```

With these commands a total of 5 arrays have been defined with their corresponding sizes. If this program is executed once then the arrays listed above will be created in the SyncPos option. If, when the program is re-started, it is determined that the definition of the arrays differs from the arrays in the SyncPos option, then this is indicated as an error. However, it is correct if a second program only contains the following line:

Example

```
DIM target1[20], target2[20], target3[20]
```

However, the sequence of definition must always be the same since the SyncPos option does not store the names of the arrays but only their position in the DIM statement. Thus the following program line is also correct and the xpos array is identical to the target1 array.

Example

```
DIM xpos[20], ypos[20], zpos[20], offs[50]
```

### Indexes

The elements of an array are designated by a corresponding index in square brackets: xpos[5]. Indexes are allowed from 1 to the size of the array defined. Thus, in the above case for xpos from 1 to 20. If an attempt is made to access elements before or after this array then an error message is generated since this could lead to data overrun and destruction of the array.

### Reading and Writing Arrays

Access to the arrays thus defined is made analog to the use of variables. Thus all of the following statements are correct:

Example

```
xpos [1] = 10000
xpos [2] = 20000
xpos [3] = 30000
i = 1
WHILE (i<20) DO
ypos [i] = i*1000
i = i+1
ENDWHILE
zpos [1] = APOS
POSA xpos [1]
offs [1] = (xpos[2]) % 20
```

### Arrays versus Variables

In general arrays can be used everywhere variables are also permitted. Furthermore, an array only occupies the location of an internal variable and thus only reduces the number of maximally permitted variables by one. The maximum number of variables can be set in the menu "SETTINGS" → "COMPILER".

### ■ Arithmetic

The compiler offers the following commands and parameters:

|                       |                                                                                                  |
|-----------------------|--------------------------------------------------------------------------------------------------|
| Operators             | plus, minus, times, divided by, XOR, Modulo, Division, Absolute amount                           |
| Bit operators         | and, or, invert, left shift, rightshift, bit, byte, word, long                                   |
| Comparison Operations | greater than, less than, greater than or equal to, less than or equal to, the same as, not equal |
| Logical Operations    | and, or, not                                                                                     |

Inform yourself about the type of assignment operation which is structured in accordance with the Bit/Byte commands and about the priorities of the operators and the operations.



### NB!

All arithmetical operations are integer number operations.

**Operators**

| Symbol | Meaning         | Syntax / Example                   | Description                                      |
|--------|-----------------|------------------------------------|--------------------------------------------------|
| +      | plus            | 3 + 3 = 6                          | Addition                                         |
| -      | minus           | 9 - 3 = 6                          | Subtraction                                      |
| *      | times           | 2 * 3 = 6                          | Multiplication                                   |
| %      | divided by      | 19 % 3 = 6                         | Division (result is truncated)                   |
| ^      | XOR             | expr1 ^ expr2<br>127 ^ 255 = 128   | Exclusive Or (binary operation)                  |
| mod    | modulo          | expr1 mod expr2<br>250 mod 16 = 10 | Mathematic modulo (rest of an integer division)  |
| rnd    | division        | expr1 rnd expr2<br>250 rnd 16 = 16 | Division with round-off (opposite to truncating) |
| abs    | absolute amount | Abs(expr)<br>abs (-5) = 5          | Absolute amount of the expression                |

**Bit operators**

| Symbol | Meaning     | Syntax / Example                                  | Description                                  |
|--------|-------------|---------------------------------------------------|----------------------------------------------|
| &      | and         | 7 & 6 = 6                                         | bit-by-bit relationship                      |
|        | or          | 2   4 = 6                                         | bit-by-bit relationship                      |
| ~      | invert      | ~(-7) = 6                                         | bit-by-bit inversion                         |
| <<     | left shift  | 3 << 1 = 6                                        | bit-by-bit shift to the left                 |
| >>     | right shift | 12 >> 1 = 6                                       | bit-by-bit shift to the right                |
| .      | Bit         | expr1.expr2<br>7.1 = 1<br>7.3 = 1<br>7.4 = 0      | Returns the Bit expr2 from expr1             |
| .b     | Byte        | expr1.b expr2<br>0x027F.b1 = 127<br>0x027F.b2 = 2 | Returns the Byte expr2 from expr1            |
| .w     | Word        | expr1.w expr2<br>0x0010FFFF.w2 = 16               | Returns the Word expr2 from expr1            |
| .l     | Long        | expr1.l expr2                                     | Returns the Long expr2 from expr1 (standard) |

**Comparison Operations and Logical Operations**

|                       |     |                          |
|-----------------------|-----|--------------------------|
| Comparison operations | >   | greater than             |
|                       | <   | less than                |
|                       | >=  | greater than or equal to |
|                       | <=  | less than or equal to    |
|                       | =   | the same as              |
|                       | !=  | not equal                |
| Logical operations    | AND | and                      |
|                       | OR  | or                       |
|                       | NOT | not                      |

Assignment Operation

|            |       |                                                                    |
|------------|-------|--------------------------------------------------------------------|
| Value      | = 0   | Standard assignment to a variable                                  |
| Field[1]   | = 0   | Standard assignment to an array value                              |
| Value.3    | = 1   | Bit 3 is set at 1, value = 4                                       |
| Field[1].8 | = 1   | Bit 8 is set at 1, field[1] = 128                                  |
| Value.b1   | = 72  | The lowest byte of value is set at 72<br>Value = 72                |
| Value.b2   | = 128 | Second byte of value is set at 128<br>Value = 0x00008048           |
| Value.w2   | = 15  | Second word of value is set at the value 15.<br>Value = 0x000F8048 |

Priority of the Operators and the Operations

Operators in the same line have the same priority, thus they are completed one after the other.

The priorities are described in decreasing order:

|     |    |                         |   |          |
|-----|----|-------------------------|---|----------|
| *   | %  | multiplicative          |   |          |
| +   | -  | additive                |   |          |
| >>  | << | bit-by-bit shifting     |   |          |
| >=  | <= | >                       | < | relation |
| ==  | != | equality                |   |          |
| &   |    | bit-by-bit and          |   |          |
|     |    | bit-by-bit inclusive or |   |          |
| AND |    | (logical and)           |   |          |
| OR  |    | (logical or)            |   |          |

**■ User units**

The units for the drive or the slave and the master, respectively, can be defined by the user in any way desired so that the user can work with meaningful measurements.

User Units [UU]

All path information in motion commands are made in user units and are converted to quad-counts internally. These also have an effect on all commands for the positioning: e.g. APOS, POS. Scaling determines how many quad-counts make up a user unit. For example, if it is 50375/1000, then one UU corresponds to exactly 50.375 qc.

The user can also select meaningful units for the cam control in order to describe the curve for the master and the slave. For example 1/100 mm, or 1/10 degrees in applications where a revolution is being observed.

In the CAM control, the maximum run distance of the slave or the slave cycle length are indicated in User Units UU (qc).

You can standardize the unit with a factor. This factor is a fraction which consists of a numerator and denominator:

$$1 \text{ UU} = \text{POSFACT\_Z (23)} / \text{POSFACT\_N (26)}$$

Master Units [MU]

The curve length or the master cycle length and other information (e.g. the marker distance) for the cam control are indicated in master units MU.

A factor (fraction) is used for the conversion into qc, as with the user unit:

$$1 \text{ MU} = \text{SYNCFAC TM (49)} / \text{SYNCFAC TS (50)}$$

**■ Software Reference**
**■ List of Commands**
Initialization of the SyncPos option (INI)

Commands to initialize the axis and the SyncPos option start up and define the zero point(s)

|               |                                                                  |
|---------------|------------------------------------------------------------------|
| DEFMORIGIN    | set the current master position as the zero point for the master |
| DEF ORIGIN    | sets the actual position as real zero point                      |
| DELETE ARRAYS | delete all arrays in the RAM                                     |
| ERRCLR        | clears error                                                     |
| HOME          | moves to machine zero point                                      |
| INDEX         | moves to next index position                                     |
| MOTOR OFF     | turns off motor control                                          |
| MOTOR ON      | turns on motor control                                           |
| RST ORIGIN    | resets temporary zero point                                      |
| SETMORIGIN    | set the current position as the zero point for the master        |
| SET ORIGIN    | sets temporary zero point                                        |
| SAVE ARRAYS   | save arrays in the EEPROM                                        |
| SAVE AXPARS   | save current axis parameters in the EEPROM                       |
| SAVE GLBPARS  | save current global parameters in the EEPROM                     |
| SAVEPROM      | saves memory in EEPROM                                           |
| SWAPMENC      | Swap master and slave encoder internally.                        |

Control Commands (CON)

Commands for controlling the program flow and structuring the programs.

|                  |                                                                                      |
|------------------|--------------------------------------------------------------------------------------|
| DELAY            | time delay                                                                           |
| DIM              | declaration of a global array                                                        |
| EXIT             | desired, premature program termination                                               |
| GOSUB            | calling up a subroutine                                                              |
| GOTO             | jumping within a program                                                             |
| IF THEN          | conditional program execution                                                        |
| ... ELSE IF THEN | .. with conditional alternative branching                                            |
| ... ELSE         | .. with alternative branching                                                        |
| ... ENDIF        | end of the conditional program execution                                             |
| LOOP             | repeats loop                                                                         |
| CONTINUE         | continues positioning from point of interruption, for example following a motor-stop |
| MOTOR STOP       | motor-stop with a programmed delay (with ramp)                                       |
| NOWAIT ON/OFF    | on/off switch for waiting for the command execution                                  |
| REPEAT           | beginning of repeat loop                                                             |

|                 |                                                    |
|-----------------|----------------------------------------------------|
| REPEAT... UNTIL | conditional loop, with an end criteria             |
| SUBMAINPROG     | commencement of the subroutine definition area     |
| ... ENDPROG     | end of the subroutine definition area              |
| SUBPROG         | begins a subroutine                                |
| ... RETURN      | ends a subroutine                                  |
| SYSVAR          | system variable (pseudo array) reads system values |
| WAITAX          | waits until target position is reached             |
| WAITI           | waits for input                                    |
| WAITNDX         | waits until the next index position is reached     |
| WAITP           | waits until a certain position is reached          |
| WAITT           | time delay in milliseconds                         |
| WHILE ... DO    | conditional loop with commencement criteria        |
| ... ENDWHILE    | end of the loop                                    |
| #INCLUDE        | compiler directive: embedding further data         |

Absolute Motion (ABS)

Commands for the absolute positioning of the axis

|      |                                                                                                    |
|------|----------------------------------------------------------------------------------------------------|
| ACC  | sets acceleration                                                                                  |
| DEC  | sets deceleration                                                                                  |
| POSA | positions axis absolutely                                                                          |
| VEL  | sets velocity for relative and absolute motions and set maximum allowed velocity for synchronizing |

Speed Control (ROT)

Commands to obtain a permanently driving axis with constant speed

|        |                                            |
|--------|--------------------------------------------|
| CSTART | starts the continuous movement in rpm mode |
| CSTOP  | stops the drive in speed mode              |
| CVEL   | sets the velocity for speed regulation     |

---

**Input/Output Commands (I/O)**


---

Commands for setting and re-setting the outputs, reading the inputs, reading movement information, reading system data and entering and outputting user information.

|           |                                                     |
|-----------|-----------------------------------------------------|
| APOS      | reads actual position                               |
| AXEND     | reads info on status of program execution           |
| CPOS      | reads set position                                  |
| MAPOS     | queries actual position of the master               |
| MIPOS     | queries last index or marker position of the master |
| IPOS      | queries last index or marker position of the slave  |
| MAVEL     | queries actual velocity of the master               |
| AVEL      | queries actual velocity of an axis                  |
| TRACKERR  | queries actual position error of the axis           |
| SYNCERR   | queries actual synchronization error of the slave   |
| ERRNO     | reads error number                                  |
| IN        | reads input bit-by-bit (individually)               |
| INB       | reads input by bytes (units of 8)                   |
| INAD      | reads analog input                                  |
| INKEY     | reads key codes of VLT                              |
| OUT       | sets output bit-by-bit (single)                     |
| OUTAN     | sets VLT reference                                  |
| OUTB      | sets output by bytes (units of 8)                   |
| OUTDA     | sets analog output (freely available)               |
| PID       | completes PID calculation                           |
| PRINT     | output (display) texts and variables                |
| STAT      | reads axis status                                   |
| TESTSETP  | Specify recording data for test run                 |
| TESTSTART | Start the recording of a test run                   |
| TIME      | reads internal controller time                      |
| _GETVEL   | changes sample time for AVEL and MAVEL              |

---

**Interrupt Functions (INT)**


---

|                    |                                                                    |
|--------------------|--------------------------------------------------------------------|
| DISABLE ...        | switches interrupt ... off                                         |
| DISABLE ALL        | switch off all interrupts except ON ERROR                          |
| ENABLE ...         | switches interrupt ... on                                          |
| ENABLE ALL         | switches all interrupts on                                         |
| ON APOS .. GOSUB   | call up a subprogram when the slave position xxx is passed.        |
| ON COMBIT .. GOSUB | call up a subprogram when Bit n of the communication buffer is set |
| ON ERROR GOSUB     | calls subroutine in the event of an error                          |
| ON IN .. GOSUB     | calls subroutine depending on input signal                         |
| ON MAPOS .. GOSUB  | call up a subprogram when the master position xxx (qc) is passed.  |
| ON MCPOS .. GOSUB  | call up a subprogram when the master position xxx (MU) is passed.  |
| ON PARAM .. GOSUB  | call up a subprogram when a parameter is altered                   |
| ON PERIOD .. GOSUB | calls subroutine at regular intervals                              |
| ON STATBIT         | call up a subprogram when bit n of the VLT status is set           |
| ON TIME            | calls subroutine after single timing                               |

Parameter (PAR)

All global and axis parameters with a parameter code can be set and read with the following commands. (Parameter code see overview in Parameter Reference.)

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| GET        | reads SyncPos parameter values (axis, global and user parameters) |
| GETVLT     | reads VLT parameter values                                        |
| GETVLTSUB  | reads VLT parameter values with index number                      |
| SET        | sets SyncPos parameter values (axis, global and user parameter)   |
| SETVLT     | sets VLT parameter values                                         |
| SETVLTSUB  | sets VLT parameter values with index number                       |
| LINKAXPAR  | links axis parameter with LCP display                             |
| LINKGPAR   | links global parameter with LCP display                           |
| LINKSYSVAR | links system variable with LCP displayCommunication option        |
| COMOPTSEND | writes in the Communication option buffer                         |
| COMOPTGET  | reads a Communication option telegram                             |
| PCD        | pseudo array for direct access to the fieldbus data area          |

Relative Motion (REL)

Commands for the relative positioning of the axis

|      |                                             |
|------|---------------------------------------------|
| ACC  | sets acceleration                           |
| DEC  | sets deceleration                           |
| POSR | positioning relative to the actual position |
| VEL  | sets velocity                               |

Synchronization (SYN)

Commands to synchronize the slave with the master or the master simulation

|                |                                                                   |
|----------------|-------------------------------------------------------------------|
| DEF SYNCORIGIN | Defines master-slave relation for the next SYNCP or SYNCM command |
| MOVESYNCORIGIN | relative shifting of the origin of synchronization                |
| SYNCV          | synchronization of velocity                                       |
| SYNCP          | synchronization of angle/position                                 |
| SYNCM          | synchronization of angle/position with marker correction          |
| SYNCSTAT       | queries flag for synchronization status                           |
| SYNCSTATCLR    | resetting of the flags MERR and MHIT                              |
| PULSACC        | sets acceleration for the virtual master                          |
| PULSVEL        | sets velocity for the virtual master                              |

CAM-Mode (CAM)

Commands for the synchronization in CAM-Mode (cam control)

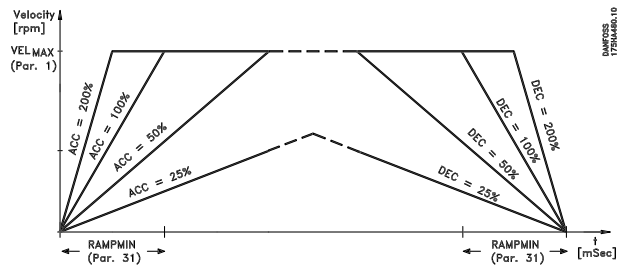
|               |                                                                                             |
|---------------|---------------------------------------------------------------------------------------------|
| CURVEPOS      | Retrieve slave curve position that corresponds to the current master position of the curve. |
| DEFMCPOS      | Define initial position of the master                                                       |
| POSA CURVEPOS | Move slave to the curve position corresponding to the master position                       |
| SETCURVE      | Set CAM curve                                                                               |
| SYNCC         | Synchronization in CAM-Mode                                                                 |
| SYNCCMM       | Synchronization in CAM-Mode with master marker correction                                   |
| SYNCCMS       | Synchronization in CAM-Mode with slave marker correction                                    |
| SYNCCSTART    | Start slave for synchronization in CAM-Mode                                                 |
| SYNCCSTOP     | Stop slave after the CAM synchronization                                                    |

## ■ All Commands from ACC to #INCLUDE

In the following section all commands are listed in alphabetical order and described in detail with syntax examples.

## ■ ACC

The ACC command defines the acceleration for the next motion command (speed control, synchronizing or positioning). The value will remain valid until a new acceleration value is set, using the ACC command. The value is related to the parameters **Shortest ramp** RAMPMIN (31) and **Maximum velocity** VELMAX (1) as well as **Velocity resolution** VELRES (22).



Program sample

ACC\_01.M

### NB!

If you work with the SyncPos option card then you should always set the ramps via the option card and not in the VLT. The VLT ramps must always be set to minimum.

### Summary

setting acceleration for motion commands

### Syntax

ACC a

### Parameter

a = acceleration

### NB!

If acceleration has not been defined previous to a motion command, then the acceleration will be the value of parameter DFLTACC(34).

### Command group

REL, ABS

### Cross Index

DEC, VEL, POSA, POSR

Parameter: RAMPMIN (31), VELMAX (1), VELRES (22)

### Syntax-Example

ACC 10 /\* Acceleration 10 \*/

### Example

minimum acceleration time: 1000 msec  
 maximum velocity: 1500 Rpm (25 Rev./s)  
 velocity resolution: 100

**■ APOS**

The APOS command can query the absolute position of the axis related to the actual zero point.

---

**Summary**

reads actual position

---

**Syntax**

res = APOS

---

**Return value**

res = absolute actual position in user units (UU) related to the actual zero point

All path information in motion commands are made in user units and are converted to quad-counts internally. (See also *Numerator and Denominator, Parameter 23 and 26.*)

The user unit (UU) corresponds in standard setting to the number of Quad counts.

Parameter POSFACT\_Z (23) / POSFACT\_N (26) = 1


**NB!**

If a temporary zero point which has been set via SET ORIGIN exists, then the position value is relative to this zero point.

---

**Command group**

I/O

---

**Cross Index**

CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR

Parameter: POSFACT\_Z (23) and POSFACT\_N (26)

---

**Syntax-Example**

PRINT APOS

/\* display the actual position of axis on the PC \*/

---

**Program sample**

APOS\_01.M,

GOSUB\_01.M, MOTOR\_01.M

**■ AVEL**

This function returns the actual velocity of the axis in User Units per second. The accuracy of the values depends on the duration of the measurement (averaging). The standard setting is 20 ms, but this can be changed by the user with the \_GETVEL command. It is sufficient to call up the command once in order to work with another measuring period from then on.

Thus, the command:

```
var = _GETVEL 100
```

sets the duration of the measurement to 100 ms, so that you have a considerably better resolution of the speed with AVEL and MAVEL, however, in contrast, quick changes are reported with a delay of a maximum of 100 ms.

---

**Summary**

queries actual velocity of axis

---

**Syntax**

res = AVEL

---

**Return value**

res = actual velocity of axis in UU/sec, the value is signed

---

**Command group**

I/O

---

**Cross Index**

MAVEL, APOS, \_GETVEL

---

**Syntax-Example**

PRINT AVEL

/\* queries actual velocity of axis and display on the PC \*/

**■ AXEND**

The AXEND command gives the actual status of the axis or the status of the program execution. This means for example that you can enquire when the “position is reached” and a positioning command (POSA, POSR) has actually been completed. When Bit 1 is set at “0” the positioning process is complete and the position reached. If, however, the positioning command has been interrupted with MOTOR STOP and continued later with CONTINUE, then the following bits would be set at “1”:

- the Bit 0 for “motor is at a standstill”
- the Bit 1 for “positioning process active”
- the Bit 3 for “motor is at STOP status”
- the Bit 6 for “axis controller switched off”

The AXEND command is especially suitable for determining whether or not a movement in the NOWAIT ON condition is terminated. With the help of the bit operators, the desired information from the axis status can be filtered out.

---

**Summary**

reads info on status of program execution

---

**Syntax**

res = AXEND

---

**Return value**

res = axis status with the following meaning:

| <u>Value</u> | <u>Bit</u> |                                                                                                                 |
|--------------|------------|-----------------------------------------------------------------------------------------------------------------|
| 128          | 7          | 1 = Motor is reset, i.e. it is ready to start and is controlling again, e.g. after ERRCLR, MOTOR STOP, MOTOR ON |
| 64           | 6          | 1 = axis controller is OFF, motor is off                                                                        |
|              | 4–5        | not in use                                                                                                      |
| 8            | 3          |                                                                                                                 |
| 4            | Bit 2      | 1 = speed mode is active                                                                                        |
| 2            | Bit 1      | 1 = positioning procedure is active                                                                             |
| 1            | Bit 0      | 1 = target position reached; motor is not in motion                                                             |

---

**Command group**

I/O

---

**Cross Index**

WAITAX, STAT

---

**Syntax-Example**

```

NOWAIT ON //do not wait until position is reached
POSA 100000
WHILE (AXEND&2) DO
// as long as the positioning process is active,
// repeat loop
 IF IN1 THEN // if input 01 is set
 VEL 100 // increase velocity
 POSA 100000
 WAIT IN1 OFF // Wait until input (key) is off
 ENDIF
ENDWHILE // position reached

```

---

**Syntax-Example**

```

IF (AXEND&64) THEN
/* set output 01, when axis controller is switched
off */
 OUT 1 1
ELSE
 OUT 1 0
ENDIF

```

---

**Program sample**

AXEND\_01.M

**■ COMOPTGET**

COMOPTGET reads from the Communication option buffer the *no* words and writes them in the array *array*, starting with the first element.

Summary

reads a Communication option telegram

Syntax

COMOPTGET *no* *array*

Parameter

*array* = the name of an array which must be at least the size of *no*

*no* = number of words to be read

Portability

built-in Communication option card

Communication option function

Parameters: Read and write parameters is not affected by the SyncPos option.

Control data:

The function of Control word (CTW) and Main Reference (MRV) depends on the setting of parameter 700; Status words (STW) and Main actual value (MAV) is always active:

|         | Parameter 700<br>"Enable SyncPos" | Parameter 700<br>"Disable SyncPos" |
|---------|-----------------------------------|------------------------------------|
| CTW/MRV | Disabled                          | Active                             |
| STW/MAV | Active                            | Active                             |

Process data:

PCD's 1 – 4 of PPO type 2/ 4 and PCD's 1 – 8 of PPO type 5 are not assigned a parameter number by parameter 915 and 916 but are used as a free data area which can be used in a SyncPos program.

The command COMOPTGET is copying the data received on the communication option into an array, where each array element contains one data word (16 bit).

The command COMOPTSEND is copying data from an array, where each array element contains one data work (16 bit) into the send buffer on the communication option, from where it is send via the network to the master.

Command group

Communication option

Cross Index

COMOPTSEND

Program sample

COM\_OPT

**■ COMOPTSEND**

COMOPTSEND writes in the Communication option buffer. In doing so the first *no* values are sent from the *array*.

Summary

writes in the Communication option buffer

Syntax

COMOPTSEND *no* *array*

Parameter

*array* = the name of an array which must be at least the size of *no*

*no* = number of words to be sent

Portability

built-in Communication option card

Communication option function

see command COMOPTGET

Command group

Communication option

Cross Index

COMOPTGET

Program sample

COM\_OPT

### ■ CONTINUE

By using CONTINUE, positioning and speed motion commands which have been aborted via the MOTOR STOP command or an error condition or halted via MOTOR OFF can be resumed. The CONTINUE command can be used especially in an error subroutine in connection with the ERRCLR command, to enable the correct continuation of a motion procedure following an error abort.



#### NB!

However CONTINUE does not continue interrupted synchronization commands.

#### Summary

continues positioning from point of interrupted motion

#### Syntax

CONTINUE

#### Command group

CON

#### Cross Index

MOTOR STOP, ERRCLR, ON ERROR GOSUB

#### Syntax-Example

CONTINUE

/\* continue interrupted motion procedure \*/

#### Program sample

MSTOP\_01.M

### ■ CPOS

The CPOS command queries the actual commanded position of the axis related to the actual zero point. The commanded position is understood to be the temporary set position which is re-calculated every ms by the positioning control during a positioning procedure or a movement in rotation mode.

The command position can be queried independently of the operating condition (position control during standstill, positioning process, speed control or synchronization).

#### Summary

reads the actual command position of the axis

#### Syntax

res = CPOS

#### Return value

res = Absolute commanded position in User Units (UU) related to the actual zero point



#### NB!

If a set and active temporary zero point (set via SET ORIGIN) exists, then the position value is relative to this zero point.

#### Command group

I/O

#### Cross Index

APOS, DEF ORIGIN, SET ORIGIN, POSA, POSR  
Parameter: POSFACT\_Z (23), POSFACT\_N (26)

#### Syntax-Example

PRINT CPOS

/\* actual command position of axis \*/

#### Program sample

CPOS\_01.M

GOSUB\_01.M

### ■ CSTART

The CSTART command is starting the drive in speed control mode. Acceleration/deceleration, as well as the speed should be set via the ACC, DEC and CVEL commands prior to starting. CSTART does not contains the command MOTOR ON which turns on the motor control. When using CSTART an explicit calling up of MOTOR ON is necessary after previous use of MOTOR OFF.

#### Summary

starts the speed mode

#### Syntax

CSTART



#### NB!

If no speed value has been defined via CVEL before the beginning of CSTART, then the default velocity 0 is used – this means that the motor will not rotate, but the PID controller is active.

All CVEL commands following the start of speed mode will be carried out immediately, i.e. a corresponding speed change will take place immediately, with the defined acceleration or deceleration (ACC/DEC).

#### Command group

ROT

#### Cross Index

ACC, DEC, CVEL, CSTOP

#### Syntax-Example

CSTART /\* rpm mode start \*/

#### Program sample

CMODE\_01.M

### ■ CSTOP

Via the CSTOP command, the speed mode is terminated and the positioning mode is started, whereby a still rotating axis is stopped with the deceleration defined with DEC and the motor is held in the stop position.

#### Summary

stops the drive in speed mode

#### Syntax

CSTOP



#### NB!

A CSTOP command carried out in the positioning mode can also cause an abrupt termination of the positioning procedure.

#### Command group

ROT

#### Cross Index

ACC, DEC, CVEL, CSTART

#### Syntax-Example

CSTOP /\* rpm mode stop \*/

#### Program sample

CMODE\_01.M

### ■ CURVEPOS

CURVEPOS returns the slave value which corresponds to the actual curve master position. The position can be retrieved independently of the operating status (position control at standstill, positioning procedure, velocity control or synchronization).

#### Summary

Retrieve slave curve position that corresponds to the current master position of the curve.

#### Syntax

res = CURVEPOS

#### Return value

res = Slave position in CAM units (UU) absolute to the current zero point.



#### NB!

The position is only defined if a SETCURVE has been set before.



#### NB!

If a temporary zero point exists which has been set with SET ORIGIN and is active, the position value will refer to this zero point.



#### NB!

DEFMCPOS and DEFMORIGIN can still modify this position.

#### Command group

CAM

#### Cross Index

APOS, DEF ORIGIN, SET ORIGIN, POSA, POSR, DEFMCPOS

Parameter: SYNCFACTM, SYNCTACTS

#### Syntax example

```
PRINT CURVEPOS
// print actual slave position of the curve
```

### ■ CVEL

The velocity for the next speed controlled motor movement is set with the CVEL command. The value remains valid until a further CVEL command sets a new velocity.

The velocity value to be given will be related to the parameters **Maximum velocity** VELMAX (1) and **Velocity resolution** VELRES (22).

#### Summary

sets velocity for speed controlled motor movements

#### Syntax

CVEL v

#### Parameter

v = velocity value (negative value for reversing)

$$\text{Command velocity} = v \cdot \frac{\text{VELMAX (1)}}{\text{VELRES (22)}}$$



#### NB!

CVEL commands which take place after CSTART will be carried out immediately i.e. the velocity will be adapted via the ACC/DEC set acceleration or deceleration to the new value of CVEL.

If a velocity has not been defined before the start of speed control mode (CSTART), then the default velocity is 0, i.e. the motor will not turn, and a velocity input via CVEL will start the movement in speed control mode.

#### Command group

ROT

#### Cross Index

ACC, DEC, CSTART, CSTOP

Parameter: VELMAX (1)

#### Syntax-Example

CVEL 100

#### Program sample

CMODE\_01.M

### ■ DEC

The DEC command defines the deceleration for the next motion command (speed control, synchronization or positioning). The value will remain valid until a new deceleration value is set with another DEC command. The value is related to the parameters **Shortest ramp** RAMPMIN (31) and **Maximum velocity** VELMAX (1) as well as **Velocity resolution** VELRES (22).



#### NB!

If you work with the SyncPos option card then you should always set the ramps via the option card and not in the VLT. The VLT ramps must always be set to minimum.

#### Summary

sets deceleration

#### Syntax

DEC a

#### Parameter

a = deceleration



#### NB!

If deceleration is not defined previous to the positioning command then deceleration will be the setting (negative) of parameter **Default acceleration** DFLTACC (34).

#### Command group

REL, ABS

#### Cross Index

ACC

Parameter: RAMPMIN (31), VELMAX (1), VELRES (22)

#### Syntax-Example

```
ACC 50 /* acceleration: 50, while braking 10 */
DEC 10
```

#### Example

```
minimum acceleration time: 1000 msec
maximum velocity: 1500 Rpm
velocity resolution: 100
```

### ■ DEFMCPOS

DEFMCPOS defines the initial position of the master (in MU) in the CAM-Mode and thus the point where the curve begins as soon as the master pulses are being counted.

#### Summary

Define initial position of the master

#### Syntax

DEFMCPOS p

#### Parameter

p = position in Master Units (MU)

#### Command group

CAM

#### Cross Index

DEFMORIGIN, SETMORIGIN, SYNCC

Parameter: SYNCMSTART

#### Syntax example

```
DEFMCPOS 1000
```

```
// Set internal MU counter to 1000
```

**■ DEF MORIGIN**

DEF MORIGIN defines the current master position as the zero point for the master. The master position (MAPOS) refers to this zero point until a redefinition takes place using DEF MORIGIN or SET MORIGIN.

---

**Summary**

Set the current master position as the zero point for the master.

---

**Syntax**

DEF MORIGIN

---

**Command group**

INI

---

**Cross Index**

MAPOS, SET MORIGIN

---

**Syntax example**

DEF MORIGIN

/\* Set current position as the zero point for the master \*/

**■ DEF ORIGIN**

With the DEF ORIGIN command the current position is set as the zero point. All absolute positioning commands (POSA) then refer to this zero point.

---

**Summary**

sets the actual position as real zero point

---

**Syntax**

DEF ORIGIN

---

**Command group**

INI

---

**Cross Index**

POSA

---

**Syntax-Example**

POSA 80000 /\* Absolute positioning \*/

DEF ORIGIN /\* define actual position as zero point \*/

---

**Program sample**

DORIG\_01.M, ORIG\_01.M

**■ DEF SYNCORIGIN**

This command allows to define the relation between master and slave for the next SYNCP or SYNCM command. It sets the internal slave command position (Mpcmd) to the slave value.

The master value is used for an internal MOVE-SYNCORIGIN. For that reason, a MOVESYNCORIGIN will be overwritten by this command. Both actions are done at the moment, when the SYNC command is activated. So it is guaranteed, that master and slave will be synchronized at the above master-slave position.

---

**Summary**

Defines master-slave relation for the next SYNCP or SYNCM command

---

**Syntax**

DEFSYNCORIGIN master slave

---

**Parameter**

master = reference position in qc

slave = reference position

---

**Portability**

Standard command with option card version 5.00 onwards

---

**Command group**

SYN

---

**Cross Index**

MOVESYNCORIGIN

### ■ DELAY

The DELAY command leads to a defined program delay. This parameter gives the delay time in milliseconds.

If an interrupt occurs during the delay time, then following the processing of the interrupt procedure, the programmed delay will take place after the correct inclusion of the interrupt time. Thus, the DELAY command gives a constant delay time, independent of whether various interrupts have to be processed during the programmed delay time.

If the interrupt requires more processing time than is available for the delay, then the interruption procedure will be carried out to the end, before the command following the DELAY instruction is commenced.

#### Summary

Time delay

#### Syntax

DELAY t

#### Parameter

t = time delay in milliseconds (maximum MLONG)

#### Command group

CON

#### Cross Index

WAITT, WAITI, WAITAX

#### Syntax-Example

DELAY 1000 /\* 1 second delay \*/

#### Program sample

DELAY\_01.M

### ■ DELETE ARRAYS

With DELETE ARRAYS you can delete all arrays in the RAM without – as was the case up to now – also deleting the parameters etc. This new command has the same effect as the menu command "CONTROL" → "RESET" → "ARRAYS".

#### Summary

Delete all arrays in the RAM

#### Syntax

DELETE ARRAYS



#### NB!

If you then execute a SAVE ARRAYS, the arrays in the EEPROM are also overwritten!



#### NB!

If DELETE ARRAYS is carried out after a DIM assignment in the program, it is then no longer possible to access the array elements.



#### NB!

If a program contains a DELETE ARRAYS command, there are no more arrays in the RAM after the program is exited.

#### Command group

INI

**■ DIM**

Via a DIM instruction at the commencement of the program, it is possible to declare one or more arrays (= Variable fields).

Arrays are valid for all programs. If arrays are not yet available in the SyncPos option memory, then the arrays are allocated via the DIM instructions. Arrays which are already available in the memory are checked to see if their size corresponds to the momentary DIM commands. If differences are found, then an error registration is made. If, additionally to the corresponding arrays, new arrays are declared, then these must also be added at the end of the DIM command.

Each array element can later be accessed, similar to a variable, calculation results, characters or other information can be stored.

An array element can be called up via the array name and an index. The indices are admissible from 1 to the defined size in the DIM allocation.

An essential difference between variables and array elements consists in the fact that arrays are stored in the non-volatile memory, and their contents are permanent even when the power supply is switched off – insofar as it is saved with SAVEPROM or SAVE ARRAYS.

In contrast to variables, arrays have a validity not only for one, but for all programs in the VLT flow. The only condition necessary is that the arrays must be accessible via a DIM command in the desired program which enables a data exchange between several programs. It is of no importance whether or not the array is identified with the same name in all the programs. What is important is the order of the array definitions. This means, for example, that the first defined array in all programs always refers to the first stored array in the memory, independent of the array name.

---

**Summary**

Definition of an array

---

**Syntax**

DIM array [n]

---

**Parameter**

array = name of the array

n = number of array elements


**NB!**

The DIM command must be the first instruction in a program, and must appear before the subroutines.

Indices from 1 to the defined size of the array are permissible.

A defined array size is valid for all programs, and cannot be altered. Only the order of the array definitions (not the names) determines which of the data-fields will be accessed.

Array definitions can only be canceled via erasure of the entire memory.

---

**Command group**

CON

---

**Syntax-Example**

```
DIM xpos[100], ypos[100]
/* define array XPOS and YPOS each with 100
elements */
```

---

**Program sample**

DIM\_01.M

### ■ DISABLE ... interrupts

DISABLE switches off all or explicitly specified interrupts – apart from ON ERROR. If the function DISABLE.. is used in the main program, it can prevent interrupts of the corresponding type.

This is particularly useful if a variable, which is set in an interrupt procedure, is used in in the main program. To do this you should first switch off the corresponding (or all) interrupts in the main program DISABLE.. alter the variable and then switch the corresponding (or all) interrupts back on with ENABLE..

#### Summary

Switches interrupts off

#### Syntax

DISABLE inttyp



#### NB!

DISABLE cannot be called up during interrupt procedures. (The system automatically switches back to enabled after an interrupt.)

#### Parameter

inttyp = ALL  
 INT  
 COMBIT  
 STATBIT  
 PARAM  
 PERIOD  
 TIME  
 POSINT



#### NB!

DISABLE ALL = all except ON ERROR



#### NB!

DISABLE ALL does indeed block the execution of all interrupts but not their detection. Therefore, all I/O interrupts or COMBIT interrupts are registered in the background, but are only executed after ENABLE ALL.



#### NB!

The detection of the interrupts is indeed switched off by means of DISABLE INT or DISABLE COMBIT and incoming interrupts are ignored. If afterwards a bit changes, it will not be registered and does not lead to an interrupt even after the ENABLE.

This applies to all edge-oriented interrupts such as COMBIT, INT and STATBIT.

This does not apply to message-oriented interrupts such as PARAM.

#### Command group

INT

#### Cross Index

ON INT, ON COMBIT, ON STATBIT, ON PARAM, ON PERIOD, ON TIME, ENABLE .. Interrupts

#### Syntax example

```
DISABLE ALL /* Switch off all interrupts */
DISABLE STATBIT
/* Switch off the interrupt for the status bit */
```

### ■ ENABLE ... interrupts

ENABLE switches all or explicitly specified interrupts on again.

#### Summary

Switches interrupts on

#### Syntax

ENABLE inttyp



#### NB!

ENABLE cannot be called up during interrupt procedures. (The system automatically switches back to enabled after an interrupt.)

#### Parameter

inttyp = ALL  
 INT  
 COMBIT  
 STATBIT  
 PARAM  
 PERIOD  
 POSINT  
 TIME

#### Command group

INT

#### Cross Index

ON INT, ON COMBIT, ON STATBIT, ON PARAM, ON PERIOD, ON TIME

#### Syntax example

```
ENABLE ALL /* Switch on all interrupts */
ENABLE COMBIT /* Switch on the interrupt for
the communication bit */
```

### ■ ERRCLR

An option card error can be cleared via the ERRCLR command. However, the cause of the error must be eliminated first, otherwise the same error alarm will occur again. If, in the meantime, another un-corrected error occurs, then only the first error will be canceled.

#### Summary

Error cancellation

#### Syntax

ERRCLR

The ERRCLR command should only be used in a subroutine for error handling (see ON ERROR GOSUB).



#### **NB!**

ERRCLR contains the command MOTOR ON, which automatically turns on the control again. (The motor is position controlled at the current position.)



#### **UP TO VERSION 3.00**

ERRCLR is only resetting option card errors, VLT alarms must be reset via a digital input or the LCP.

#### **NEW**

As of Version 3.1x ERRCLR also resets Bit 7 of the control word of the VLT. Then the VLT messages no longer need to be deleted.

#### Command group

INI, CON

#### Cross Index

ON ERROR GOSUB, ERRNO, CONTINUE, MOTOR ON  
Messages and Error reference

#### Syntax-Example

ERRCLR /\* erase actual error alarm \*/

#### Program sample

ERROR\_01.M, IF\_01.M, INDEX\_01.M

### ■ ERRNO

ERRNO is a system variable which is available in all the programs, and contains the momentary error code. All error codes are explained in the chapter Messages and Error Reference.

If, at the time of inquiry no error has occurred, then ERRNO will contain a 0.

#### Summary

System variable with the actual error code

#### Syntax

res = ERRNO

#### Command group

I/O

#### Cross Index

ON ERROR GOSUB, ERRCLR  
Messages and Error reference

#### Syntax-Example

/\* display actual error code \*/  
PRINT ERRNO

#### Program sample

ERROR\_01.M, IF\_01.M, INDEX\_01.M

### ■ EXIT

The EXIT command ends a program where active positioning procedures are being carried out to the end.

The EXIT command is especially intended for use in an error treatment routine, and permits an unplanned program termination in the case of an un-correctable error occurrence.

After an abort with EXIT, programs marked with "AUTOSTART" will start up again automatically if SET PRGPAR = -1.

#### Summary

Premature program termination

#### Syntax

EXIT



#### NB!

A program should only be terminated in the case of a serious error, e.g. when reacting to a limit switch.

#### Cross Index

ON ERROR GOSUB

SET PRGPAR

"AUTOSTART"

#### Syntax-Example

EXIT /\* Program termination \*/

#### Program sample

EXIT\_01.M, ERROR\_01.M, GSVEL\_01.M

### ■ GET

Reads the value of an axis parameter, a global parameter or an user parameter of the SyncPos option.

Global and axis parameter are addressed with a code, for example KPROP (11) for the **Proportional factor**, KDER (12) for the **Derivative Factor** or POSERR (15) for the **Tolerated position error**. A complete list of the codes can be found in the Parameter reference.

User parameter are addressed with a number, 130 – 229. See also the [Parameter reference](#) for details.

#### Summary

Reads an axis parameter, a global parameter or an user parameter

#### Syntax

res = GET par

#### Parameter

par = parameter identification

#### Return value

res = parameter value

#### Command group

PAR

#### Cross Index

SET, GETVLT, SETVLT, LINKGPAR, LINKAXPAR

[Parameter reference](#)

#### Syntax-Example

PRINT GET POSLIMIT

/\* Print-out positive positioning limit \*/

posdiff = GET POSERR

/\* Read actual setting tolerated position error \*/

#### Syntax-Example

PRINT GET I\_BREAK /\* reads input for abort \*/

#### Program sample

GETP\_01.M

### ■ GETVLT

GETVLT reads VLT parameters and return the corresponding value. Thus, with GETVLT you have access to the operating data (e.g. motor current P520) or to the configurations (e.g. max. command value P205) of the VLT.

Since only integer values are transmitted, it is necessary to take the conversion index into consideration when evaluating the return value.

Thus an LCP value of 50.0 Hz (conversion index = -1) is equivalent to a return value of 500.

The list of VLT parameters with their respective conversion index can be found in the VLT5000 and VLT5000 Flux operating instructions.

#### Summary

Reads a VLT parameter

#### Syntax

res = GETVLT par



#### NB!

Use GETVLTSUB to read parameters with index numbers, e.g. VLT parameter 606...617.

#### Parameter

par = parameter number

#### Return value

res = parameter value

#### Command group

PAR

#### Cross Index

SETVLT

#### Syntax-Example

```
PRINT GETVLT 202
```

```
/* reads parameter 202 output frequency high */
```

### ■ GETVLTSUB

GETVLTSUB reads VLT parameters with index numbers, e.g. VLT parameter 606...617, and return the corresponding value.

Since only integer values are transmitted, it is necessary to take the conversion index into consideration when evaluating the return value.

Thus an LCP value of 50.0 Hz (conversion index = -1) is equivalent to a return value of 500.

The list of VLT parameters with their respective conversion index can be found in the VLT5000 and VLT5000 Flux operating instructions.

#### Summary

Reads a VLT parameter with index number

#### Syntax

res = GETVLTSUB par indxno

#### Parameter

par = parameter number

indxno = index number

#### Return value

res = parameter value

#### Portability

With option card version 5.00 onwards.

#### Command group

PAR

#### Cross Index

SETVLTSUB

#### Syntax example

```
PRINT GETVLTSUB 25 1
```

```
// reads index 1 of the Quick menu parameter
```

```
// (VLT5000Flux only)
```

### ■ GOSUB

The GOSUB command will call up a subroutine, and the accompanying program will be carried out. The main program will be continued following the completion of the last subroutine command (RETURN).

#### Summary

calls a subroutine

#### Syntax

GOSUB name

#### Parameter

name = subroutine name



#### NB!

The subroutine must be defined at the beginning or end of a program within the SUBMAINPROG area.

#### Command group

CON

#### Cross Index

SUBMAINPROG...ENDPROG, SUBPROG...RETURN  
ON ERROR GOSUB, ON INT n GOSUB

#### Syntax-Example

```
GOSUB testup /* Call-up the subroutine testup */
Command line 1
Command line n
SUBMAINPROG
/* Subroutine testup must be defined */
SUBPROG testup
Command line 1
Command line n
RETURN
ENDPROG
```

#### Program sample

GOSUB\_01.M  
AXEND\_01.M, INCL\_01.M, STAT\_01.M

### ■ GOTO

The GOTO command enables an unconditional jump to the indicated program position and the program processing at this position will be carried out.

The jumped-to position is identified with a label. A label can consist of one or more characters and may not be identical to a variable name or a command word. A label must also be unique, i.e. it may not be used for different program positions. It is therefore possible to program a continuous loop via the GOTO command.

#### Summary

Jump to a program label

#### Syntax

GOTO label

#### Parameter

label = identification of program target position



#### NB!

The label for the program target position must be followed by a colon (:).

#### Command group

CON

#### Cross Index

LOOP

#### Syntax-Example

```
endless: /* Label to be jumped to */
Command line 1
Command line n
GOTO endless
/* jump command to label endless */
```

#### Program sample

GOTO\_01.M  
EXIT\_01.M, IF\_01.M

### ■ HOME

The HOME command is moving the drive to the machine reference switch, which must be placed at the machine zero or reference position. Velocity and acceleration/deceleration for HOME positioning is defined in the parameters HOME\_VEL (7) and HOME\_RAMP (41).

To achieve accurate positioning HOME\_VEL (7) should not be higher than 10 % of maximum velocity.

The sign of HOME\_VEL (7) determines the direction in which the reference switch is searched.

When the HOME position is reached, this position will be defined as 0.

The reference switch can be approached in 4 different ways defined in parameter HOME\_TYPE(40):

- 0 Moves to reference switch, moves in opposite direction leaving the references switch and stops at the next index pulse (encoder zero pulse or external marker signal).
- 1 Like 0 but without searching for the index pulse.
- 2 Like 0 but leaving the switch without reversing the direction.
- 3 Like 2 but without searching for the index pulse.

If HOME is aborted via an Interrupt, HOME will not be continued automatically at the end of the interrupt routine function. Instead the program continues with the next command. This makes it possible for HOME to also be aborted after an error.

#### Summary

Move to device zero point (reference switch) and set as the real zero point.

#### Syntax

HOME

#### Peculiarities

The system must be fitted with a reference switch, when possible with an encoder with an index pulse.



#### NB!

The HOME command will also be carried out to the end in the NOWAIT ON mode, before other program processing will be begun.

Please note, that ON PERIOD xx GOSUB xx must be disabled during homing.

E.g. ON PERIOD n GOSUB x and the resetting after homing is completed.

#### Command group

INI

#### Cross Index

INDEX

Parameter: HOME\_VEL (7), HOME\_RAMP (41), HOME\_TYPE (40), HOME\_FORCE (3)

#### Syntax-Example

HOME /\* move to reference switch and index \*/

#### Program sample

HOME\_01.M

**■ IF ..THEN .., ELSEIF .. THEN .. ELSE .. ENDIF**

Conditional program branching can be realized with the IF..ENDIF construction.

When the conditions following IF or ELSEIF are fulfilled, then the commands leading to the next ELSEIF, ELSE or ENDIF are carried out – and the program will be continued after the ENDIF instruction.

When the conditions are not fulfilled, then the following ELSEIF branching will be checked and, in as much as the conditions are fulfilled, the corresponding program part will be carried out, and the program continued after ENDIF.

The branching conditions that are checked after IF or ELSEIF can be made up of one or more comparison operations.

Any number of ELSEIF branching can occur within an IF..ENDIF construction – however, only one ELSE instruction should be available. Following the ELSE instruction is a program part that must be carried out, in as much as none of the conditions are fulfilled.

The ELSEIF and ELSE instructions can, but do not have to be, contained within an IF ENDIF construction.

---

**Summary**


---

Conditional single or multiple program branching. (When the conditions are fulfilled, then ..., else ...)

---

**Syntax**


---

IF condition THEN command  
 ELSEIF condition THEN command  
 ELSE command  
 ENDIF

---

**Parameter**


---

condition = Branching criteria  
 command = one or mor program commands


**NB!**

After a condition has been fulfilled, the appropriate program part will be carried out and the program following the ENDIF instruction continued. Further conditions will no longer be checked.

---

**Command group**


---

CON

---

**Cross Index**


---

REPEAT .. UNTIL, WHILE ..ENDWHILE

---

**Syntax-Example**


---

*/\* simple branch \*/*

```
IF (a == 1) THEN /* Variable a = 1, then */
command line 1
command line n
ENDIF
```

*/\* multiple branch \*/*

```
IF (a == 1 AND b != 1) THEN
command lines
ELSEIF (a == 2 AND b != 1) THEN
command lines
ELSEIF (a == 3) THEN
command lines
ELSE
command lines
ENDIF
```

---

**Program sample**


---

```
IF_01.M; ERROR_01.M, EXIT_01.M, HOME_01.M,
IN_01.M, ...
```

### ■ IN

The status of a digital input can be read with the IN command. Depending on the signal level, a 0 or 1 will be given.

#### Summary

reads status of digital input

#### Syntax

res = IN n

#### Parameter

n = input number

SyncPos option: 1 ... 8

VLT control card: 16, 17, 18, 19, 27, 29, 32, 33

#### Return value

res = input status

0 = Low-level or undefined

1 = High-level



#### NB!

The definition of a high or low level, as well as the input circuit, can be taken from the chapter Input/output terminals, as well as the VLT5000 manual.

#### Command group

I/O

#### Cross Index

INB, OUT, OUTB

#### Syntax-Example

in4 = IN 4

/\* store condition input 4 in variable in4 \*/

IF (IN 2) THEN

/\* If high level on terminal 2, set output 01 \*/

OUT 1 1

ELSE

OUT 1 0

ENDIF

#### Program sample

IN\_01.M

### ■ INAD

The INAD command reads the value of the analog inputs.

#### Summary

reads analog input

#### Syntax

res = INAD n

#### Parameter

n = number of the analog input: 53, 54 and 60

#### Return value

res = analog value



#### NB!

The resolution of the analog inputs was increased with the introduction of software version SyncPos VLT 3.12.

For these versions the following return values are true:

terminal 53/54    -1000 ... 1000    = -10 V ... 10 V

terminal 60:            0 ... 2000    = 0 ... 20 mA

For the software version SyncPos VLT2.03 following return value are valid:

terminal 53/54    0 ... 10 V        res = 0 ... 100

terminal 60:            0 ... 20 mA      res = 0 ... 200

You will find the number of the software version in parameter 624 on the VLT display.

#### Command group

I/O

#### Syntax-Example

an1 = INAD 53

PRINT "Analog input 53 " ,an1

**■ INB**

The condition of the digital inputs can be read as byte via the INB command.

The values reflect the condition of the individual inputs.

---

**Summary**

reads one byte from digital inputs

---

**Syntax**

res = INB n

---

**Parameter**

n = input byte  
 0 = input 1 (LSB) ... 8 (MSB)  
 1 = input 33 (LSB) ... 16 (MSB)

---

**Return value**

res = value of the input byte (0 ... 255)

The least significant bit corresponds to the condition of input 1/33.


**NB!**

The definition of the high and low level, as well as the input circuit, can be taken from the chapter Input/output terminal or the VLT5000 manual.

---

**Command group**

I/O

---

**Cross Index**

IN, OUT, OUTB

---

**Syntax-Example**

in = INB 0  
 /\* store the condition of the first 8 inputs \*/

---

**Example**

IN1 = low, IN2 = high, IN3 = high,  
 all other inputs are low  
 res = 2<sup>1</sup> + 2<sup>2</sup> = 6

---

**Program sample**

INB\_01.M, INB\_02.M,  
 OUTB\_01.M

**■ INDEX**

Movement to the index position of the encoder will be started via the INDEX command. The Index search takes places from the *Home velocity* defined in parameter HOME\_VEL (7). The HOME\_VEL sign determines the rotational direction in which the Index signal will be searched.

---

**Summary**

Move to index position of the encoder

---

**Syntax**

INDEX

---

**Peculiarities**

The utilized encoder must have an index channel.


**NB!**

Only encoders with a low active index pulse can be used.

If an index pulse is not found within a complete revolution, then an alarm signal occurs.

The INDEX command will also be carried out to the end in NOWAIT ON, before further program processing can be begun.

---

**Command group**

INI

---

**Cross Index**

HOME, POSA, DEF ORIGIN

---

**Syntax-Example**

INDEX /\* move to index \*/

---

**Program sample**

INDEX\_01.M

**■ INKEY**

With the INKEY command it is possible to read a key signal from the VLT keyboard. The parameter entered with INKEY determines whether the program waits unconditionally for a key signal, for a certain period of time or not at all.

One key signal is read in per successful INKEY command respectively. To input a string of characters it is necessary to repeat the INKEY command (p<>0) in a loop until no further key signals exist.

---

**Summary**

Read in a key signal

---

**Syntax**

INKEY (p)

---

**Parameter**

p is the maximum waiting time, defined ...

- p = 0 wait for key code
- p > 0 wait of max. p milli-seconds
- p < 0 no wait for key code

---

**Return value**

key code for the received character  
 resp -1 in case no character available

Following key codes are sent back, as long as the key is pressed. If more than one key were pressed simultaneously the corresponding sum of the values will be sent back:

| <u>key:</u>      | <u>value:</u> |
|------------------|---------------|
| MENU             | 1             |
| QUICK MENU       | 2             |
| DISPLAY / STATUS | 8             |
| OK               | 16            |
| CANCEL           | 32            |
| CHANGE DATA      | 128           |
| →-key            | 256           |
| ↑-key            | 512           |
| ↓-key            | 1024          |
| ←-key            | 2048          |
| START            | 4096          |
| FWD. REV.        | 8192          |
| JOG              | 16384         |
| STOP / RESET     | 32768         |

Combinations send the corresponding values:

|                 |      |
|-----------------|------|
| OK and CANCEL   | 48   |
| START and ↑-key | 4608 |


**NB!**

The keys keep their VLT-functions, unless they are disabled in Parameter 014–017.

A negative parameter must be given in brackets.

---

**Command group**

I/O

---

**Cross Index**

PRINT

Chapter Fundamentals of the SyncPos program

---

**Syntax-Example**

```
input = INKEY 0
/* wait until key signal is read */
input = INKEY 5000
/* wait max. 5 seconds to input */
input = INKEY (-1)
/* do not wait for input */
```

---

**Program sample**

```
INKEY_01.M
EXIT_01.M, WHILE_01.M
```

### ■ IPOS

The command IPOS returns the last index or marker position of the slave absolute to the current zero point.

#### Summary

queries last index or marker position of the slave

#### Syntax

res = IPOS

#### Return value

res = last slave position (index or marker) absolute to actual zero point.

The position input is made in user units (UU) and corresponds in the standard setting (parameter POSFACT\_Z (23) and POSFACT\_N (26) = 1) to the number of quad-counts.



#### NB!

If a temporary zero point, set and activated via SET ORIGIN, exists, then the position is respective to this zero point.

#### Command group

I/O

#### Cross Index

CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR, MIPOS  
Parameter: POSFACT\_Z (23), POSFACT\_N (26)

#### Syntax-Example

```
PRINT IPOS
/* queries last index position and display on PC */
```

### ■ LINKAXPAR

With LINKAXPAR it is possible to link an axis parameter with the LCP display. Subsequently it is possible to change this parameter via the LCP or read out the set value.

When a linked parameter is changed with a SET command, the new value is also automatically transferred to the LCP, but is not changed in the factory settings since the SET command only has a temporary effect.

If the user changes a linked parameter on the LCP, the new value is accepted and executed. After turning the VLT off and then on again the factory settings or the last user parameter to be stored is activated.

Only after the user has confirmed the parameter with OK is the new value accepted as the user parameter and saved permanently. (Corresponds to the OK function in the menu "CONTROLLER" → "PARAMETERS" → "AXIS".)

#### Summary

links axis parameter with LCP display

#### Syntax

LINKAXPAR param optpar "text" min max typ

#### Parameter

param = axis parameter name  
optpar = LCP parameter number (710–779 and 795–799)  
text = descriptive text for display  
min = minimum value for this parameter  
max = maximum value for this parameter  
typ = type of parameter  
0 = offline, i.e. changes are only active after they have been confirmed with OK.  
1 = online, that means changes via the LCP display are active at once

#### Command group

PAR

#### Cross Index

LINKGPAR, SET, GET  
User parameter, Parameter reference

#### Syntax-Example

```
LINKAXPAR POSERR 712 "position error" 300
5000 0
```

### ■ LINKGPAR

With LINKGPAR it is possible to link a pre-defined global parameter or a free internal user parameter with the LCP display. Subsequently it is possible to change this parameter via the LCP or read out the set value.

When a linked parameter is changed with a SET command, the new value is also automatically transferred to the LCP, but is not changed in the factory settings since the SET command only has a temporary effect.

If the user changes a linked parameter on the LCP, the new value is executed. Only after the user has confirmed this value with OK is the new value saved permanently as a user parameter in the EEPROM.

**NEW:** The command LINKGPAR tests whether the value of the user parameter is within the specified range. If not, the corresponding limit is used and this value saved. This ensures that a display appears.

#### Summary

links global parameter with LCP display

#### Syntax

LINKGPAR param optpar "text" min max typ

#### Parameter

param = global parameter name or user parameter number (130–229)

optpar = LCP parameter number (710–779 and 795–799)

text = descriptive text for display

min = minimum value for this parameter

max = maximum value for this parameter

typ = type of parameter

0 = offline, i.e. changes are only active after they have been confirmed with OK.

1 = online, that means changes via the LCP display are active at once

#### Command group

PAR

#### Cross Index

LINKXPAR, SET, GET

User parameter, Parameter Reference

#### Syntax-Example

```
LINKGPAR I_ERRCLR 710 " " 0 33 0
```

```
/* Predefined global parameter */
```

```
LINKGPAR 132 711 "name" 0 100000 0
```

```
/* Free internal user parameter */
```

```
LINKGPAR PRGPAR 701 "auto-program" -1 10 0
```

```
/* Define autostart */
```

### ■ LINKSYSVAR

The command LINKSYSVAR links the system variable SYSVAR[indx] with the VLT Parameter vltno (795–799) and the display text text. This means that you can link internal values on the display without using LINKGPAR.

If, for example, you are compiling with #DEBUG NOSTOP, link the internal line number with the VLT parameter 795. Then you can selectively observe the program execution.

#### Summary

Link system variable with LCP display

#### Syntax

LINKSYSVAR indx vltno "text"

#### Parameter

indx = Index of the system variable SYSVAR

vltno = LCP-Parameter number (795...799)

text = descriptive text for display



#### NB!

The parameter vltnr is updated every 40 ms. Therefore, if five parameters are linked in this way, it takes at least 200 ms until the same parameter is updated.

#### Command group

PAR

#### Cross Index

LINKGPAR, SYSVAR

User parameter, Parameter Reference

#### Syntax-Example

```
LINKSYSVAR 33 795 "internal line number"
```

```
LINKSYSVAR 30 796 "Motor voltage"
```

```
// Determine with user parameter 11 and 12:
```

```
// line number in the first display line and
```

```
// motor voltage in the second display line
```

### ■ LOOP

A single or multiple repetition of a certain program part can be realized by using the LOOP command. The number of loop repetitions can be given as either an absolute value or in the form of a variable.

The program position to be jumped to is identified via a label. A label can be made up of one or more characters, and must not be identical with a variable name or a command word. A label must also be unique, i.e. the same label may not be used more than once for different program positions.

#### Summary

Defined loop repetition

#### Syntax

LOOP n label

#### Parameter

n = number of loop repetitions  
label = identification of target program position



#### NB!

The label on the target program position must be followed by a colon(:).

Because the internal loop counter monitors only at the end of the loop and then decreases by one, the commands within the loop will be carried out with one more sequence than keyed in (keyed in loop repetitions 10 = 11 real repetitions).

#### Command group

CON

#### Cross Index

GOTO, WHILE...ENDWHILE, REPEAT...UNTIL

#### Syntax-Example

```
next_in: /* jump to label */
command line 1
command line n
LOOP 9 next_in
/* repeat loop contents 10 times */
```

#### Program sample

```
LOOP_01.M
APOS_01.M, IN_01.M, MOTOR_01.M, NOWAI_01.M
```

### ■ MAPOS

With the MAPOS command it is possible to query the actual master position (absolute to the actual zero position).

#### Summary

Queries current actual position of the master

#### Syntax

res = MAPOS

#### Return value

res = master position to absolute actual zero point in qc

#### Command group

I/O

#### Cross Index

CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR  
Parameter: POSFACT\_N (26), POSFACT\_Z (23)

#### Syntax-Example

```
PRINT MAPOS
/* queries actual master position and print to PC */
```

**■ MAVEL**

This function returns the actual velocity of the master drive in qc/sec, with qc referring to the master encoder.

The accuracy of the values depends on the duration of the measurement (averaging). The standard setting is 20 ms, but this can be changed by the user with the `_GETVEL` command. It is sufficient to call up the command once in order to work with another measuring period from then on. Thus, the command:

```
var = _GETVEL 100
```

sets the duration of the measurement to 100 ms, so that you have a considerably better resolution of the speed with AVEL / MAVEL, however, in contrast, quick changes are reported with a delay of a maximum of 100 ms.

---

**Summary**

queries actual velocity of the master

---

**Syntax**

```
res = MAVEL
```

---

**Return value**

res = actual velocity of the master in qc/sec, the value is signed

---

**Command group**

AVEL

---

**Cross Index**

PRINT MAVEL

/\* queries actual velocity of the master and print to PC \*/

**■ MIPOS**

The command MIPOS returns the last index or marker position of the master absolute to the current zero point.

---

**Summary**

Query last index or marker position of the master

---

**Syntax**

```
res = MIPOS
```

---

**Return value**

res = last index or marker position of the master absolute to actual zero point in qc

---

**Command group**

I/O

---

**Cross Index**

CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR

Parameter: POSFACT\_N (26), POSFACT\_Z (23)

---

**Syntax-Example**

```
PRINT MIPOS
```

/\* print to the PC the last index position of the master \*/

**■ MOTOR OFF**

The motor control can be disabled by using the MOTOR OFF command. After MOTOR OFF, the drive axis can be moved freely, as long as there is no motor brake. A monitoring of the actual position will continue to take place, i.e. the actual position (APOS) can still be queried after MOTOR OFF.


**NB!**

For a restart of a motion process after MOTOR OFF the command MOTOR ON must be used. Only the command ERRCLR automatically activates MOTOR ON.

Summary

turns off motor control

Syntax

MOTOR OFF

Command group

INI

Cross Index

MOTOR ON

Syntax-Example

MOTOR OFF

/\* switch off controller of the axis \*/

Program sample

MOTOR\_01.M

POS\_01.M

**■ MOTOR ON**

The motor control can be enabled again, following a previous MOTOR OFF, by use of the MOTOR ON command. When carrying out the MOTOR ON, the commanded position is set to the actual position, i.e. the motor remains at the actual position. Thus the positioning error is reset at the execution of MOTOR ON.

Summary

turns on motor control

Syntax

MOTOR ON


**NB!**

The MOTOR ON command is not suitable for re-activation of position control following an error. For this purpose, the ERRCLR command is to be used.

Command group

INI

Cross Index

MOTOR OFF

Syntax-Example

MOTOR ON

/\* switch on controller of the axis \*/

Program sample

MOTOR\_01.M

POS\_01.M

### ■ MOTOR STOP

By using the MOTOR STOP command, a drive in positioning, speed or synchronizing mode can be decelerated with programmed acceleration and arrested at the momentary position. A drive arrested with this command can, at a later point, via the CONTINUE command, resume its original motion. (Exception: CONTINUE does not continue an interrupted synchronization command.)



#### NB!

If MOTOR STOP is executed in a subprogram or if NOWAIT is set to ON, then the next lines in the program are already processed while MOTOR STOP is being processed; the braking process runs in the background.

Therefore, in order to slow the drive down to a speed of zero it is necessary to ascertain that no new positioning command is given during braking.

#### Summary

stops the drive

#### Syntax

MOTOR STOP

#### Command group

CON

#### Cross Index

POSA, POSR, CSTART, CONTINUE, CSTOP

#### Syntax-Example

MOTOR STOP /\* interrupt motion of the axis \*/

#### Program sample

MSTOP\_01.M

### ■ MOVESYNCORIGIN

The command shifts the origin of synchronization in relation to the master. While SET SYNCPOSOFFS sets the offset position absolutely, MOVESYNCORIGIN always relates to the last one and shifts the offset position relatively. If you have to shift the offset position continually, you can prevent too large numbers or an overflow in this way.

#### Summary

Relative shifting of the origin of synchronization

#### Syntax

MOVESYNCORIGIN mvalue

#### Parameter

mvalue = Relative offset in relation to the Master  
in qc



#### NB!

Valid for position synchronization SYNCP and position synchronization with marker correction SYNCM.

#### Command group

SYN

#### Cross Index

SET, SYNCPOSOFFS (54)

#### Syntax example

MOVESYNCORIGIN 1000

### ■ NOWAIT

The NOWAIT command defines the program flow for positioning commands. When NOWAIT OFF is set, the positioning commands are carried out in their entirety, i.e. until the target position is reached, before any following commands will be executed. When starting a positioning command with NOWAIT ON, further command procedures are continued and the positioning work takes place in the background (so to say). In the NOWAIT ON condition it is thus possible to query the momentary position, or to alter the velocity or the target position during the positioning procedure.

#### Summary

Wait / Do not wait after a POSA/POSR command

#### Syntax

NOWAIT s

#### Parameter

s = condition:  
 ON = no wait for positioning till target position is reached  
 OFF = wait till positioning achieved



#### NB!

The default condition is NOWAIT OFF, i.e. if no NOWAIT instruction is contained within a program, then the positioning procedure is carried out in its entirety before the processing of the next command is begun. If, when in NOWAIT condition during an active positioning procedure, a further positioning command follows, then the new target position will be tracked without interruption. The HOME as well as the INDEX commands will be processed to the end in the NOWAIT ON condition, before the next command can be begun.

#### Command group

CON

#### Cross Index

WAITAX, AXEND, POSA, POSR, HOME, INDEX

#### Syntax-Example

```
NOWAIT ON
/* no waiting after POS-commands */
NOWAIT OFF
/* wait after POS-commands till target reached */
```

#### Program samples

NOWAI\_01.M, MSTOP\_01.M, OUT\_01.M, VEL\_01.M

### ■ ON APOS .. GOSUB

It is possible to call up a subprogram with the instruction ON APOS, if a specific slave position (UU) has been passed in positive or negative direction. The instruction can be useful for positioning and synchronization controls, as well as for cam controls and cam boxes. For example, in order to replace the increasing slave position in the case of open curves after each cycle by a recurring reference point.

#### Summary

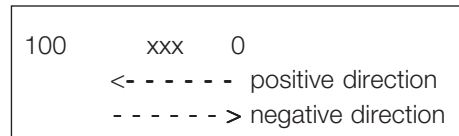
Call up a subprogram when the slave position xxx is passed.

#### Syntax

ON sign APOS xxx GOSUB name

#### Parameter

sign + when the slave position xxx is passed in positive direction  
 - when the slave position xxx is passed in negative direction  
 xxx = slave position in qc  
 name = name of the subprogram



#### Note

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.



#### NB!

During the execution of subprograms triggered by an interrupt, NOWAIT ON is set automatically.

#### Command group

INT

#### Cross Index

SUBPROG...RETURN, DISABLE... , ENABLE, Priorities of interrupts

#### Syntax Example

```
ON -APOS 800 GOSUB name
// Call up the subroutine name when slave
// position 800 is passed in negative direction
```

### ■ ON COMBIT .. GOSUB

The instruction ON COMBIT is used to call up a subprogram when Bit n of the communication buffer is set.

#### Summary

Call up a subprogram when Bit n of the communication buffer is set.

#### Syntax

ON COMBIT n GOSUB name

#### Parameter

n = Bit n of communication buffer  
 $-32 \leq n \leq 32, n \neq 0$   
 name = name of subprogram



#### NB!

ON COMBIT refers to the first 32 Bits of the process data memory.

#### Priority

If a number of interrupts occur simultaneously, the subprogram assigned to the lowest bit is worked through first. The other interrupts will be processed afterwards. If, during an interrupt subroutine, the same interrupt occurs (exception: error interrupt), then it will be ignored and thus lost.

#### Peculiarities

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.



#### NB!

During the execution of an ON COMBIT subroutine NOWAIT is set to ON.

#### Portability

In the case of COMOPTGET and COMOPTSEND, the offset of 2 Word is retained for compatibility reasons.

#### Command group

INT

#### Cross Index

SUBPROG...RETURN, COMOPTGET, COMOPTSEND

#### Syntax-Example

```
ON COMBIT 5 GOSUB test
 // set interrupt on fieldbus bit 5
```

### ■ ON ERROR GOSUB

By using the ERROR GOSUB instruction, a subroutine will be defined, which can be called up in case of error. If an error occurs after the definition, then an automatic program abort will not take place – instead, the defined subroutine will be called up.

Within this subroutine, it is possible to target the re-action to the error, to wait for user intervention via ERRCLR (clear error) or, in the case of non-correctable errors, to abort the program via the EXIT instruction.

If the program is not aborted, then the processing will continue from the point where the interruption occurred.

By using the CONTINUE command, it is possible to continue the error-interrupted motion. (Exception: synchronization commands)

#### Summary

Definition of an error subroutine

#### Syntax

ON ERROR GOSUB name

#### Parameter

name = name of the subroutine

#### Priority

Error subroutines cannot be interrupted through any other interrupts.

#### Peculiarities

The ON ERROR GOSUB instruction should be at the start of a program, so that it has validity for the entire program.

The subroutine to be called up must be defined within the identified SUBMAINPROG and ENDPROG program.

The identification of an error condition and the call up of the corresponding subroutine requires a maximum of 2 milliseconds.



#### NB!

During the execution of an error routine NOWAIT is automatically set to ON.

If the error subroutine is exited with the error still active because e.g. ERRCLR was not carried out or another error has occurred, then a new call takes place.



**NB!**

The ON ERROR GOSUB xx routine does not terminate the HOME and INDEX command.

This means they will be executed after the error has been cleared. To prevent this an ON TIME 1 can be included in the error routine.

Command group

INT

Cross Index

SUBPROG...RETURN, ERRCLR, ERRNO, CONTINUE, EXIT

Syntax-Example

```
ON ERROR GOSUB errhandle
/* definition of an error subroutine */
command line 1
command line n
SUBMAINPROG
/* subroutine errhandle must be defined */
SUBPROG errhandle
command line 1
command line n
RETURN
ENDPROG
```

Program sample

ERROR\_01.M  
IF\_01.M, INDEX\_01.M

■ **ON INT .. GOSUB**

By using the ON INT GOSUB instruction, a subroutine must be defined which will be called up when an edge is detected at the monitored input.

A maximum of one subroutine per input can be defined. It is not possible to define an interrupt for the falling and the rising edges of the same input. This definition can take place at any time. If, following this definition, a corresponding interrupt occurs, then the accompanying subroutine is called up and processed. After the last subroutine command (RETURN), the program will continue from the point of interrupt.

Summary

Defining an interrupt input

Syntax

ON INT n GOSUB name

Parameter

n = number of the input to be monitored;  
(input area -8 ... 8  
and VLT inputs 16 ... 33)  
positive input numbers (1 ... 8) =  
reaction to the rising edge  
negative input numbers (-1 ... 8) =  
reaction to the falling edge  
name = subroutine name

Priority

If a number of interrupts occur simultaneously, the subprogram assigned to the lowest bit is worked through first. The other interrupts will be processed afterwards. If, during an interrupt subroutine, the same interrupt occurs (exception: error interrupt), then it will be ignored and thus lost.

Peculiarities

The ON INT GOSUB instruction should be at the start of the program, so that it has validity for the entire program.

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.

The identification of an interrupt and the call up of the corresponding subroutine requires a maximum of 2 milliseconds.

A minimal signal length of 1 msec is necessary for the sure identification of a level change! The chapter input/output terminal contain more information concerning the input circuit and input technical data.



**NB!**

The instruction for ON INT GOSUB is edge and not level triggered.



**NB!**

During the execution of a subroutine called by an interrupt NOWAIT is automatically set to ON.

Command group

INT

Cross Index

SUBPROG...RETURN, ON ERROR GOSUB, WAITI

Syntax-Example

```
ON INT 4 GOSUB posin
/* Definition of Input 4 (positive edge) */
ON INT -5 GOSUB negin
/* Definition of input 5 (negative edge) */
 command line 1
 command line n
subroutine must be defined
SUBMAINPROG
SUBPROG posin
 command line 1
 command line n
RETURN
SUBPROG negin
 command line 1
 command line n
RETURN
ENDPROG
```

Program sample

ONINT\_01.M  
DELAY\_01.M, GSVEL\_01.M

■ ON MAPOS .. GOSUB

It is possible to call up a subprogram with the instruction ON APOS, if a specific slave position (UU) has been passed in positive or negative direction. For example, in order to set an output at any point in the case of a linear drive (slave) with a traversing range from 0 to 10000 UU.

Summary

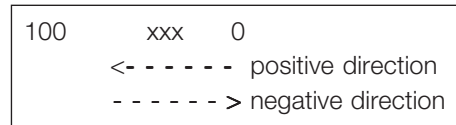
Call up a subprogram when the master position xxx (qc) is passed.

Syntax

ON sign MAPOS xxx GOSUB name

Parameter

sign = + when the slave position xxx is passed in positive direction  
 - when the slave position xxx is passed in negative direction  
 xxx = slave position in qc  
 name = name of the subprogram



Peculiarities

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.



**NB!**

During the execution of subprograms triggered by an interrupt, NOWAIT ON is set automatically.

Command group

INT

Cross index

SUBPROG...RETURN, DISABLE ENABLE, Priorities of interrupts

Syntax Example

ON +MAPOS 1200 GOSUB name  
 // Always call up subprogram at position 1200

### ■ ON MCPOS .. GOSUB

It is possible to call up a subprogram with the instruction ON MCPOS which is typical for cam controls if a specific master position (MU) has been passed in positive or negative direction. This allows not only the realization of cam boxes, but also the execution of tasks that are much more complex. For example, one could change parameters online depending on the position.

#### Summary

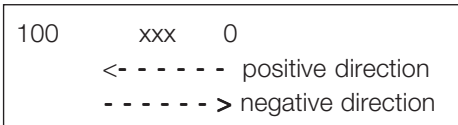
Call up a subprogram when the master position xxx (MU) is passed.

#### Syntax

ON sign MCPOS xxx GOSUB name

#### Parameter

sign = + when the slave position xxx is passed in positive direction  
 - when the slave position xxx is passed in negative direction  
 xxx = slave position in qc  
 name = name of the subprogram



#### NB!

A DEFMCPOS or a SETCURVE must always be placed in front of the command ON MCPOS .. GOSUG, since otherwise the curve position is not known.

#### Peculiarities

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.



#### NB!

During the execution of subprograms triggered by an interrupt, NOWAIT ON is set automatically.

#### Command group

INT

#### Cross index

SUBPROG...RETURN, DISABLE ENABLE, Priorities of interrupts

#### Syntax Example

Cardboard boxes are transported irregularly on a conveyor belt. By setting an output, the slave will always be started when the position xxx is reached.

```
ON +MCPOS 4500 GOSUB output
 // call subprogramm output
 // always on position 4500
SUBMAINPROG // set subprogram output
SUBPROG output
 OUT 3 1 // 03 on
RETURN
ENDPROG
ON +MCPOS 4500 GOSUB output
 // call subprogramm output
 // always on position 4500
```

### ■ ON PARAM .. GOSUB

The instruction ON PARAM can be used to respond when parameters are altered via the VLT display and to call up a subprogram.

#### Summary

Call up a subprogram when a parameter is altered.

#### Syntax

ON PARAM n GOSUB name

#### Parameter

n = 701 ... 779  
name = subroutine name

#### Priority

If, during an interrupt subroutine, the same interrupt occurs (exception: error interrupt), then it will be ignored and thus lost.

#### Peculiarities

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.

A maximum of 10 ON PARAM functions are possible.



#### NB!

During the execution of a subroutine called by an interrupt NOWAIT is automatically set to ON.

#### Command group

INT

#### Cross Index

SUBPROG...RETURN

#### Syntax-Sample

```
LINKAXPAR POSERR 712 „position error“ 300
5000 0
ON PARAM 712 GOSUB poserr
// when position error is changed
SUBMAINPROG
SUBPROG poserr
 PRINT „New position error: „, GET POSERR
RETURN
```

### ■ ON PERIOD

With ON PERIOD it is possible to call up a subprogram at regular intervals (time triggered). ON PERIOD works like an interrupt.

#### Summary

calls up a subroutine at regular intervals

#### Syntax

ON PERIOD n GOSUB name

#### Parameter

n > 100 ms = time in ms, after which the subroutine is called up again  
n = 0 = switch off the function



#### NB!

The precision with which the time is kept depends on the remaining program. Typically the precision is  $\pm 1$  ms.

#### Peculiarities

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.



#### NB!

During the execution of an ON PERIOD subroutine NOWAIT is set to ON.

#### Command group

INT

#### Cross Index

ON TIME, GOSUB

**■ ON STATBIT .. GOSUB**

The instruction ON STATBIT is used to call up a subprogram when bit n of the VLT status is set. These 32 bits of the VLT status consist of the VLT status word, the byte 2 of the internal status (e.g. MOVING) and the bit n of SYNCSTAT.

Summary

Call up a subprogram when bit n of the VLT status is set.

Syntax

ON STATBIT n GOSUB name

Parameter

n =       Bit n of the VLT status  
           Bit 1...16    VLT status word  
           Bit 17...24   Byte 2 of the internal status:  
           Bit 17 = MOVING  
           Bit 18 = Overflow Slave Encoder  
           Bit 19 = Overflow Master Encoder  
           Bit 20 = POSFLOAT active \*)  
           Bit 25...32   SYNCSTAT  
           Bit 25 = SYNCREADY  
           Bit 26 = SYNCFAULT  
           Bit 27 = SYNCACCURACY  
           Bit 28 = SYNCMMHIT  
           Bit 29 = SYNC SMHIT  
           Bit 30 = SYNCMMERR  
           Bit 31 = SYNC SMERR  
  
 name =   subroutine name

\*) Explanation: i.e. the axis is within the tolerance range of the control window REGWINMAX / REGWINMIN. As soon as the control window is set, the axis controller is switched on again.

Priority

If a number of interrupts occur simultaneously, the subprogram assigned to the lowest bit is worked through first. The other interrupts will be processed afterwards. If, during an interrupt subroutine, the same interrupt occurs (exception: error interrupt), then it will be ignored and thus lost.

Peculiarities

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.


**NB!**

During the execution of a subroutine called by an interrupt NOWAIT is automatically set to ON.

Command group

INT

Cross Index

SUBPROG...RETURN

Syntax-Example

```
ON STATBIT 30 GOSUB markererror
/* Interrupt, if error flag Master */
SUBMAINPROG
 SUBPROG markererror
 SYNCSTATCLR 32
 /* clear error flag SYNCMMERR */
 /* use value 32 of Parameter SYNCSTATCLR,
 not the bit-number! */
RETURN
ENDPROG
```

### ■ ON TIME

After expiration of the time set the corresponding subroutine is called up. In the meantime the program flow continues normally.

#### Summary

One-time access of a subroutine

#### Syntax

ON TIME n GOSUB name

#### Parameter

n = time in ms, after which the subroutine is called up (maximum MLONG)  
 name = name of the subroutine



#### NB!

The precision with which the time is kept depends on the remaining program. Typically the precision is  $\pm 1$  ms.

#### Peculiarities

The subroutine to be called up must be defined within the SUBMAINPROG and ENDPROG identified program.



#### NB!

During the execution of an ON TIME subroutine NOWAIT is set to ON.

#### Command group

INT

#### Cross Index

ON PERIOD, GOSUB

#### Syntax-Example

```
OUT 1 1 /* light on */
ON TIME 200 GOSUB off1
/* light off again after 200 ms */
SUBMAINPROG
SUBPROG off1
OUT 1 0
RETURN
ENDPROG
```

### ■ OUT

The 8 digital outputs of the SyncPos option and the 2 relay outputs of the VLT5000 can be set and reset by using the OUT command.

#### Summary

Set or reset digital outputs

#### Syntax

OUT n s

#### Parameter

n = output number  
 SyncPos option: O1 ... O8  
 Relay O1 = 11  
 Relay O4 = 14  
 s = condition (0 = OFF, 1 = ON)



#### NB!

After switching on the system, all outputs are OFF.

These outputs, which have pre-defined functions according to the I/O parameter settings, will also be influenced by the OUT commands! The actual output status remains as it is, even after program end or program abort. The output circuit and maximum load current can be taken from the chapter input/output terminal and the VLT5000 manual.

#### Command group

I/O

#### Cross Index

OUTB, IN, INB

#### Syntax-Example

```
OUT 3 1 /* 03 on */
OUT 6 0 /* 06 off */
```

#### Program sample

OUT\_01.M

### ■ OUTAN

The OUTAN command can set VLT bus reference (speed or torque reference depending on setting of VLT parameter 100).



#### NB!

The command MOTOR OFF must be executed previously. Thus, monitoring of the position error is no longer active.

With OUTAN it is also possible to turn off the controller in OPEN LOOP using MOTOR OFF and to operate the VLT without return as a pure frequency converter. In this manner you can use SyncPos to directly output set values, to read inputs, etc.

#### Summary

sets speed reference

#### Syntax

OUTAN v

#### Parameter

v = speed reference  
range: -0X4000 ... 0X4000 = -100 % ... 100 %



#### NB!

Remember that with older versions a different scale may exist.

#### Command group

I/O

#### Cross Index

MOTOR OFF  
Chapter input/output terminal; VLT5000 manual

#### Syntax-Example

MOTOR OFF  
OUTAN 0X2000 /\* set speed reference 50 % \*/

### ■ OUTB

With the OUTB command the condition of the digital outputs can be changed byte-by-byte. The byte value transferred determines the condition of the individual outputs. The bit with the lowest value in the byte corresponds to the set condition of output 1.

#### Summary

Alteration of the condition of a digital output byte

#### Syntax

OUTB n v

#### Parameter

n = output byte (0 = O1 ... O8)  
v = value (0 ... 255)



#### NB!

After switching on the system, all outputs are OFF. Outputs which have pre-defined functions according to the I/O parameter settings, will also be influenced by the OUTB command! The actual output status remains as it is even after program end or program abort. Output circuit and maximum load current [see](#) chapter input/output terminal.

#### Command group

I/O

#### Cross Index

OUT, IN, INB

#### Syntax-Example

OUTB 0 10  
/\* switch through outputs 2 and 4, disable other outputs \*/  
OUTB 0 245  
/\* disable outputs 2 and 4, switch through all other outputs \*/  
OUTB 0 128  
/\* switch through output 8 only, disable others \*/

#### Program sample

OUTB\_01.M

**■ OUTDA**

With the OUTDA command it is possible to control the analogue and digital outputs of the VLT control card.

**VLT5000** has two configurable outputs that can be analogue or digital, they are configured via parameter 319 (output 42) and parameter 321 (output 45).

**VLT5000Flux** has two analogue outputs (output 42 configured in parameter 319 and output 45 configured in parameter 321) and two digital/pulse outputs (output 26 configured in parameter 341 and output 46 configured in parameter 355).

A VLT control card output can only be controlled from the application program when it is configured as option output in the appropriate parameter.

---

**Summary**

Sets VLT output

---

**Syntax**

OUTDA no value

---

**Parameter VLT5000**

no = output number (42 or 45)

value = if parameter 319/321 =

"Option digital": value = 0 or 1

0 = output low (0 V)

1 = output high (24 V)

= if parameter 319/321 =

"Option 0...20 mA": then value = 0...100000 corresponding to 0...20 mA

"Option 4...20 mA": then value = 0...100000 corresponding to 4...20 mA

"Option 0...32 kHz": then value = 0...100000 corresponding to 0...32 kHz

---

**Parameter VLT5000Flux**

no = output number (26, 42, 45 and 46)

value = if parameter 319/321 (output 42/45) =

"Option 0...20 mA": then value = 0...100000 corresponding to 0...20mA

or

"Option 4...20 mA": then value = 0...100000 corresponding to 4...20 mA

or

= if parameter 341/355 (output 26/46) =

"Option digital": then value = 0 or 1

0 = output low (0 V)

1 = output high (24 V)

or

"Option 0...50 kHz": then value = 0...100000 corresponding to 0...50 kHz

---

**Command group**

I/O

---

**Cross Index**

VLT parameter 319, 321, 341 (only Flux) and 355 (only Flux)

---

**Syntax-Example**

```
OUTDA 42 5000 // set analog output to 10 mA
 // condition: parameter 319 is
 // set to "option 0...20 mA"
```

**■ PCD**

You can directly access the fieldbus data area with the command PCD without an additional command COMOPTGET or COMOPTSEND. The communications memory is written or read word by word (16-Bit).

---

**Summary**

Pseudo array for direct access to the fieldbus data area

---

**Syntax**

PCD[n]

n = index

---

**Command group**

Communication option

---

**Cross Index**

SYSVAR

---

**Syntax-Example**

```
Variable = PCD[1] // Word 1
Variable = PCD[1].2 // Bit 2 of Word 1
Variable = PCD[2].b1 // Byte 1 of Word 2
PCD[1] = Variable
PCD[1].3 = Variable
```

---

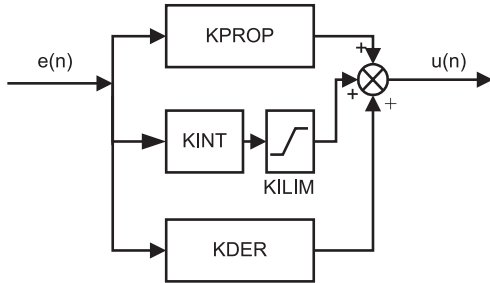
**Program sample**

```
_IF (PCD 2 ==256) THEN // compare value
_IF (PCD 3.2) THEN // is bit 2 of PCD3 high?
```

## ■ PID

A PID filter can be calculated with this function. The PID filter works according to the following formula:

$$u(n) = ( k_p * e(n) + k_d*(e(n)-e(n-1)) + k_i*\sum e(n) ) / \text{timer}$$



where the following is true:

- e(n) error occurring at time n
- k<sub>p</sub> proportional factor of the PID control
- k<sub>d</sub> derivative factor
- k<sub>i</sub> integral factor (limited by Integration Limit)
- Timer controller sample time

The corresponding factors can be set with the following commands:

```
SET PID KPROP 1 /* set kp 1 */
SET PID KDER 1 /* set kd 1 */
SET PID KINT 0 /* set ki 0 */
SET PID KILIM 0 /* Integration limit 0 */
SET PID TIMER 1 /* Sample time = 1 */
```

The following syntax examples which also show the default allocation of the factors.

### Summary

calculates PID filter

### Syntax

u(n) = PID e(n)

### Parameter

e(n) = actual deviation (error) for which the PID filter should be used

### Return value

u(n) = result of the PID calculation

### Command group

I/O

### Syntax-Example

```
e = INAD 53
u = PID e
PRINT "input = ",e, "output = ",u
```

## ■ POSA

The axis can be moved to a position absolute to the actual zero position.

When the POSA command exceeds the software limit switch NEGLIMIT (4) or POSLIMIT (5) the program continue with the next command after an error.

### Summary

Positioning in relation to actual zero point

### Syntax

POSA p

### Parameter

p = Position in user units (UU) absolute to the actual zero point; the UU corresponds in the standard setting the number of Quadcounts.



### NB!

If a temporary zero point, set via SET ORIGIN, exists and is active, then the position result refers to this zero point.



### NB!

If an acceleration and/or velocity has not been defined at the time of the POSA command, then the procedure will take place with the values of parameter DFLTVEL (33) and DFLTACC (34).

### Command group

ABS

### Cross Index

VEL, ACC, POSR, HOME, DEF ORIGIN, SET ORIGIN

Parameter: POSFACT\_N (26), POSFACT\_Z (23)

### Syntax-Example

```
POSA 50000 /* move axis to position 50000 */
```

### Program sample

POS\_01.M

### ■ POSA CURVEPOS

This command acts like POSA and moves the slave to the corresponding position on the curve, which is given by the actual master position.

#### Summary

Move slave to the curve position corresponding to the master position

#### Syntax

POSA CURVEPOS



#### NB!

If a temporary zero point, set via SET ORIGIN, exists and is active, then the position result refers to this zero point.



#### NB!

If an acceleration and/or velocity has not been defined at the time of the POSA command, then the procedure will take place with the values of parameter DFLTVEL (33) and DFLTACC (34).

#### Command group

ABS, CAM

#### Cross Index

CURVEPOS, SET ORIGIN

#### Syntax example

```
POSA CURVEPOS
 // Move slave to the curve position
 // corresponding to the master position
```

### ■ POSR

The axis can be moved to a position relative to the actual position by use of the POSR command.

#### Summary

Positioning in relation to actual position

#### Syntax

POSR d

#### Parameter

d = distance to actual position in user units (UU); this corresponds in the standard setting for the number of Quadcounts.



#### NB!

If acceleration and/or velocity has not been defined at the time of the POSR command, then the procedure will take place with the values of parameter DFLTVEL (33) and DFLTACC (34).

#### Command group

REL

#### Cross Index

VEL, ACC, POSA

Parameter: POSFACT\_N (26), POSFACT\_Z (23)

#### Syntax-Example

```
POSR 50000 /* move axis relative 50000 UU */
```

#### Program sample

POS\_01.M

### ■ PRINT

Calculation results, variables contents and text information can be displayed on the connected PC via the RS485 communication interface by use of the PRINT command, if the SyncPos software is open and the connection active.

To obtain multiple data with a single PRINT command, the individual elements must be separated with a comma (,). Text information must be given in quotation marks ("").

A line feed is normally created following each PRINT instruction. This automatic line feed can be suppressed with a semi-colon (;) after the last output element.

#### Summary

Information output

#### Syntax

PRINT i or PRINT i;

#### Parameter

i = information, for example, variables, text, CHR (n) separated by commas. The CHR command returns the ASCII characters corresponding to a certain number.



#### NB!

If there is no PC connected all prints are acted as a dummy output. But if a PC sends a XOFF because it is busy or the serial communication is not connected the messages are buffered. When the buffer is full, the next print command will wait until the buffer can be written again. That means, that the program stands still.

#### Command group

I/O

#### Cross Index

INKEY

#### Syntax-Example

```
PRINT "Information is important !"
/* print text information */
PRINT "Information is important !";
/* print information without line feed */
variable = 10
PRINT variable /* print contents of variables */
PRINT APOS /* print returned value function */
PRINT "Variable", variable,"Pos.:",APOS
/* print mixed information */
```

#### Program sample

Uses – see all [program samples](#) in Online Help

### ■ PULSACC

With PULSACC it is possible to set the acceleration/deceleration for the virtual master (encoder output).

The virtual master signal simulates an encoder signal. To calculate the pulse acceleration PULSACC the parameters **Encoder resolution**, master velocity and the ramp times must be taken into consideration.

Example:

The virtual master signal should correspond to an encoder signal of 1024 counts/revolution. The maximum speed of 25 encoder revolutions/s should be achieved in 1 s.

$$\begin{aligned} \text{PULSACC} &= \frac{\text{pulse velocity (PULSVEL) [Hz]}}{t[\text{s}]} \\ &= \frac{25 \frac{\text{counts}}{\text{s}} \cdot 1024 \text{ counts/turn}}{1 \text{ s}} \\ &= 25600 \frac{\text{counts}}{\text{s}^2} = 25600 \frac{\text{Hz}}{\text{s}} \end{aligned}$$

The signals generated are evaluated simultaneously as master input so that MAPOS, MIPOS etc. function as they would in an external master.

PULSACC = 0 is the condition for switching off the virtual master mode.



#### NB!

Changes in the acceleration in PULSACC are only valid after the next PULSVEL command.

#### Summary

Set acceleration for the virtual master

#### Syntax

PULSACC a

#### Parameter

a = acceleration in Hz/sec

#### Command group

SYN

#### Cross Index

PULSVEL

### ■ PULSVEL

With PULSVEL it is possible to set the velocity for the virtual master (encoder output).

The virtual master signal simulates an encoder signal. To calculate the pulse velocity the parameters **Encoder resolution** and master velocity must be taken into consideration.

Example:

The virtual master signal should correspond to an encoder signal of 2048 counts /revolution within a encoder speed of 50 revolutions/sec.

$$\begin{aligned} \text{PULSVEL} &= \text{encoder counts per turn} \cdot \frac{\text{turns}}{\text{s}} \\ &= 2048 \cdot 50 \text{ Hz} = 102400 \text{ Hz} \end{aligned}$$

#### Summary

Set the velocity for the virtual master

#### Syntax

PULSVEL v

#### Parameter

v = velocity in pulses per second (Hz)

#### Command group

SYN

#### Cross Index

PULSACC

### ■ REPEAT .. UNTIL ..

The REPEAT..UNTIL construction enables any number of repetitions of the enclosed program section, dependent on abort criteria. The abort criteria consist of one or more comparative procedures and are always checked at the end of a loop. As long as the abort criteria are not fulfilled, the loop will be processed repeatedly.

#### Summary

Conditional loop with end criteria  
(Repeat ... until condition fulfilled)

#### Syntax

REPEAT  
UNTIL condition

#### Parameter

condition = Abort criteria



#### NB!

Because the abort criteria are checked at the end of the loop, the commands within the loop will be carried out at least once. To avoid the possibility of an endless loop, the processed commands within the loop must have a direct or indirect influence on the result of the abort monitoring.

#### Command group

CON

#### Cross Index

LOOP, WHILE .. DO .. ENDWHILE

#### Syntax-Example

```
REPEAT /* start loop */
command line 1
command line n
UNTIL (A != 1) /* Abort condition */
```

#### Program sample

REPEA\_01.M  
DIM\_01.M, ONINT\_01.M, OUT\_01.M, INKEY\_01.M

### ■ RST ORIGIN

A previously with SET ORIGIN set temporary zero point can be erased by use of the RST ORIGIN command. This means that all the following absolute positioning commands (POSA) again refer to the real zero point.

#### Summary

Erase temporary zero point

#### Syntax

RST ORIGIN

#### Command group

INI

#### Cross Index

SET ORIGIN, DEF ORIGIN, POSA

#### Syntax-Example

```
/* reset temporary zero point */
RST ORIGIN
```

#### Program sample

TORIG\_01.M, OUT\_01.M, VEL\_01.M

### ■ SAVE part

If you alter array elements, axis parameters while the program is running you can save the altered values in the EEPROM with SAVE ARRAYS. To save the range of global parameters and user parameters (130–229) individually in the EEPROM, use the command SAVE GLBPARS.

#### Summary

Save arrays, current axis parameters or global parameters in the EEPROM

#### Syntax

SAVE part

```
part = ARRAYS
 AXPARS
 GLBPARS
```

#### Command group

INI

#### Cross Index

DELETE ARRAYS, SAVEPROM

### ■ SAVEPROM

When changing array elements or user parameters (130–229) while the program is running SAVEPROM offer the possibility of saving the values which have been changed. This must be done by triggering the command SAVEPROM explicitly.

SAVEPROM triggers the same process, which can also be started in the menu "CONTROLLER". If you want to save only array elements or only global and user parameters, use the commands SAVE ARRAY and SAVE GLBPARS.

#### Summary

saves memory in EEPROM



#### NB!

The execution time of SAVEPROM depends on the amount of data to be saved. It can be up to 4 seconds.



#### NB!

Please note that AXE parameters are not saved by SAVEPROM. To do this you must use the command SAVE AXPARS.

#### Command group

INI

#### Syntax-Example

```
PRINT "please wait"
SAVEPROM
PRINT "Thanks"
```

### ■ SET

With the SET command certain axis and global parameters can be temporarily changed while the program is running.

The parameter codes permitted can be found in the respective overviews of the [Parameter reference](#) in the left-hand column, for example, KPROP (11) for the **Proportional factor**, KDER (12) for the **derivative factor** or POSERR (15) for the **Tolerated position error**. Sample for global parameters: SET PRGPAR (102) for the **Activated program number** and SET I\_BREAK (105) for the **Input for abort**.

#### Summary

sets a parameter

#### Syntax

SET par v

#### Parameter

par = Parameter identification

v = parameter value



#### NB!

The parameter alterations are only valid while the program is running. After program end or abort, the original parameter values are valid again.

The parameter alterations can be made permanent by using the command SAVEPROM.

#### Command group

PAR

#### Cross Index

GET

Parameter-reference

#### Syntax-Example

```
SET POSLIMIT 100000
```

```
/* set positive positioning limit */
```

```
SET KPROP 150
```

```
/* change proportional factor */
```

#### Syntax-Example

```
SET KPROP 150
```

```
/* change proportional factor */
```

### ■ SETCURVE

This command defines the actual used curve, which is described in *array*. This command has to be used, before you can use the commands CURVEPOS, SYNCCxx, SYNCCSTART or SYNCCSTOP.

When this command is executed, the necessary pre calculations are done.

#### Summary

Set CAM curve

#### Syntax

SETCURVE array

#### Parameter

array = name of the array or of the curve



#### NB!

The DIM instruction with the name of the curve or array and the number of array elements must stand in front of the command SETCURVE or at the beginning of the program. If there are several arrays or curves in the cnf file, then the order in the DIM instruction must match the order of the arrays in the cnf file.

#### Command group

PAR

#### Cross Index

DIM

#### Syntax example

```
DIM curve [280]
```

```
// See number of elements in the title bar
```

```
// of the CAM-Editor
```

```
SETCURVE curve
```

### ■ SETMORIGIN

With the SETMORIGIN command you can set any position as the new zero point for the master.

#### Summary

Set any position as the zero point for the master.

#### Syntax

SETMORIGIN value

#### Parameter

value = absolute position



#### NB!

The command SETMORIGIN cancels the command DEFMORIGIN.



#### NB!

Thus, to alter the zero point for the master again, you have to reset it with SETMORIGIN or DEFMORIGIN. RST ORIGIN does not have any effect on the zero point for the master.

#### Command group

INI

#### Cross Index

DEFMORIGIN, MAPOS

#### Syntax-Example

SETMORIGIN 10000

/\* Set the zero point for the master at 10000 \*/

### ■ SET ORIGIN

Any absolute position can temporarily be set as a new reference point for absolute positioning command (POSA) by use of the SET ORIGIN command. This position is called temporary zero point. In combination with the command CURVEPOS, one can fix in this way that the current slave position matches the corresponding value of the curve.

#### Summary

Set absolute position as temporary zero point

#### Syntax

SET ORIGIN p

#### Parameter

p = absolute position in relation to the real zero point



#### NB!

It is possible to carry out several SET ORIGIN commands without out a previous RST ORIGIN. The absolute position value always refers to the real zero point. The last carried out SET ORIGIN command therefore determines the position of the temporary zero point in relation to the real zero point.

#### Command group

INI

#### Cross Index

RST ORIGIN, DEF ORIGIN, POSA, CURVEPOS, POSA CURVEPOS

#### Syntax Example

SET ORIGIN 50000

// set temporary zero point to 50000

#### Syntax Example

SET ORIGIN (-CURVEPOS)

// Set temporary zero to the beginning

// of the curve

#### Program sample

TORIG\_01.M

OUT\_01.M, VEL\_01.M

**■ SETVLT**

With the SETVLT command VLT parameters can be changed temporarily and thus the configuration of the VLT can also be changed temporarily. Since only integer values can be transmitted the parameter value to be transmitted must be adjusted with the associated conversion index. A list of the VLT parameters with the corresponding conversion index can be found in the VLT5000 manual.

Summary

sets a VLT parameter

Syntax

SETVLT par v

Parameter

par = Parameter number  
v = Parameter value


**NB!**

The parameter alterations are only stored in RAM. After power down the original parameter values are restored.

Command group

PAR

Cross Index

GETVLT

Syntax-Example

```
/* change parameter 202 "maximum reference"
high to 60 Hz */
-Conversion index = -3 (Multiplied with 103 during
transmission)
SETVLT 202 60000
```

**■ SETVLTSUB**

With the SETVLT commands VLT parameters can be changed temporarily and thus the configuration of the VLT can also be changed temporarily, in this case parameters with index numbers too. Since only integer values can be transmitted the parameter value to be transmitted must be adjusted with the associated conversion index. A list of the VLT parameters with the corresponding conversion index can be found in the VLT5000 manual.


**NB!**

The parameter alterations are only stored in RAM. After power down the original parameter values are restored.

Summary

Sets a VLT parameter with index number

Syntax

SETVLTSUB par indxno v

Parameter

par = parameter number  
indxno = index number  
v = parameter value

Command group

PAR

Cross Index

GETVLTSUB

Syntax example

```
SETVLT 25 1 100
// sets index 1 of the parameter 25
// "Quick menu" to 100 "configuration"
// (VLT5000Flux only)
```

### ■ STAT

The STAT command reports the actual status of the axis control unit as well as that of the axis. For example, whether the axis controller shuts down, ends the motion or the end switch is active. The status of the program execution cannot be called up with STAT, but only with AXEND.

The status consists of a total of four bytes.

#### Summary

Query axis and control status

#### Syntax

res = STAT

#### Return value

res = Axis- and Control status (4-Byte value):

|        |                             |                               |
|--------|-----------------------------|-------------------------------|
| Byte 3 | MSB                         |                               |
|        | Bit 0                       | 1 = MOVING                    |
|        | Bit 1                       | 1 = OVERFLOW Slave Encoder    |
|        | Bit 2                       | 1 = OVERFLOW Master Encoder   |
|        | Bit 3                       | 1 = POSFLOAT active *)        |
| Byte 2 | Status byte of axis control |                               |
|        | Bit 7                       | 1 = axis control switched off |
|        | Bit 2                       | 1 = position reached          |
|        | Bit 0,1,3-6                 | has no meaning                |
| Byte 1 | not used                    |                               |
| Byte 0 | LSB                         |                               |
|        | Bit 7                       | 1 = limit switch active       |
|        | Bit 6                       | 1 = Reference switch active   |
|        | Bit 2                       | 1 = axis control switched off |
|        | Bit 0,1,3,4                 | not in use                    |

\*) Explanation: i.e. the axis is within the tolerance range of the control window REGWINMAX / REGWINMIN. As soon as the control window is set, the axis controller is switched on again.

#### Command group

I/O

#### Cross Index

AXEND

#### Syntax-Example

PRINT STAT                    /\* print status word \*/

#### Program sample

STAT\_01.M

### ■ SUBMAINPROG .. ENDPROG

The code word SUBMAINPROG begins the sub-routine section, and the code word ENDPROG ends this specific program. The term subroutine means command sequences that, via the GOSUB instructions, can be called up and executed from various program positions.

All necessary subroutines must be contained within the subroutine section. It is possible to insert a subroutine anywhere within a main program; however, for reasons of clarity, it is advisable to insert it either at the beginning or end of a program.

#### Summary

Subroutine section definition

#### Syntax

SUBMAINPROG

ENDPROG



#### NB!

Only one subroutine area may be inserted within a program.

#### Command group

CON

#### Cross Index

SUBPROG ...RETURN,

GOSUB, ON ERROR GOSUB, ON INT n GOSUB

#### Syntax-Example

SUBMAINPROG

/\* Begin the subroutine section \*/

subroutine 1

subroutine n

ENDPROG                    /\* End the subroutine section \*/

#### Program sample

GOSUB\_01.M

AXEND\_01.M, ERROR\_01.M, INCL\_01.M,

STAT\_01.M

### ■ SUBPROG name .. RETURN

The instruction SUBPROG identifies the beginning of a subroutine. The name of the subroutine must directly follow SUBPROG code word. The name can be made up of one or more characters, and must be unique, i.e. only one subroutine may have that name.

A subroutine can be called up and executed at any time by use of the GOSUB instruction.

A subroutine can have any number of command lines and can refer to all program variables. The last command in each subroutine must be the RETURN instruction, which permits exiting the subroutine and continuing the program with the command following the GOSUB instruction.

#### Summary

Subroutine definition

#### Syntax

SUBPROG name  
RETURN

#### Parameter

name = subroutine name



#### NB!

All subroutines must be contained within the SUBMAINPROG and ENDPROG defined areas. It is not admissible to declare a second subroutine within an existing subroutine.

#### Command group

CON

#### Cross Index

SUBMAINPROG ... ENDPROG,  
GOSUB, ON ERROR GOSUB, ON INT n GOSUB

#### Syntax-Example

```
SUBMAINPROG /* begin SP-section */
SUBPROG sp1 /* begin sp1 */
command line 1
command line n
RETURN /* end sp1 */
ENDPROG /* end SP-section */
```

#### Program sample

GOSUB\_01.M, AXEND\_01.M, ERROR\_01.M,  
IF\_01.M, STAT\_01.M

### ■ SWAPMENC

This command allows swapping of the master and slave encoders. This is particularly meaningful if one wishes to alternately use two motors with one control.

Prior to the command SWAPMENC ON/OFF, a MOTOR OFF must always be executed in order to avoid a tolerated position error exceeded. Also, the controller parameters or axis parameters must be changed if both motors are different.

The motor leads could be switched via relay.

#### Summary

Swap master and slave encoder internally.

#### Syntax

SWAPMENC s

#### Parameter

s = condition  
ON = master encoder input is feedback input  
OFF = slave encoder input is feedback input



#### NB!

In this changeover, no positions are lost, even if the motors are moved by hand while the other motor is controlled. It is possible to always access the uncontrolled motor through MAPOS as well.

#### Command group

INI

#### Cross Index

MAPOS

#### Syntax Example

```
SWAPMENC ON // Swap slave encoder internally
// with master encoder
```

#### Sample

```
MOTOR OFF
OUT 1 1 // Switch motor leads
SET KPROP ... // Change axis parameters
SWAPMENC ON // Swap encoder internally
MOTOR ON // Turn on control again
POSA 10000 // Move the motor which is
// connected to master encoder
```

```
MOTOR OFF
OUT 1 0 // Switch motor leads
SET KPROP ... // Change axis parameters
SWAPMENC OFF // Swapencoder
MOTOR ON // Turn on control again
POSA 0 // Move the motor again which is
// connected to the slave encoder
```

### ■ SYNCC

The command SYNCC starts the CAM-Mode (cam control). From this moment, the curve positions of the master are counted depending on the actual master positions and the defined starting behavior in SYNCMSTART (62): Where and when counting is started. With the parameter SYNCMSTART = 2000, the curve positions of the master are only counted after the next master marker.

#### Summary

Synchronization in CAM-Mode

#### Syntax

SYNCC num

#### Parameter

num = number of curves to be processed  
 0 = the drive remains in CAM-Mode until another mode is started with commands such as MOTOR STOP, CSTART, POSA etc.



#### NB!

SYNCC does not start the slave drive nor does it interrupt on-going motions (e.g. CVEL), only SYNCCSTART does.



#### NB!

The drive remains in CAM-Mode until num curves have been processed successfully. If the synchronization (after num curves) is being closed normally, the start stop point pair 2 will be used – if no SYNCCSTOP with a corresponding point pair is defined – in order to stop the drive. It will then come to a stop at the position slavepos (see parameters).

#### Command Group

CAM

#### Cross Index

SYNCCSTART

#### Syntax Example

```
DIM curve [280] // see number of elements in
 // the title bar of CAM-Editor
SETCURVE curve // Set curve
SYNCC
 // Synchronization in
 // CAM-Mode
```

### ■ SYNCCMM

Like SYNCC, the command SYNCCMM brings about a synchronization in CAM-Mode, but beyond that it also performs a marker correction (only if the master moves forward).

In order to save the distance between sensor and processing point, the parameter SYNCMPULSM is used. It allows the correction of the marker position without changing the curve. Also, larger sensor distances than the actual curve length are possible. In this case, a FIFO is used for the marker correction (see example).

#### Summary

Synchronization in CAM-Mode with master marker correction

#### Syntax

SYNCCMM num

#### Parameter

num = number of curves to be processed;  
 0 = the drive remains in CAM-Mode until another mode is started with commands such as MOTOR STOP, CSTART, POSA etc.



#### NB!

SYNCCMM does not start the slave drive nor does it interrupt on-going motions (e.g. CVEL), only SYNCCSTART does.



#### NB!

The drive remains in CAM-Mode until num curves have been processed successfully. If the synchronization (after num curves) is being closed normally, the start stop point pair 2 will be used – if no SYNCCSTOP with a corresponding point pair is defined – in order to stop the drive. It will then come to a stop at the position slavepos (see parameters).

#### Marker signal

The marker can be the zero pulse from the encoder or an external 24 volt signal:

I5 = master  
 I6 = slave

#### Command Group

CAM

#### Cross Index

SYNCMPULSM

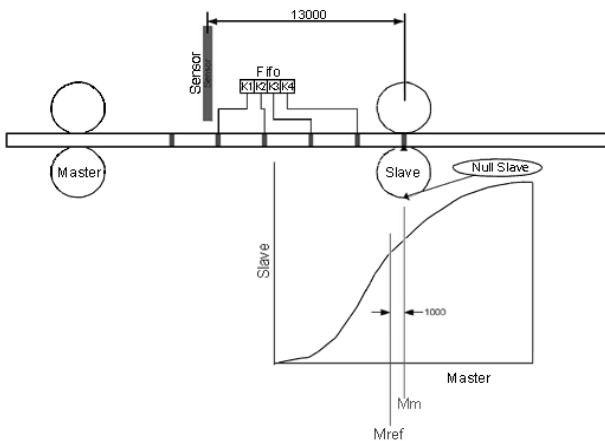
### Syntax Example

```
SETCURVE curve
SYNCCMM 1 // Synchronize 1 x in CAM mode
 // with marker correction
```

### Sample

If for example curve length is 3000 and distance of sensor to working point is 13000, we will have a FIFO with 4 register and an offset of 1000 which has to be concerned.

See the following diagram:



### ■ SYNCCMS

Like SYNCC, the command SYNCCMS brings about a synchronization in CAM-Mode, but beyond that it also performs a marker correction of the slave. Here, the slave position is corrected, not the curve position.

In contrast to SYNCCMM, no FIFO is created.

### Summary

Synchronization in CAM-Mode with slave marker correction

### Syntax

```
SYNCCMS num
```

### Parameter

num = number of curves to be processed;  
 0 = the drive remains in CAM-Mode until another mode is started with commands such as MOTOR STOP, CSTART, POSA etc.



### NB!

SYNCCMS does not start the slave drive nor does it interrupt on-going motions (e.g. CVEL), only SYNCCSTART does.



### NB!

The drive remains in CAM-Mode until num curves have been processed successfully. If the synchronization (after num curves) is being closed normally, the start stop point pair 2 will be used – if no SYNCCSTOP with a corresponding point pair is defined – in order to stop the drive. It will then come to a stop at the position slavepos (see parameters).

### Marker signal

The marker can be the zero pulse from the encoder or an external 24 volt signal:  
 I5 = master; I6 = slave

### Command Group

CAM

### Syntax Example

```
SETCURVE curve
SYNCCMS 0 // Synchronization in CAM-Mode
 // with slave marker correction
```

### ■ SYNCCSTART

The command starts the movement of the slave. With *pnum*, the point pair is selected that determines in which master position the synchronization begins and where it should be finished. When moving forward, the synchronization begins at point A and is finished up to point B. When moving backward, it begins at point B and is finished up to point A.

#### Summary

Start slave for synchronization in CAM-Mode

#### Syntax

SYNCCSTART *pnum*

#### Parameter

*pnum* = Start stop points number

*pnum* > 0 Engaging begins when the corresponding point A is reached, provided the master moves in positive direction; the engage curve is finished at point B.

If point A and B are identical, the slave will be engaged with the set maximum velocity – i.e. without curve – as soon as the master has reached this point.

*pnum* = 0 The slave will be engaged immediately with the set maximum velocity. It does not matter in what direction the master moves or whether it moves at all.

*pnum* < 0 Again, the corresponding point pair is used, however, engaging begins at point B and is finished at point A, i.e. in negative direction.

#### Command Group

CAM

#### Syntax Example

SETCURVE *curve*

SYNCC 0 // CAM mode synchronization

SYNCCSTART 1 // Engage slave at point A from  
// start stop point pair 1

### ■ SYNCCSTOP

This command stops synchronization without leaving SYNCC mode. The slave will be disengaged according to the point pair defined in *pnum*. Only then will the slave actually be stopped. When the stop point has been reached, the slave must be at *slavepos*.

#### Summary

Stop slave after the CAM synchronization

#### Syntax

SYNCCSTOP *pnum slavepos*

#### Parameter

*pnum* = Start stop points number

*pnum* > 0 Engaging begins when the corresponding point A is reached, provided the master moves in positive direction; the engage curve is finished at point B. If point A and B are identical, the slave will be engaged with the set maximum velocity – i.e. without curve – as soon as the master has reached this point.

*pnum* = 0 The slave will be engaged immediately with the set maximum velocity. It does not matter in what direction the master moves or whether it moves at all.

*pnum* < 0 Again, the corresponding point pair is used, however, engaging begins at point B and is finished at point A, i.e. in negative direction.

*slavepos* = Position where the slave is supposed to stand after disengaging.



#### NB!

When moving forward, disengaging begins at point A and is finished at point B; vice versa when moving backward.



#### NB!

If the program is closed without SYNCCSTOP command, disengaging occurs by default with the second point pair and a stop occurs at the Slave Stop Position defined in the curve data.

#### Command Group

CAM

### Cross Index

Curve Data, Slave Stop Position

### Syntax Example

SETCURVE curve

SYNCC 0 // Synchronize in CAM-Mode

SYNCCSTART 1

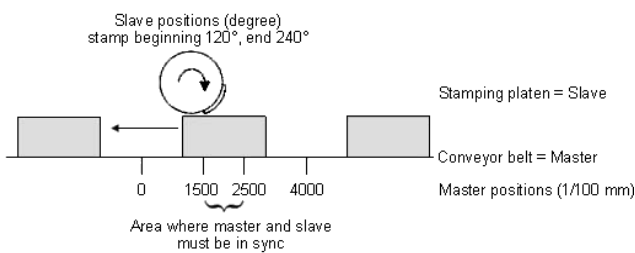
// Start slave with start point pair 1

SYNCCSTOP 2 0

// Stop slave with stop point pair 2 at the

// slave position 0 or 3600

### Sample



### ■ SYNCERR

SYNCERR returns the actual synchronization error in qc or in CAM mode in User Units UU. This is the distance between the actual master position (converted with drive factor and offset) and the actual position of the slave.

If the parameter SYNCACCURACY (55) is defined by a minus sign, you can also determine whether the synchronization is running ahead (negative result) or running behind (positive result).

### Summary

Queries actual synchronization error of the slave.

### Syntax

res = SYNCERR

### Return value

res = actual synchronization error of the slave in (qc) or in CAM mode in UU and

- as an absolute value when the value of the accuracy window is defined with a plus sign in the parameter SYNCACCURACY;
- with polarity sign when in SYNCACCURACY the value of the window is defined with a minus sign.



### NB!

Up to option card version < 5.00: SYNCERR only functions in synchronization mode. As soon as you exit SNYCM or SYNCP, the pulses are no longer counted. SYNCERR is only updated within a SYNC command.

With option card software 5.00 onwards the SYNCERR (as well as the internal master command position G\_Mpcmd) is also updated when SYNCP or SYNCM are not longer active, e.g. after a MOTOR STOP.

### Command group

I/O

### Cross Index

TRACKERR, MAPOS, APOS,  
SYNCPISOFFS (54), SYNCFACTM (49),  
SYNCFACTS (50), SYNCACCURACY (55)

### Syntax-Example

PRINT SYNCERR

/\* query actual synchronization error of the slave \*/

■ **SYNCM**

The SYNCM command functions just like the SYNCP command by making a angle/position synchronization with the master, but also makes a marker correction. Thus, during the starting of synchronization the program is synchronized to the next marker calculated. In this manner it is possible to compensate for differing running behaviors, for example slippage.

Once synchronization has been completed, deviation is determined at every marker (or every n-th marker if the number of markers is not identical for the master and slave). This is input into the control as the new offset and the program immediately attempts to compensate for this. However, in doing so the values set for velocity, VEL, and acceleration, ACC or DEC are not exceeded.

Summary

angle/position synchronization with the master with marker correction

Syntax

SYNCM

Peculiarities

In addition to the parameters used by SYNCP, SYNCREADY (56) and SYNCFAULT (57) are also of significance.



Since the following parameters could lead to overdefinition, it is important to ensure that these values are logical, match one another and are consistent with the information on the gear factors. SYNCMARKM (52) and SYNCMARKS (53) SYNCMPULSM (58) and SYNCMPULSS (59) SYNCMTYPM (60) and SYNCMTYPS (61)



SYNCM should only be called up once since the synchronizing continues until the next motion or stop command. All additional SYNCM commands cause the synchronization to start over again from the beginning and this is not normally intended, as you reset the actual SYNCERR.



When defined in SYNCMSTART (62), the system waits for the first evaluation of the marker pulses on starting SYNCM and only then the offset SYNCPOSOFFS (54) is applied.

Marker signal

The marker can be the zero pulse from the encoder or an external 24 volt signal (I5 = master; I6 = slave).

Command group

SYN

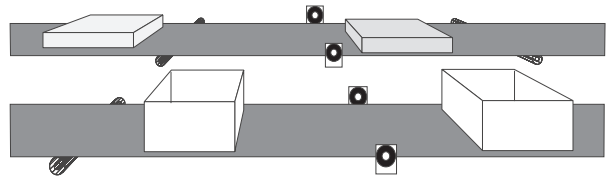
Cross Index

parameter of the AXS group

Syntax-Example

SYNCM /\* synchronization of the position with marker correction \*/

Example



Even when both belts are running synchronously the lids may not be aligned with the boxes at the right time. With SYNCM the difference between master and slave is detected by means of the external markers and the possible position deviation is corrected.

### ■ SYNCNP

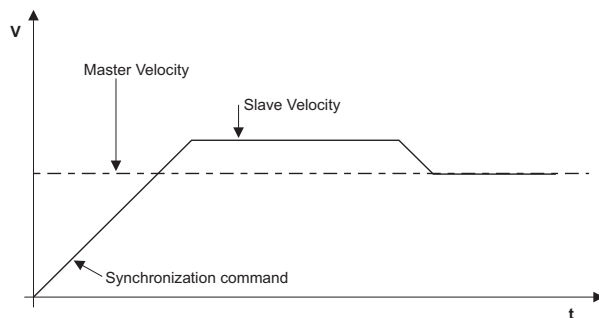
The SYNCNP command completes an angle/ position synchronization with the master. In doing so the position according to the gear factors to the master is kept synchronous, that means after an external disturbance the program subsequently tries to recover the corresponding stretch. However, in doing so the values set for velocity, VEL, and acceleration, ACC or DEC are not exceeded.

The following parameters affect the behavior: SYNCFACTM (49), SYNCFACTS (50)

|                   |                   |
|-------------------|-------------------|
|                   | gear factors      |
| SYNCPOSOFFS (54)  | position offset   |
| SYNCACCURACY (55) | accuracy for flag |
| REVERS (63)       | reverse behavior  |

During synchronization the program proceeds as follows:

When the SYNCNP command is started, the actual master position is determined and is retained. From the master velocity, and in consideration of the acceleration allowed, the necessary slave velocity is calculated in order to reach the master position. The slave is accelerated so long until the calculated position has been reached or until it is close enough to the reference position to reach it.



### Summary

angle/position synchronization with the master

### Syntax

SYNCNP



### NB!

As soon as the deviation between the position of the slave and master is less than SYNCACCURACY (55), the ACCURACY flag is set.

If REVERS (63) is set so that it is not possible to drive in reverse, but for some reason the slave is further than the master (e.g. because only the master has moved in reverse) then the slave will wait at velocity 0.

In doing so the slave takes its own acceleration time into consideration and will start moving if necessary before the correct position has been reached if the master already has a higher velocity.

Instead of using this catch-up procedure it is also possible to move the slave with CVEL to approximately the same velocity as the master and then trigger SYNCNP.

A change in the SYNCPOSOFFS (54) during the synchronization leads to a new synchronization procedure with ramps (see above).



### NB!

SYNCNP should only be called up once since the synchronizing continues until the next motion or stop command. All additional SYNCNP commands cause the synchronization to start over again from the beginning and this is not normally intended, as you reset the actual SYNCERR.

### Command group

SYN

### Cross Index

parameter of the AXS group

### Syntax-Example

SYNCNP

/\* normal synchronization of the position \*/  
CVEL 50

/\* achieve velocity before synchronization \*/

CSTART  
WAITT 500  
SYNCNP

### ■ SYNCSTAT

The following flags are defined and can be queried with SYNCSTAT: READY, FAULT, ACCURACY and MHIT and MERR for both the master and the slave.

#### Summary

Flag to query synchronization status

#### Syntax

res = SYNCSTAT

#### Return value

res = synchronization status with the following meaning:

|              | Value | Bit |
|--------------|-------|-----|
| SYNCREADY    | 1     | 0   |
| SYNCFAULT    | 2     | 1   |
| SYNCACCURACY | 4     | 2   |
| SYNCSMHIT    | 8     | 3   |
| SYNCSMERR    | 16    | 4   |
| SYNCSMHIT    | 32    | 5   |
| SYNCSMERR    | 64    | 6   |

#### SYNCACCURACY

The controller checks whether SYNCERR < SYNCACCURACY (55) is true every ms. If this is true, then SYNCACCURACY is set, otherwise the flag is deleted. This check is made for both SYNCP and SYNCM.

This flag is not used with SYNCV.

When executing a SYNCP or SYNCM command the flag is deleted.

#### SYNCFULT / SYNCREADY

For every SYNCP or SYNCM command these flags are deleted. Subsequently the program checks whether SYNCACCURACY is set or not at every marker pulse of the slave (SYNCP) or when a marker pulse of the master and a marker pulse of the slave exist (SYNCM).

If it is set the ready counter is increased and the fault counter is set to 0, otherwise the fault counter is increased and the ready counter set to 0.

If the ready counter is greater than the value determined by the parameter SYNCREADY (56), then the flag SYNCREADY is set. Otherwise it is deleted.

If the fault counter is greater than the value determined by the parameter SYNCFULT (57) then the flag SYNCFULT is set. Otherwise it is deleted.

#### SYNCSMHIT / SYNCSMERR

SYNCSMHIT and SYNCSMERR are set, when the master marker resp. the slave marker is occurred. These flags are deleted for every SYNCM command. Subsequently the flag SYNCSMHIT is set after the first occurrence of a master marker pulse or after the n-th marker pulse (parameters SYNCMARKM 52).

The same is true for SYNCSMERR with the slave.

#### SYNCSMERR / SYNCSMERR

If in the **marker window** SYNCSMWINM (68) or SYNCSMWINS (69) a tolerance range is defined, then SYNCSMERR or SYNCSMERR are set as soon as the maximum distance allowed has been achieved and no marker was identified.

Example:

```
Distance between two master markers
SYNCSMPULSM (58) = 30000
Marker Window SYNCSMWINM (68) = 1000
The flag is set at 31000 if no marker is
identified.
```

These flags are deleted for every SYNCM command.

If the **marker window** is 0 and thus no tolerance range is defined, the program checks every marker pulse (or after every n-th pulse) whether the distance between the two last markers registered is less than 1.8 times the value defined by the parameter SYNCSMPULSM (58). If not the corresponding flag is set.

The same applies analogously for SYNCSMERR in the slave.



#### NB!

These flags are automatically reset: during the next successful marker correction and in the event of a new start of SYNCM or through the command SYNCSTATCLR.

#### Command group

SYN

#### Cross Index

SYNCSTATCLR

#### Syntax-Example

```
IF (SYNCSTAT & 4) THEN OUT 1 1
/* If ACCURACY then set output */
ENDIF
```

### ■ SYNCSTATCLR

The corresponding bits can be reset in SYNCSTAT with SYNCSTATCLR value thus resetting the error flags MERR and the HIT flags MHIT. None of the other flags can be altered.

#### Summary

Resetting of the flags MERR and MHIT

#### Syntax

SYNCSTATCLR value

The SYNCSTATCLR command should only be used in a subprogram for dealing with errors. (see ON ERROR GOSUB).

#### Parameter

value = 8 = SYNCMMHIT  
 16 = SYNCMMHIT  
 32 = SYNCMMERR  
 64 = SYNCMMERR



#### NB!

ERRCLR contains the command MOTOR ON, which automatically switches the controller on again. (The motor is position-controlled at the current position.)

#### Command group

SYN

#### Cross Index

ON STATBIT, ON ERROR GOSUB, ERRNO, CONTINUE, MOTOR ON

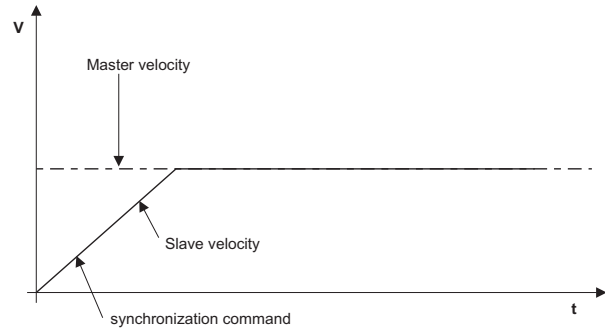
#### Syntax example

SYNCSTATCLR 32

/\* clear current error message \*/

### ■ SYNCV

With SYNCV a velocity synchronization with the master is completed. In doing so only the velocity is taken into consideration and the controller does not attempt to recover the position.



For synchronization and during the synchronization process neither the pre-set velocity, VEL, nor the pre-set acceleration, ACC or DEC, are exceeded. The parameters of the gear factors used for synchronization are: SYNCFACTM (49), SYNCFACTS (50).

Furthermore, the speeds are not simply determined on the basis of the difference between the current position minus the last position (master/slave), rather the values are filtered according to the settings in SYNCVFTIME (65). This means the filter for the slave is determined by the maximum speed.

In other words:

$VELMAX * 5$  corresponds to the encoder resolution for the filter table, where VELMAX is the speed in qc/ms. (The formula is a result of the assumption that the filter table for the encoder resolution was made with a maximum speed of 3000 Rpm.)

During the transition from the speed controller to the position controller this is done as smoothly as possible. In addition the new set position is defined in such a manner that the following is true:

$$\text{command\_pos} = \text{actual\_pos} + \text{error}$$

old\_error, cvel, avel are maintained. Sumerr is also accepted unchanged.

### Summary

velocity synchronization with the master

### Syntax

SYNCV



#### NB!

SYNCV should only be called up once since the synchronizing continues until the next motion or stop command. All additional SYNCV commands cause the synchronization to start over again from the beginning and this is not normally intended, as you reset the actual SYNCERR.



#### NB!

Position error is not monitored in SYNCV mode, it is therefore recommendable to use the hardware encoder monitor.

### Command group

SYN

### Cross Index

parameter of the AXS group

### ■ SYSVAR

The system variable SYSVAR – a prepared pseudo array – provides detailed system information. You also need this index if you link the system variable with the LCP display using LINKSYSVAR or specify recording data of a test run with TESTSETP.

### Summary

System variable (Pseudo array) reads system values.

### Syntax

SYSVAR[n]

n = index

### System Process Data

| Index | Description                                                         |
|-------|---------------------------------------------------------------------|
| 1     | Input Byte 0 (I1 ... I8)                                            |
| 2     | Input Byte 1 (VLT input 16 ... 33)                                  |
| 9     | Output Byte 0                                                       |
| 17    | Top 2 bytes which are provided by the SyncPos command STAT          |
| 22    | Value which is also supplied by the SyncPos command TIME            |
| 28    | Actual motor current [1/100 Amp.]                                   |
| 30    | Motor voltage voltage [1/10 V]                                      |
| 31    | VLT status word                                                     |
| 32    | Actual output frequency                                             |
| 33    | current line number of the SyncPos program in case of #DEBUG NOSTOP |

### Axis Process Data

| Index | Description                                                                                                                                                                     |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4096  | Current position slave in qc (without conversion to UU) (see APOS)                                                                                                              |
| 4097  | Set position slave in qc (without conversion to UU) (see CPOS)                                                                                                                  |
| 4098  | Last slave index position in qc (without conversion to UU) (see IPOS)                                                                                                           |
| 4099  | Actual velocity in qc/st, where st is the sample time set by _GETVEL.                                                                                                           |
| 4100  | Current velocity Master (as above)                                                                                                                                              |
| 4101  | Current position error in qc                                                                                                                                                    |
| 4102  | Contains the number of revolutions of the encoder after the first overflow of the absolute encoder, provided the graduation per revolution is entered correctly in ENCODER (2). |
| 4103  | As above for the master                                                                                                                                                         |
| 4105  | Current master position without conversion (qc) (see MAPOS)                                                                                                                     |
| 4106  | Last Master index position without conversion (qc) (see MIPOS)                                                                                                                  |
| 4107  | Internal current velocity (ACTPOS – last ACTPOS) (qc/1 ms)                                                                                                                      |
| 4108  | Internal Master velocity (see above)                                                                                                                                            |

|      |                                                                                                                                                           |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4109 | Current frequency of the master simulation (1/ 1000 Hz) (see PULSVEL)                                                                                     |
| 4110 | Determines whether the master simulation is active or not (1 or 0)                                                                                        |
| 4111 | REG_REFERENCE Actual set value which is output by the control through the position controller (between -FFFF and FFFF resp. -1048575 and 1048575 decimal) |

**Axis Process Data, Profile generator value**

| Index              | Description                                               |
|--------------------|-----------------------------------------------------------|
| 4218               | Returns all 32 flags of the profile generator. These are: |
| PG_FLAG_BUSY       | 1L<br>// Flag for busy information                        |
| PG_FLAG_COMMANDERR | 2L<br>// Flag for Command Error occurred (not used)       |
| PG_FLAG_POSREACHED | 4L<br>// Flag for Position reached                        |
| PG_FLAG_INDEX_HIT  | 8L<br>// Flag for Index observed                          |
| PG_FLAG_WRAP_OCC   | 16L<br>// Flag for Wraparound occurred (not used)         |
| PG_FLAG_POS_ERR    | 32L<br>// Flag for Position Error occurred                |
| PG_FLAG_BRKPT_RCHD | 64L<br>// Flag for Breakpoint reached (not used)          |
| PG_FLAG_FLOATING   | 128L<br>// Flag for MOTOR OFF                             |
| PG_FLAG_MOVING     | 1L << 8<br>// Flag for axes is moving                     |
| PG_FLAG_OVERFLOW   | 2L << 8<br>// Flag for overflow of slave position         |
| PG_FLAG_OVERFLOWM  | 4L << 9<br>// Flag for overflow of master position        |
| PG_FLAG_POSFLOAT   | 8L << 8<br>// Flag for floating in pos control            |
| PG_FLAG_INTERNTST  | 64L << 8<br>// Flag for internal use                      |
| PG_FLAG_SYNCREADY  | 1L << 24<br>// Flag for Synchronization ready             |
| PG_FLAG_SYNCFAULT  | 2L << 24<br>// Flag for Synchronization failed            |
| PG_FLAG_SYNCACCUR  | 4L << 24<br>// Flag for Accuracy reached (syn)            |
| PG_FLAG_SYNCOMMHit | 8L << 24<br>// Flag for master marker hit                 |
| PG_FLAG_SYNCSMHIT  | 16L << 24<br>// Flag for slave marker hit                 |
| PG_FLAG_SYNCOMMERR | 32L << 24<br>// Flag for master marker distance exceeded  |
| PG_FLAG_SYNCSMERR  | 64L << 24<br>// Flag for slave marker distance exceeded   |
| PG_FLAG_TESTFLAG   | 128L << 24<br>// Flag for internal use                    |

**Axis Process Data, CAM profile**

| Index | Description                                                                                                                                                  |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4220  | CINDEX Actual index into the curve interpolation area (number of actual interpolation point 0..Intno-1)                                                      |
| 4221  | CVINDEX Actual value index used. Is equal to CINDEX if not starting or stopping is active. In that case the CVINDEX points into the start or stop path data. |
| 4222  | CMAXINDEX Maximum index allowed (Intno - 1)                                                                                                                  |
| 4223  | CIMPS Curve position within the actual interpolation interval (integer part of 64 bit value)                                                                 |
| 4224  | CMILEN Length of interpolation interval in MU units (integer part of 64 bit value)                                                                           |
| 4225  | CWRAP Actual curve counter (0 .. CCOUNTER)                                                                                                                   |
| 4226  | CSSTART Slave start offset of actual curve in qc (for closed curves always 0).                                                                               |
| 4227  | CCOUNTER Demanded value of number of curves to do (last SYNCC command)                                                                                       |
| 4228  | CCURVEPOS Slave curve position in UU units (updated in SETCURVE and when SYNCCxx active).                                                                    |
| 4229  | CSLAVECPOSQ Actual slave curve position in qc (relative to CSSTART)                                                                                          |
| 4230  | CMASTERCPOS Actual master curve position in MU units (initialised in SETCURVE and updated after SYNCCxx).                                                    |
| 4231  | GETCMDVEL Gives command velocity (intgr part * 128) with sign (see also 4186)                                                                                |


**NB!**

The values of the system variables are internal, hardware-dependent values which may change.

**Command group**

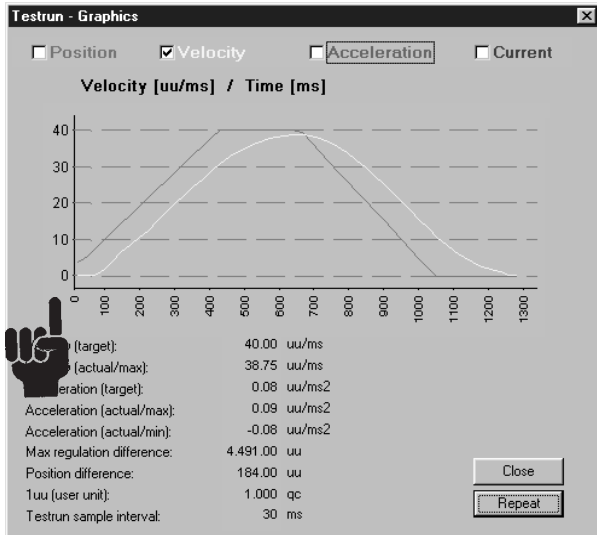
CON

**Cross index**

LINKSYSVAR, TESTSETP

### ■ TESTSETP

As standard you can use the menu "TESTRUN" to trigger a test run which records desired and current position, speed, acceleration and current and whose result can be seen in the test run graphic:



In addition, with the two commands TESTSETP and TESTSTART you can record other or additional parameters, for example you can record the master position. And, in contrast to a test run, you can record this data while executing the program. With TESTSETP you determine the parameters of the recording (which parameters are to be recorded, how often and in which array) and with TESTSTART you then start recording.

#### Summary

Specify recording data for test run

#### Syntax

TESTSETP ms vi1 vi2 vi3 arrayname

#### Parameter

|            |                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------|
| ms         | = interval in milliseconds between two measurements                                                                        |
| vi 1...3   | = indices of the three values to be recorded. The agreements for the system array apply. Three values are always recorded. |
| array name | = Name of the array used for the recording                                                                                 |

#### Array format

The values are stored as follows within the array (all values 4 Byte):

| Designation | Content | Meaning                                        |
|-------------|---------|------------------------------------------------|
| version     | 000     | Version of the data structure                  |
| ms          | 1       | Interval between two measurements in ms        |
| vi1         | i       | Value, recorded at point 1 (Index)             |
| vi2         | i       | Value, recorded at point 2 (Index)             |
| vi3         | i       | Value, recorded at point 3 (Index)             |
| number      | nn      | Specifies how many measurements follow         |
| data        | ...     | Measurement data                               |
| ...         | ...     | (total nn*3)                                   |
| number      | 0-mm    | Number of measurements (if others are present) |
| data        | ...     | (see above)                                    |



#### NB!

Please make sure that the size of the array is sufficient for the recording. You need 6 elements for the header, 1 element for the number, and 3 elements for each measurement. Thus for 100 measurements you need 307 elements.

#### Command group

I/O

#### Cross index

TESTSTART, DIM, SYSVAR, "DISPLAY RECORDING"

#### Syntax-Example

```
DIM tstfahrtarray[307] //Array with 307 elements
TESTSETP 3 4096 4105 4101 tstrunarray
// Record current slave position, current master
// position and current tolerated position error
```

... Start positioning run ...

```
TESTSTART 100 // Start recording
```

### ■ TESTSTART

This command is used to start the recording of a test run with the contents as defined in TESTSETP. The recorded data can then – as required – be graphically represented using "TESTRUN" → "DISPLAY RECORDING". There are four graphics: Position, Speed, Acceleration and Current available for this.

#### Summary

Start the recording of a test run

#### Syntax

TESTSTART no

#### Parameter

no = number of measurements to be carried out



#### NB!

If an array does not have sufficient space for no. measurements, the error O\_ERR 71 „Field limits exceeded“ is triggered.

#### Command group

I/O

#### Cross references

TESTSETP, "DISPLAY RECORDING"

#### Syntax example

```
SYNCP // Synchronization of the position
WAITI 1 ON // When the key is pressed
TESTSTART 200
// Start recording (200 measurements)
```

#### Syntax example

```
NOWAIT ON
// Do not wait until the position is reached
VEL 50
POSA 100000
// Start positioning with velocity 50%
WHILE (APOS<50000) DO
// Wait until position 50000 is reached
ENDWHILE
VEL 100 // Increase velocity to 100%
TESTSTART 200
// Start recording (200 measurements)
DELAY 20// Wait 20 ms
POSA 100000
// Start positioning with new velocity
NOWAIT OFF
// Wait until positioning is finished
```

### ■ TIME

The internal system-time can be read out using the TIME command. The TIME command is most suitable for calculating the execution time of a command sequence or device cycle time.

#### Summary

reads system-time

#### Syntax

res = TIME

#### Return value

res = system-time in milliseconds after switching on



#### NB!

Please note that after counting up to MLONG the value will change to -MLONG.

#### Command group

I/O

#### Syntax-Example

```
PRINT TIME /* print current system-time */
timestop1 = TIME
/* store current entry system-time */
```

#### Program sample

ACC\_01.M, DELAY\_01.M, EXIT\_01.M, GOSUB\_01.M

### ■ TRACKERR

With TRACKERR it is possible to query the actual position error of the axis in User Units (in consideration of the signs). This is the difference between the set value (CPOS) and the actual position (APOS).

#### Summary

queries actual position error of the axis

#### Syntax

res = TRACKERR

#### Return value

res = actual trailing of the axis in UU

#### Command group

I/O

#### Cross Index

APOS, CPOS, POSERR (15)

#### Syntax-Example

```
PRINT TRACKERR
/* query actual position error of the axis */
```

### ■ VEL

The velocity for the next absolute and relative positioning procedures and the maximum allowed velocity for synchronizing procedures is determined with the VEL command.

The value remains valid until a new velocity is set via another VEL command. The new velocity value will be set in reference to the parameters VELMAX (1) and VELRES (22). If the velocity value equals the **Velocity resolution**, then the rpm value set in **Maximum velocity** VELMAX (1) will be used.

Note: Slave velocity in synchronizing mode is also limited by the VEL command.

#### Summary

Set velocity for relative and absolute motion as well as the maximum allowed velocity for synchronizing

#### Syntax

VEL v

$$\text{command velocity (rpm)} = v * \frac{\text{VELMAX (1)}}{\text{VELRES (22)}}$$

#### Parameter

v = scaled velocity value



#### NB!

If no velocity has been set prior to a positioning or synchronizing command, then the value of parameter DFLTVEL (33) will be used.

If the velocity needs to be altered during positioning, it is possible in the NOWAIT ON mode, when the VEL command is followed by another POSA command targeting to the desired position.

The maximum speed allowed can be changed at any time with the command VEL, if a SYNCV, SYNCP or SYNCM follows the VEL command.

#### Command group

REL, ABS

#### Cross Index

ACC, POSA, POSR

Parameter: Maximum velocity VELMAX (1)

#### Syntax-Example

VEL 100 /\* Velocity 100 \*/

#### Program sample

VEL\_01.M

### ■ WAITAX

The WAITAX command has been designated for use with an active NOWAIT mode. By use of this command in NOWAIT ON condition, it is possible to wait for further program processing after a positioning command, until the axis has achieved its set position.

#### Summary

Wait till target position is achieved

#### Syntax

WAITAX

#### Command group

CON

#### Cross Index

NOWAIT ON/OFF, POSA, POSR, AXEND, STAT, WAITI

#### Syntax-Example

WAITAX /\* Wait till the axis has ended motion \*/

WAIT AX /\* Alternative form \*/

#### Program sample

WAIT\_01.M

VEL\_01.M

### ■ WAITI

The WAITI command waits before continuing the processing until the specified input has got the desired signal level.

#### Summary

Wait for defined input condition

#### Syntax

WAITI n s

#### Parameter

n = input number                    1 – 8 or 16 – 33  
 s = expected condition:        ON = High-Signal  
                                               OFF = Low-Signal



#### NB!

If the expected input condition does not occur, then the program will remain 'stuck' at this point. A minimal signal length is required for the sure identification of a signal condition! The VLT manual and the chapter input/output terminals give information about the circuit and technical data for the inputs.

#### Command group

CON

#### Cross Index

ON INT GOSUB, DELAY, WAITT, WAITAX  
 VLT5000 manual

#### Syntax-Example

```
WAITI 4 ON
/* Wait till High level reached input 4 */
WAITI 4 1 /* 3 alternative forms: */
WAIT I 4 ON
WAIT I 4 1
```

```
WAITI 6 OFF
/* Wait till Low level reached input 6 */
WAITI 6 0 /* 3 alternative forms: */
WAIT I 6 OFF
WAIT I 6 0
```

#### Program sample

WAIT\_01.M

### ■ WAITNDX

Waits for the index while checking timeout. The program waits until either the index of the axis is found or the time (timeout) is exceeded.

#### Summary

waits until the next index position is reached

#### Syntax

WAITNDX t

#### Parameter

t = timeout in ms



#### NB!

If the time is exceeded then an error is triggered which can be evaluated with a ON ERROR function.

#### Command group

CON

#### Cross Index

WAITI, WAITP, INDEX

#### Syntax-Example

```
CVEL 1
CSTART
WAITNDX 10000
OUT 1 1
/* Waits a maximum of 10 seconds for the axis to
reach the index position */
```

### ■ WAITP

The WAITP command causes the program to wait until position p is reached.

If, from the speed and the current position, it follows that the point p has already been exceeded then the command is also terminated.

#### Summary

waits until a certain position is reached

#### Syntax

WAITP p

#### Parameter

p = absolute position being waited for



#### **NB!**

Active ON INT or ON PERIOD commands can affect the precision and reproducibility.

#### Command group

CON

#### Cross Index

DELAY, WAITI, WAITAX

#### Syntax-Example

```
NOWAIT ON
POSA 100000
WAITP 50000 /* wait on position 50000 */
OUT 1 1 /* set output 1 */
NOWAIT OFF
```

### ■ WAITT

The WAITT command is suitable for achievement of a defined program time delay. The inputted parameter shows the delay time in milliseconds.

#### Summary

Time delay

#### Syntax

WAITT t

#### Parameter

t = delay time in milliseconds (maximum MLONG)



#### **NB!**

If an interrupt occurs during the delay time, then the entire delay procedure will be re-begun following the processing of the interrupt. The DELAY command is preferable to the WAITT command, because of its constant time behavior.

#### Command group

CON

#### Cross Index

DELAY, WAITI, WAITAX

#### Syntax-Example

```
WAITT 5000 /* wait 5000 seconds */
WAIT T 5000 /* Alternative form: */
```

#### Program sample

WAIT\_01.M

### ■ WHILE .. DO .. ENDWHILE

By using the WHILE. .. ENDWHILE construction you can repeat the enclosed program area one or more times, dependent on any criteria. The loop criteria is made up of one or more comparison operations, and is always monitored at the start of a loop. When a negative result already appears at the first monitoring, this can cause an omission of the commands within the loop, and the program will continue after the ENDWHILE instruction.

#### Summary

Conditional loop with start criteria  
(While condition is fulfilled, repeat ...)

#### Syntax

```
WHILE condition DO
ENDWHILE
```

#### Parameter

condition = abort criteria



#### NB!

Depending on the loop criteria, it can happen that the contents of the loop will never be processed.

To avoid an endless loop, the processed commands within the loop must have a direct or indirect influence on the result of the abort check.

#### Command group

CON

#### Cross Index

LOOP, REPEAT .. UNTIL ..

#### Syntax-Example

```
WHILE (A != 1 AND B == 0) DO
command line 1
command line n
ENDWHILE
```

#### Program sample

```
WHILE_01.M, INKEY_01.M
```

### ■ \_GETVEL

With the \_GETVEL command it is possible to change the sampling time for AVEL and MAVEL. AVEL and MAVEL usually work with a sampling time of 20 ms. With this sampling time the resolution is better. However, a new measurement is only sampled every 20 ms.

The command \_GETVEL lasts exactly as long as the assigned value, e.g.

\_GETVEL 200 ca. 200 ms.

#### Summary

Changes sample time for AVEL and MAVEL

#### Syntax

```
var = _GETVEL t
```

The values are displayed in UU/sec for AVEL or qc/sec for MAVEL.

#### Parameter

t = sample time in msec

#### Command group

I/O

#### Cross Index

AVEL, MAVEL

#### Syntax-Example

```
var = _GETVEL 200
```

Thus, the measurement resolution is considerably better; however changes are only seen after a delay of 200 ms.

### ■ #DEBUG

With introducing the debug mode into the PC user interface, the #DEBUG commands are no longer executed from firmware version

VLT5000/SyncPos Software Version 3.xx/4.2x

VLT5000/FluxSyncPos Software Version 5.xx/4.2x onwards.

Existing #DEBUG commands may not necessarily be removed from the program as they will be ignored.

#### Cross Index

Look up user Interface: "DEVELOPMENT" → "PREPARE SINGLESTEP"

### ■ INCLUDE

The #INCLUDE instruction tells the Compiler to include the contents of the given file in the corresponding program position during the translation of the program.

The INCLUDE instruction is therefore not a genuine command that causes a reaction within the SyncPos option, but an instruction for the translation program – a Compiler Directive.

The #INCLUDE instruction can be placed in any program position, as many times as desired within a program. However, attention must be given to the fact that the data to be included contains commands that may be used in the momentary program position, and that the command syntax is correct. The #INCLUDE instruction is especially suitable for storing frequently used subroutines in separate files and to include them in the application within the SUBMAINPROG ..ENDPROG area.

#### Summary

Inclusion of file contents in the indicated program position

#### Syntax

#INCLUDE file

#### Parameter

file = complete name of the file to be included (path commands inadmissible)



#### **NB!**

The file to be included must be in the current directory. The given data names must end with ".M". The commands within the file to be included must have correct syntax.

#### Command group

CON

#### Syntax-Example

```
#INCLUDE INC_UP01.M
/* Include contents from file INC_UP01.M */
```

#### Program sample

```
INCL_01.M + INCSTA01.M + INCPOS01.M +
INCIN01.M
```

**■ Parameter Reference**
**■ VLT parameter and SyncPos parameter**

Basically there are 2 main parameter types: VLT parameters and SyncPos option parameters. The parameters of VLT parameter group 7.. are both VLT parameters and SyncPos option parameters.

VLT parameters except groups 7.., 8.. and 9.. are described in VLT5000/VLT5000Flux operating instruction, this manual describes changes to the VLT parameters including group 7.. and the SyncPos option parameters.

VLT parameter groups 8.. and 9.. are only available if a field bus option is installed and they are described in the manuals covering these option cards (see field bus option manual).

Factory settings of the parameter

All parameters has a default value called the factory setting. It is possible to reset the parameters to factory setting:

VLT parameters can be reset to factory settings by manual initialization of the VLT or by means of the setup copy function (see in VLT5000/VLT5000Flux operating instruction for further details).

SyncPos option parameters can be reset to factory settings in menu "CONTROLLER" → "RESET" → "PARAMETERS" or "CONTROLLER" → "RESET" → "COMPLETE". Deleting the entire memory in menu "CONTROLLER" → "MEMORY" → "DELETE EEPROM" will reset the parameters to factory setting as well.

User parameter

The parameters 710 though 779 and 795 though 799 are user parameters which can be defined and used in a SyncPos program.

The user parameters are defined in a SyncPos program using the commands LINKXPAR and LINKGPAR, it is possible to specify a text which will be displayed as parameter name in the LCP, a minimum limit, a maximum limit and the way how parameters will be executed:

- 0 = Online executes a changes to the parameter value immediately after the input in display.
- 1 = Offline change means that a change to the parameter value is activated when the OK key is pressed.

An user parameter can be linked to a global option card parameter or an axis option card parameter. There are two types of global parameters:

- Predefined global parameters as for example I\_ERRCLR (107)
- Free internal parameters which can be accessed from the SyncPos program.  
Parameters 130–229 are used for this purpose.

Syntax examples:

```
/* Predefined global parameter */
LINKGPAR I_ERRCLR 710 " " 0 33 0
/* Free internal parameter */
LINKGPAR 132 711 "name" 0 100000 0
/* Axis parameter */
LINKXPAR POSERR 712 "position error" 0
100000 0
```

Parameters 795 – 799 are read only parameters which can be used for data read out on the LCP. These parameters can be read out in display line 1 or 2. You can select which parameters from 795 to 799 are displayed at which position on the VLT display in the VLT parameters 9, 10, 11 and 12:

- User parameter 95 [31]
- User parameter 96 [32]
- User parameter 97 [33]
- User parameter 98 [34]
- User parameter 99 [35]

Parameter changes and storage

VLT parameters changed via the Local Control Panel and SyncPos option parameter changed via menu "CONTROLLER" → "PARAMETERS" → "AXIS" are saved in an EEPROM and thus retained at power down.

VLT parameters changed from the SyncPos application program with the command SETVLT are only stored in RAM and thus lost at power down. SyncPos option parameters changed from the SyncPos application program with the command SET are only active while the application program is running. SyncPos option parameters changed from the application program can be saved in an EEPROM and thus retained at power down by means of the command SAVEPROM or press the OK-key on the VLT display.


**NB!**

Please note that an EEPROM has limited lifetime; but it can be reprogrammed approximately 10000 times.

|     |
|-----|
| 0.. |
| 1.. |
| 2.. |
| 3.. |
| 4.. |
| 5.. |
| 6.. |
| 7.. |
| 8.. |
| 9.. |

**Parameter 009, 010, 011 and 012 (Display line 1 and 2) has got 5 additional selections:**

|                   | VLT5000 | VLT5000Flux |
|-------------------|---------|-------------|
| User parameter 95 | [31]    | [90]        |
| User parameter 96 | [32]    | [91]        |
| User parameter 97 | [33]    | [92]        |
| User parameter 98 | [34]    | [93]        |
| User parameter 99 | [35]    | [94]        |

**Parameter 115 (Slip compensation):** Factory setting changed from 100% to 0%.

**Factory settings changed:**

| From                                   | to                         |
|----------------------------------------|----------------------------|
| Parameter 200 Only clockwise 132Hz [0] | Both directions 132 Hz [1] |
| Parameter 203 Min ... Max [0]          | -Max ... +Max [1]          |
| Parameter 207 Power dependent          | 0,05 sec                   |
| Parameter 208 Power dependent          | 0,05 sec                   |

**Factory settings VLT5000:**

|                                                  |                          |      |
|--------------------------------------------------|--------------------------|------|
| Parameters 300 - 303, 305 - 308, 314 changed to: | "No operation"           | [0]  |
| Parameters 319 + 321 changed to:                 | "Option 0 ... 20mA"      | [91] |
| Parameters 323 + 326 changed to:                 | "Control word bit 11/12" | [33] |

**4 Additional selections in parameters 319+321:**

|                           |      |
|---------------------------|------|
| Option digital            | [90] |
| Option 0 ... 20mA         | [91] |
| Option 4 ... 20mA         | [92] |
| Option 0 ... 32000 pulses | [93] |

**Factory settings VLT5000Flux:**

**Changed factory settings:**

|                                                 |                         |      |
|-------------------------------------------------|-------------------------|------|
| Parameters 300...303, 305...308, 314 changed to | "No operation"          | [0]  |
| Parameters 319 + 321 changed to                 | "Option 0 ... 20mA"     | [90] |
| Parameters 323 + 326 changed to                 | "Control word bit11/12" | [33] |
| Parameters 341 + 355 changed to                 | "Option digital"        | [90] |

**2 Additional selections in parameters 319 + 321:**

|                   |      |
|-------------------|------|
| Option 0 ... 20mA | [90] |
| Option 4 ... 20mA | [91] |

**2 Additional selections in parameters 341 + 355:**

|                     |      |
|---------------------|------|
| Option digital      | [90] |
| Option pulse output | [91] |

Only available if a field bus option is installed (see field bus option manual).

Group 7: next page

Group 7:

| VLT par. #<br>LCP access                                    | Option parameter<br># and name                                                                                    | Parameter Read/Write<br>Application program             |
|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
| 700                                                         | —                                                                                                                 | GETVLT/SETVLT parameter number<br>Ex.: var = GETVLT 700 |
| 701<br>702<br>703<br>704<br>705<br>706<br>707<br>708<br>709 | 102 PRGPAR<br>11 KPROP<br>12 KDER<br>13 KINT<br>21 KILIM<br>35 BANDWIDTH<br>36 FFVEL<br>37 FFACC<br>65 SYNCVFTIME | GET/SET parameter name<br>Ex.: var = GET KPROP          |
| 710-779                                                     | 130 – 229                                                                                                         | GET/SET parameter number<br>Ex.: var = GET 131          |
| 780                                                         | —                                                                                                                 | GETVLT/SETVLT parameter number<br>Ex.: var = GETVLT 780 |
| 795-799                                                     | 130 – 229                                                                                                         | GET/SET parameter number<br>Ex.: var = GET 132          |


**NB!**

In general it is very important to optimize the VLT parameters to the motor, i.e. by using AMA, in order to obtain a good control behavior.

Parameter 205

Parameter 205, maximum reference must be set in accordance with maximum velocity (option parameter #1) before the controller parameters are optimized.

**System Control 700**

When "enable SyncPos" is selected the VLT is controlled by the SyncPos motion controller.

When "disable SyncPos" is selected the VLT is not controlled by the SyncPos PID controller. It is thus possible to control the VLT by traditional inputs or serial interface.

One can select "ENABLE SP. W/O MONI." as third setting and thus avoid Option Error 13 (status error "VLT NOT READY").

---

**Content**

Controlled by the option or not

---

**Value range**

0 ... 2 ★ 0

0 = enable SyncPos

1 = disable SyncPos

2 = enable SyncPos without VLT monitoring

**D.TIME CMP.ACT. 780**

The active inverter dead time compensation which is part of the VLT5000 control algorithm (VCC+) is causing instability at standstill when working in closed loop control. The purpose of this parameter is switching off the active dead time compensation to avoid instability.

Select OFF when the VLT is controlled from the SyncPos option in speed, position or synchronizing control. Select ON if the VLT is operated in open loop speed control. The inverter dead time compensation is providing better control performance in open loop control.

---

**Content**

active inverter dead time compensation

---

**Compatibility**

Not available in VLT5000Flux

---

**Value range**

0 ... 1 ★ 0

0 = OFF

1 = ON

**■ VLT parameter list**

| PNU | Parameter description                                              | Factory settings | Range                        | Changes during Operation | 4-Setup | Con-version index | Data type |
|-----|--------------------------------------------------------------------|------------------|------------------------------|--------------------------|---------|-------------------|-----------|
| 700 | System control                                                     | 0                | 0 - 2                        | Yes                      | No      | 0                 | 6         |
| 701 | Program number                                                     | -1               | -1 - 127                     | Yes                      | No      | 0                 | 4         |
| 702 | PID, Proportional factor                                           | 30               | 0 - 65000                    | Yes                      | No      | 0                 | 4         |
| 703 | PID, Derivative factor                                             | 0                | 0 - 65000                    | Yes                      | No      | 0                 | 4         |
| 704 | PID, Integral factor                                               | 0                | 0 - 65000                    | Yes                      | No      | 0                 | 4         |
| 705 | PID, Integral bandwidth                                            | 1000             | 0 - 1000                     | Yes                      | No      | 0                 | 4         |
| 706 | PID, BANDWIDTH                                                     | 1000             | 0 - 65000                    | Yes                      | No      | 0                 | 4         |
| 707 | PID, Velocity Feed-forward                                         | 0                | 0 - 65000                    | Yes                      | No      | 0                 | 4         |
| 708 | PID, Acceleration Feed-forward                                     | 0                | 0 - 65000                    | Yes                      | No      | 0                 | 4         |
| 709 | PID, Velocity filter                                               | 0                | 0 - 65000                    | Yes                      | No      | 0                 | 4         |
| 710 | User parameter 10                                                  | 0                | Defined by user <sup>1</sup> | Yes                      | No      | 0                 | 4         |
| 711 | User parameter 11                                                  | 0                | Defined by user <sup>1</sup> | Yes                      | No      | 0                 | 4         |
| ... |                                                                    |                  |                              |                          |         |                   |           |
| 778 | User parameter 78                                                  | 0                | Defined by user <sup>1</sup> | Yes                      | No      | 0                 | 4         |
| 779 | User parameter 79                                                  | 0                | Defined by user <sup>1</sup> | Yes                      | No      | 0                 | 4         |
| 780 | Activated dead time compensation<br>(Not available in VLT5000Flux) | OFF              | 0 - 1                        | No                       | No      | 0                 | 6         |
| 795 | User parameter 95 (read only)                                      | 0                | Defined by user <sup>1</sup> | Read only                | No      | 0                 | 4         |
| 796 | User parameter 96 (read only)                                      | 0                | Defined by user <sup>1</sup> | Read only                | No      | 0                 | 4         |
| 797 | User parameter 97 (read only)                                      | 0                | Defined by user <sup>1</sup> | Read only                | No      | 0                 | 4         |
| 798 | User parameter 98 (read only)                                      | 0                | Defined by user <sup>1</sup> | Read only                | No      | 0                 | 4         |

---

**Changes during operation**

"Yes" means that the parameter can be changed, while the VLT frequency converter is in operation. "No" means that the VLT frequency converter must be stopped before a change can be made.

---

**4-Setup**

"Yes" means that the parameter can be programmed individually in each of the four setup's, i.e. the same parameter can have four different data values. "No" means that the data value will be the same in all four setup's.

---

**Conversion index**

The conversion index is used when transferring parameter values with decimals by means of serial communication. Conversion index 0 means that the parameter value is multiplied by 1.

---

**Data type**

Indicates the data type of the parameter value. **Data type 4** is an **Integer 32bit** and **data type 6** is an **Unsigned 16bit**.

---

1) The maximum range is  $-2^{32}$  to  $2^{32}-1$  (-2147483648 to 2147483647) but the actual range is specified when defining a user parameter in the application program using LINKGPARG or LINKAXPAR.

**■ All SyncPos parameters in Alphabetical Order**

For a better overview the parameters for the SyncPos program are divided into groups:

Global Parameters GL

The program language is prepared for multi axis applications. Therefore all parameters independent of an axis such as the **Activated Program Number** PRGPAR (102) and the I/O parameters such as **clear error** I\_ERRCLR (107) are combined in the group global parameters (GL).

Axis parameters AX...

This group contains all axis-specific parameters which can be processed with the GET and SET commands and need axis information in the command:

- AXE Axis parameter encoder
- AXV Axis parameter velocity (velocity and acceleration values)
- AXI Axis parameter I/O (I/O configuration)
- AXH Axis parameter home (parameters affecting home motion)
- AXR Axis parameter control (everything concerning control behavior) and
- AXS Axis parameter synchronization

The column parameter group (AXV, AXR etc.) references to the dialog fields "CONTROLLER" → "PARAMETERS" → "AXIS", resp. → "GLOBAL", where you can define and change the parameters.

| Parameter Code | Parameter Name                                               | VLT- No. | SyncPos Par.No. | Par. Group | Unit                 | Factory setting |
|----------------|--------------------------------------------------------------|----------|-----------------|------------|----------------------|-----------------|
| ACCMAXQC       | Maximum acceleration                                         |          | 10              | AXV        | qc/ms <sup>2</sup> * | 1/65536         |
| BANDWIDTH      | Bandwidth within which the PID filter is active              | 706      | 35              | AXR        | %                    | 1000            |
| DFLTACC        | Default acceleration                                         |          | 34              | AXV        | %                    | 50              |
| DFLTVEL        | Default velocity                                             |          | 33              | AXV        | %                    | 50              |
| ENCODER        | Encoder counts per revolution                                |          | 2               | AXE        | counts/revolution    | 500             |
| ENCODERTYPE    | Encoder type for encoder 0 (Slave)                           |          | 27              | AXE        | –                    | 0               |
| ENDSWMOD       | Behavior at limit switch                                     |          | 44              | AXI        | –                    | 0               |
| ERRCOND        | Behavior after error                                         |          | 43              | AXI        | –                    | 0               |
| ESCCOND        | Behavior after error                                         |          | 70              | AXI        | –                    | 0               |
| FFACC          | Acceleration feed-forward                                    | 708      | 37              | AXR        | %                    | 0               |
| FFVEL          | Velocity feed-forward                                        | 707      | 36              | AXR        | %                    | 0               |
| HOME_FORCE     | Force HOME?                                                  |          | 3               | AXH        | –                    | 0               |
| HOME_OFFSET    | Zero point offset in relation to machine zero point or index |          | 42              | AXH        | qc                   | 0               |
| HOME_RAMP      | Ramp for home motion                                         |          | 41              | AXH        | %                    | 10              |
| HOME_VEL       | Velocity of home motion                                      |          | 7               | AXH        | %                    | 10              |
| HOMETYPE       | Behavior during home motion                                  |          | 40              | AXH        | –                    | 0               |
| I_BREAK        | Input for abort                                              |          | 105             | GLI        | –                    | 0               |
| I_CONTINUE     | Continue program                                             |          | 106             | GLI        | –                    | 0               |
| I_ERRCLR       | Clear error                                                  |          | 107             | GLI        | –                    | 0               |
| I_NEGLIMITSW   | Negative limit switch                                        |          | 47              | AXI        | –                    | 0               |
| I_POSLIMITSW   | Positive limit switch                                        |          | 46              | AXI        | –                    | 0               |
| I_PRGCHOICE    | Input for beginning program choice                           |          | 104             | GLI        | –                    | 0               |
| I_PRGSTART     | Input for program start                                      |          | 103             | GLI        | –                    | 0               |
| I_REFSWITCH    | Input for reference switch                                   |          | 45              | AXI        | –                    | 0               |
| KDER           | Derivative value for PID control                             | 703      | 12              | AXR        | –                    | 1               |

## Programmable SyncPos motion controller

| Parameter Code | Parameter Name                                                      | VLT-SyncPos |         | Par. Group | Unit                       | Factory Setting |
|----------------|---------------------------------------------------------------------|-------------|---------|------------|----------------------------|-----------------|
|                |                                                                     | No.         | Par.No. |            |                            |                 |
| KILIM          | Limit value for integral sum for PID control                        | 705         | 21      | AXR        | –                          | 0               |
| KINT           | Integral value for PID control                                      | 704         | 13      | AXR        | –                          | 0               |
| KPROP          | Proportional value for PID control                                  | 702         | 11      | AXR        | –                          | 30              |
| MENCODER       | Resolution of the encoder 1 (master)                                |             | 30      | AXE        | counts/<br>revolution      | 500             |
| MENCODERTYPE   | Encoder type for encoder 0 (Master)                                 |             | 67      | AXE        | –                          | 0               |
| NEGLIMIT       | Negative software limit switch                                      |             | 4       | AXI        | qc                         | –500000         |
| O_AXMOVE       | Output for motion command active                                    |             | 64      | AXI        | –                          | 0               |
| O_BRAKE        | Output for mechanical brake                                         |             | 48      | AXI        | –                          | 0               |
| O_ERROR        | Output for error                                                    |             | 108     | GLS        | –                          | 0               |
| POSDRCT        | Rotational direction                                                |             | 28      | AXE        | –                          | 1               |
| POSERR         | Maximum tolerated position error                                    |             | 15      | AXR        | qc                         | 20000           |
| POSFAC_T_N     | Denominator user factor                                             |             | 26      | AXE        | –                          | 1000            |
| POSFAC_T_Z     | Numerator user factor                                               |             | 23      | AXE        | –                          | 1000            |
| POSLIMIT       | Positive software limit switch                                      |             | 5       | AXI        | qc                         | 500000          |
| PRGPAR         | Activated program number                                            | 701         | 102     | GLS        | –                          | –1              |
| PROFTIME       | Scan time for profile generator                                     |             | 29      | AXE        | ms                         | 1000            |
| RAMPMIN        | Shortest ramp                                                       |             | 31      | AXV        | msec                       | 1000            |
| RAMPTYPE       | Ramp type                                                           |             | 32      | AXV        |                            | 0               |
| REGWMAX        | Size of the control window (activation)                             |             | 38      | AXR        | qc                         | 0               |
| REGWMIN        | Size of the control window (deactivation)                           |             | 39      | AXR        | qc                         | 0               |
| REVERS         | Reverse behavior for slave                                          |             | 63      | AXR        | –                          | 0               |
| SWNEGLIMACT    | Negative software limit switch active                               |             | 19      | AXI        | –                          | 0               |
| SWPOSLIMACT    | Positive software limit switch active                               |             | 20      | AXI        | –                          | 0               |
| SYNCACCURACY   | Size of the precision window for position synchronization           |             | 55      | AXS        | qc                         | 1000            |
| SYNCFACTM      | Synchronization factor master (M:S)                                 |             | 49      | AXS        | qc                         | 1               |
| SYNCFACTS      | Synchronization factor slave (M:S)                                  |             | 50      | AXS        | qc                         | 1               |
| SYNCFAULT      | Marker number for fault                                             |             | 57      | AXS        | –                          | 10              |
| SYNCMARKM      | Marker number for master                                            |             | 52      | AXS        | –                          | 1               |
| SYNCMARKS      | Marker number for slave                                             |             | 53      | AXS        | –                          | 1               |
| SYNCOMPULSM    | Marker interval for master                                          |             | 58      | AXS        | qc                         | 500             |
| SYNCOMPULSS    | Marker interval for slave                                           |             | 59      | AXS        | qc                         | 500             |
| SYNCOMSTART    | Start behavior for marker synchronization                           |             | 62      | AXS        | –                          | 0               |
| SYNCOMTYPM     | Marker type for master                                              |             | 60      | AXS        | –                          | 0               |
| SYNCOMTYPS     | Marker type for slave                                               |             | 61      | AXS        | –                          | 0               |
| SYNCOMWINM     | Tolerance window for master marker monitoring                       |             | 68      | AXS        | qc                         | 0               |
| SYNCOMWINS     | Tolerance window for slave marker monitoring                        |             | 69      | AXS        | qc                         | 0               |
| SYNCPPOSOFFS   | Position offset for positioning synchronization                     |             | 54      | AXS        | qc                         | 0               |
| SYNCREADY      | Marker number for ready                                             |             | 56      | AXS        | –                          | 1               |
| SYNCSVREL      | Tolerated deviance of the slave drive from the master velocity in % |             | 66      | AXS        | %                          | 0               |
| SYNCSVFTIME    | Velocity filter                                                     | 709         | 65      | AXS        | $\tau_{filt}$ ( $\mu$ sec) | 0               |
| TESTTIM        | Time in target window                                               |             | 24      | AXI        | ms                         | 0               |
| TESTVAL        | Limit value for reading in target window                            |             | 25      | AXI        | qc                         | 1               |
| TESTWIN        | Size of target window                                               |             | 8       | AXI        | qc                         | 0               |
| TIMER          | Sampling time for PID control                                       |             | 14      | AXR        | msec                       | 1               |
| VELMAX         | Maximum velocity                                                    |             | 1       | AXV        | Rpm                        | 3000            |
| VELMAXQC       | Maximum velocity (internal parameter)                               |             | 9       | AXV        | qc/ms *<br>1/65536         |                 |
| VELRES         | Velocity resolution                                                 |             | 22      | AXV        |                            | 100             |

### ■ All SyncPos Parameters in Detail

The sequence of parameters is determined by the internal parameter number. We recommend using the overview as a guide; then you will be able to find detailed information very quickly using the internal parameter number.

#### General Information on the Parameter Values

Some limiting values are listed at 1 billion to make them more easily readable. However, the exact value is 1,073,741,824.



#### **NB!**

The unit defined by the parameter **User factor** POSFACT\_Z (23) and POSFACT\_N (26) is not used for all path and position parameters. For various parameters (for example a software limit switch like I\_POSLIMITSW (46) and the parameter **Tolerated position error** POSERR(15) the quad-count unit (qc) is valid. 4 quad-counts correspond to one sensor unit.

#### Abbreviation

For a description which is not dependent on the version or the hardware, variables were used in the following section for some limit values: thus the value of  
MLONG = 1.073.741.824

#### Factory settings

Each factory setting is marked with ★.

#### Input range

Whether the input range listed is exceeded is not checked by the program, since due to the large domain there is no suitable test possibilities.



#### **NB!**

Even within the areas indicated illogical inputs can result from the large differences in performance of the motors and the wide variety of possible applications. Therefore, it is the responsibility of the programmer and the user to observe the performance ranges of the drive and of the system.

### 1 VELMAX

VELMAX defines the rated speed of the drive. This value is listed in Rpm and is needed for the calculation of ramps and actual velocities.



#### **NB!**

The nominal speed refers to the speed of the encoder.

#### Content

Rated speed of the drive

#### Parameter Group

Axis parameter Velocity AXV

CAM-Editor: index card Velocity

#### Unit

Rpm

#### Value range

1 ... 65535

★ 3000

**2 ENCODER**

The parameter **Encoder count per revolution** contains the position feedback transmitter of the actual encoder (incremental or absolute encoder) in relation to one encoder revolution.

The number of quad-counts (qc) per revolution is calculated from this information. Quadcounts are the basic units for all path measurements.

Quadcounts are generated by extracting of all edges of the A and B tracks. One encoder count corresponds to four quad-counts. In the case of absolute encoders, the absolute values are returned 1 : 1.

The number of quad-counts per revolution is needed for the index pulse search during the reference drive and for the conversion of velocity and acceleration to internal units.

The **Encoder counts per revolution** also supplies information whether during a HOME or INDEX movement the index signal has been missed. If more than a complete revolution is executed without registering an index signal then the corresponding error message will be made.

---

**Content**

Encoder counts per revolution

---

**Parameter Group**

Axis parameter Encoder AXE  
CAM-Editor: index card Encoder

---

**Unit**

counts/revolution

---

**Value range**

1 ... MLONG ★ 500


**NB!**

No negative values are allowed. Whether index pulses are being used or not is entered in the parameter HOMETYPE (40).

---

**Limit values**

In order to guarantee perfect functioning of the SyncPos option the product of the **Encoder counts per revolution** ENCODER (2) and the **Maximum velocity** in encoder revolution/sec of the limit frequency of the encoder input level (220 kHz) may not be exceeded.

$$\text{ENCODER} * \text{VELMAX} \left[ \frac{\text{Encoder revolution}}{\text{sec}} \right] \leq 220 \text{ kHz}$$

**3 HOME\_FORCE**

If this parameter is set to yes = 1, then movement to the home position must be completed before any other positioning movement can be completed.

For a motion command that is not executed with a terminated home run the error O.ERR\_6 is triggered.

---

**Content**

Forced movement to home position

---

**Parameter Group**

Axis parameter Home AXH  
CAM-Editor: index card Home

---

**Value range**

0; 1 ★ 0

0 = Homerun is not forced.

After being turned on the current position is valid as the real index point

1 = Homerun is forced.

After turning on the VLT and after changing axis parameters a forced tracking of the home position must be made before a motion command is executed directly or by the program.

Internally the parameters can also contain the value 255, which indicates that a forced tracking of the HOME position is necessary and has already taken place.


**NB!**

For safety reasons and to avoid false positioning the parameter should always be set to 1 and thus forcing tracking of the home position. However, in this case it is necessary to consider that all programs must complete a HOME command before the first motion command in order to receive perfect functioning.

### 4 NEGLIMIT

NEGLIMIT indicates the **Negative position limit** for all movements. If this value is exceeded then an error is triggered. NEGLIMIT is only active if SWNEGLIMACT (19) has been set. If a positioning command is entered which exceeds the limits set, then it is not executed.

Content

Negative software limit switch

Parameter Group

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

Unit

qc

Value range

0 ... -MLONG ★ -500000



**NB!**

When using the command DEF ORIGIN the path limitation is automatically adapted so that the original position of the positioning range is maintained.

### 5 POSLIMIT

POSLIMIT indicates the **Positive position limit** for all movement. If this value is exceeded then an error is triggered. POSLIMIT is only active if SWPOSLIMACT (20) is set. If a positioning command is entered which exceeds the limits set, then it is not executed.

Content

Positive software limit switch

Parameter Group

Inputs/Outputs

Unit

qc

Value range

0 ... MLONG ★ 500000



**NB!**

When using the command DEF ORIGIN the path limitation is automatically adapted so that the original position of the positioning range is maintained.

### 7 HOME\_VEL

HOME\_VEL determines the **Home velocity**, with which the movement to the reference switch is made. The velocity statement refers to the rated speed and depends on the VELRES (22) parameters. Generally this statement is made in % of the rated speed.

$$\text{Home velocity} = \text{HOME\_VEL} \cdot \frac{\text{VELMAX (1)}}{\text{VELRES (22)}}$$

Content

Velocity for movement to home position

Parameter Group

Axis parameter Home AXH  
CAM-Editor: index card Home

Unit

% (VELRES = 100)

Value range

-VELRES ... VELRES ★ 10

A negative sign means the search will be made in the other direction.



**NB!**

Since the program always searches for the reference switch in the same direction of rotation (depending on sign) this should be set at the limits of the motion area. Only in this manner is it possible to guarantee that the drive actually moves towards the reference switch when moving home and not away from it. In order to maintain a good repeatability of the reference motion no more than 10% of the maximum speed should be used.

**8 TESTWIN**

TESTWIN indicates the size of the target window. A position is only viewed as reached when the difference between the target and the actual position is less than the window. The controller wait to execute the next command until the actual position is within the target window.

If TESTWIN is not active (parameter 0), then the target has been reached if the set position is the target position. However, this does not necessarily correspond with the actual position of the drive.

---

**Content**

Size of the target window

---

**Parameter Group**

Axis parameter Inputs/Outputs AXI  
 CAM-Editor: index card Inp/Outp

---

**Unit**

qc

---

**Value range**

0 ... MLONG ★ 0

0 = inactive

TESTWIN must always be less than TESTVAL.


**NB!**

If the target window surrounding the end position is selected to be too small, the drive could move in a very small area around the end position without reaching the target window. Thus the program would be 'stuck' after the corresponding positioning command.

A target window of 0 deactivates the monitoring of the actual position and only monitors the command position.

**9 VELMAXQC**

VELMAXQC determines the **Maximum velocity**. All statements made with the assistance of VELRES (22) refer to this speed. The sample time is 1 ms.

VELMAXQC is an internal parameter which cannot be changed. It is automatically calculated from VELMAX (1) which the user enters in Rpm. These internal values are decisive for the permitted value range. However, in practice these limit values are of no importance since they exceed the encoder input frequency determined by the hardware by far.

---

**Content**

Maximum velocity

---

**Parameter Group**

Axis parameter velocity AXV  
 CAM-Editor: index card velocity

---

**Unit**

qc/ms \* 1/65536

---

**Value range**

1 ... MLONG

### 10 ACCMAXQC

ACCMAXQC determines the maximum acceleration. This is the amount of time the drive needs with a connected load to achieve the maximum rotation. All other statements made with the assistance of the scaling VELRES (22) refer to this acceleration.

This internal parameter – which cannot be changed – is automatically calculated from RAMPMIN (31), which the user enters in ms. These internal values are decisive for the value range permitted. However, in practice these limit values are of no importance.

#### Content

Maximum acceleration

#### Parameter Group

Velocity

#### Unit

qc/(ms)<sup>2</sup> \* 1/65536

#### Value range

1 ... MLONG



#### NB!

If a starting time is entered which is too short, which under the existing mechanical conditions causes non-achievable acceleration, usually a position error will occur.

### 11 KPROP (702)

The **Proportional factor** KPROP indicates the linear correction factor with which the deviation between the current set and actual position is evaluated and a corresponding correction of the motor torque is made.

Rule of Thumb:

KPROP greater = Drive will become 'stiffer'

KPROP too high = Tendency to overswing

#### Content

Proportional value for PID control

#### Parameter Group

Axis parameter PID-Controller AXR

CAM-Editor: index card PID

#### Value range

1 ... 65000 ★ 30

### 12 KDER (703)

The **Derivative factor** KDER is the correction factor with which the changing speed of a motor position error is evaluated.

The derivative factor works against the tendency to overswing due to a high proportional share and 'dampens' the system. However, if the derivative factor selected is too large this will lead to a 'nervous' drive.

#### Content

Derivative value for PID control

#### Parameter Group

Axis parameter PID-Controller AXR

CAM-Editor: index card PID

#### Value range

0 ... 65000 ★ 0

### 13 KINT (704)

The **Integral Factor** KINT is the weighting factor, with which at time n the sum of all motor position errors are evaluated.

The integral factor of the PID filter causes a corresponding corrective motor torque which increases over time. Through the integral share a static position error is reduced to zero, even if a constant load is affecting the motor.

However, an integral factor which is too large leads to a 'nervous' drive.

#### Content

Integral value for PID control

#### Parameter Group

Axis parameter PID-Controller AXR

CAM-Editor: index card PID

#### Value range

0 ... 65000 ★ 0

### 14 TIMER

The TIMER parameter determines the sampling time of the control algorithm.

For example, increase the value of the factory settings

- for very low pulse frequency, such as from 1 to 2 qc per sampling time. You need at least 10 to 20 qc per sampling time.
- Or for very slow systems with a long dead time. If 1 ms is used here for control, large motors will vibrate.

#### Content

Sampling time for PID control

#### Parameter Group

Axis parameter PID-Controller AXR  
CAM-Editor: index card PID

#### Unit

msec

#### Value range

1... MLONG ★ 1

Accordingly, the value should not be set higher than 1000 (= 1 s). This would be a very slow control.



#### NB!

Note that is has a direct effect on the PID loop e.g. if you double the TIMER the KPROP (11) has twice the effect.

### 15 POSERR

The maximum **Tolerated position error** POSERR defines the tolerance allowed between the current actual position and the calculated command position. If the value defined with POSERR is exceeded then the position control is turned off and a position error is triggered.

#### Content

Maximum **Tolerated position error**

#### Parameter Group

Axis parameter PID-Controller AXR  
CAM-Editor: index card PID

#### Unit

qc

#### Value range

1 ... MLONG ★ 20000

The maximum **Tolerated position error** does not affect the positioning accuracy, but merely determines how precisely the theoretically calculated path of motion must be followed, without an error being triggered.



#### NB!

For safety reasons the maximum **Tolerated position error** selected should not be too large since this could be dangerous for both the machine and its operator.

On the other hand, if the values for the **Tolerated position error** are too small this could result in frequent errors. As a guideline, it is wise to set the quadruple of Encoder counts per revolution. This corresponds to one encoder rotation.

### 19 SWNEGLIMACT

By setting this parameter (to 1) the VLT is informed that the negative software limit switch should be monitored. Then it is checked whether the target position is located outside of the permissible movement range during every movement. In this case an error message is issued and the drive control is switched off.

In the positioning mode this means that the corresponding positioning process is not started and the error can be cleared with the ERRCLR command. In the impulse mode an error can only be recognized after the limit has been exceeded, thus when the error message is issued the drive is already outside of the permissible area of movement. In this case, it is necessary to move the drive, by hand, back to the admissible area, and to erase the error – or in the menu "CONTROLLER" → "PARAMETERS" → "AXIS" to temporarily turn off the corresponding **Software limit switch** and then delete the error.

#### Content

Negative limit enable

#### Parameter Group

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

#### Value range

0 ... 1 ★ 0  
0 = inactive  
1 = active

### 20 SWPOSLIMACT

By setting this parameter (to 1) the VLT is informed that the positive software limit switch is to be monitored. In this case it is checked whether the target position is located outside of the permissible movement range during every movement. If necessary an error message is issued and the drive control is switched off.

In the positioning mode this means that the corresponding positioning process is not started and the error can be cleared with the ERRCLR command.

In the impulse mode an error can only be recognized after the limit has been exceeded, thus when the error message is issued the drive is already outside of the permissible area of movement. In this case, it is necessary to move the drive, by hand, back to the admissible area, and to erase the error – or in the menu "CONTROLLER" → "PARAMETERS" → "AXIS" to temporarily turn off the corresponding **Software limit switch** and then delete the error.

#### Content

Positive software limit switch active

#### Parameter Group

Inputs/Outputs

#### Value range

0 ... 1 ★ 0  
0 = inactive  
1 = active

### 21 KILIM (705)

The **Integration limit** KILIM indicates the maximum value that can be inputted into the PID filter via the integral factor. Thus it can be prevented that due to a high additive error too strong of counter control is used which would cause the system to swing.

#### Content

Limit value for integral sum for PID control

#### Parameter Group

PID-Controller

#### Value range

0 ... 65000 ★ 0  
0 = OFF

### 22 VELRES

The **Velocity resolution** VELRES defines a relative size for the velocity values of the motion commands and parameters.

The information concerning speed and acceleration can then be made in whole numbers in relation to this scaling. The value 100 means that the information in the commands are related to 100, thus in percent.

#### Content

Velocity resolution

#### Parameter Group

Axis parameter velocity AXV

CAM-Editor: index card velocity

#### Value range

1 ... MLONG ★ 100

### 23 POSFACT\_Z

All path information in motion commands are made in user units and are converted to quad-counts internally. By choosing these scaling units correspondingly it is possible to work with any technical measurement unit (for example mm). This factor is a fraction which consists of a numerator and denominator.

$POSFACT\_Z / POSFACT\_N = 1 \text{ UU}$

Scaling determines how many quad-counts make up a user unit. For example, if it is 50375/1000, then one UU corresponds to exactly 50.375 qc.

In CAM-Mode, the parameter is used to fix the unit for the slave drive so that it is possible to work with meaningful units in the CAM-Editor. See sample 2.

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} \text{ qc} = 1 \text{ UU}$$

provided that:

$$\text{Gearing factor} = \frac{\text{Motor revolutions}}{\text{Revolutions on output}}$$

Encoder = Incremental encoder (in the case of absolute encoders, the multiplier 4 is omitted)

Scaling factor = Number of user units UU (qc) that correspond to one revolution at the drive

In addition, it is possible to compress or expand the curves with this factor without having to define new curves each time. The use of numerator and denominator for the gearing factor leads to a very precise result since transmission ratios can be represented as a fraction in virtually all cases.

#### Content

Numerator user factor

or in CAM mode conversion of the units qc into UU

#### Parameter Group

Axis parameter Encoder AXE

CAM-Editor: index card Encoder

#### Value range

1 ... MLONG ★ 1000

The upper limit depends on the maximal number of motion commands used within a program. The following context is valid: max. position (UU) \* POSFACT\_Z < MLONG

#### Example 1: Shaft or spindle

25 motor revolutions result in 1 spindle revolution; gearing factor = 25/1

Encoder resolution (incremental encoder) = 500

Spindle gradient = 1 revolution of the spindle = 5 mm

Scaling factor if we want to work with 1/10 mm resolution = 5 \* 10 = 50

$$\frac{25/1 * 500 * 4}{50} \text{ qc} = \frac{25 * 10 * 4}{1} \text{ qc} = \frac{1000}{1} \text{ qc} = 1 \text{ UU}$$

Numerator user factor [23] = 1000

Denominator user factor [26] = 1

#### Example 2: Cylinder

Gear factor = 5/1

Encoder resolution (incremental encoder) = 500

One revolution of the cylinder is 360 degrees. We want to work with a resolution of 1/10 degrees. This means that we divide one revolution of the cylinder into 3600 units.

Scaling factor = 3600

$$\frac{5/100 * 500 * 4}{3600} \text{ qc} = 1 \text{ UU}$$

$$\frac{5 * 500 * 4}{3600} \text{ qc} = \frac{25}{9} \text{ qc} = 1 \text{ UU} = \frac{POSFACT\_Z (23)}{POSFACT\_N (26)}$$

Numerator user factor [23] = 25

Denominator user factor [26] = 9

### 24 TESTTIM

Once the target window has been reached the position is read twice and compared with the parameter TESTVAL (25). If the result is less than TESTVAL (25), then the position has been reached, otherwise a new reading is taken. TESTTIM indicates the time interval between the measurements.

Content

Measuring time in target window

Parameter Group

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

Unit

ms

Value range

0 ... 10 ★ 0

### 25 TESTVAL

Once the target window has been reached the position is read twice with an interval of TESTTIM (24) and the interval is compared with the **target window limit value** TESTVAL. The result determines whether the position is viewed as having been reached or not.



**NB!**

For longer time intervals it must be taken into consideration that reaching this target position will be delayed by this amount of time in any case.

Content

Limit value for readings in target window

Parameter Group

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

Unit

qc

Value range

1 ... 65535 ★ 1  
TESTVAL must always be set to be greater than TESTWIN.

### 26 POSFACT\_N

All path information in motion commands are made in user units and are converted to quad-counts internally. By choosing these scaling units correspondingly it is possible to work with any technical measurement unit (for example mm). This factor is a fraction which consists of a numerator and denominator.

$$\text{POSFACT\_Z} / \text{POSFACT\_N} = 1 \text{ User unit [UU]}$$

Scaling determines how many quad-counts make up a user unit. For example, if it is 50375/1000, then one UU corresponds to exactly 50.375 qc.

In CAM mode, the parameter is used to determine the unit for the slave drive so that it is possible to work with meaningful units in the CAM-Editor. See prerequisites of the formula and example under POSFACT\_Z (23).

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} \text{ qc} = 1 \text{ UU}$$

In addition, it is possible to compress or expand the curves with this factor without having to define new curves each time.

The use of numerator and denominator for the gearing factor leads to a very precise result since transmission ratios can be represented as a fraction in virtually all cases.

Content

Denominator user factor  
or in CAM mode conversion of the units qc in UU

Parameter Group

Axis parameter Encoder AXE  
CAM-Editor: index card Encoder

Value range

1 ... MLONG ★ 1000

Sample

See POSFACT\_Z (23)

### 27 ENCODERTYPE

This parameter determines the type of encoder: incremental or absolute.

Errors can occur when a linear absolute encoder is used. A possible leap in the position data can be detected with the encoder types "3" and "4" if it is larger than the encoder resolution/2. The correction is made by means of an artificial position value which is calculated from the last velocity. If the error continues for more than 100 read-outs (> 100 ms), there will be no further correction which will then indeed lead to a tolerated position error exceeded. The total number of errors will be saved in an internal variable which can be read out via SYSVAR[16].

#### Content

Type of encoder for slave encoder (encoder 0)

#### Parameter Group

Axis parameter Encoder AXE  
CAM-Editor: index card Encoder

#### Value range

|                                                                                                                                                      |     |
|------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 0 ... 4, 100 ... 104                                                                                                                                 | ★ 0 |
| 0 = Incremental encoder                                                                                                                              |     |
| 1 = Absolute encoder, standard ca. 262 kHz                                                                                                           |     |
| 2 = Absolute encoder, ca. 105 kHz                                                                                                                    |     |
| 3 = absolute encoder without overflow (linear) but with error correction, approx. 262 kHz                                                            |     |
| 4 = absolute encoder without overflow (linear) but with error correction, approx. 105 kHz                                                            |     |
| 100 ... 104 = like 0 ... 4, however, hardware monitoring of the encoder will be activated. Error 92 will be issued in case of open or short circuit. |     |



#### NB!

The following commands cannot be used with absolute encoders: DEF ORIGIN, HOME, INDEX and WAITNDX.



#### NB!

The following commands can only be used with absolute encoders if external markers are used: IPOS, MIPOS.

### 28 POSDRCT

A normal requirement for the axis controller is that a positive reference value brings about a positive change of the position. If this is not the case (e.g. encoder transposed or motor connected improperly), the reference value can be reversed internally by the parameter POSDRCT = 2.

In some applications it is desirable to reverse the direction of rotation in user units. This is achieved by the negative sign in POSDRCT.

Die Richtung der Synchronisation (Verhältnis zum Master) kann durch negativen SYNCFACTM umgedreht werden.

The direction of rotation (relation to master) can be turned by negative SYNCFACTM.

In the case of a synchronization in CAM-Mode, you can determine a positive direction of rotation for the slave with POSDRCT. This is a prerequisite for the CAM functionality.

#### Content

Positive direction of rotation

#### Parameter Group

Axis parameter Encoder AXE  
CAM-Editor: index card Encoder

#### Value range

|                                                                                                                                                                                                                                                                                                                            |     |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| -2 ... 2                                                                                                                                                                                                                                                                                                                   | ★ 1 |
| 1 = No change, i.e. positive reference values produce positive encoder values.                                                                                                                                                                                                                                             |     |
| -1 = The sign of the user unit is reversed. Thus, positive reference values produce positive encoder values which are indicated as negative values, however. This applies to all outputs (APOS, CPOS, ...), all user inputs (POSA, POSR, ...) and all synchronization factors, as well as the velocities (CVEL, HOME_VEL). |     |
| 2 = The sign of the reference value is reversed internally (plus becomes minus and vice versa). This is equal to a reversal of the motor leads, or a transposition of the A and B tracks on the encoder.                                                                                                                   |     |
| -2 = Same as "2", i.e. the sign of the reference value is reversed internally; in addition, the sign of the user unit is negated as in "-1".                                                                                                                                                                               |     |

**PROFTIME - 29**

The Parameter gives the possibility to set the sample time for the profile generator, which is independent of the sample time for the PID controller.

For demanding control tasks in the background (SYNCP, SYNCM, SYNCC), the execution time of the SyncPos program may rise drastically. In such cases, the scan time of the profile generator can be increased to 2000 in order to have more time available for the SyncPos program. Values higher than 2000 provide hardly any benefits.

Content

Scan time for profile generator

Parameter Group

Axis parameter PID-Controller AXR  
CAM-Editor: index card PID

Unit

ms

Value range

1000, 2000 ★ 1000  
1000 = 1 ms  
2000 = 2 ms

**30 MENCODER**

MENCODER indicates the number of counts of the master encoders in pulses counts per revolution.

Content

Resolution of master encoder

Parameter Group

Axis parameter Encoder AXE  
CAM-Editor: index card Encoder

Unit

Encoder counts per revolution

Value range

1 ... MLONG ★ 500

**31 RAMPMIN**

The RAMPMIN parameter determines the shortest ramp (maximum acceleration). It indicates how long the acceleration phase lasts at the very least in order to achieve the rated velocity.


**NB!**

If you work with the SyncPos option card then you should always set the ramps via the option card and not in the VLT. The VLT ramps must always be set to minimum.

Content

Shortest ramp

Parameter Group

Axis parameter Velocity AXV  
CAM-Editor: index card Velocity

Unit

msec

Value range

1 ... 65535 ★ 1000

**32 RAMPTYPE**

RAMPTYPE determines the type of ramp used (trapeze or sinusoidal).

Content

Ramp type

Parameter Group

Axis parameter Velocity AXV  
CAM-Editor: index card Velocity

Value range

0 - 1 ★ 0  
0 = trapeze  
1 = sinusoidal

**33 DFLTVEL**

DFLTVEL indicates the default velocity which is always used when no velocity is defined in the process set. This value refers to the **Velocity resolution** VELRES (22).

---

**Content**

Default velocity

---

**Parameter Group**

Axis parameter Velocity AXV  
 CAM-Editor: index card Velocity

---

**Unit**

$1/V_{ELRES}$ ;  
 Standard =  $1/100 = \%$

---

**Value range**

1 ... VELRES ★ 50

**34 DFLTACC**

DFLTACC indicates the acceleration used when no explicit statements have been made. This statement is made in relation to RAMPMIN (31) and refers to the VELRES (22) parameter.

---

**Content**

Default acceleration

---

**Parameter Group**

Axis parameter Velocity AXV  
 CAM-Editor: index card Velocity

---

**Unit**

$1/V_{ELRES}$ ;  
 Standard =  $1/100 = \%$

---

**Value range**

1 ... VELRES ★ 50

**35 BANDWIDTH (706)**

You can limit the bandwidth in which the PID controller should function, for example to avoid the built-up of a vibration if you are operating a system which could be jeopardized by vibrations. However, then it is necessary to enter considerably higher values for the parameters FFVEL (36) and FFACC (37) in order to achieve the corresponding control. A system adjusted in such a manner is not as dynamic as it could be, but is considerably more stable and tends to experience less uncontrolled vibrations.

---

**Content**

Bandwidth within which the PID filter is active

---

**Parameter Group**

Axis parameter PID-Controller AXR  
 CAM-Editor: index card PID

---

**Unit**

%

The value 1000 means that the PID filter can output the full command value. For a BANDWIDTH of 500 only 50 % of the set value is output. Thus, values less than 1000 limit the P-share accordingly.

---

**Value range**

0 ... 1000 ★ 1000

**36 FFVEL (707)**

When a control has a limited bandwidth then a base velocity must be set so that it can be ruled out that the control will entirely prevent the drive from running due to the limit set. FFVEL indicates the value with which the velocity feed-forward is completed.

When working with a normal PID algorithm the FFVEL must always be the same as the KDER factor in order to achieve typical dampening.

---

**Content**

Velocity feed-forward

---

**Parameter Group**

PID-Controller

---

**Unit**

%

---

**Value range**

0 ... 65000 ★ 0

### 37 FFACC (708)

Set the base velocity whenever you have limited the bandwidth. Thus you will prevent the control from not accelerating at all due to the limit set. FFACC indicates the value with which the acceleration feed-forward is completed. For a normal PID algorithm this value is equal to 0.

Content

Acceleration feed-forward

Parameter Group

Axis parameter PID-Controller AXR  
CAM-Editor: index card PID

Unit

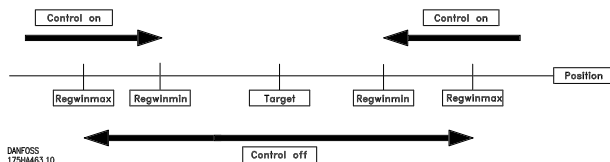
%

Value range

0 ... 65000 ★ 0

### 38 REGWMAX

The parameters REGWMAX and REGWMIN (39) are used to turn the position control within defined areas (control windows) on and off: REGWMAX indicates the size of the window outside of which the control should begin again.



Content

Size of the control window (activation)

Parameter Group

Axis parameter PID-Controller AXR  
CAM-Editor: index card PID

Unit

qc

Value range

0 ... MLONG ★ 0

### 39 REGWMIN

REGWMIN indicates the size of the window inside of which the control is to be deactivated until the REGWMAX (38) control window is reached again.

Content

Size of the control window (deactivation)

Parameter Group

Axis parameter PID-Controller AXR  
CAM-Editor: index card PID

Unit

qc

Value range

0 ... MLONG ★ 0

### 40 HOMETYPE

Determines behavior during movement to home:

- 0 = Moves to reference switch with HOME velocity and direction, then reverses and slowly leaves the switch, subsequently moves to the next index impulse.
- 1 = like 0, but does not search for index impulse
- 2 = like 0 but without reversing, rather continues movement in the same direction out of the switch
- 3 = like 1 but without reversing

Content

Behavior during movement to home

Parameter Group

Axis parameter Home AXH  
CAM-Editor: index card Home

Value range

0 ... 3 ★ 0

### 41 HOME\_RAMP

Acceleration to be used during movement to home position. This statement refers to the minimum ramp, which is defined under the RAMPMIN (31) parameter. This unit results from the parameter VELRES (22) usually in % of the minimal ramp; 50% means half as fast, i.e. twice as long.

The following cohesion for HOME\_RAMP results:

$$\text{HOME\_RAMP [ms]} = \frac{\text{VELRES (22)}}{\text{HOME\_RAMP (41)}} \cdot \text{RAMPMIN (31) [ms]}$$

#### Content

Ramp for home tracking normalized to the unit VELRES (22)

#### Parameter Group

Axis parameter Home AXH  
CAM-Editor: index card Home

#### Unit

VELRES

#### Value range

1 ... 65535 ★ 10



#### NB!

HOME\_RAMP can never have a higher value than DFLTACC (34).

### 42 HOME\_OFFSET

HOME\_OFFSET is used to introduce an offset compared to the reference switch or index pulse. After homing, the drive is positioned to HOME\_OFFSET. At this point the zero point is set.

#### Content

Zero point offset for machine zero point or index

#### Parameter Group

Axis parameter Home AXH  
CAM-Editor: index card Home

#### Unit

qc

#### Value range

-MLONG ... MLONG ★ 0

### 43 ERRCOND

ERRCOND determines the behavior in event of an error as follows:

0 = Standard, i.e. drive moves in COASTING, control loop is interrupted.

1 = like 0, but brake is activated, see O\_BRAKE (48)

2 = motor stop with max. deceleration (stop ramp), subsequently standstill control

3 = like 2, brake is activated in addition, but only after MOTOR STOP.

All other activities such as MOTOR OFF etc. must be set in the ON\_ERROR routine.

#### Content

Behavior in event of an error

#### Parameter Group

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

#### Value range

0 ... 3 ★ 0

### 44 ENDSWMOD

ENDSWMOD indicates how the control resp. the VLT should behave when a limit switch has been reached.

Error behavior see ERRCOND (43)

#### Content

Behavior at limit switch

#### Parameter Group

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

#### Value range

0 – 1 ★ 0

0 = Trigger error

1 = Stop motor with max. deceleration

**45 I\_REFSWITCH**

I\_REFSWITCH determines which input of the option card should serve as reference switch. It is possible to react to a positive or negative edge, using a positive or negative number. Behavior after reaching see HOMETYPE (40).

---

**Content**

Input reference switch

---

**Parameter Group**

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

---

**Value range**

1 ... 8 = Reaction to a positive edge on input 1 ... 8  
0 = no function ★ 0  
-1 ... -8 = Reaction to a negative edge on input 1 ... 8

**46 I\_POSLIMITSW**

I\_POSLIMITSW determines which input should be interpreted as the positive limit switch. It is possible to react to a positive or negative edge, using a positive or negative number.

---

**Content**

Positive limit switch

---

**Parameter Group**

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

---

**Value range**

1 ... 8 = Reaction to a positive edge on input 1 ... 8  
0 = no function ★ 0  
-1 ... -8 = Reaction to a negative edge on input 1 ... 8

**47 I\_NEGLIMITSW**

I\_NEGLIMITSW determines which input should be interpreted as the negative limit switch. It is possible to react to a positive or negative edge, using a positive or negative number.

---

**Content**

Negative limit switch

---

**Parameter Group**

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

---

**Value range**

1 ... 8 = Reaction to a positive edge on input 1 ... 8  
0 = no function ★ 0  
-1 ... -8 = Reaction to a negative edge on input 1 ... 8

### 48 O\_BRAKE

O\_BRAKE indicates the output with which the brake can be activated.

**NEW:** If an output is defined for the brake, this remains active even when the program is terminated with ESC.

O\_BRAKE is activated in the case of an abort or option error if ERRCOND (43) is set to 1 or 3. A positive number means that the output is high (24 V) when the function is active. A negative number means that the output is low (0 V) when the function is active.



**NB!**

The brake output must always be reset by an OUT command in the program.

Content

Output for brake

Parameter Group

Axis parameter Inputs/Outputs AXI  
CAM-Editor: index card Inp/Outp

Value range

-8 ... 8 and  
-14, -11, 11, 14 ★ 0

Example

```
ON ERROR GOSUB err_handle
SET O_BRAKE -1
SET ERRCOND 1
/* Main program loop */
...
SUBPROG err_handle
 WAITI 1
 ERRCLR
 OUT 1 1
RETURN
```

### 49 SYNCFACTM

The synchronization is described with a ratio of qc (Master : Slave); SYNCFACTM determines the synchronization factor for the master.

SYNCFACTM (49) and SYNCFACTS (50) make the compensation of different drive factors possible or the adaptation of the slave speed in relation to the master speed set.

$$\text{Slave velocity} = \text{Master velocity} * \frac{\text{SYNCFACTS (50)}}{\text{SYNCFACTM (49)}}$$

In conjunction with curve synchronization the parameters SYNCFACTM and SYNCFACTS are used to transform qc into MU units. This allows the user to work with meaningful units in the CAM-Editor. See example 2 below.

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} \text{ qc} = 1 \text{ MU}$$

provided that:

$$\text{Gearing factor} = \frac{\text{Motor revolutions}}{\text{Revolutions on output}}$$

Encoder = Incremental encoder (the multiplier 4 is omitted in the case of absolute encoders)

Scaling factor = Number of user units UU (qc) that correspond to one revolution at the drive.

Content

Synchronization factor master (M:S)  
or in CAM mode conversion qc in MU units

Parameter Group

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

Unit

qc

Value range

1 ... MLONG ★ 1  
-MLONG ... -1 = turns the direction of synchronization (ratio to the master)

Example 1

If the master is to run twice as fast as the slave, then the ratio is 2 : 1

SYNCFACTM = 2

SYNCFACTS = 1

Example 2: Conveyor belt

The input should be possible in 1/10 mm resolution.

The drive is connected to the conveyor belt with a gearing of 25:11; this means that the motor makes 25 revolutions and the drive pulley 11.

Gear factor = 25/11

Incremental encoder directly on the master drive; encoder resolution = 4096

The drive pulley has 20 teeth/revolution, 2 teeth correspond to 10 mm; thus, 1 revolution = 100 mm conveyor belt feed.

Thus, the scaling factor is 1000

$$\frac{25/11 * 4096 * 4}{1000} q_c = 1 \text{ MU}$$

$$\frac{25 * 4096 * 4}{1000 * 11} q_c = \frac{2048}{55} q_c = 1 \text{ MU} = \frac{\text{SYNCFACTM (49)}}{\text{SYNCFACTS (50)}}$$

Set the following parameters in order to work with 1/10 degree division

SYNCFACTM = 2048

SYNCFACTS = 55

Example 3: Calculation of the scaling factor for a friction drive

Assume that the output is equipped with a friction wheel (radius 60 mm); we want to work with a resolution of 1/10 mm:

One revolution on the output is thus calculated as follows:

$$\text{Scaling factor} = 2 \Pi r * 10 = 2 \Pi * 60 * 10 = 3969,91$$

$$\text{Scaling factor} = 3970$$

Since an error will occur in any case due to the rounding, a marker adjustment must be performed after each full revolution.

**50 SYNCFACTS**

The synchronization is described with a ratio of  $q_c$  (Master : Slave); SYNCFACTS determines the synchronization factor for the slave.

SYNCFACTM (49) and SYNCFACTS (50) make the compensation of different drive factors possible or the adaptation of the slave speed in relation to the master speed set.

$$\text{Slave velocity} = \text{Master velocity} * \frac{\text{SYNCFACTS (50)}}{\text{SYNCFACTM (49)}}$$

In conjunction with CAM synchronization the parameters SYNCFACTM and SYNCFACTS are used to transform  $q_c$  into MU units. This allows the user to work with meaningful units in the CAM-Editor. See example 2 below.

See prerequisites of the formula and example under SYNCFACTM (49).

$$\frac{\text{Gearing factor} * \text{Encoder resolution} * 4}{\text{Scaling factor}} q_c = 1 \text{ MU}$$

Content

Synchronization factor slave (M:S)

or in CAM mode conversion  $q_c$  in MU units

Parameter Group

Axis parameter Synchronization AXS

CAM-Editor: index card Synchron.

Unit

$q_c$

Value range

1 ... MLONG/max. master velocity ★ 1

Example

See SYNCFACTM (49)

**52 SYNCMARKM**

SYNCMARKM and SYNCMARKS must be set according to the ratio between the number of marker signals from master and slave.

A ration of 1:1 means that each slave marker will be aligned with each master marker. A ratio of 2:1 means that each slave marker will be aligned with each second master marker.

---

**Content**

Marker number of the master

---

**Parameter Group**

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

---

**Value range**

0 ... 65535 ★ 1

**53 SYNCMARKS**

SYNCMARKM and SYNCMARKS must be set according to the ratio between the number of marker signals from master and slave.

A ration of 1:1 means that each slave marker will be aligned with each master marker. A ratio of 2:1 means that each slave marker will be aligned with each second master marker.

---

**Content**

Marker number of the slave

---

**Parameter Group**

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

---

**Value range**

0 ... 65535 ★ 1

---

**Example**

The master marker is an external signal which reports when a transport article arrives; the corresponding slave marker is the index impulse of the motor. If the motor always requires 3 rotations until the article arrives, then this means that 3 index impulses must elapse before a marker comes. Thus, this results in a ratio of 3 : 1; only every 3rd slave pulse is evaluated.

**54 SYNCPOSOFFS**

Defines the offset for position synchronization (SYNCP, SYNCP). The offset is also valid for position synchronization with marker correction.

This position offset can be altered online at any time during the synchronization with a command.


**NB!**

The offset will be executed immediately when the command SYNCP follows.

**NEW**

When SYNCP is started, however, the system waits for the first evaluation of the marker pulses. Only then is the offset applied.

---

**Compatibility**

To avoid compatibility problems you should determine the start-up behavior of SYNCP with SYNCPSTART.

---

**Content**

Position offset for position synchronization

---

**Parameter Group**

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

---

**Unit**

qc

---

**Value range**

-MLONG ... MLONG ★ 0

**55 SYNCACCURACY**

Defines how large the difference between the actual master and slave position can be during a position synchronization (SYNCP and SYNCM), so that the required accuracy is still fulfilled. In contrast SYNCERR provides the actual synchronization error of the slave in user units. In the program you can query whether SYNCACCURACY will be fulfilled using SYNCSTAT. SYNCACCURACY is important for the marker synchronization in order to be able to report READY, since otherwise SYNCERR would have to be queried and compared beforehand.

---

**Content**

Size of the accuracy window for position synchronization

---

**Parameter Group**

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

---

**Unit**

qc  
or in CAM mode in UU

---

**Value range**

|                  |                                                                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -MLONG ... MLONG | ★ 1000                                                                                                                                                             |
| 0 ... MLONG      | A plus sign supplies the absolute value to SYNCERR.                                                                                                                |
| -MLONG ... -1    | A minus sign supplies the synchronization error to SYNCERR with polarity sign. It is then possible to tell whether the synchronization is running ahead or behind. |

**56 SYNCREADY**

SYNCREADY defines how often during a marker synchronization (SYNCP and SYNCCMM) a synchronization evaluation with ACCURACY must be completed with accuracy so that ready is fulfilled. ACCURACY is checked during every correction. If ACCURACY is fulfilled then 1 is added until the set marker number has been achieved. Synchronization evaluation is always executed after n marker pulses by the master SYNCMARKM (52). ACCURACY and READY can be queried using SYNCSTAT.

---

**Content**

Number of markers for READY

---

**Parameter Group**

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

---

**Value range**

0 ... 65535 ★ 1

**57 SYNCFAULT**

Defines how often during a marker synchronization (SYNCP and SYNCCMM) an inaccuracy may occur during a synchronization evaluation before a FAULT is registered. In the program this condition can be queried using SYNCSTAT.

---

**Content**

Marker number for fault

---

**Parameter Group**

Axis parameter synchronization AXS

---

**Value range**

0 ... 65535 ★ 10

**58 SYNCMPULSM**

SYNCMPULSM indicates how many qc (master) lie between two master markers.

If you use the encoder index impulse as a marker signal, then the distance between two markers is the resolution (qc) of the encoder.

If external marker signals are used, then it is possible to measure the marker distance with the program "marker count" (refer to program sample in Chapter 7), if it is unknown.

SYNCMPULSM is only valid for synchronization with marker correction (and SYNCCMM).

In a CAM synchronization, the distance of the sensor to the working position in MU will be indicated instead of the distance between two master markers. (The distance is brought about automatically by the master cycle length (Mt))

If the parameter is larger than one master cycle length (Mt), a marker-FIFO register will be created automatically for the handling of the marker correction.

---

**Content**

Distance between two master markers or in CAM-Mode the distance between sensor and working position

---

**Parameter Group**

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

---

**Unit**

qc  
in CAM mode: MU

---

**Value range**

0 ... MLONG ★ 500

**59 SYNCMPULSS**

SYNCMPULSS defines how many qc (slave) lie between two markers (slave) or in CAM-Mode the distance of the sensor to the working position in UU.

SYNCMPULSS is only valid for synchronization with slave marker correction (SYNCM and SYNCCMS).

---

**Content**

Marker interval slave (marker distance slave) or in CAM-Mode the distance of the sensor to the working position in UU.

---

**Parameter Group**

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

---

**Unit**

qc  
in CAM mode: UU

---

**Value range**

0 ... MLONG ★ 500

**60 SYNCMTYPM**

Defines the signal resp. the marker type for the master: Index pulse of the encoder or external marker.

SYNCMTYPM is only valid for synchronization with marker correction (SYNCM and SYNCCMM) or if you want to use the command MIPOS in your program.

Master marker signal: Input 5.

Content

Marker type for master

Parameter Group

Axis parameter Synchronization AXS

CAM-Editor: index card Synchron.

Value range

- |                                      |     |
|--------------------------------------|-----|
| 0 ... 3                              | ★ 0 |
| 0 = index pulse (positive flank)     |     |
| 1 = index pulse (negative flank)     |     |
| 2 = external marker (positive flank) |     |
| 3 = external marker (negative flank) |     |

**61 SYNCMTYPS**

Defines the signal resp. the marker type for the slave: Index pulse of the encoder or external marker.

SYNCMTYPS is only valid for marker synchronizations (SYNCM).

Slave marker signal: Input 6

Content

Marker type slave

Parameter Group

AXS

Value range

- |                                      |     |
|--------------------------------------|-----|
| 0 ... 3                              | ★ 0 |
| 0 = index pulse (positive flank)     |     |
| 1 = index pulse (negative flank)     |     |
| 2 = external marker (positive flank) |     |
| 3 = external marker (negative flank) |     |

**62 SYNCMSTART**

SYNCMSTART defines whether at start the synchronization should be made to the leading, subsequent or closest marker impulse of the master. SYNCMSTART is only valid for synchronization with marker correction (SYNCM and SYNCCMM).

Content

Start behavior for synchronization with marker correction

Parameter Group

Axis parameter Synchronization AXS

CAM-Editor: index card Synchron.

Value range

- |                                                                                                                                                                                    |     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 0 ... 6 and 1000 ... 1006, 2000                                                                                                                                                    | ★ 0 |
| 0 = The slave marker following the first master marker (after SYNCM) is aligned with the first master marker.                                                                      |     |
| 1 = The first slave marker (after SYNCM) is aligned with the following master marker.                                                                                              |     |
| 2 = After reaching the master velocity the next 2 markers will be aligned (correction can be forward or backward)                                                                  |     |
| 3 = After reaching the master velocity the next slave marker will be aligned with the master marker in front (correction is forward)                                               |     |
| 4 = After reaching the master velocity the next slave marker will be aligned with the marker behind (correction is backward)                                                       |     |
| 5 = After reaching the master velocity the next slave marker will be aligned with the closest master marker (correction can be forward or backward, always the shortest distance). |     |
| 6 = After the command SYNCM the first two markers are taken and the program synchronizes to these markers.                                                                         |     |
| 1000 ... 1006 = as above, an offset (SYNCPPOSOFFS) is not active before the first marker correction ist done.                                                                      |     |
| 2000 = in CAM mode: Counting of the master pulses in MU begins with the master marker.                                                                                             |     |


**NB!**

Only the parameter 2000 is effective in curve synchronizations.

**63 REVERS**

REVERS determines the behavior while moving in reverse (moving in a negative direction): whether reverse is allowed, only allowed when the master is reversed or not allowed in general.

REVERSE is always valid even for the positioning errors VEL and "TESTRUN".

In order to prevent automatic reversing during the "TESTRUN" adjust the value to 1 or 2.

---

**Content**

Reverse behavior for the slave

---

**Parameter Group**

Axis parameter PID-Controller AXR

CAM-Editor: index card PID

---

**Value range**

- |                       |                                                                          |
|-----------------------|--------------------------------------------------------------------------|
| 0 ... 2 and 10 ... 12 | ★ 0                                                                      |
| 0                     | = reverse always allowed, reference $\pm 10$ V                           |
| 1                     | = reverse only allowed when the master is reversed, reference $\pm 10$ V |
| 2                     | = reverse not allowed, reference $\pm 10$ V                              |

**64 O\_AXMOVE**

Set the output number which must be controlled by the AXMOVE function. The output is always activated as soon as a motion command is active, regardless in which mode (position, velocity or synchronization command).

This function is not suitable for monitoring the motor, since the motor could be standing still although the control is in motion.

It is possible to react to a positive or negative edge.

A positive number means that the output is high (24 V) when the function is active. A negative number means that the output is low (0 V) when the function is active.

---

**Content**

Output for motion command active

---

**Parameter Group**

Axis parameter Inputs/Outputs AXI

CAM-Editor: index card Inp/Outp

---

**Value range**

- |          |     |
|----------|-----|
| -8 ... 8 | ★ 0 |
|----------|-----|

**65 SYNCVFTIME (709)**

This parameter configures the velocity filter which is used for the velocity synchronization. Since the velocity synchronization only uses the currently active master velocity and the values can decrease to very small values (e.g. 2 qc/ms) a small fluctuation in velocity can have dramatic effects. In order to even this out the following filter function is applied:

$$\text{Cmdvel} = \text{Old\_Cmdvel} + (\text{Actvel} - \text{Old\_Cmdvel}) * \text{ms} / \tau_{\text{filt}}$$

With the following:

- Cmdvel = set velocity
- Old\_Cmdvel = last set velocity
- Actvel = actual velocity of the master
- ms = sampling time (fixed 1ms)
- $\tau_{\text{filt}}$  = filter time constants

Generally the value for  $t_{\text{filt}}$  is taken from a table, depending on the Encoder counts per revolution of the master. This value can be overwritten by the SYNCVFTIME parameter and is always used when SYNCVFTIME is not equal zero.

If the speed filter is defined with a negative number, the corresponding value also applies for angle/position synchronization SYNCP and for marker correction SYNCM.

In this case filtering takes place as described above, but the errors made are summed up. This error sum is taken into the calculation with  $1000/(\tau * 10)$  in each case, so that no position deviation can occur over prolonged periods.

The value returned by SYNCERR always contains the error made so that this is also used for the evaluation of the synchronicity. In the case of marker correction the correction value is balanced more slowly and with the same factor as the error sums. If, for example, a filter factor of -100000 (100 ms) is used, a marker correction is balanced within 1 sec. (100 ms \* 10). This allows a "taming" of the synchronization without restricting the acceleration.

---

**Content**

velocity filter

---

**Syntax**

SET SYNCVFTIME value  
 value = filter time constants

---

**Parameter Group**

Axis parameter Synchronization AXS  
 CAM-Editor: index card Synchron.

---

**Unit**

$\tau_{\text{filt}}$  ( $\mu\text{sec}$ )

---

**Value range**

-MLONG ... MLONG ★ 0  
 -999 ... 999 = Standard table

---

**Standard table**

| Encoder resolution | $t_{\text{filt}}$ (msec) |
|--------------------|--------------------------|
| 250                | 39500                    |
| 256                | 38600                    |
| 500                | 19500                    |
| 512                | 19000                    |
| 1000               | 9500                     |
| 1024               | 9300                     |
| 2000               | 4500                     |
| 2048               | 4400                     |
| 2500               | 3500                     |
| 4096               | 1900                     |
| 5000               | 1400                     |

**66 SYNCVELREL**

This parameter indicates by how many percent the slave drive can deviate from the velocity of the master while attempting re-synchronization. For example, during changes in SYNCPOSOFFS (54) or at the start of synchronization or during the correction of deviation for marker evaluation. The following is valid:

If the slave need to catch up it runs with the maximum speed allowed; this is either the speed set with

VEL or the master velocity calculated with

$MAVEL + MAVEL * SYNCVELREL/100$  depending which of the two is less. (MAVEL is the actual master velocity).

If the slave needs to slow down and wait for the master it will run with at least the following speed  $MAVEL - MAVEL * SYNCVELREL/100$ .

That means, if SYNCVELREL is 50, for example, the slave will not run slower than MAVEL/2.

Content

tolerated deviance of the slave drive from the master velocity in %

Syntax

set syncvelrel value  
value = percent value

Parameter Group

Axis parameter Synchronization AXS  
CAM-Editor: index card Synchron.

Unit

%

Value range

0 ... MLONG ★ 0  
0 = OFF, i.e. no restriction

**67 MENCODERTYPE**

This parameter determines the type of master encoder: incremental or absolute encoder.

Errors can occur when a linear absolute encoder is used. A possible jump in the position data can be detected with the encoder types "3" and "4", as long as it is larger than the encoder resolution/2. The correction is made by means of an artificial position value which is calculated from the last velocity. If the disruption is present for more than 100 read-outs (> 100 ms), no further corrections are made, which will then in fact lead to a tolerated position error exceeded.

The total number of disruptions that have occurred will be stored in an internal variable which can be read out via SYSVAR[16].

It is possible to simulate a master by means of APOSS command with the encoder type "6", for example when the master position is read via the bus. The simulated master positions are set and read with the system variable SYSVAR[4105].

Content

Type of encoder for master encoder

Parameter Group

Axis parameter encoder AXE  
CAM-Editor: index card encoder

Value range

0 ... 2 and 6 and 100 ... 102 ★ 0  
0 = Incremental encoder  
1 = Absolute encoder, standard ca. 262 kHz  
2 = Absolute encoder, ca. 105 kHz  
6 = The master position is not read by the encoder; instead, it is set with the system variable SYSVAR[4105].  
100 ... 102 = like 0 ... 2, however, hardware monitoring of the encoder will be activated. Error 92 will be issued in case of open or short circuit.


**NB!**

With absolute encoders the following commands can not be used: DEF ORIGIN, HOME, INDEX and WAITNDX.


**NB!**

The following commands can only be used with absolute encoders if external markers are used: IPOS, MIPOS

### 68 SYNCMWINM

The **Marker Window Master** SYNCMWINM shows how large the permitted tolerance for the occurrence of the markers is.

With the works setting "0" the window is not monitored, which means that it is always synchronized to the next marker even if this has a considerably larger interval.

At every other setting only those markers are accepted which are within the window. If there is no marker within the tolerance window the corresponding flag (SYNCSTAT) is set and no marker correction takes place. The corresponding other marker is also ignored and only corrected the next time – i.e. no catching up to the next marker.

When SYNCM (or SYNCCSTART) is started the monitoring only begins when the first marker has been found.

#### Content

Tolerance window for marker monitoring

#### Parameter Group

Axis parameter Synchronization AXS

CAM-Editor: index card Synchron.

#### Unit

qc

or in CAM mode: MU

#### Value range

0 ... MLONG or. max. Marker interval

SYNCPULSM (58) ★ 0

0 = The window is not monitored.

1 ... MLONG = Only one marker is accepted within the window. If no marker is within the tolerance window, the corresponding flag (SYNCSTAT) is set and no marker correction carried out. This flag can be reset with an interrupt (ON STATBIT).



#### NB!

Changes of the parameter will become active immediately – not only after the next SYNCM command.

#### Sample

Marker interval SYNCPULSM = 30000

Tolerance window SYNCMWINM = 1000

Only one marker within an interval of 29000 to 31000 is accepted.

### 69 SYNCMWINS

The **Marker Window Slave** SYNCMWINS shows how large the permitted tolerance for the occurrence of the markers is.

With the works setting „0“ the window is not monitored, which means that it is always synchronized to the next marker even if this has a considerably larger interval.

At every other setting only those markers are accepted which are within the window. If there is no marker within the tolerance window the corresponding flag (SYNCSTAT) is set and no marker correction takes place. The corresponding other marker is also ignored and only corrected the next time – i.e. no catching up to the next marker.

When SYNCM (or SYNCCSTART) is started the monitoring only begins when the first marker has been found.

#### Content

Tolerance window for slave marker monitoring

#### Parameter Group

Axis parameter Synchronization AXS

CAM-Editor: index card Synchron.

#### Unit

qc

or in CAM mode: UU

#### Value range

0 ... MLONG or. max. Marker interval

SYNCPULSS (59) ★ 0

0 = The window is not monitored.

1 ... MLONG = Only one marker is accepted within the window. If no marker is within the tolerance window, the corresponding flag (SYNCSTAT) is set and no marker correction carried out. This flag can be reset with an interrupt (ON STATBIT).



#### NB!

Changes of the parameter will become active immediately – not only after the next SYNCM command.

### 70 ESCCOND

With ESCCOND you can determine how the VLT will react to a program termination using ESC.

#### Content

Condition on program termination

#### Parameter Group

Axis parameters Inputs/Outputs AXI

CAM-Editor: index card Inp/Outp

#### Value range

0 ... 2 ★ 0

0 = The motor is stopped with maximum deceleration, the brake is activated (if defined), the master simulation is stopped. The outputs remain in the current status.

1 = As 0, but all outputs including the VLT5000 outputs 11, 14, 42 and 45 (resp. the VLT5000Flux outputs 26 and 46) are set at "0".

**Exception:** The brake output – if defined – is always activated.

2 = As "0", but all outputs including the VLT5000 outputs 11, 14, 42 and 45 (resp. the VLT5000Flux outputs 26 and 46) are set at "1".

**Exception:** The brake output – if defined – is always activated

### 102 PRGPAR (701)

With PRGPAR it is possible to set which program should be started after the conclusion of a program executed via autostart (auto identification).

This parameter can also be changed and stored with other programs or via the display.

If the program parameters are set within a program with the command SET, for example

SET PRGPAR 5 then after the program is run program no. 5 will be started. Thus it is possible to start other programs from an SyncPos program and link several programs together.

If no program number is activated and no input for the program start I\_PRGSTART (103) is set, then the program with auto identification will be started.



#### NB!

If no autostart program is defined then it is not possible to start a program via PRGPAR (102); this always requires a terminated autostart program.

#### Content

Activated program number

#### Parameter Group

Global parameters GLS

#### Value range

-1 ... 127 (optionally + 1000) ★ -1

-1 = program number is not activated, i.e. no program to start after autoexec

0...127 = activated program number is started after power up (and autoexec)

0...127 +1000 = activated program number is started, but at power on the motor is turned off (motor off)

-1 + 1000 = (999) same as -1, but with motor disabled after power up

#### Portability

Parameter value +1000 with option card version 5.00 onwards.

#### Example

```
SET PRGPAR 5 // program number 5 is started
 // immediately after power up
 // and after execution of autoexec
```

```
SET PRGPAR 1005
 // program number 5 is started,
 // but at power on the motor is off
```

**103 I\_PRGSTART**

If the input for I\_PRGSTART  $\neq$  0, then the autostart program is run first and then the program waits until input I\_PRGSTART comes. This is then evaluated relevant to the program selection I\_PRGCHOICE (104) in order to determine the number of the program to be run.

If no input for the program start I\_PRGSTART is set, then the program with auto identification will be started.

---

**Content**

Input for program start

---

**Parameter Group**

Global parameters GLI

---

**Value range**

0 ... 8 ★ 0

**104 I\_PRGCHOICE**

If I\_PRGCHOICE  $>$  0 then this parameter indicates the input number starting at which the inputs for the **Program selection** are used. This includes all numbers up to I\_PRGSTART (103).

Example:

If I\_PRGCHOICE = 3 and I\_PRGSTART = 7, then upon activation of input 7 the inputs 3, 4, 5, and 6 will be evaluated binary and the result will be used as a program number.

| Input                     | Level | Binary value   |
|---------------------------|-------|----------------|
| 3                         | low   | 0              |
| 4                         | high  | 2              |
| 5                         | high  | 2 <sup>2</sup> |
| 6                         | low   | 0              |
| => program to be started: |       | 6              |

Thus it is possible to choose between a maximum of 128 programs, identified with the numerals 0 to 127.

---

**Content**

Input for program selections start number

---

**Parameter Group**

Global parameters GLI

---

**Value range**

0 ... 8 ★ 0

**105 I\_BREAK**

If this input is defined, then when this input is activated the program running will be aborted immediately. Such a program can be continued with CONTINUE. It is possible to react to a positive or negative edge, using a negative or positive number.

---

**Content**

Input for abort

---

**Parameter Group**

Global parameters GLI

---

**Value range**

1 ... 8 = Reaction to a positive edge on input 1 ... 8  
 0 = no function ★ 0  
 -1 ... -8 = Reaction to a negative edge on input 1 ... 8

**106 I\_CONTINUE**

I\_CONTINUE determines which input is used to continue aborted programs. It is possible to react to a positive or negative edge, using a negative or positive number.

---

**Content**

Continue program

---

**Parameter Group**

Global parameters GLI

---

**Value range**

1 ... 8 = Reaction to a positive edge on input 1 ... 8  
 0 = no function ★ 0  
 -1 ... -8 = Reaction to a negative edge on input 1 ... 8

**107 I\_ERRCLR**

I\_ERRCLR determines which input is used to clear an error. It is possible to react to a positive or negative edge, using a negative or positive number.

---

**Content**

Clear error

---

**Parameter Group**

Global parameters GLI

---

**Value range**

|           |                                                   |     |
|-----------|---------------------------------------------------|-----|
| 1 ... 8   | = Reaction to a positive edge<br>on input 1 ... 8 |     |
| 0         | = no function                                     | ★ 0 |
| -1 ... -8 | = Reaction to a negative edge<br>on input 1 ... 8 |     |

**108 O\_ERROR**

The output defined by O\_ERROR is set when an option error has occurred. When the error is cleared this output is reset.

A positive number means that the output is high (24 V) when the function is active. A negative number means that the output is low (0 V) when the function is active.

---

**Content**

Output for error

---

**Parameter Group**

Global parameters GLI

---

**Value range**

|          |  |     |
|----------|--|-----|
| -8 ... 8 |  | ★ 0 |
|----------|--|-----|


**NB!**

The setting of the O\_ERROR parameter does not influence the use of the OUT and OUTB commands. With these commands it is also possible to change the outputs which have predefined functions.



**Chapter 6**

■ **Messages and Error Reference**

VLT and SyncPos messages ..... 200  
SyncPos messages in detail ..... 201  
VLT software SyncPos messages (PC software) ... 205

**■ VLT and SyncPos Motion Controller messages**

All messages are shown in the LCP display of the VLT in short and in the SyncPos software in plain text.

You can find brief information on the error messages in the table or detailed information in the following section.

**■ Table of Messages**

The tables contain the messages in numerical order. Letters following a % sign represent variables which can be used in plain text at the corresponding locations.

| O.ERR_ | LCP-Display          | SyncPos message                            |
|--------|----------------------|--------------------------------------------|
| 3      | AXES NOT IN SYSTEM   | Axis no. %bu not in system                 |
| 5      | ERROR NOT CLEARED    | Axis no: 1 Error not cleared               |
| 6      | HOME NOT EXECUTED    | Axis no: 1 Failed to move to HOME position |
| 7      | HOME_VEL ZERO        | Axis no: 1 Home vel 0                      |
| 8      | POSITION ERROR       | Axis no: 1 Position error                  |
| 9      | INDEX NOT FOUND      | Axis no: 1 Index pulse (encoder) not found |
| 10     | UNKNOWN COMMAND      | Unknown command                            |
| 11     | SOFTWARE LIMIT ACT.  | Axis no.1 Software limit switch activated  |
| 12     | ILLEGAL PARAMET. NO  | Illegal parameter number                   |
| 13     | General VLT error    | General VLT error                          |
| 14     | TOO MANY NEST. LOOP  | Too many nested loops                      |
| 16     | PARAM. ERROR EEPROM  | Parameters in EEPROM are corrupted         |
| 17     | PROGR. ERROR EEPROM  | Programs in EEPROM are corrupted           |
| 18     | RESET BY CPU         | Reset by CPU                               |
| 19     | USER ABORT           | User abort                                 |
| 25     | LIMIT SWITCH ACTIV.  | Axis no. 1 Limit switch activated          |
| 49     | TOO MANY INTERRUPTS  | Too many interrupt functions               |
| 51     | TOO MANY NEST. GOSUB | Too many nested GOSUB commands             |
| 52     | TOO MANY RETURN      | Too many RETURN commands                   |
| 62     | ERROR IN VERIFYING   | EEPROM: address % defect                   |
| 70     | ERROR IN DIM COMMD   | Error in DIM command                       |
| 71     | ARRAYBOUNDS CROSSED  | Attempt was made to cross arraybounds      |
| 79     | TIMEOUT WAITNDX      | Timeout while waiting for index            |
| 84     | TOO MANY ONTIME      | Too many time interrupts                   |
| 87     | OUT OF MEMORY (VAR)  | No more room for variables                 |
| 90     | MEMORY locked        | The program memory is write-protected.     |
| 91     | ILLEGAL CURVE ARRAY  | Curve array wrong                          |
| 92     | ENCODER ERROR        | Error from encoder monitoring              |
| xx     | INTERNAL ERROR       | Internal error ##                          |

**■ SyncPos option messages in detail**

As in the table above, the messages are in numerical order. Here you can find additional information about possible causes and tips on troubleshooting.

**O.ERR\_3**

SyncPos

---

Axis no. %bu not in system

Cause

---

An attempt has been made to find another axis which does not exist in the controller. (The program allows for input of axis numbers greater than 1 since it is configured for multi-axes applications.)

Tip

---

Check to see if the program axis command has an invalid number or a general axis command (...X(\*)).

**O.ERR\_5**

SyncPos

---

Axis no: 1 - Error not cleared

Cause

---

An attempt has been made to execute a motion command, although a momentary error message has not been cleared.

**O.ERR\_6**

SyncPos

---

Axis no: 1 - Failed to move to HOME position

Cause

---

According to the axis parameter HOME\_FORCE (3), a forced move to the machine zero-point is demanded, before other motion commands can be executed. This move to the machine zero-point has not been executed.

**O.ERR\_7**

SyncPos

---

Axis no: 1 - Home vel 0

Cause

---

HOME was executed with HOME\_VEL set to zero.

**O.ERR\_8**

SyncPos

---

Axis no: 1 - Position error

Meaning

---

The distance between the set and the real position was greater than the **Tolerated position error** defined in parameter POSERR (15).

Cause

---

Mechanically blocked or overloaded drive, **Tolerated position error** POSERR (15) too small, commanded speed greater than VLT Parameter 202 and 205, commanded acceleration too great, **Proportional factor** KPROP (11) too small or VLT not enabled.

**O.ERR\_9**

SyncPos

---

Axis no: 1 - Index pulse (encoder) not found

Meaning

---

At reference resp. index search, the encoder index pulse could not be found within a motor rotation.

Cause

---

An encoder without an index pulse has been used, index pulse not connected correct, index pulse incorrect (all three channels must have a simultaneous low) or the parameter ENCODER (2) is set too low.

**O.ERR\_10**

SyncPos

---

Unknown command

Cause

---

A communication or program error.

Tip

---

The program must be re-compiled and re-loaded.

**O.ERR\_11**


---

SyncPos

Axis no. 1 Software limit switch activated

---

Cause

A motion command will cause / has caused the software limit switch to be activated.

---

Tip

Identification of attainment of software limit at a motion in the speed mode will only be made after the current position is identical to the software limit switch.

The control unit will be switched off and the drive must be manually moved back to within the admissible area, or the monitoring of the software limit switch must be temporarily de-activated via the axis parameter SWPOSLIMACT (20) resp. SWNEGLIMACT (19). Only then is it possible to clear the error.

In positioning mode, it will be known before motion start that the target position lies outside the path.

In this case, the movement will not be executed and the error message can be cleared.

**O.ERR\_12**


---

SyncPos

Illegal parameter number

---

Cause

An attempt has been made to change a parameter (SET or SETVLT command), which does not exist.

**O.ERR\_13**


---

SyncPos

General VLT error

---

Cause

Every 20 ms the VLT status is tested and in the event of an error status this fault alarm is given.

Bits with the value of 512 and 2048 are checked. This test only takes place in the controlled status, not at MOTOR OFF and not when parameter 700 is deactivated.

---

Note

You can enquire in the VLT status as to the exact cause of the fault alarm (e.g. „Trip“ or „Not ready“).

**O.ERR\_14**


---

SyncPos

Too many nested loops

---

Cause

Too many nested loops exist in the executed program.

**O.ERR\_16**


---

SyncPos

Parameters in EEPROM are corrupted

---

Meaning

The parameters in EEPROM are no longer correct.

---

Cause

EEPROM defective or power outage while saving.

---

Tip

You have to re-initialize the parameter with "CONTROLLER" → "PARAMETERS" → "RESET" and then overwrite these parameters with your own user parameters. Otherwise motion programs which require user parameters will no longer function correctly.

**O.ERR\_17**


---

SyncPos

Programs in EEPROM are corrupted

---

Meaning

The program data stored in EEPROM cannot be found resp. are no longer correct.

---

Cause

EEPROM defective or power outage while saving.

---

Tip

Delete the EEPROM with "CONTROLLER" → "MEMORY" → "DELETE EEPROM" and then re-load the programs and parameters.

**O.ERR\_18**SyncPos

Reset by CPU

Meaning

The processor has been stopped and a re-set has automatically been executed (watchdog).

Cause

Short term voltage drop,  
voltage peak or  
short circuit.

**O.ERR\_19**SyncPos

User-Abort

Cause

The autostart-program has been aborted by the user.  
Or the [CANCEL] key was pressed during switching on and a Master Reset triggered.

**O.ERR\_25**SyncPos

Axis 1: Limit switch activated

Cause

A motion command has caused an axis limit switch to be activated.

Tip

Through activation of a limit switch, the controller – depending on the parameter ENDSWMOD (44) – is automatically switched off and the drive must be manually moved out of this position, before the error message can be cleared.

**O.ERR\_49**SyncPos

Too many interrupt functions

Cause

More interrupt functions than the maximum possible number were used ...

Tip

...permitted are:

32 ON INT

32 ON STATBIT

32 ON COMBIT

10 ON PARAM

20 ON APOS, ON MAPOS, ON MCPOS

**O.ERR\_51**SyncPos

Too many nested GOSUB commands

Cause

In the program, too many calls from one subroutine to another subroutine exist.  
The error usually occurs when there is a recurrent reference to one of the sub-programs in a sub-program.

Tip

Avoid too many (10 is maximum) opposing subroutine calls,  
avoid subroutines which call themselves (re cursive subroutine procedures).

**O.ERR\_52**SyncPos

Too many RETURN commands

Cause

There are either more RETURN than corresponding GOSUB commands in the program, or there is a direct jump from a subroutine with a GOTO command.  
Only one RETURN is allowed per sub-program.  
It is always better to jump to the beginning of a sub-program and then to jump with IF... to a previously defined label.

**O.ERR\_62**SyncPos

Error in verifying

Meaning

After saving something in the EEPROM (a program or parameters) an error was detected during verification.

Tip

Delete the EEPROM with "CONTROLLER" → "MEMORY" → "DELETE EEPROM" and try to save the program or parameters again. If this is not successful please call the technical service department.

### O.ERR\_70

SyncPos

Error in DIM command

#### Meaning

The definition of an array in a DIM command does not correspond to an already existing array in the SyncPos option.

#### Cause

The fields are from older SyncPos programs. The current program has other definitions.

#### Tip

Either adapt the SyncPos program to the correct array size or delete the old arrays with "CONTROLLER" → "MEMORY" → "DELETE EEPROM".



#### NB!

Remember to follow the recommendations concerning saving programs and parameters before deleting the EEPROM.

### O.ERR\_71

SyncPos

Attempt was made to cross arraybounds

#### Meaning

An attempt was made to describe an array element that is located outside of the defined array limits.

#### Cause

Error in the SyncPos program. Array sizing does not agree with the space required (e.g. due to an incorrectly programmed loop).

Or the array is too small for the number of test runs triggered by TESTSTART.

#### Tip

Check loop variables.

### O.ERR\_79

SyncPos

Timeout while waiting for index.

#### Meaning

The command WAITNDX was executed and the timeout listed was exceeded.

#### Cause

The timeout is probably too short or the index impulse could not found (see also O.ERR\_9).

### O.ERR\_84

SyncPos

Too many time interrupts

#### Meaning

Too many ON TIME or ON PERIOD commands were used within the program.

#### Tip

A maximum of 12 of these ON TIME and/or ON PERIOD commands are allowed within one program.

### O.ERR\_87

SyncPos

No more space for variables

#### Meaning

When the SyncPos program is started the space for the necessary variables is reserved dynamically. This space is now no longer available.

#### Cause and tip

You may have selected a maximum number of variables which is too high. Reduce the maximum number in "SETTINGS" → "COMPILER" (Standard = 92).

Or the memory available is occupied with programs or arrays. Delete the programs with "CONTROLLER" → "PROGRAMS" → "DELETE ALL"

or delete both the programs and arrays, i.e. by deleting the entire memory with "CONTROLLER" → "MEMORY" → "DELETE EEPROM".



#### NB!

Remember to follow the recommendations concerning saving programs and parameters before deleting the EEPROM.

### O.ERR\_90

SyncPos

MEMORY locked

#### Meaning

The program memory is write-protected and cannot be altered.

#### Tip

This means that autorecognition can neither be set nor deleted and programs can neither be saved or deleted. Equally, → "RAM SAVE" and → "EEPROM DELETE" will not be executed.

### O.ERR\_91

SyncPos

Curve array wrong

Meaning

An incorrect or old array is defined in the DIM instruction for SETCURVE.

Tip

An old array may exist if the cnf file with all parameters and arrays has not been loaded into the CAM-Editor.

An incorrect array could be caused by the following:

- It was not created by the curve editor.
- Previous version of a curve editor. Such an array must first be converted by the current curve editor (→ "LOAD" and "SAVE").
- Or the order of the arrays in the DIM instruction does not match the order in the cnf file. Refer to the number of the array in the title bar of the CAM-Editor in this respect.

### O.ERR\_92

SyncPos

Error from encoder monitoring

Meaning

Open or short circuit in accordance with the displayed LED



**NB!**

An error will be indicated even if no encoder is connected.

### O.ERR\_xx

SyncPos

Internal error ##

Meaning

If such an error should occur, please contact your dealer and report the error number displayed to the technical service department.

### ■ VLT software SyncPos messages

The VLT software SyncPos messages are arranged in alphabetical order. Letters following a % sign represent variables which can be used in plain text at the corresponding locations.

### ■ Compilation error ...

SyncPos

Compilation error(s): program not saved!

Meaning and tip

A file is always compiled first and then saved. If you want to save the program, for example in the menu "CONTROLLER" → "SAVE PROGRAM" and a syntax error is found during compilation this message will be displayed.

Start the "SYNTAX CHECK" in the menu "DEVELOPMENT", correct the syntax error and then save the program.

### ■ Connection to ... already exists ...

SyncPos

Connection to %d already exists [%s] - change to new Window?

Meaning

When opening a new window or when trying to connect a window with a controller that is already linked to a window.

Yes The controller is disconnected from the old window and linked to the new window.

No The controller stays connected to the old window, the new window is not linked to a controller.

### ■ Controller is executing a program ...

SyncPos

Controller is executing a program or command!

Meaning

When the controller is executing a command or program it is not available for additional commands. You have to → "BREAK" the new command and restart it once the previous command has been completely executed.

**■ Error in array ...**SyncPos

Error in array part of file.

Meaning

When re-saving a configuration ("CONTROLLER" → "PARAMETERS" → "RESTORE FROM FILE") the computer recognizes that the data in the array area is formatted incorrectly.

Cause

In order to be able to save a file, the following conditions must be fulfilled:

- Identical software versions
- same configuration (e.g. same number of axes)
- In the case that arrays have already been inputted, these must match the ones that are to be saved in terms of type and size.

**■ Error in axis parameter ...**SyncPos

Error in axis parameter part of file.

Meaning

When re-saving a configuration ("CONTROLLER" → "PARAMETERS" → "RESTORE FROM FILE") the computer recognizes that the data in the area of the axis parameters is formatted incorrectly. The parameter number and the sequence must be correct and numbering must be continuous.

Cause

In order to be able to save a file, the following conditions must be fulfilled:

- Identical software versions, that provides same number and order of the parameters
- same configuration (e.g. same number of axes)

**■ Error in global parameter ...**SyncPos

Error in global parameter part of file.

Meaning

When re-saving a configuration ("CONTROLLER" → "PARAMETERS" → "RESTORE FROM FILE") the computer recognizes that the data in the area of the global parameters is formatted incorrectly.

Causes

In order to be able to save a file, the following conditions must be fulfilled:

- Identical software versions, that provides same number and order of the parameters
- same configuration (e.g. same number of axes)

**■ Lost connection to ...**SyncPos

Lost connection to #%d!

Meaning

If the VLT is turned off or the plug is pulled, etc. the window is disconnected from the VLT and the lost connection is registered.

**■ Timeout: no reply from VLT**SyncPos

Timeout: no reply from VLT

Meaning

The VLT does not answer; check the connection.

**Chapter 8**

■ **Program samples**

|                                              |     |
|----------------------------------------------|-----|
| Introduction and warning .....               | 208 |
| COM_OPT .....                                | 209 |
| Marker count .....                           | 209 |
| Master simulation via software command ..... | 210 |

**■ Program samples****■ Introduction****WARNING!**

The entire unit must be installed according to the installation instructions and the functionality of the individual components and connections must be tested.

An EMERGENCY STOP must be installed according to the specific guidelines of the respective country.

The program samples may only be used with a drive which can turn freely without a path limitation.

The drive should be fixed in a stable holding device.

When using a linear unit limit switches for path limitation must be mounted and connected.

When using a linear unit the area of movement of the program samples must be calculated roughly and compared with the paths of movement which are mechanically permissible.

Cross references to the online help program samples

---

In the online help you will find all the other program samples, which are mentioned in the chapter software reference. You can copy these programs or parts of them directly into your program.

**■ COM\_OPT**

```

/* Program for sending and receiving 8 bytes of */
/* data via a communication option using */
/* PPO type 2 */

/* Definition of arrays */
DIM send [4]
DIM receive [4]

/* Definition of user parameters */
LINKGPAR 133 710 "DATA WORD 1" 0 255 0
LINKGPAR 134 711 "DATA WORD 2" 0 255 0
LINKGPAR 135 712 "DATA WORD 3" 0 255 0
LINKGPAR 136 713 "DATA WORD 4" 0 255 0

/* Initialize arrays (all elements = 0) */
i = 1
WHILE (i<=4) DO
 receive [i] = 0
 i = i+1
ENDWHILE
j = 1
WHILE (j<=4) DO
 send [j] = 0
 j = j+1
ENDWHILE

/* Main program loop */
main:
send [1] = GET 133
 /* send array, element 1 = value of par. 710 */
send [2] = GET 134
 /* send array, element 2 = value of par. 711 */
send [3] = GET 135
 /* send array, element 3 = value of par. 712 */
send [4] = GET 136
 /* send array, element 4 = value of par. 713 */

COMOPTGET 4 receive
 /* Copy 4 words from comm. option */
 /* to receive array */
COMOPTSEND 4 send
 /* Copy 4 words from send array to */
 /* communication option */

/* Print data of receive array */
print "RECEIVED(4 WORDS)", " ,receive [1],"
" ,receive [2]," " ,receive [3]," " ,receive [4]

GOTO main
 /* End of program */

```

**■ Marker count**

```

/* Measurement of marker distance in */
/* connection with SYNCM */
/* Master and slave must run when this program */
/* is executed */

DIM test[4] /* Definition of array with 4 elements */

/* Definition of variables */
dones = 0
donem = 0

/* Save last marker position for slave and master */
test [1] = IPOS
test [2] = MIPOS

/* Start of main program loop */
main:
/* Monitor the next slave marker and */
/* save it's position */
IF (IPOS != test [1] AND dones == 0) THEN
 test [3] = IPOS
 dones = 1
 res = test [3] - test [1]
 /* Calculate distance between 2 consecutive */
 /* slave markers */
 PRINT "Slave distance ", res
 /* Print result */
ENDIF
/* Monitor the next master marker and */
/* save it's position */
IF (MIPOS != test [2] AND donem == 0) THEN
 test [4] = MIPOS
 donem = 1
 res = test [4] - test [2]
 /* Calculate distance between 2 consecutive */
 /* master markers */
 PRINT "Master distance ", res
 /* Print result */
ENDIF
GOTO main

```

**■ syncc\_msim.m**

```
// Simulation of a master via software command

DIM curve[112], test [1000]
SET MENCODERTYPE 6
 // switch to internal software simulation
SETCURVE curve
simpos = 0
SYNCC 0
SYNCCSTART 0
start:
 SYSVAR[4105] = simpos
 // set master position (in qc)
 // Subsequently, gear system will be employed

 simpos = simpos + 100

 // ATTENTION, MAPOS continues
 // to supply the encoder values

GOTO start
TORIG_01
```

Chapter 9

- **Appendix**
- **Glossary terms** ..... 212
- **What's new in PC Software ...**
  - 2.1x / Option Card Software 3.1x? ..... 213
  - 2.2x and in SyncPos Motion Controller Software? 215
  - New hardware ..... 216
  - 2.3x and in SyncPos Motion Controller Software? .217
  - What's new in the PC Software Version
  - 2.5x and in Option Card Version 5.00? .....217
- **Technical Reference** ..... 218
- **Index** ..... 222

**■ Glossary terms**
FPGA

Field programmable gate array

Inverted value

notA = /A =

$\bar{A}$

LED

Light Emitting Diode

Master Units [MU]

The curve length or the master cycle length and other information (e.g. the marker distance) for the cam control are indicated in master units MU.

A factor (fraction) is used for the conversion into qc, as with the user unit:

$$1 \text{ MU} = \text{SYNCFAC TM (49)} / \text{SYNCFAC TS (50)}$$

MLONG

For a description which is not dependent on the version or the hardware, variables were used for some limit values: thus the value of MLONG = 1,073,741,824

Mt

Master (working) cycle length

MTC10

Motion Control Tool

Quadcounts

The unit defined by the parameter **User factor** POSFACT\_Z (23) and POSFACT\_N (26) is not used for all path and position parameters. For various parameters (for example a software limit switch like I\_POSLIMITSW (46) and the parameter **Tolerated position error** POSERR (15) the quad-count unit (qc) is valid.

4 quad-counts correspond to one sensor unit (incremental encoder). In the case of absolute encoders, the absolute values are returned 1 : 1.

st

sample time = 1 ms

User units [UU]

The units for the drive or the slave and the master, respectively, can be defined by the user in any way desired so that the user can work with meaningful measurements. The user units and are converted to quad-counts internally.

In the CAM control, the maximum run distance of the slave or the slave cycle length are indicated in User Units UU (qc).

The User Units are standardized with a factor:

$$1 \text{ UU} = \text{POSFACT\_Z (23)} / \text{POSFACT\_N (26)}$$

Zero impulse

= Index impulse

**■ What's new in PC Software 2.1x / Option card Software 3.1x?**
**■ Extension of the user interface**
Greater comfort in the editing window

Various colors are used to distinguish between comments, program sections, operators, numbers etc. You can change the colors with "SETTINGS" → "COLORS EDITOR".

You can also split the window vertically or horizontally for the display of a number of files.

Further editing tools and shortcuts

There are new editing commands, key combinations and function keys. The following editing tools have been added:

Unlimited undo and redo and the function find and replace.

If you press the CANCEL key on the VLT when switching on, the SyncPos option card will not start any program.

**■ Extension of the compiler, commands and parameters**
Extension of the compiler

Identification of comments using // analogously to the programming language C.

Further operators, such as XOR and Modulo and further bit operators.

Assignment Operation

Further interrupts

|            |                                         |
|------------|-----------------------------------------|
| ON COMBIT  | Interrupt when Bit n is set             |
| ON STATBIT | Interrupt when Bit n is set             |
| ON PARAM   | Interrupt when a parameter n is changed |

**■ New commands**

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| DEFMORIGIN         | Set the current master position as the zero point for the master.   |
| DELETE ARRAYS      | Delete all arrays in the RAM.                                       |
| DISABLE .. inttyp  | Switches interrupts off                                             |
| DISABLE ALL        | Switches all interrupts off (except ON ERROR)                       |
| ENABLE ...         | Switches interrupts on                                              |
| ENABLE ALL         | Switches all interrupts on                                          |
| LINKSYSVAR         | Link system variable with LCP display                               |
| MOVESYNCORIGIN     | Relative shifting of the origin of synchronization                  |
| ON COMBIT .. GOSUB | Call up a subprogram when Bit n of the communication buffer is set. |
| ON PARAM .. GOSUB  | Call up a subprogram when a parameter is altered.                   |
| ON STATBIT         | Call up a subprogram when bit n of the VLT status is set.           |
| PCD                | Pseudo array for direct access to the profibus data area            |
| SAVE ARRAYS        | Save arrays in the EEPROM                                           |
| SAVE AXPARS        | Save current axis parameters in the EEPROM                          |
| SAVE GLBPARS       | Save current global parameters in the EEPROM                        |
| SETMORIGIN         | Set any position as the zero point for the master.                  |
| SYNCSTATCLR        | Resetting of the flags MERR and MHIT                                |
| SYSVAR             | System variable (Pseudo array) reads system values.                 |
| TESTSETP           | Specify recording data for test run                                 |
| TESTSTART          | Start the recording of a test run                                   |

**■ Commands which have been extended**SYNCERR

The return value informs, whether the synchronization is running ahead (negative result) or behind (positive result), if SYNCACCURACY is set to a negative value.

ERRCLR

As of Version 2.1x ERRCLR also resets Bit 7 of the control word of the VLT. Then the VLT messages no longer need to be deleted.

LINKGPAR

The command LINKGPAR tests whether the value of the user parameter is within the specified range. If not, the corresponding limit is used and this value saved. This ensures that a display appears.

SYNCM

When defined in SYNCMSTART (62), the system waits for the first evaluation of the marker pulses on starting SYNCM and only then the offset SYNCPOSOFFS (54) is applied.

#DEBUG

In addition to ON and OFF you can define NO STOP. The system is not stopped, but internal break commands update the line number when the program is executed incrementally with "DEVELOPMENT" → "SINGLESTEP F9"  
The line numbers generated in this way can then be laid on a VLT parameter 795-799 with LINKSYSVAR and observed on the LCP display.

**■ New Parameters**ESCCOND (70)

With ESCCOND you can determine how the VLT will react to a program termination using [ESC].

SYNCMWINM (68) / SYNCMWINS (69)

The Marker Window Master and Slave shows how large the permitted tolerance for the occurrence of the markers is.

**■ Parameters which have been extended**POSDRCT (28)

You can invert the evaluation of the encoder information without changing the wiring (turning around the motor leads or exchanging the A and B tracks in the encoder).

O\_BRAKE (48)

If an output is defined for the brake, this remains active even when the program is terminated with ESC.

SYNCPOSOFFS (54)

When SYNCM is started – in opposite to SYNCP – the system waits for the first evaluation of the marker pulses. Only then is the offset applied.

SYNCACCUARY (55)

Extended input range: A minus sign supplies the synchronization error to SYNCERR with polarity sign. It is then possible to tell whether the synchronization is running ahead or behind.

SYNCMSTART (62)

Additional possibilities to define the start conditions of the marker synchronization.

SYNCVFTIME (65)

Master filter for SYNCP and SYNCM

■ **What's new in PC Software 2.2x and in corresponding SyncPos Motion Controller Software:**

VLT5000/SyncPos software version 3.xx/4.1x  
 VLT5000Flux/SyncPos software version 5.xx/4.1x

■ **CAM-Editor**

This PC software version contains a CAM-Editor for the creation of the curves for cam control with and without marker correction of the master and slave. All parameters required for this application are set in the CAM-Editor.

The curve definition is possible alphanumeric as well as interactive. The curve path and requisite parameters are visualized in the curve profile.

■ **Extension of the commands and parameters**

New interrupts

The new position interrupts POSINT have been developed especially for the cam box function: ON APOS, ON MAPOS and ON MCPOS.

■ **New commands**

The new commands have been developed for the realization of cam control and cam box:

|                   |                                                                                             |
|-------------------|---------------------------------------------------------------------------------------------|
| CURVEPOS          | Retrieve slave curve position that corresponds to the current master position of the curve. |
| DEFMCPOS          | Define initial position of the master                                                       |
| ON APOS .. GOSUB  | Call up a subprogram when the slave position xxx is passed                                  |
| ON MAPOS .. GOSUB | Call up a subprogram when the master position xxx (qc) is passed.                           |
| ON MCPOS .. GOSUB | Call up a subprogram when the master position xxx (MU) is passed.                           |
| POSA CURVEPOS     | Move slave to the curve position corresponding to the master position                       |
| SETCURVE          | Set CAM curve                                                                               |
| SYNCC             | Synchronization in CAM-Mode                                                                 |
| SYNCCMM           | Synchronization in CAM-Mode with master marker correction                                   |
| SYNCCMS           | Synchronization in CAM-Mode with slave marker correction                                    |
| SYNCCSTART        | Start slave for synchronization in CAM-Mode                                                 |

|           |                                                |
|-----------|------------------------------------------------|
| SYNCCSTOP | Stop slave after the CAM synchronization       |
| SWAPMENC  | Transpose master and slave encoder internally. |

■ **Modified and Expanded Commands**

SET ORIGIN

In combination with the command CURVEPOS, one can fix in this way that the current slave position matches the corresponding value of the curve.

SYNCCSTAT

The flags SYNCCMMERR and SYNCCSMERR are automatically reset: during the next successful marker correction and in the event of a new start of SYNCCM or through the command SYNCCSTATCLR.

SYSVAR

Completion with the axis processing data for the curve profile.

■ **New parameter**

PROFTIME (29)

Scan time for profile generator

■ **Modified and Expanded Parameters**

Parameter 700

One can select „ENABLE SP. W/O MONI.“ as third setting and thus avoid Option Error 13 (VLT not ready).

ENCODERTYPE (27)

The disruption can be corrected in the case of absolute encoders without overflow (linear encoders).

POSFACZ\_Z (23) and POSFACT\_N (26)

Is used in CAM-Mode to convert the units qc into user units.

SYNCCFACTM (49) and SYNCCFACTM (50)

Is used in CAM-Mode to convert the units qc into user units MU.

SYNCCPULSM (58) and SYNCCPULSS (59)

In CAM-Mode the distance between sensor and working position in MU resp. UU.

SYNCMTYPM (60) and SYNCMTYPS (61)

The signal resp. marker type applies also for the command MIPOS.

SYNCMSTART (62)

Additional parameter for the starting behavior:  
2000 = in CAM-Mode: Counting of the master pulses in MU begins with the master marker.

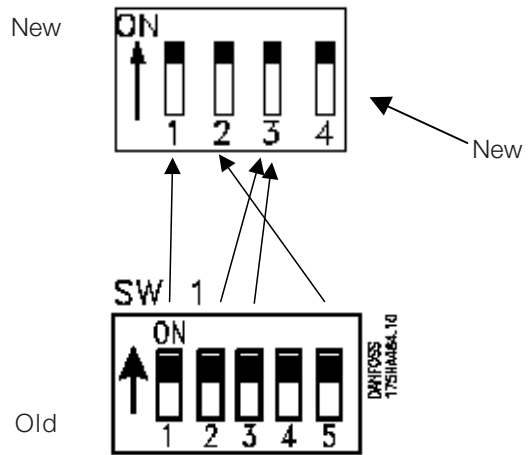
SYNCMWINM (68) and SYNCMWINS (69)

Changes of the parameter will become active immediately – not only after the next SYNCM command.

**■ New hardware**

The hardware of the Programmable SyncPos motion controller has been updated:

- Memory for programs has been extended from 38 kByte to 62 kByte.
- Hardware encoder monitor has been added.
- New LED's
- Dip switch changed, switches 2, 3 and 4 are combined to one switch (3) in the new hardware:



SyncPos for VLT5000Flux is always based on the new hardware.

SyncPos for VLT5000 based on the new hardware can be identified via the serial number which can be found on the bar-code label. Serial number xxxxxxGxxx means new hardware, serial number xxxxxxNxxx means old hardware.

Software compatibility

The new hardware is supported by option card software version 4.xx or newer.

The old hardware is supported by option card software version 3.12 or older.

**■ What's new in PC Software 2.3x and in SyncPos Motion Controller Software**

VLT5000/SyncPos Software Version 3.xx/4.2x  
 VLT5000FluxSyncPos Software Version 5.xx/4.2x?

**■ Debug Modus**

A comfortable debug mode and online status information as well as the possibility to change the variables during program execution.

"PREPARE SINGLESTEP"

All executable program lines are marked by blue dots.

"SINGLESTEP" or [F9]

In the debug mode: executes the next program line.

Breakpoints

In the debug mode: → "EXECUTE" or press [F5] in order to process the program until the next breakpoint.

Change variables online

In the debug mode: set value

Read variables

In the debug mode: read the current value of the variables after the program execution.

"SHOW WATCH"

Enables online monitoring of the variables, arrays, system and axis processing data (according to the SYSVAR indices) and axis parameters.

**■ "FIND" and "REPLACE"**

Find and replace in accordance with the Windows conventions and:

- |            |                                                               |
|------------|---------------------------------------------------------------|
| "MARK ALL" | marks all sites found with a blue triangle at the left margin |
| [F2]       | jumps back and forth between the sites found                  |
| [F3]       | jumps from one site found to the next one                     |

**■ What's new in the PC Software Version 2.5x and in Option Card Version 5.00?**

This version contains primarily the necessary extensions and a modified GUI to use the SyncPos program in MCT10 mode.

Further on there is an additional feature to set or change the axis parameters with the modified CAM-Editor.

**■ New and modified Commands**
DEF SYNCORIGIN

Defines master-slave relation for the next SYNCOP or SYNCOM command

SYNCERR

The SYNCERR (as well as the internal master command position G\_Mpcmd) is also updated when SYNCOP or SYNCOM are not active, e.g. after a MOTOR OFF.

**■ Modified and Expanded Parameter**
PRGPAR - 102

Extended with the possibility to start the activated program number, but at power on the motor is turned off (motor off).

**■ Technical Reference**

This section documents data structures and compiler details which are only required in exceptional cases by the user. For example, if an automatically generated programming is to be modified like a curve profile.

**■ Array Structure of CAM Profiles**
Header

The header contains general information like

- Identification for curve array
- Version number for curve structure
- Type of curve
- Name of curve
- Index to curve information section
- Index to start/stop point section
- Index to fixed point section
- Index to interpolation point section
- Index to start/stop point indices (in interpolation section)
- Index to start/stop velocities (times 100000)
- Index to startpath interpolation points
- Index to stoppath interpolation points

Curve information section

This section of the array contains all information about the type of curve like

- Length of curve (master)
- Length of curve (slave)
- Number of fix points
- Number of Interpolation points (this gives the resolution)
- Type of interpolation
- Slave stop point, point where slave is positioned, when synchronization is stopped
- Correction start point (only valid for marker synchronization)
- Correction end point (only valid for marker synchronization)
- Maximum correction which is allowed (only valid for marker synchronization)
- Maximum start/stop path length (Size of start/stop path area)(min. 2)
- No of start/stop point pairs
- Maximum number of cycles per minute (Application information)

Curve start/stop point section

This section contains the start/stop points.

Because the use of this point is up to the user, we just speak of a path, which can be a start or a stop sequence. Every path consists of 2 points. If we are moving forward, the path starts (start or stop) with the a-point and ends with the b-point. If we are moving backward, the path starts with the b-point and ends with the a-point. So the user is able to tell us in the program, which pair of points to use for starting or stopping, when he uses a STARTCURVE or STOPCURVE command.

- Path 1 (a – point)
- Path 1 (b – point)
- Path 2 (a – point)
- Path 2 (b – point)
- ...

These points have to lie on interpolation points, so possibly the PC software has to adjust them according to the interpolation resolution. This should not be a real restriction, because the interpolation points are normally very dense. So for example if we have rotating master which makes one revolution per cycle and we choose a cycle length of 3600 MU (1 MU = 1/10 degree). Let us further assume, that we choose the number of interpolation points as 1200, than you have a resolution of 3 MU = 3/10 degree for defining your start and stop points.

Fixed point section

This section contains the fix points, which were the basis for the interpolation calculation. These points always consist of the following triple

- Master coordinate
- Slave coordinate
- Type of point (tangent, curve)

These points are defined by the user in MU units (see internal description). If you want to avoid, that the real interpolation curve misses your fix points, you have to choose them in such a manner that they lay on an interpolation point (see above). This can be forced through a snap function within the PC software.

Interpolation point section

This section contains a list of slave coordinates. They belong to master coordinates which are of equal distance, given by the interpolation resolution.

### Indices of start/stop points

---

Here we have the indices of the start/stop points (see above) within the interpolation array. These are necessary for the ease of start and stop recognition. We are waiting until start index for example equals the actual index and direction of movement is correct. If both is true, we start synchronization. The same is true for stopping.

### Start Stop Velocities

---

To be able to calculate an appropriate starting or stopping path, we need the velocity we have to reach at end (start) or we will have at the beginning (stop) in UU/MU units (Slave units per Master units).

### Start / Stop paths

---

This is the place for the interpolation points of the actual start and stop path. These points are calculated when a SYNCCSTART or SYNCCSTOP command is executed, but we have to reserve the room right now.

## ■ CAM Array Definition

| Part           | Index | Name           | Unit    | Value                       | Description                              |
|----------------|-------|----------------|---------|-----------------------------|------------------------------------------|
| <b>General</b> | 1     | Identification | (dec)   | 999.000.001                 | Number to identify array                 |
|                | 2     | VersioNumber   | (dec)   | 100                         | Version as decimal (1.00 = 100)          |
|                | 3     | CurveType      | (dec)   | 0                           | 0 = symmetrical, 1 = compatible          |
|                | 4     | CurveName 1    | (4char) | Nona                        | Name of curve total 16 char.             |
|                | 5     | CurveName 2    | (4char) | meCu                        | default is:                              |
|                | 6     | CurveName 3    | (4char) | rve0                        | NonameCurve00001                         |
|                | 7     | CurveName 4    | (4char) | 0001                        |                                          |
|                | 8     | IndexCIF       | (dec)   | 16                          | Index to Curve Information Part          |
|                | 9     | IndexSTP       | (dec)   | 27                          | Index to Start/Stop Point Part           |
|                | 10    | IndexFIP       | (dec)   | IndexSTP + STPno*2          | Index to Fix Point Part                  |
|                | 11    | IndexINP       | (dec)   | IndexFIP + FixPointNo * 3   | Index to Interpolation Point Part        |
|                | 12    | IndexSTPInd    | (dec)   | IndexINP + InterpolPointNo  | Index to StartStop Interpolation Indices |
|                | 13    | IndexSTPVel    | (dec)   | IndexSTPInd + STPno*2       | Index to StartStop Velocities            |
|                | 14    | IndexSTIP      | (dec)   | IndexSTPVel + STPno*2       | Index to Startpath interpolation points  |
|                | 15    | IndexSTPIP     | (dec)   | IndexSTIP + MaxStartStopLen | Index to Stoppath interpolation points   |

| Part                     | Index | Name                 | Unit  | Value          | Description                                                                                      |
|--------------------------|-------|----------------------|-------|----------------|--------------------------------------------------------------------------------------------------|
| <b>Curve Information</b> | 1     | MasterCycleLen       | MU    | -              | Length of Curve in Curve Master units                                                            |
|                          | 2     | SlaveCycleLen        | UU    | -              | Slave max. travel distance in CurveSlave units                                                   |
|                          | 3     | FixPointNo           | (dec) | 4              | Number of fix points (min. 4)                                                                    |
|                          | 4     | InterpolPointNo      | (dec) | -              | Number of interpolation points (including first and last, which correspond to the same location) |
|                          | 5     | InterpolType         | (dec) | 0              | 0 = cubic spline 1 = periodic cubic spline                                                       |
|                          | 6     | SlaveStopPosition    | UU    | 0              | Position, where slave stands after stopping                                                      |
|                          | 7     | CorrectionStartPoint | MU    | 0              | Position, where Correction may start                                                             |
|                          | 8     | CorrectionStopPoint  | MU    | MasterCycleLen | Position, where Correction has to be finished                                                    |
|                          | 9     | MaximumCorrection    | UU    | -              | Maximum Correction which is allowed in one cycle                                                 |
|                          | 10    | MaxStartStopLen      | (dec) | 0              | Maximum length of start/stop path (no of int. points)                                            |
|                          | 11    | StartStopNo          | (dec) | 0              | Number of start stop point pairs (n) (see below)                                                 |
| <b>Start/Stop Point</b>  | 1     | STPoint_1.a          | MU    | 0              | Start (forward) / Stop (backward) Point no. 1                                                    |
|                          | 2     | STPoint_1.b          | MU    | 0              | Stop (forward) / Start (backward) Point no. 1                                                    |
|                          | 3     | STPoint_2.a          | MU    | 0              | Start (forward) / Stop (backward) Point no. 2                                                    |
|                          | 4     | STPoint_2.b          | MU    | 0              | Stop (forward) / Start (backward) Point no. 2                                                    |
|                          | 5     | ...                  | MU    | 0              |                                                                                                  |
|                          | 6     | ...                  | MU    | 0              |                                                                                                  |
|                          | 2*n-1 | STPoint_n.a          | MU    | 0              | Start (forward) / Stop (backward) Point no. n                                                    |
|                          | 2*n   | STPoint_n.b          | MU    | 0              | Stop (forward) / Start (backward) Point no. n                                                    |
| <b>Fix Point</b>         | 1     | FixPoint_1.master    | MU    | 0              | Fix Point no. 1 - master coordinate                                                              |
|                          | 2     | FixPoint_1.slave     | UU    | -              | Fix Point no. 1 - slave coordinate                                                               |
|                          | 3     | FixPoint_1.type      | (dec) | C              | Fix Point no. 1 - type of point (1 = Curve Point, 2 = Tangent Point)                             |
|                          | 4     | ...                  |       |                |                                                                                                  |
|                          | 3*n-2 | FixPoint_n.master    | MU    | MasterCycleLen | Fix Point no. n - master coordinate                                                              |
|                          | 3*n-1 | FixPoint_n.slave     | UU    | -              | Fix Point no. n - slave coordinate                                                               |
|                          | 3*n   | FixPoint_n.type      | (dec) | C              | Fix Point no. n - type of point (1 = Curve Point, 2 = Tangent Point)                             |

| <b>Part</b>                           | <b>Index</b> | <b>Name</b>        | <b>Unit</b> | <b>Value</b> | <b>Description</b>                                        |
|---------------------------------------|--------------|--------------------|-------------|--------------|-----------------------------------------------------------|
| <b>Interpolation Point</b>            | 1            | IntPoint_1         | UU          | 0            | Interpolation Point no. 1 - slave coordinate              |
|                                       | ...          |                    |             |              |                                                           |
|                                       | n            | IntPoint_n         | UU          | -            | Interpolation Point no. n - slave coordinate              |
| <b>StartStop Indices</b>              | 1            | STPoint_1.a-index  | (dec)       | 0            | Index in Interpolation Array, corresponding to Startpoint |
|                                       | 2            | STPoint_1.b-index  | (dec)       | 0            | Index in Interpolation Array, corresponding to Sartpoint  |
|                                       | 3            | .                  |             |              |                                                           |
| <b>StartStop Velocities</b>           | 1            | STPoint_1.a-veloc. | (dec)       | (*100000)    | Velocity (UU/MU * 100000) in startpoint                   |
|                                       | 2            | STPoint_1.b-veloc. | (dec)       | (*100000)    | Velocity (UU/MU * 100000) in startpoint                   |
|                                       | ...          |                    |             |              |                                                           |
| <b>StartPath Interpolation Points</b> | 1            | StartPoint_1       | UU          | 0            | Interpolation Point no. 1 - for start path                |
|                                       | ...          |                    |             |              |                                                           |
|                                       | n            |                    |             |              |                                                           |
| <b>StopPath Interpolation Points</b>  | 1            | StopPoint_1        | UU          | 0            | Interpolation Point no. 1 - for stop path                 |
|                                       | ...          |                    |             |              |                                                           |
|                                       | n            |                    |             |              |                                                           |

**■ Index**
**Symbole**

|                |     |
|----------------|-----|
| #DEBUG .....   | 162 |
| #INCLUDE ..... | 162 |
| _GETVEL .....  | 161 |
| 4-Setup .....  | 167 |

**A**

|                                                     |        |
|-----------------------------------------------------|--------|
| Abbreviation .....                                  | 170    |
| About Program .....                                 | 79     |
| About this manual .....                             | 6      |
| ACC .....                                           | 99     |
| Acceleration feed-forward FFACC (37) .....          | 182    |
| Acceleration Limit .....                            | 76     |
| ACCMAXQC (10) .....                                 | 174    |
| Analog inputs .....                                 | 10     |
| Analog outputs .....                                | 10     |
| APOS .....                                          | 100    |
| Arithmetic .....                                    | 93     |
| Array Structure of CAM Profiles .....               | 218    |
| Arrays .....                                        | 92     |
| Arrays versus Variables .....                       | 93     |
| Reading and Writing Arrays .....                    | 93     |
| reset .....                                         | 64     |
| Assignment Operation .....                          | 95     |
| Autostart .....                                     | 59     |
| Linking autostart programs .....                    | 60     |
| Repeated execution .....                            | 59, 60 |
| AVEL .....                                          | 100    |
| AXEND .....                                         | 101    |
| Axis parameters .....                               | 168    |
| change the parameters of a configuration file ..... | 62     |
| factory settings .....                              | 61     |
| set or change online .....                          | 61     |
| Axis Process Data .....                             | 154    |

**B**

|                        |         |
|------------------------|---------|
| BANDWIDTH (35) .....   | 34, 181 |
| Before you begin ..... | 22      |
| Breakpoints .....      | 55      |

**C**
**CAM**

|                                                   |     |
|---------------------------------------------------|-----|
| Accuracy .....                                    | 38  |
| Array Definition .....                            | 219 |
| Diagram of the principle .....                    | 38  |
| Example: Slave Synchronization with Marker .....  | 45  |
| Example: Stamping of boxes with use-by date ..... | 39  |
| Example: with Marker Correction .....             | 42  |
| How a Cam box with SyncPos operates .....         | 48  |
| How a CAM control with SP operates .....          | 38  |
| Internal Realization as Array .....               | 38  |
| Interpolation .....                               | 38  |
| Problematic situations ... marker distance .....  | 44  |

|                                         |        |
|-----------------------------------------|--------|
| Quick Start Tutorial .....              | 39     |
| Tangent Points .....                    | 38     |
| CAM Box .....                           | 48     |
| CAM Control .....                       | 38     |
| CAM Profile .....                       | 72     |
| scroll, zoom .....                      | 72     |
| CAM-Editor menu .....                   | 69     |
| Edit curve menu .....                   | 71     |
| File menu CNF .....                     | 70     |
| Menu Bar .....                          | 69     |
| Messages .....                          | 73     |
| Permanent Recalculation .....           | 71     |
| Recalculation .....                     | 71     |
| Snap on Grid .....                      | 71     |
| Title Bar .....                         | 69     |
| View menu .....                         | 71     |
| Window .....                            | 69     |
| Window menu .....                       | 72     |
| Change variables online .....           | 49, 56 |
| Changes during operation .....          | 167    |
| Close SyncPos .....                     |        |
| in MCT10 mode .....                     | 83     |
| Colors Editor .....                     | 78     |
| COM_OPT .....                           | 209    |
| Command List .....                      | 57     |
| Command run times .....                 | 89     |
| Command Structure .....                 | 89     |
| Commands which have been extended ..... | 214    |
| Communication Setup .....               | 27     |
| COMOPTGET .....                         | 102    |
| COMOPTSEND .....                        | 102    |
| Compiler options .....                  | 77     |
| Connection examples .....               | 15     |
| Constants .....                         | 92     |
| Context menu .....                      | 53     |
| Context-sensitive help .....            | 79     |
| CONTINUE .....                          | 103    |
| Controller menu .....                   | 58     |
| Autostart .....                         | 59     |
| Axis parameters .....                   | 61     |
| Delete all programs .....               | 60     |
| Delete EEPROM .....                     | 64     |
| Global parameters .....                 | 60     |
| Parameter Name .....                    | 63     |
| Parameters .....                        | 60     |
| Programs .....                          | 58     |
| Reset .....                             | 64     |
| Restore from file .....                 | 63     |
| Save programs .....                     | 58     |
| Save RAM .....                          | 64     |
| Start programs .....                    | 59     |
| Conventions .....                       | 5, 7   |
| Conversion index .....                  | 167    |
| Correction Start / End .....            | 75     |
| CPOS .....                              | 103    |
| Cross-references .....                  | 79     |

|                         |     |
|-------------------------|-----|
| CSTART .....            | 104 |
| CSTOP .....             | 104 |
| Curve Data .....        | 75  |
| Curve Info .....        | 76  |
| Curve Points .....      | 73  |
| CURVEPOS .....          | 105 |
| CVEL .....              | 105 |
| Cycles/min Master ..... | 76  |

**D**

|                                        |            |
|----------------------------------------|------------|
| D.TIME CMP.ACT. 780 .....              | 166        |
| Dampening of vibrations .....          | 36         |
| Data sampling interval .....           | 65         |
| Data type .....                        | 167        |
| Dead time compensation .....           | 166        |
| Debug Mode                             |            |
| Change variables online .....          | 49, 56     |
| Compatibility .....                    | 56         |
| End Debug .....                        | 49, 56     |
| Go .....                               | 49, 56     |
| Interrupt program execution .....      | 56         |
| Read variables .....                   | 56         |
| Show Watch .....                       | 49, 56     |
| DEC .....                              | 106        |
| DEF ORIGIN .....                       | 107        |
| DEF SYNCORIGIN .....                   | 107        |
| DEFMCPOS .....                         | 106        |
| DEFMORIGIN .....                       | 107        |
| DELAY .....                            | 108        |
| Delete file .....                      | 52         |
| Derivative factor KDER (12) .....      | 33, 174    |
| Determining the maximum velocity ..... | 37         |
| Development menu .....                 | 54         |
| Break and break all .....              | 55         |
| Command List .....                     | 57         |
| Continue program .....                 | 55         |
| Execute .....                          | 54         |
| Execute now .....                      | 57         |
| Insert .....                           | 57         |
| Move to position .....                 | 58         |
| Preparing single-step .....            | 55         |
| Select controller .....                | 27, 57     |
| Single-step .....                      | 55         |
| DFLTACC (34) .....                     | 181        |
| DFLTVEL (33) .....                     | 181        |
| Dialog fields .....                    | 23         |
| Digital inputs .....                   | 10, 11, 14 |
| Digital outputs .....                  | 10, 11, 14 |
| DIM .....                              | 109        |
| DIM Statement .....                    | 92         |
| Dip switch .....                       | 13, 216    |
| DISABLE ... interrupts .....           | 110        |
| Display Recording .....                | 67         |

**E**

|                                                |     |
|------------------------------------------------|-----|
| Edit menu .....                                | 53  |
| Copy .....                                     | 53  |
| Cut .....                                      | 53  |
| Find .....                                     | 53  |
| Paste .....                                    | 53  |
| Replace .....                                  | 53  |
| Undo .....                                     | 53  |
| ENABLE ... interrupts .....                    | 110 |
| Encoder                                        |     |
| Connection check .....                         | 30  |
| Direction of rotation .....                    | 30  |
| Input 1 (master) .....                         | 14  |
| Input 2 (slave) .....                          | 15  |
| Output (Virtual master) .....                  | 15  |
| ENCODER (2) .....                              | 171 |
| Encoder monitor .....                          | 12  |
| ENCODERTYPE (27) .....                         | 179 |
| ENDSWMOD (44) .....                            | 183 |
| ERRCLR .....                                   | 111 |
| ERRCOND (43) .....                             | 183 |
| ERRNO .....                                    | 111 |
| ESCCOND (70) .....                             | 195 |
| Evaluating motion figures .....                | 67  |
| Examples                                       |     |
| Cam box .....                                  | 48  |
| Master simulation via software command .....   | 210 |
| PID-control optimization .....                 | 36  |
| Printing of boxes with Marker Correction ..... | 42  |
| Slave Synchronization with Marker .....        | 45  |
| Stamping of boxes with use-by date .....       | 39  |
| Execute Testrun .....                          | 66  |
| EXIT .....                                     | 112 |
| Exit program .....                             | 52  |
| External 24 V DC supply .....                  | 14  |

**F**

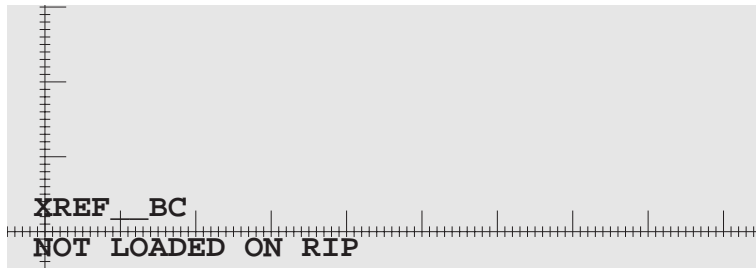
|                                            |         |
|--------------------------------------------|---------|
| Factory settings .....                     | 170     |
| Axis parameters .....                      | 61      |
| Global parameters .....                    | 60      |
| Parameter .....                            | 163     |
| Feed-forward Acceleration FFACC (37) ..... | 34      |
| FFACC (37) .....                           | 34, 182 |
| FFVEL (36) .....                           | 33, 181 |
| FILE menu .....                            | 51      |
| Close .....                                | 51      |
| Exit program .....                         | 52      |
| New .....                                  | 51      |
| Open .....                                 | 51      |
| Print .....                                | 52      |
| Save and save as .....                     | 52      |
| Find and Replace .....                     | 53      |
| Mark all .....                             | 54      |
| Fixpoints                                  |         |
| Index Card .....                           | 73      |
| Inserting fixpoints .....                  | 72      |

|                                                     |        |                                                      |         |
|-----------------------------------------------------|--------|------------------------------------------------------|---------|
| Forward Feel Acceleration FFACC (37) .....          | 182    | Input values .....                                   | 90      |
| Forward Feel Acceleration FFVEL ( 36) .....         | 181    | Installing SyncPos .....                             | 26      |
| FPGA .....                                          | 212    | Integral factor KINT (13) .....                      | 33, 174 |
| Function keys .....                                 | 25     | Integration limit KILIM (21) .....                   | 33, 176 |
| <b>G</b>                                            |        | Interface Settings                                   |         |
| GET .....                                           | 112    | Scan Range .....                                     | 78      |
| GETVLT .....                                        | 113    | Interpolation .....                                  | 38      |
| GETVLTSUB .....                                     | 113    | Interrupt .....                                      | 90      |
| Global Parameters .....                             | 168    | General processing of interrupt procedures .....     | 90      |
| Go .....                                            | 49, 56 | Interrupt nesting .....                              | 91      |
| GOSUB .....                                         | 114    | Use of variables within interrupt procedures .....   | 90      |
| GOTO .....                                          | 114    | Interval Size .....                                  | 76      |
| <b>H</b>                                            |        | Interval Time (ms) .....                             | 76      |
| Help in Command list menu .....                     | 57     | Inverted value .....                                 | 212     |
| Help menu .....                                     | 79     | IPOS .....                                           | 120     |
| Context-sensitive help .....                        | 79     | <b>K</b>                                             |         |
| Cross-references .....                              | 79     | KDER (12) .....                                      | 33, 174 |
| Printing or copying help texts .....                | 79     | Keyboard .....                                       | 24      |
| HOME .....                                          | 115    | KILIM (21) .....                                     | 33, 176 |
| HOME_FORCE (3) .....                                | 171    | KINT (13) .....                                      | 33, 174 |
| HOME_OFFS (42) .....                                | 183    | KPROP (11) .....                                     | 33, 174 |
| HOME_RAMP (41) .....                                | 183    | <b>L</b>                                             |         |
| HOME_VEL (7) .....                                  | 172    | Labels, maximum number .....                         | 77      |
| HOMETYPE (40) .....                                 | 182    | Language .....                                       | 78      |
| How a Cam box with SyncPos operates .....           | 48     | Line number .....                                    | 53      |
| How SyncPos functions .....                         | 21     | LINKXPAR .....                                       | 120     |
| <b>I</b>                                            |        | LINKGPAR .....                                       | 121     |
| I_BREAK (105) .....                                 | 196    | LINKSYSVAR .....                                     | 121     |
| I_CONTINUE (106) .....                              | 196    | List of commands .....                               | 96      |
| I_ERRCLR (107) .....                                | 197    | LOOP .....                                           | 122     |
| I_NEGLIMITSW (47) .....                             | 184    | Lost files .....                                     | 51      |
| I_POSLIMITSW (46) .....                             | 184    | <b>M</b>                                             |         |
| I_PRGCHOICE (104) .....                             | 196    | MAPOS .....                                          | 122     |
| I_PRGSTART (103) .....                              | 196    | Mark all .....                                       | 54      |
| I_REFSWITCH (45) .....                              | 184    | Marker count .....                                   | 209     |
| IF ..THEN .., ELSEIF .. THEN .. ELSE .. ENDIF ..... | 116    | Marker signal .....                                  | 150     |
| IN .....                                            | 117    | Marking text .....                                   | 53      |
| INAD .....                                          | 117    | Master Length / Slave Length .....                   | 76      |
| INB .....                                           | 118    | Master Marker Position / Slave Marker Position ..... | 75      |
| Include source code .....                           | 59     | Master simulation .....                              | 210     |
| INDEX .....                                         | 118    | Master Units [MU] .....                              | 95      |
| Index .....                                         | 79     | MAVEL .....                                          | 123     |
| Index Card                                          |        | Max. Actual Acceleration .....                       | 76      |
| Curve Data .....                                    | 75     | Max. Actual Velocity .....                           | 76      |
| Curve Info .....                                    | 76     | Max. Slave Acceleration .....                        | 76      |
| Fix points .....                                    | 73     | Max. Slave Verlocity .....                           | 76      |
| Start Stop Points .....                             | 74     | MCT10 mode .....                                     | 81      |
| Synchronization .....                               | 77     | CAM Editor .....                                     | 85      |
| Velocity .....                                      | 77     | CAM-Editor window without curve data .....           | 85      |
| Indexes .....                                       | 93     | Development menu .....                               | 84      |
| Initializing .....                                  | 89     | File Handling .....                                  | 83      |
| INKEY .....                                         | 119    | File menu .....                                      | 84      |
| Input range .....                                   | 170    | Operation Mode: Offline or Online .....              | 83      |

|                                             |     |                                           |             |
|---------------------------------------------|-----|-------------------------------------------|-------------|
| Settings Menu .....                         | 86  | New commands .....                        | 213         |
| SyncPos Window .....                        | 83  | New hardware .....                        | 216         |
| Memory                                      |     | New Parameters .....                      | 214         |
| Delete EEPROM .....                         | 64  | NOWAIT .....                              | 126         |
| Save RAM .....                              | 64  | in interrupts .....                       | 91          |
| MENCODER (30) .....                         | 180 | Number of Intervals .....                 | 75          |
| MENCODERTYPE (67) .....                     | 193 | <b>O</b>                                  |             |
| Menus .....                                 | 23  | O.ERR_3 ... xx .....                      | 201         |
| Messages                                    |     | O_AXMOVE (64) .....                       | 191         |
| Attempt was made to cross arraybounds ..... | 204 | O_BRAKE (48) .....                        | 185         |
| Axis no. ... not in system .....            | 201 | O_ERROR (108) .....                       | 197         |
| Compilation error ... .....                 | 205 | ON APOS .. GOSUB .....                    | 126         |
| Connection to ... already exists ... .....  | 205 | ON COMBIT .. GOSUB .....                  | 91, 127     |
| Controller is executing a program ... ..... | 205 | ON ERROR GOSUB .....                      | 127         |
| Curve array wrong .....                     | 205 | ON INT .. GOSUB .....                     | 128         |
| Error from encoder monitoring .....         | 205 | ON MAPOS .. GOSUB .....                   | 129         |
| Error in array ... .....                    | 206 | ON MCPOS .. GOSUB .....                   | 130         |
| Error in axis parameter ... .....           | 206 | ON PARAM .. GOSUB .....                   | 91, 131     |
| Error in DIM command .....                  | 204 | ON PERIOD .....                           | 90, 91, 131 |
| Error in global parameter ... .....         | 206 | Interrupt .....                           | 90          |
| Error in verifying .....                    | 203 | ON STATBIT .. GOSUB .....                 | 91, 132     |
| Error not cleared .....                     | 201 | ON TIME .....                             | 91, 133     |
| Failed to move to HOME position .....       | 201 | Optimizing of the PID controller .....    | 32          |
| Illegal parameter number .....              | 202 | Option card layout .....                  | 13          |
| Index pulse (encoder) not found .....       | 201 | Option card terminals .....               | 10          |
| Internal error ## .....                     | 205 | ORIGINOFFS (42) .....                     | 183         |
| Limit switch activated .....                | 203 | OUT .....                                 | 133         |
| Lost connection to ... .....                | 206 | OUTAN .....                               | 134         |
| No more space for variables .....           | 204 | OUTB .....                                | 134         |
| Parameters in EEPROM are corrupted .....    | 202 | OUTDA .....                               | 135         |
| Position error .....                        | 201 | <b>P</b>                                  |             |
| Programs in EEPROM are corrupted .....      | 202 | Parameter 205 .....                       | 165         |
| Reset by CPU .....                          | 203 | Parameter changes and storage .....       | 163         |
| Software limit switch activated .....       | 202 | Parameter Reference .....                 | 163         |
| Timeout while waiting for index. ....       | 204 | Parameter Values .....                    | 170         |
| Timeout: no reply from VLT .....            | 206 | Parameters which have been extended ..... | 214         |
| Too many interrupt functions .....          | 203 | Path of motion .....                      | 65          |
| Too many nested GOSUB commands .....        | 203 | PC Software Interface .....               | 49          |
| Too many RETURN commands .....              | 203 | PCD .....                                 | 135         |
| Too many time interrupts .....              | 204 | Permanent Recalculation .....             | 71          |
| Unknown command .....                       | 201 | PID .....                                 | 136         |
| User-Abort .....                            | 203 | PID control parameters .....              | 32          |
| MIPOS .....                                 | 123 | Point Type .....                          | 72          |
| MK3A .....                                  | 11  | POSA .....                                | 136         |
| MK3B .....                                  | 11  | POSA CURVEPOS .....                       | 137         |
| MK3C .....                                  | 11  | POSDRCT (28) .....                        | 179         |
| MK3D .....                                  | 11  | POSERR (15) .....                         | 175         |
| MOTOR OFF .....                             | 124 | POSFAC <sub>T</sub> _N (26) .....         | 178         |
| MOTOR ON .....                              | 124 | POSFAC <sub>T</sub> _Z (23) .....         | 177         |
| MOTOR STOP .....                            | 125 | POSLIMIT (5) .....                        | 172         |
| Move to position .....                      | 58  | POSR .....                                | 137         |
| MOVESYNCORIGIN .....                        | 125 | Preparing singlestep .....                | 55          |
| <b>N</b>                                    |     | PRGPAR (102) .....                        | 195         |
| Name of Curve .....                         | 75  | PRINT .....                               | 138         |
| NEGLIMIT (4) .....                          | 172 |                                           |             |

|                                      |         |                                               |          |
|--------------------------------------|---------|-----------------------------------------------|----------|
| Printing or copying help texts ..... | 79      | Set Breakpoints .....                         | 55       |
| Priorities                           |         | SET ORIGIN .....                              | 142      |
| Interrupts .....                     | 91      | SETMORIGIN .....                              | 142      |
| Operators and the Operations .....   | 95      | Setting of SyncPos parameters .....           | 28       |
| PROFTIME (29) .....                  | 180     | Encoder .....                                 | 28       |
| Program Layout .....                 | 89      | Home .....                                    | 29       |
| Definitions .....                    | 89      | Synchronization .....                         | 29       |
| Initializing .....                   | 89      | Velocity .....                                | 29       |
| Main program .....                   | 89      | Setting of VLT parameters .....               | 27       |
| Sub programs .....                   | 89      | Setting parameters for a test run             |          |
| Program samples .....                | 208     | Number of readings .....                      | 65       |
| Programming Language elements .....  | 92      | Path of motion .....                          | 65       |
| Proportional factor KPROP (11) ..... | 33, 174 | Sampling interval .....                       | 65       |
| PULSACC .....                        | 138     | Velocity, acceleration and deceleration ..... | 65       |
| PULSVEL .....                        | 139     | Setting up communication .....                | 27       |
| <b>Q</b>                             |         | Settings menu .....                           | 77       |
| Quadcounts .....                     | 212     | Colors Editor .....                           | 78       |
| <b>R</b>                             |         | Compiler .....                                | 77       |
| RAMPMIN (31) .....                   | 180     | Interface .....                               | 78       |
| RAMPTYPE (32) .....                  | 180     | Language .....                                | 78       |
| Read variables .....                 | 56      | Maximum number of labels .....                | 77       |
| Recalculation .....                  | 71      | Maximum number of variables .....             | 77       |
| Regular Expression .....             | 54      | SETVLT .....                                  | 143      |
| REGWMAX (38) .....                   | 182     | SETVLTSUB .....                               | 143      |
| REGWMIN (39) .....                   | 182     | Shortcuts .....                               | 25       |
| REPEAT .. UNTIL ..                   | 139     | Show Watch .....                              | 49, 56   |
| Replace .....                        | 53, 54  | Close Watch window .....                      | 57       |
| Reset Arrays .....                   | 64      | Significance of the control parameters .....  | 32       |
| Reset parameters .....               | 64      | Singlestep .....                              | 49, 56   |
| Response times .....                 | 91      | Slave Stop Position .....                     | 75       |
| Restore from file .....              | 63      | Snap on Grid .....                            | 71       |
| REVERS (63) .....                    | 191     | Software Reference .....                      | 89       |
| RS485 connection .....               | 27, 78  | Start Stop Points .....                       | 74       |
| RST ORIGIN .....                     | 140     | Start SyncPos                                 |          |
| Run programs in several VLTs .....   | 55      | in MCT10 mode .....                           | 83       |
| <b>S</b>                             |         | Starting SyncPos .....                        | 26       |
| Safety regulations .....             | 4       | STAT .....                                    | 144      |
| Safety tips .....                    | 26      | Sub programs .....                            | 89       |
| Sampling time .....                  | 34      | SUBMAINPROG .. ENDPROG .....                  | 144      |
| Save                                 |         | SUBPROG name .. RETURN .....                  | 145      |
| include source code .....            | 59      | Supply voltages .....                         | 12       |
| SAVE part .....                      | 140     | SWAPMENC .....                                | 145      |
| Save part                            |         | SWNEGLIMACT (19) .....                        | 176      |
| ARRAYS .....                         | 140     | SWPOSLIMACT (20) .....                        | 176      |
| AXPARS .....                         | 140     | Symbol bar .....                              | 22       |
| GLBPARS .....                        | 140     | SYNCCURV (55) .....                           | 152, 188 |
| Save to file .....                   | 63      | SYNCC .....                                   | 146      |
| SAVEPROM .....                       | 140     | SYNCCMM .....                                 | 146      |
| Scan Range .....                     | 78      | SYNCCMS .....                                 | 147      |
| Search via Index .....               | 79      | SYNCCSTART .....                              | 148      |
| Sequential command processing .....  | 89      | SYNCCSTOP .....                               | 148      |
| SET .....                            | 141     | SYNCERR .....                                 | 146      |
| Set axis parameters online .....     | 61      | SYNCFAC (49) .....                            | 185      |
|                                      |         | SYNCFAC (50) .....                            | 186      |
|                                      |         | SYNCFAC (57) .....                            | 152, 188 |
|                                      |         | SYNCM .....                                   | 150      |

|                                                |          |                                               |     |
|------------------------------------------------|----------|-----------------------------------------------|-----|
| SYNCMARKM (52) .....                           | 187      | TIMER (14) .....                              | 175 |
| SYNCMARKS (53) .....                           | 187      | Tips for Increasing Program Readability ..... | 89  |
| SYNCMMERR / SYNC SMERR .....                   | 152      | tolerated position error is exceeded .....    | 31  |
| SYNCMMHIT / SYNC SMHIT .....                   | 152      | TRACKERR .....                                | 157 |
| SYNCMPULSM (58) .....                          | 189      | Type of Curve .....                           | 75  |
| SYNCMPULSS (59) .....                          | 189      | <b>U</b>                                      |     |
| SYNCMSTART (62) .....                          | 190      | Upload source code .....                      | 59  |
| SYNCMTYPM (60) .....                           | 190      | User unit .....                               | 95  |
| SYNCMTYPS (61) .....                           | 190      | Using the mouse .....                         | 24  |
| SYNCMWINM (68) .....                           | 194      | <b>V</b>                                      |     |
| SYNCMWINS (69) .....                           | 194      | Variables .....                               | 92  |
| SYNCP .....                                    | 151      | Maximum number of variables .....             | 77  |
| SyncPos motion controller messages .....       | 200      | Special Variables .....                       | 92  |
| ... messages in detail .....                   | 201      | Variable Identification Names .....           | 92  |
| SyncPos parameters in Alphabetical Order ..... | 168      | VEL .....                                     | 158 |
| SyncPos Parameters in Detail .....             | 170      | VELMAX (1) .....                              | 170 |
| SyncPos Window .....                           | 22       | VELMAXQC (9) .....                            | 173 |
| SyncPos-Parameter .....                        | 163      | Velocity Forward Feed FFVEL (36) .....        | 33  |
| SYNCPOSOFFS (54) .....                         | 187      | Velocity Limit .....                          | 76  |
| SYNCREADY (55) .....                           | 152, 188 | VELRES (22) .....                             | 177 |
| SYNCSTAT .....                                 | 152      | VLT control card terminals .....              | 10  |
| SYNCSTATCLR .....                              | 153      | VLT parameters .....                          | 163 |
| SYNCV .....                                    | 153      | VLT software SyncPos messages .....           | 205 |
| SYNCVELREL (66) .....                          | 193      | <b>W</b>                                      |     |
| SYNCVFTIME (65) .....                          | 192      | WAITAX .....                                  | 158 |
| Syntax check .....                             | 57       | WAITI .....                                   | 159 |
| System Control 700 .....                       | 166      | WAITNDX .....                                 | 159 |
| SYSVAR .....                                   | 154      | WAITP .....                                   | 160 |
| <b>T</b>                                       |          | WAITT .....                                   | 160 |
| Table of Messages .....                        | 200      | What to do if ...                             |     |
| Tangent Points .....                           | 38, 73   | encoder doesn't work .....                    | 30  |
| Technical data .....                           | 14       | maximum acceleration is not achieved .....    | 36  |
| Technical Reference .....                      | 218      | motor doesn't move .....                      | 31  |
| Terminal description .....                     | 10       | motor sets off uncontrolled .....             | 31  |
| Terminals .....                                | 14       | motor vibrates heavily .....                  | 31  |
| Test run program .....                         | 31       | stationary precision is required .....        | 36  |
| Testrun                                        |          | there is a tendency towards instability ..... | 36  |
| execute .....                                  | 66       | tolerated position error is exceeded .....    | 36  |
| Testrun graphs                                 |          | What's new in PC Software ...                 |     |
| Acceleration graph .....                       | 68       | 2.1x / Option Card Software 3.1x .....        | 213 |
| Current graph .....                            | 68       | 2.2x and in corresponding MTC Software? ..... | 215 |
| Positioning graph .....                        | 67       | 2.3x and in corresponding MTC Software? ..... | 217 |
| Velocity graph .....                           | 67       | 2.5x and in corresponding Option card? .....  | 217 |
| Testrun menu .....                             | 65       | WHILE .. DO .. ENDWHILE .....                 | 161 |
| Evaluating motion figures .....                | 67       | Window menu .....                             | 79  |
| Loading test run parameters .....              | 66       | Cascade .....                                 | 79  |
| Repeat .....                                   | 67       | Printing or copying help texts .....          | 79  |
| Save test run parameters .....                 | 36, 66   | Tile vertically .....                         | 79  |
| TESTSETP .....                                 | 156      | <b>Z</b>                                      |     |
| TESTSTART .....                                | 157      | Zero impulse .....                            | 212 |
| TESTTIM (24) .....                             | 178      |                                               |     |
| TESTVAL (25) .....                             | 178      |                                               |     |
| TESTWIN (8) .....                              | 173      |                                               |     |
| Text retrieval .....                           | 79       |                                               |     |
| TIME .....                                     | 156, 157 |                                               |     |



## VLT<sup>®</sup> 5000 / 5000 FLUX



## Instruction Manual

VLT<sup>®</sup> frequency converters **BAUER** geared motors

Danfoss can accept no responsibility for possible errors in catalogues, brochures and other printed material. Danfoss reserves the right to alter its products without notice. This also applies to products already on order provided that such alterations can be made without subsequential changes being necessary in specifications already agreed. All trademarks in this material are property of the respective companies. Danfoss and the Danfoss logotype are trademarks of Danfoss A/S. All rights reserved.

