SEW EURODRIVE

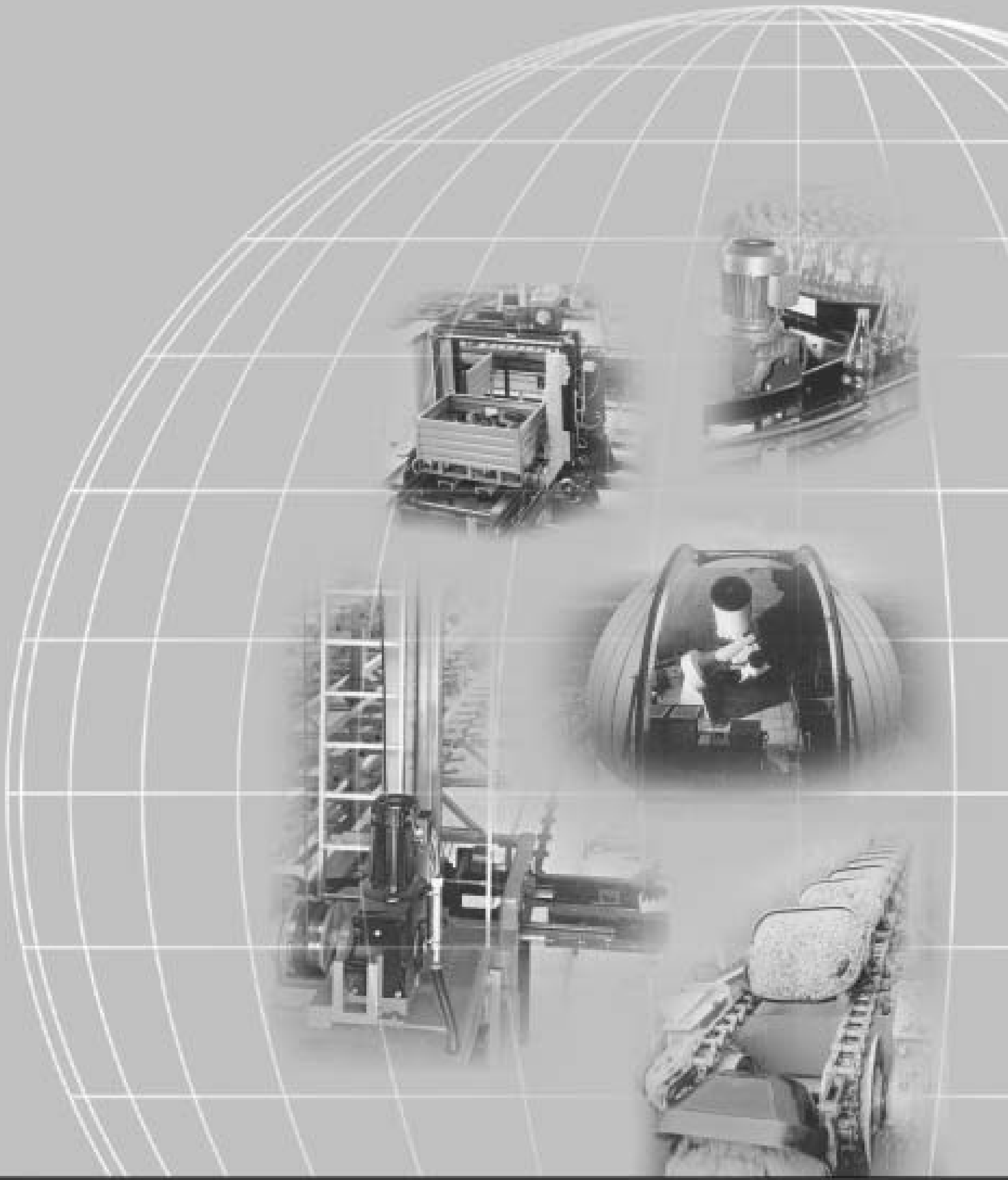
# MOVIDRIVE® MDX61B
# Internal Synchronous Operation (ISYNC)

Edition

02/2004

**Manual**

**1125 2626 / EN**



SEW-EURODRIVE

# 1 Important Notes

- **This manual does not replace the detailed operating instructions!**
- **Only electrical specialists are allowed to perform installation and startup observing applicable accident prevention regulations and the MOVIDRIVE® operating instructions!**

***Documentation***

- Read through this manual carefully before you commence installation and startup of MOVIDRIVE® drive inverters with internal synchronous operation.
- This manual was written assuming that the user has access to and is familiar with the MOVIDRIVE® documentation, in particular the MOVIDRIVE® system manual.
- In this manual, cross references are marked with "→". For example, (→ Sec. X.X) means: Further information can be found in section X.X of this manual.
- A requirement of fault-free operation and fulfillment of any rights to claim under guarantee is that you observe the information in the documentation.

***Bus systems***

**General safety notes on bus systems:**

This communication system allows you to match the MOVIDRIVE® drive inverter to your specific application to a very high degree. As with all bus systems, there is a danger of invisible, external (as far as the inverter is concerned) modifications to the parameters which give rise to changes in the inverter's behavior. This may result in unexpected (not uncontrolled) system behavior.

***Safety and warning notes***

**Always observe the safety and warning notes in this publication!**

**Electrical hazard**
Possible consequences: Severe or fatal injuries.

**Hazard**
Possible consequences: Severe or fatal injuries.

**Hazardous situation**
Possible consequences: Slight or minor injuries.

**Harmful situation**
Possible consequences: Damage to the unit and the environment.

Tips and useful information.

# 2 System Description

## 2.1 Application fields

The internal synchronous operation function enables a group of motors to be operated at a synchronous angle in relation to one another or with an adjustable proportional relationship (electronic gear). Internal synchronous operation is particularly suited to the following sectors and applications:

- **Beverage industry**
  - Filling stations
- **Multiple column hoist**
- **Synchronous material transport**
- **Extruder applications, cutting to length material off the roll**
  - Flying saw
  - Rotating knife
- **Packaging systems**

***Advantages of internal synchronous operation***

- Possibility of position-dependent synchronization → smooth synchronizing without overshooting
- Possibility of position-dependent offset
- Signed input of the master gear factor
- Possibility of synchronizing with a virtual encoder
- Possibility of synchronized SBus connection between master and slave
- Software solution → no option card required

## 2.2 Functional description

The internal synchronous operation function is a special firmware/technology function, which only expects increments from a master. The master can either be

- the X14 input **or**
- any IPOS$^{plus®}$ variable (virtual master drive), for example in conjunction with the SBus or a virtual encoder.

***Synchronization***

The time-controlled synchronization mechanism has been implemented. A variation between the angle of the slave drive and the master drive resulting from free running is reduced to zero.

In addition, a special type of synchronization can be employed. The slave drive moves at a synchronous angle to the master drive following a specified number of master increments (position-dependent synchronization). In this synchronization type, the slave drive moves with a quadratic ramp.

***Synchronous operation***

Synchronous operation comprises various functions. For example, it is possible to operate with a specified offset over a specific travel distance. The offset between the master and slave drive comes into effect after a specified number of master increments.

***Decoupling***

The slave exits synchronous operation. This process can be triggered manually by setting a system variable, or event-driven by an external signal.

## 2.3 State machine of internal synchronous operation

The individual functions of internal synchronous operation are controlled using a state machine. The state machine has the six main states shown in the following figure (→ Sec. 6 "Operating Principle and Functions").

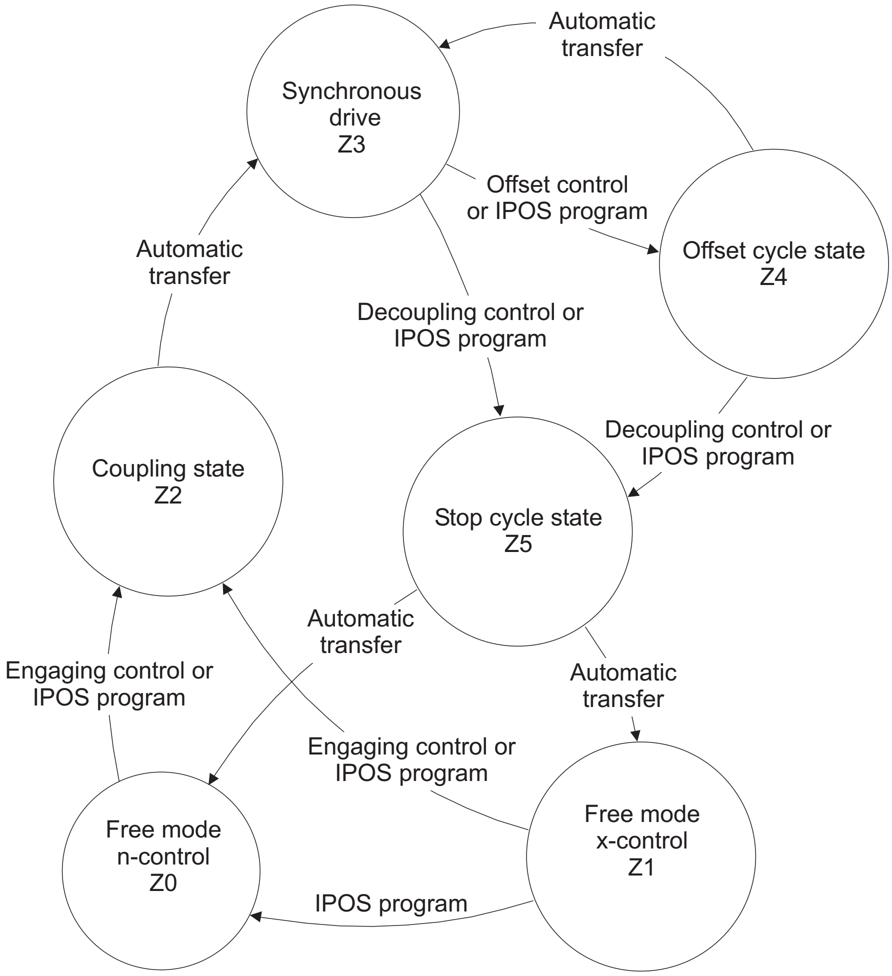



*Figure 1: Overview of the state machine for internal synchronous operation*

***6 Main states***

The state machine distinguishes between the six states Z0 to Z5 (→ Sec. 6 "Operating Principles and Functions").

- **State Z0 = Free mode n-control**

  The slave drive moves in free running mode with speed control. The reference to the master drive can be stored in a difference counter.

- **State Z1 = Free mode x-control**

  The slave drive stops with position control and therefore does not drift out of position. The reference to the master drive can be stored.

- **State Z2 = Coupling state**

  The slave drive is synchronized to the master drive either time controlled or position-dependent.

- **State Z3 = Synchronous drive**

  The slave drive moves synchronously with the master drive.

- **State Z4 = Offset cycle state**

  In synchronous operation, an offset can be set either time-controlled or position-dependent.

- **State Z5 = Stop cycle state**

  The slave drive exits synchronous operation.

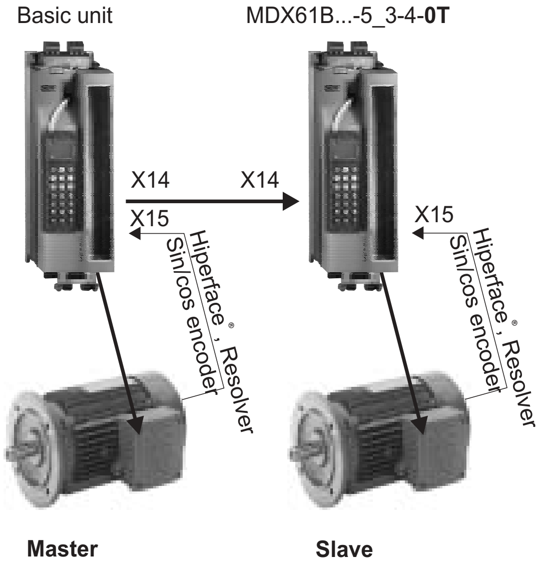
## *2.4 Controlling internal synchronous operation*

Internal synchronous operation is controlled using IPOS$^{plus®}$ variables within the IPOS$^{plus®}$ application program. All states can be viewed and set in a variable range from H360 to H446, which is reserved for internal synchronous operation.

# 3 Project Planning

## 3.1 Application examples

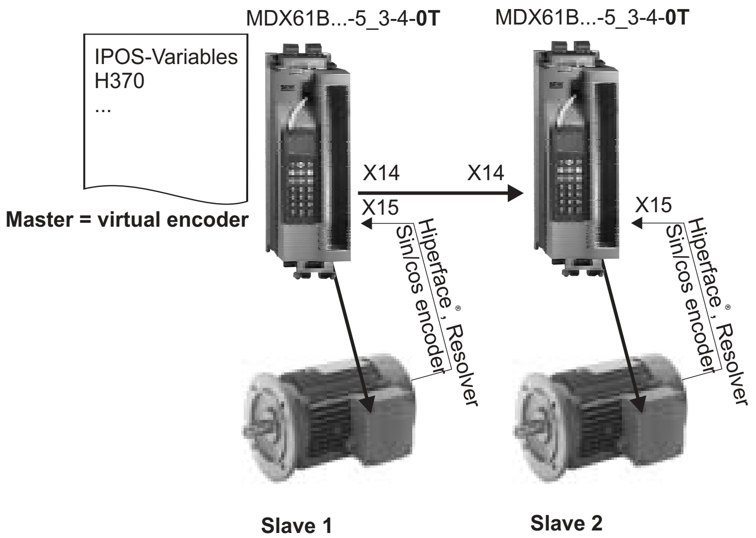
### Master/slave mode of two drives




06671AEN

*Figure 2: Master/slave mode*

### Master/slave mode of two drives with virtual encoder as master




06672AEN

*Figure 3: Master/slave mode with virtual encoder*

***Group configuration: Master and equivalent slaves, e.g. multiple column hoist***

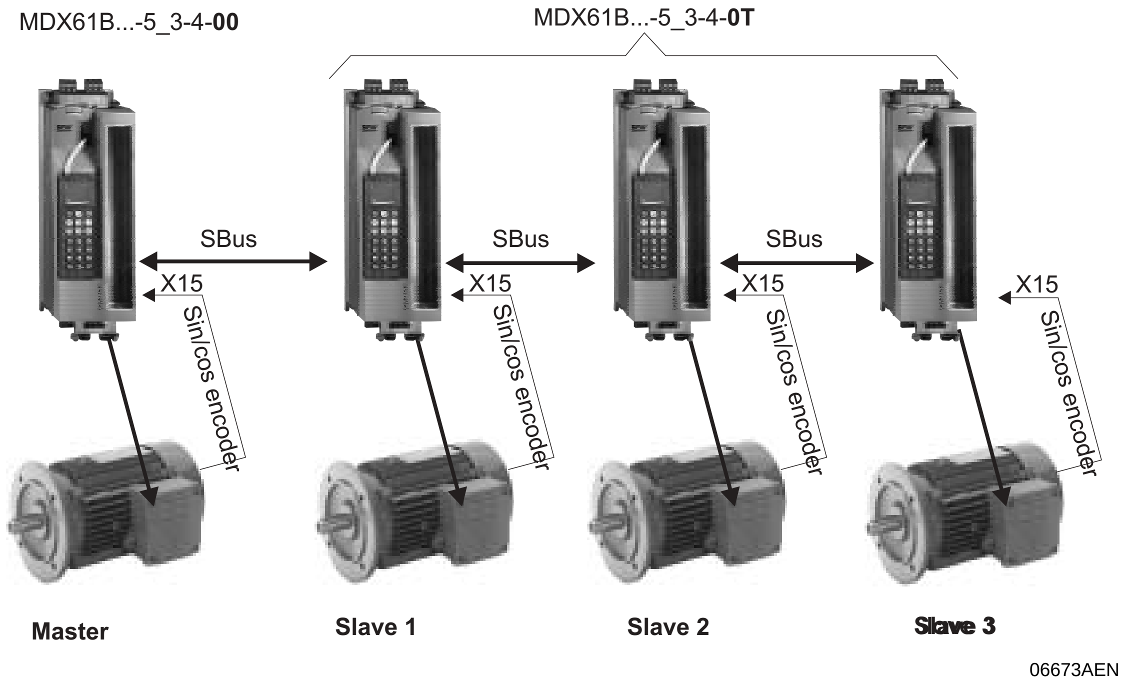



*Figure 4: Group configuration*

***Group configuration with virtual encoder***

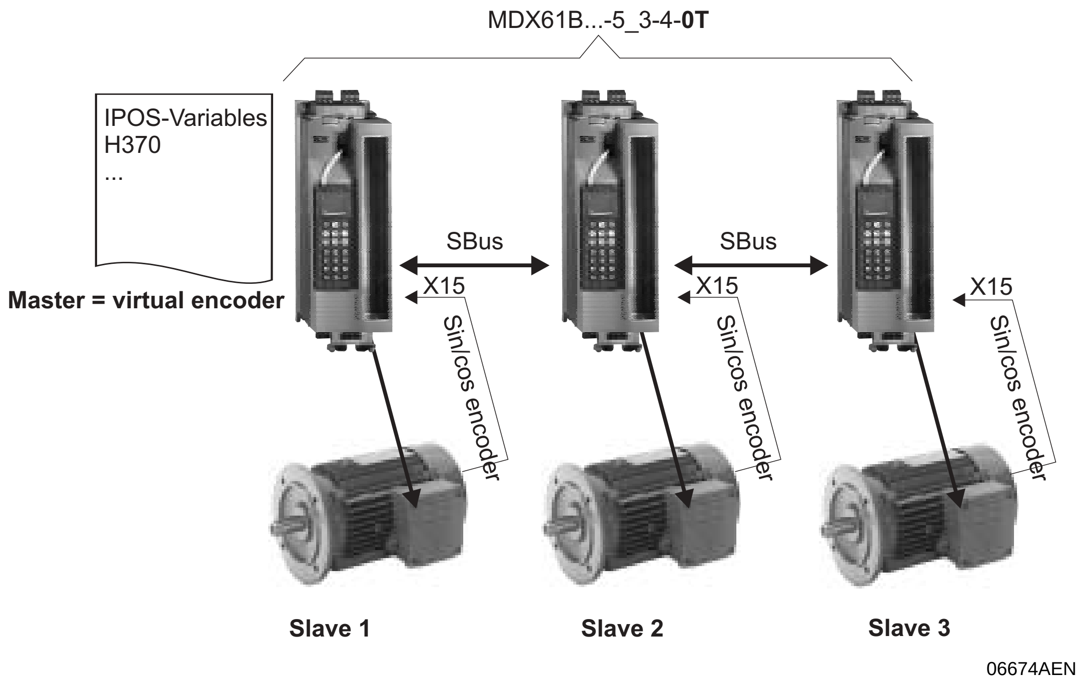
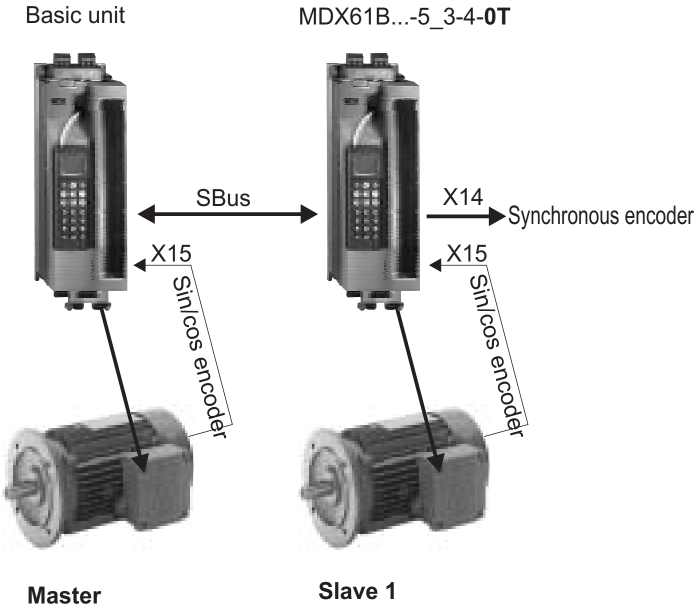



*Figure 5: Group configuration with virtual master encoder*

***Slave drive with slip and absolute encoder***




06679AEN

*Figure 6: Slave drive with slip*

## 3.2 Requirements

***PC and software***

You need the SEW MOVITOOLS® software package to be able to use internal synchronous operation. To use MOVITOOLS®, you must have a PC with one of the following operating systems: Windows® 95, Windows® 98, Windows NT® 4.0 or Windows® 2000.

***IPOS$^{plus®}$ compiler***

The application program for internal synchronous operation must be created using the IPOS$^{plus®}$ compiler. Do not use the assembler (on-screen programming) for this purpose.

IPOS$^{plus®}$ variables H360 to H450 are defined for internal synchronous operation.

***Inverter***

- The MOVIDRIVE® MDX61B...-5_3-4-0T application version already includes the technology function for internal synchronous operation.
- Internal synchronous operation has been implemented for MOVIDRIVE® MDX61B and places the following requirements on the drive system:
  - Encoder feedback
  - "CFC", "SERVO" or "VFC-n control" operating modes with master/slave connection via X14-X14
- Only parameter set 1 is available; parameter set 2 cannot be used.
- The DRS11 synchronous operation card is not supported and therefore may not be used.

Internal synchronous operation **cannot** be realized with

- MOVIDRIVE® MDX60B

***Motors and encoders***

- For operation on MOVIDRIVE® MDX61B:
  - CT/CV asynchronous servomotor, high-resolution sin/cos encoder installed as standard or HIPERFACE® encoder.
  - AC motor DT/DV/D with incremental encoder option, preferably high-resolution sin/cos encoder or HIPERFACE® encoder
  - DS/CM synchronous servomotors, resolver (installed as standard) or HIPERFACE® encoder

High-resolution speed measurement is required for optimum operation of internal synchronous operation. The encoders installed as standard on CT/CV and DS/CM motors fulfill these requirements. If you use DT/DV/D motors, we recommend using HIPERFACE® encoders or high-resolution sin/cos encoders ES1S, ES2S or EV1S.

## 3.3 Project planning notes

- Do not use internal synchronous operation with systems that have a rigid mechanical connection.
- Equip slave inverters with a braking resistor.
- During project planning, bear in mind that the slave must be able to reduce the angle differential between itself and the master to zero at any time. For this reason, set the maximum speed (P302) of the slave to a greater value than the maximum speed of the master by taking into account the scaling factors of master and slave.
- Provide for a sufficient torque reserve for the slave drive.
- During the time-controlled synchronization process, the synchronization speed of the slave drive must be faster than the maximum speed of the master drive.
- If possible, always use the same type of drives for internal synchronous operation.
- In the case of multiple column hoists, always use the same motors and the same gear units (identical ratios).
- When drives of the same type are operating as a synchronized group (e.g. multiple column hoist), then the drive which carries the highest proportion of the load during operation must be selected as the master.
- Connect the slave motor encoder to terminal X15 (ENCODER IN) and the master incremental encoder to terminal X14 (ENCODER IN/OUT) (→MOVIDRIVE® MDX60B/61B operating instructions).
- Master is incremental encoder on terminal X14. Use an incremental encoder with the maximum possible resolution (max. 200 kHz).
- Operation with SBus → Setting up a **cyclical** data transfer in an IPOSplus® program:
  - Group configuration: SBus connection between the master and all slave drives is permitted
  - SBus synchronization with transfer of the SBus synchronization ID
  - Transferring the position of the master drive
- **Direct cable-break monitoring** is possible for X14-X14 connection via the parameter *encoder monitoring X14*. Indirect cable-break monitoring is possible during operation with SBus by way of the SBus timeout response (P836).

## 3.4 Synchronous start / stop

In certain applications such as a two-column hoist, it is essential to make sure that the master and slave can start and stop synchronously. This is a prerequisite for correct operation.

**As a result, combinations in which the master is more dynamic than the slave are not permitted.**

The following table shows the possible master/slave combinations and the required settings for synchronous start/stop.

| Master | Slave | Master parameter | Slave parameter | Comment |
|---|---|---|---|---|
| **MDX61B** | **MDX61B** | DO∅2 = Output stage ON | DI∅3 = Enable / rapid stop (factory setting)<br>DI∅1 and DI∅2 = No function | Connect master binary output DO∅2 to slave binary output DI∅3. |

**Strictly observe the following points:**

- The brake function must be active in the master and the slave (P730 "Brake function 1" = ON).
- With asynchronous motors: The brake release time (P731) of the master must be increased by the premagnetizing time (P323) of the slave drive.

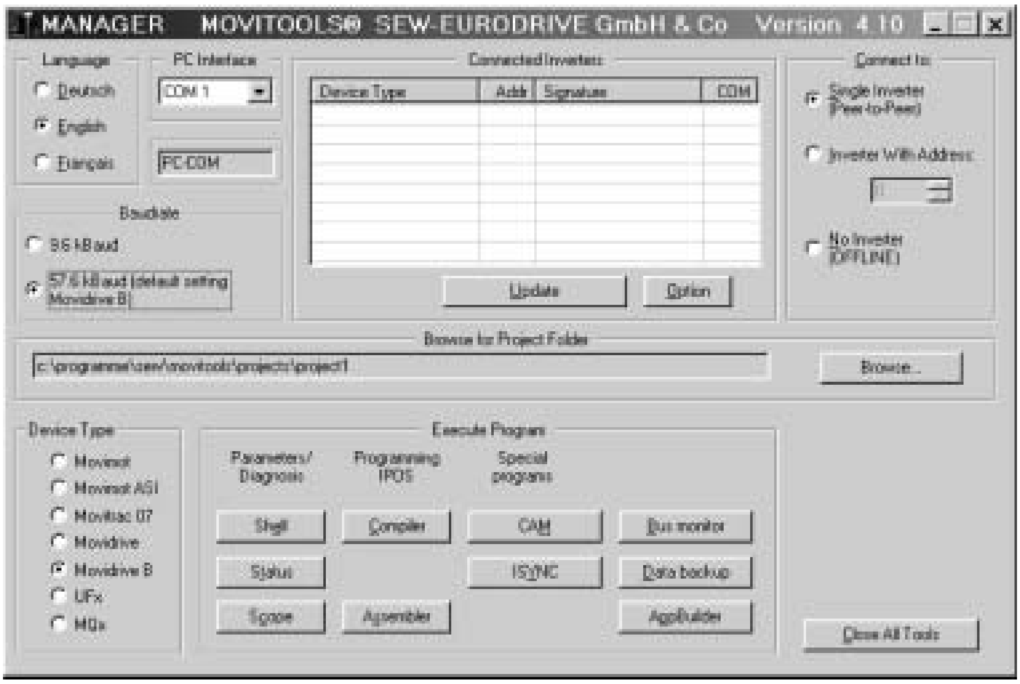
# 4 Installation

## *4.1 Software*

Proceed as follows to install MOVITOOLS® 4.10 on your computer:

1. Insert the MOVITOOLS® CD into the CD ROM drive of your PC.
2. Select "Start\Run...".
3. Type "{Drive letter of your CD drive}:setup" and press the Enter key.
4. The MOVITOOLS® setup menu appears. Follow the instructions of the installation wizard. You will be automatically guided through the installation.

You can now use the program Manager to start MOVITOOLS®. If a MOVIDRIVE® B unit is connected to your PC, select the correct port (PC COM port) and select peer-to-peer connection. Select <Update> to display the inverter in the "Connected Inverters" window.




10383AEN

*Figure 7: MOVITOOLS® startup window*

## 4.2 Connection of encoder and resolver

STOP

The following wiring diagrams do not show the view onto the cable end but

- with plugs / sockets: View on motor plugs/sockets
- with unit sockets: View on unit socket

The conductor colors specified in the wiring diagrams are in accordance with IEC 757 and correspond to the conductor colors used in the pre-fabricated cables from SEW.

For more details, refer to the "SEW Encoder Systems" manual, which can be obtained from SEW-EURODRIVE.

***General installation notes***

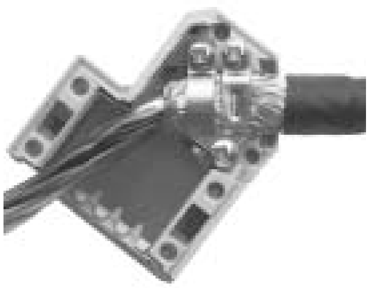
- Max. line length of inverter - encoder/resolver: 100 m (330 ft) with a capacitance per unit length ≤ 120 nF/km (193 nF/mile).
- Core cross section: 0.20 ... 0.5 mm$^2$ (AWG 24 ... 20)
- If you cut off a core of the encoder/resolver cable: Isolate the cut-off end of the core.
- Use shielded cables with twisted pairs of insulated conductors and connect the shield at both ends
  - to the encoder in the cable gland or in the encoder plug
  - to the inverter in the housing of the sub D plug
- Route the encoder/resolver cable separately from the power cables.

***Shielding***

Connect the shield of the encoder/resolver cable over a large area.

*On the inverter*

Connect the shield on the inverter end in the housing of the sub D plug.

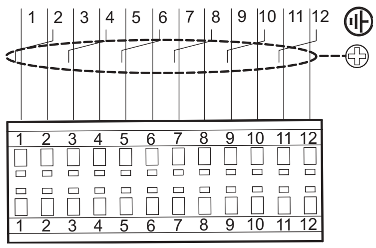


01939BXX

*Figure 8: Connect the shield in the sub D connector*

*On the encoder/ resolver*

Connect the shield on the encoder/resolver side only on the respective earthing clamps, not on the cable gland. For drives with plug connector, connect the shield on the encoder plug.




52392AXX

*Figure 9: Connect the shield on the cable gland of the encoder*

***Prefabricated cables***

- SEW-EURODRIVE offers prefabricated cables for connecting encoders/resolvers. We recommend using these prefabricated cables.

## 4.3 Connecting incremental encoder master to MOVIDRIVE® slave

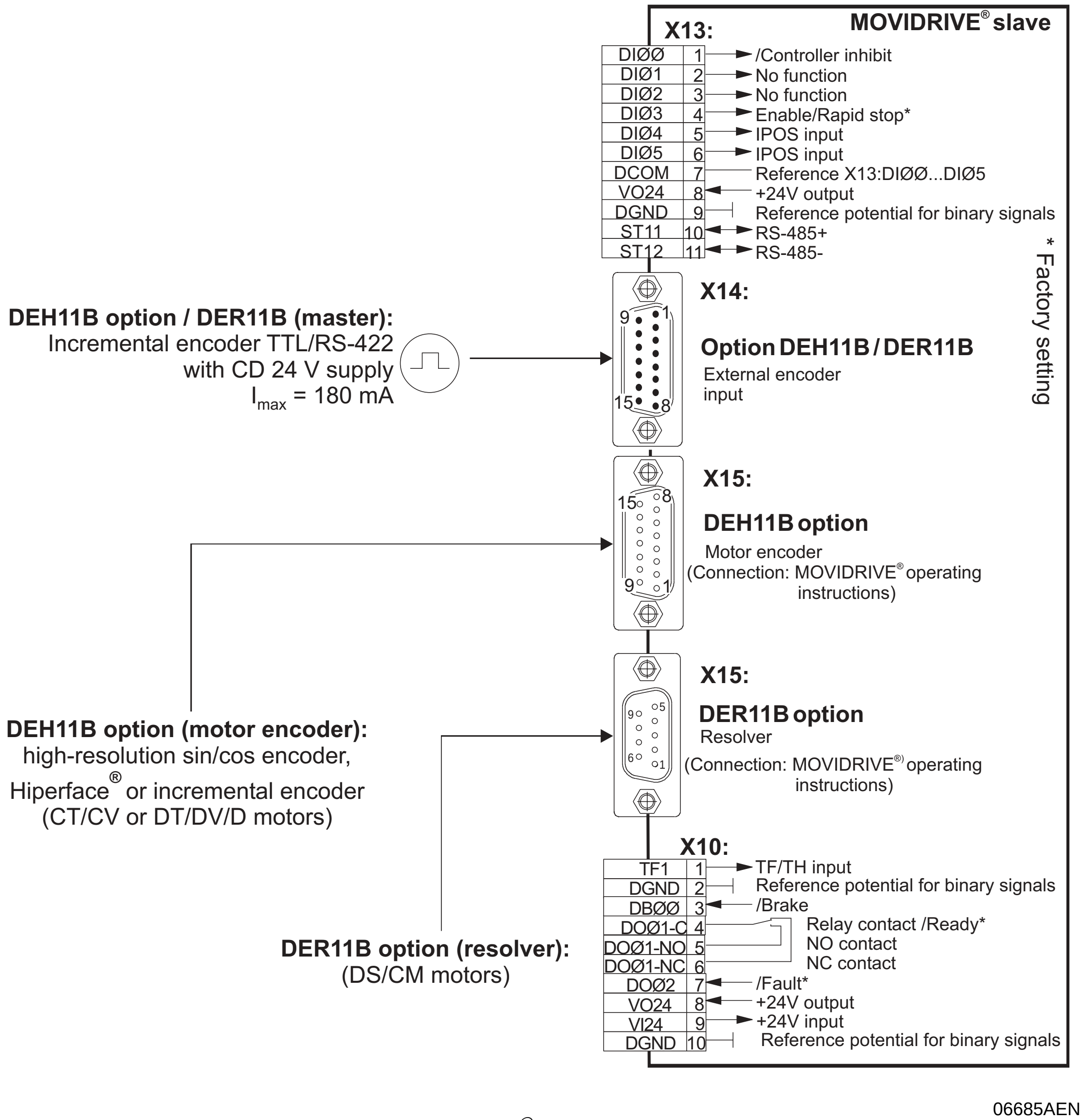
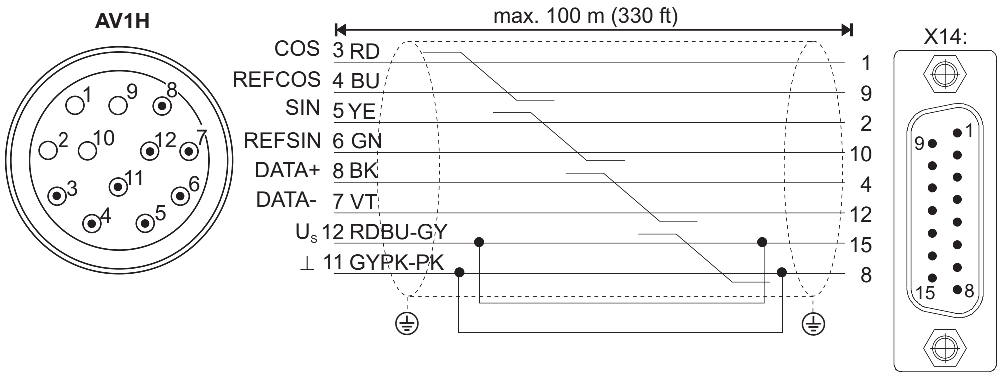



*Figure 10: Connecting incremental encoder master to MOVIDRIVE® slave*

The two options DEH11B and DER11B on X15 **cannot** be connected at the same time.

***Connecting HIPERFACE® encoder AV1H***

Connect the HIPERFACE® encoder AV1H as follows:




06570BXX

*Figure 11: Connecting HIPERFACE® encoder AV1H as external encoder to MOVIDRIVE® master*

Part numbers of the prefabricated cables:

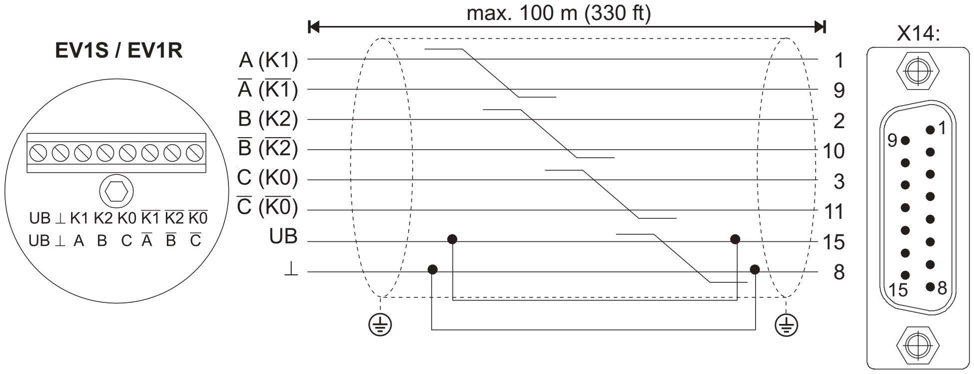
- For fixed routing: 818 015 6
- For cat track routing: 818 165 9

Part numbers of the prefabricated extension cables:

- For fixed routing: 199 539 1
- For cat track routing: 199 540 5

***Sin/cos encoder connection***

Connect the sin/cos encoder as follows:



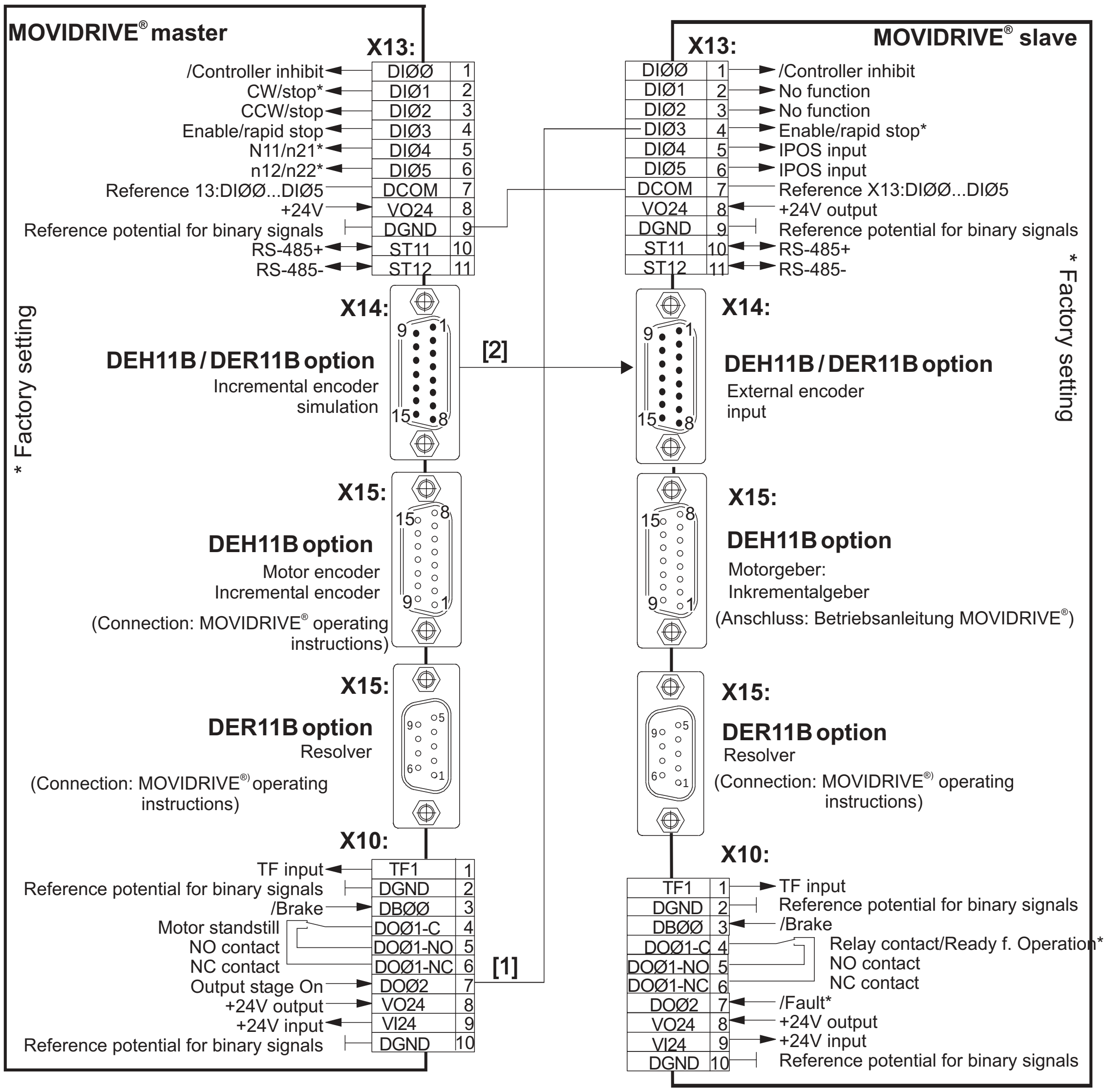

06230CXX

*Figure 12: Connecting sin/cos encoder as external encoder to MOVIDRIVE® master*

Part numbers of the prefabricated cables:

- For fixed routing: 817 960 3
- For cat track routing: 818 168 3

## 4.4 Connecting MOVIDRIVE® master to MOVIDRIVE® slave




06684AEN

*Figure 13: Connecting MOVIDRIVE® master to MOVIDRIVE® slave and necessary connection for synchronous start/stop*

1 Necessary connection for synchronous start/stop (→ Sec. Synchronous start/stop)

2 For the X14-X14 connection, it is essential to comply with section "X14-X14 connection" and the following instructions!

The two options DEH11B and DER11 on X15 **cannot** be connected at the same time.

*X14 X14 connection*

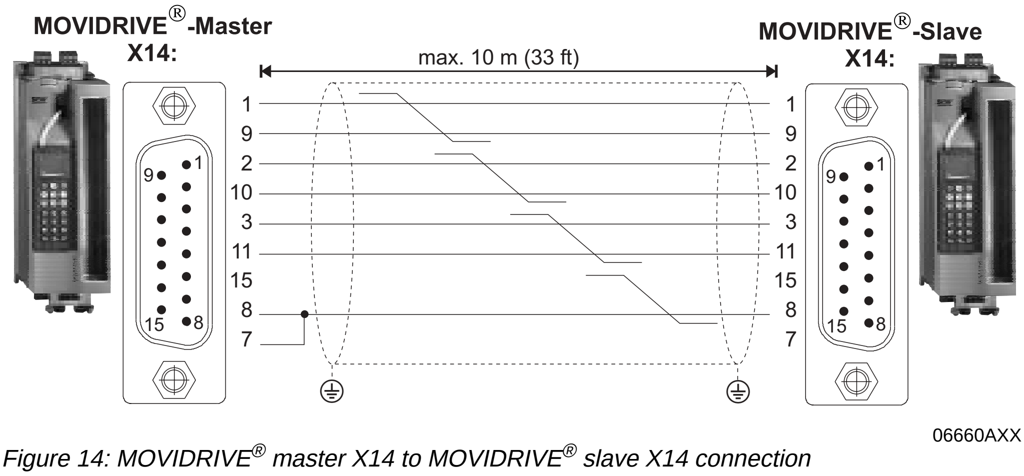



*Figure 14: MOVIDRIVE® master X14 to MOVIDRIVE® slave X14 connection*

**Please note:**

- With MOVIDRIVE® master: Jumper X14:7 with X14:8.
- **Do not** connect MOVIDRIVE® master X14:15 and MOVIDRIVE® slave X14:15.
- SEW-EURODRIVE offers a prefabricated cable for a straightforward and trouble-free master (X14) and slave (X14) connection. You can order this cable from SEW-EURODRIVE under part number 817 958 1.

## 4.5 Connecting absolute encoder master to MOVIDRIVE® slave

***Internal DC 24 V supply***

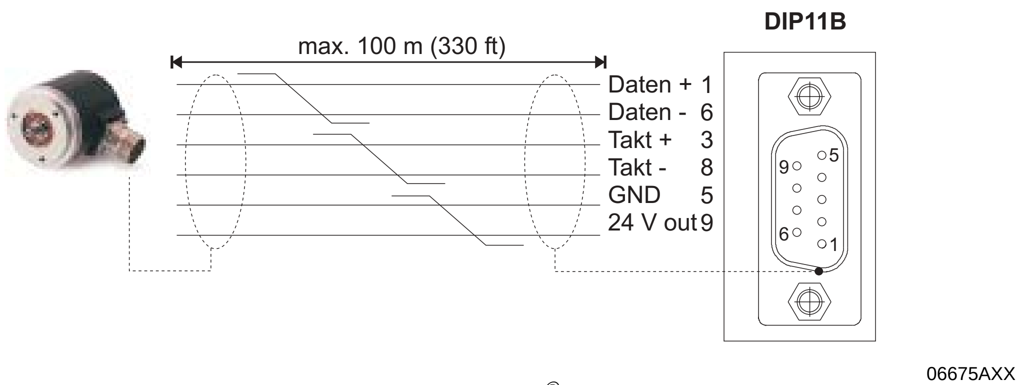



*Figure 15: Absolute encoder master to MOVIDRIVE® slave connection*

***External DC 24 V supply***

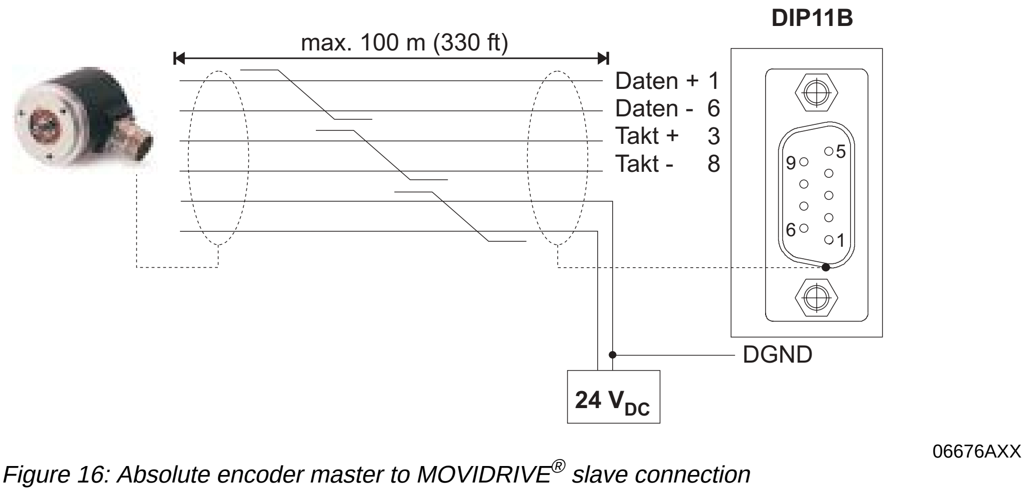



*Figure 16: Absolute encoder master to MOVIDRIVE® slave connection*

***Connecting HIPERFACE® encoder to DT../ DV../D, CT../CV motors***

HIPERFACE® encoders AS1H, ES1H and AV1H are recommended for operation with MOVIDRIVE MDX61B. The encoder is connected using plug connectors.

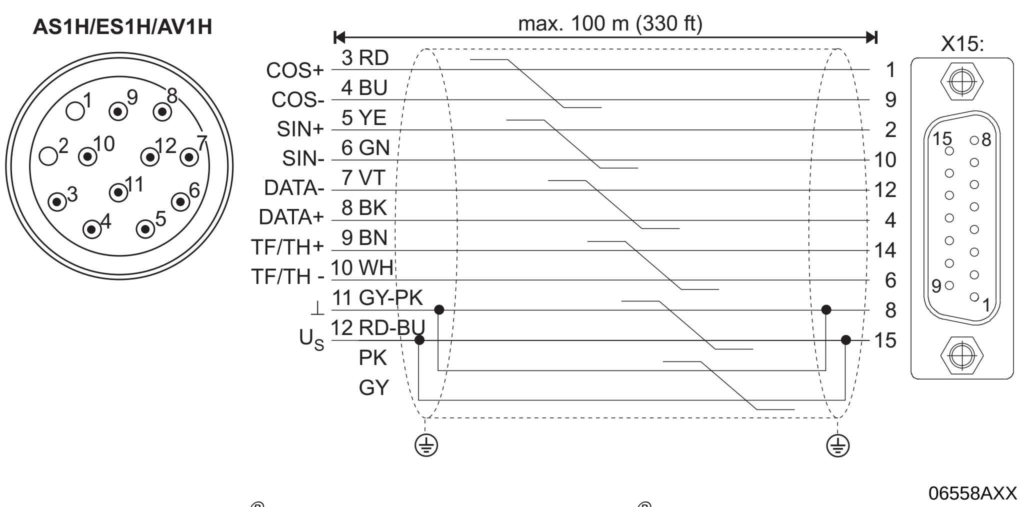



*Figure 17: HIPERFACE® encoder as master to MOVIDRIVE® slave connection*

Important for DT/DV or CT/CV motors: TF or TH is **not** connected with the encoder cable but must be connected using an additional 2-core shielded cable.

Part numbers of the prefabricated cables:

- For fixed routing: 1332 453 5
- For cat track routing: 1332 455 1

Part numbers of the prefabricated extension cables:

- For fixed routing: 199 539 1
- For cat track routing: 199 540 5

## 4.6 *System bus connection of master/slave*

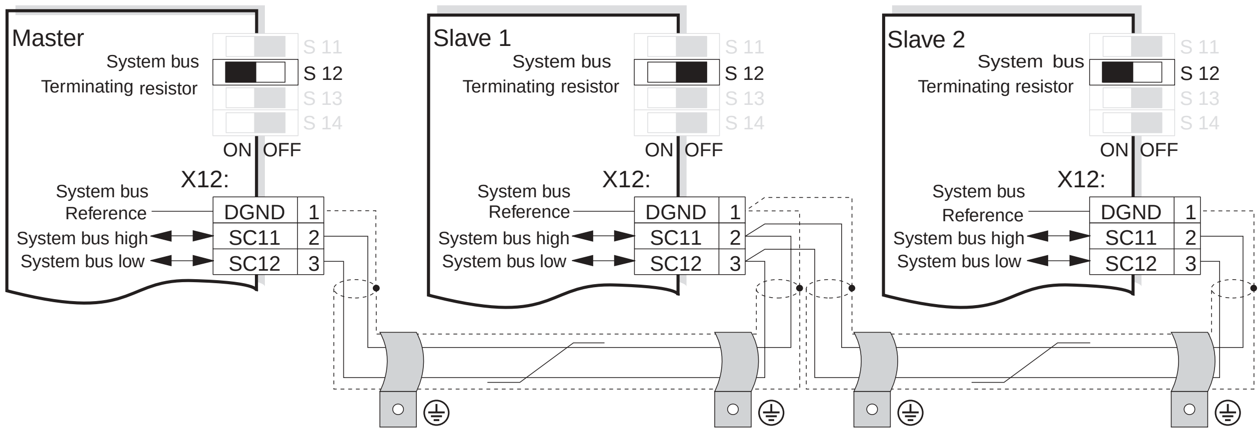
**Only when P816 "SBus baud rate" = 1000 kbaud:**

MOVIDRIVE® *compact* MCH4_A units are not allowed to be combined with other MOVIDRIVE® units in the same system bus combination.

The units are allowed to be combined at baud rates ≠ 1000 kbaud.

Max. 64 CAN bus stations can be interconnected using the system bus (SBus). The SBus supports transmission systems compliant with ISO 11898.

The "Serial Communication" manual contains detailed information about the system bus. This manual can be obtained from SEW-EURODRIVE.




06615AEN

*Figure 18: System bus connection (example: one master and two slaves)*

***Cable specification***

- Use a 2-core twisted and shielded copper cable (data transmission cable with braided copper shield). The cable must meet the following specifications:
  - Conductor cross section 0.75 $mm^2$ (AWG 18)
  - Cable resistance 120 Ω at 1 MHz
  - Capacitance per unit length ≤ 40 pF/m (12 pF/ft) at 1 kHz

  Suitable cables include CAN bus or DeviceNet cables.

***Cable length***

- The permitted total cable length depends on the baud rate setting of the SBus (P816):
  - 125 kBaud → 320 m (1056 ft)
  - 250 kBaud → 160 m (528 ft)
  - **500 kbaud → 80 m (264 ft)**
  - 1000 kBaud → 40 m (132 ft)

***Terminating resistor***

- Switch on the system bus terminating resistor (S12 = ON) at the start and end of the system bus connection. Switch off the terminating resistor on the other units (S12 = OFF).

- There must not be any difference of potential between the units connected with the SBus. Take suitable measures to avoid a difference of potential, such as connecting the unit ground connectors using a separate lead.

***Shield contact***

- Connect the shield at either end to the electronics shield clamp of the inverter or the master control and ensure the shield is connected over a large area. Also connect the ends of the shield to DGND.

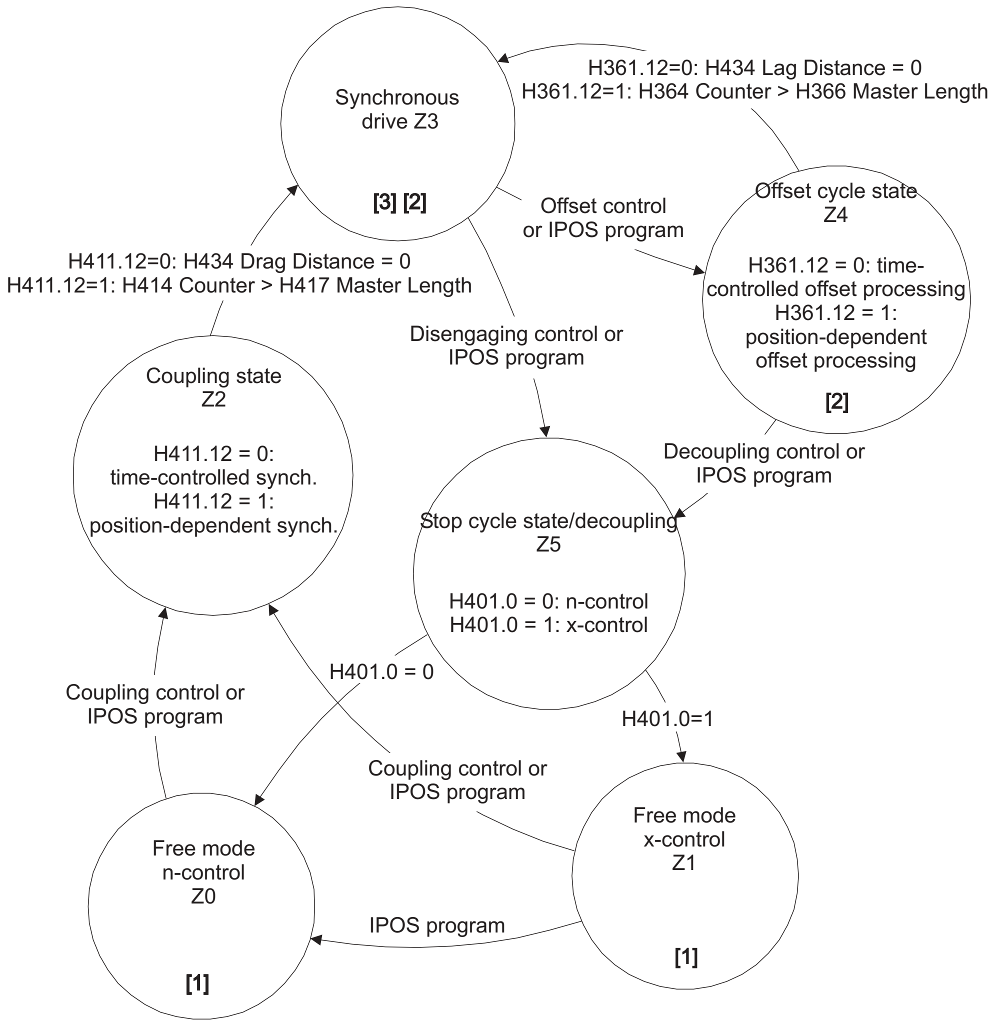
# 5 Operating principle and functions

## *5.1 Controlling internal synchronous operation*

Internal synchronous operation is controlled using IPOSplus® variables within the IPOSplus® program, in the following referred to as "application." All states of internal synchronous operation can be viewed and set in a variable range from H360 to H450 which is reserved for internal synchronous operation (→ Sec. "System variables"). All variables that are connected to internal synchronous operation have symbolic names. These variables are shown below in bold and italics.

## *5.2 Main state machine*

The following figure shows the main state machine and possible state changes of internal synchronous operation (*H427 → SynchronousState*).




06629AEN

*Figure 19: Main state machine of internal synchronous operation with sub-state machines*

[1] Startup cycle state machine
[2] Stop cycle state machine
[3] Offset state machine

***6 main states***

The state machine distinguishes between six states (Z0 to Z5). These states are stored in the ***H427 SynchronousState*** IPOS$^{plus®}$ variable (→ Sec. 7 "System variables").

| State H427 | Description |
|---|---|
| *SynchronousState* = 0 | **Free mode n-control**<br>The slave drive can be moved with speed control at the speed entered in *H439 SpeedFreeMode*. |
| *SynchronousState* = 1 | **Free mode x-control**<br>The slave drive is held in its current position. |
| *SynchronousState* = 2 | **Coupling phase**<br>Synchronization takes place either time-controlled or position-controlled depending on bit 12 in *H411 StartupCycleModeControl*. |
| *SynchronousState* = 3 | **"Hard" synchronous operation**<br>The slave drive follows the master drive at the same angle. |
| *SynchronousState* = 4 | **Offset**<br>The offset is set time-controlled or position-controlled depending on bit 12 in *H361 OffsetCycleModeControl*. |
| *SynchronousState* = 5 | **Decoupling phase (stop cycle)**<br>The slave drive is decoupled with the **t11 ramp (P130)**. |

*Additional functions with H426*

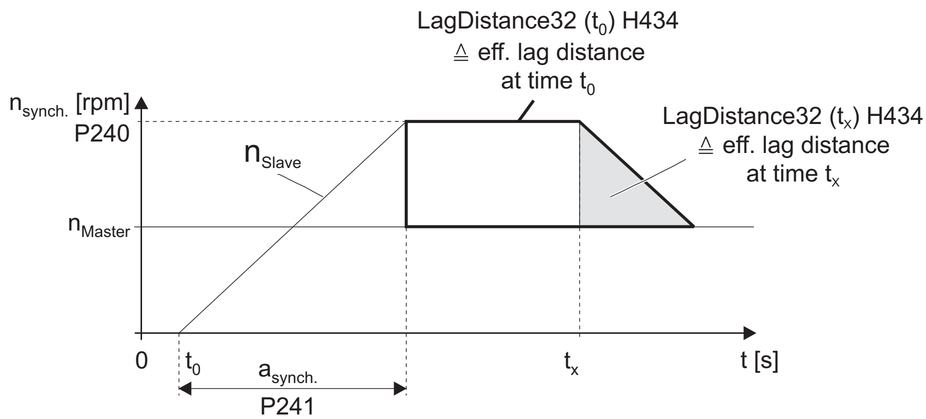
Additional functions can be selected using the bits of the ***H426 SynchronousModeControl*** IPOS$^{plus®}$ variable (→ Sec. 7 "System variables").

| Bit | Name | Description |
|---|---|---|
| 0 | PosTrim | = 0: Deactivated<br>= 1: During position control in free running mode (main state 1); causes the slave drive to move to *TargetPos (H492)* without ramp. This setting should therefore only be used for position corrections. |
| 1 | LagError | = 0: State 3, operating mode not equal to positioning, ramp type not equal to internal synchronous operation → Lag error monitoring.<br>= 1: State 3, operating mode not equal to positioning, ramp type not equal to internal synchronous operation → No lag error monitoring. |
| 2 | RegisterScale | = 0: The values of the correction mechanism are written to the difference counter one-to-one.<br>= 1: The values are scaled with *GFSlave*. |
| 3 | ZeroPointMode | = 0: Feedforward is disabled with "Set DRS zero point".<br>= 1: Feedforward is maintained (speed synchronization with reference to the drive), which means the slave continues to move at the same speed as the master. |

## 5.3 Startup cycle mode control

***Time-controlled synchronization process***

During the time-controlled synchronization process, the existing difference of position between the master and slave drive (64-bit counter) is compensated by accelerating or decelerating the slave drive to the synchronization speed. The time needed depends on the synchronization speed, the synchronization ramp and the lag distance (*H434 LagDistance32*). The following diagram shows the speed profile of the slave drive during the entire process, e.g. at a constant master speed.




06630AEN

*Figure 20: Speed profile of the time-controlled synchronization process*

The synchronization speed $n_{synch}$ and the synchronization ramp $a_{synch}$ are set using parameters *P240 synchronization speed* and *P241 Synchronization ramp*. These two parameters are also used by the DRS11B synchronous operation card.

Synchronization takes place in two steps:

- First, the speed of the slave drive is adjusted to the speed of the master drive by using a specified ramp.
- In the second step, the still existing angle differential (*H434 LagDistance32*) is reduced to zero by accelerating or decelerating the drive (positional synchronism).

**Observe the following points for controller setting:**

- $n_{max\_Slave}$ (P302) ≥ $n_{synch}$ (P240)

  The output speed of the slave drive must be equal to or higher than the output speed of the master drive.

**Observe the following points during project planning:**

- Provide for a sufficient torque reserve for the slave drive.

***Position-dependent synchronization process***

In this synchronization mechanism, the slave drive moves synchronously with the master drive once the master drive has covered a specified distance. The specified distance must be stored in increments in relation to the master in the *H417 StartupCycleMasterLength* variable. Observe that the slave drive must start with speed zero.

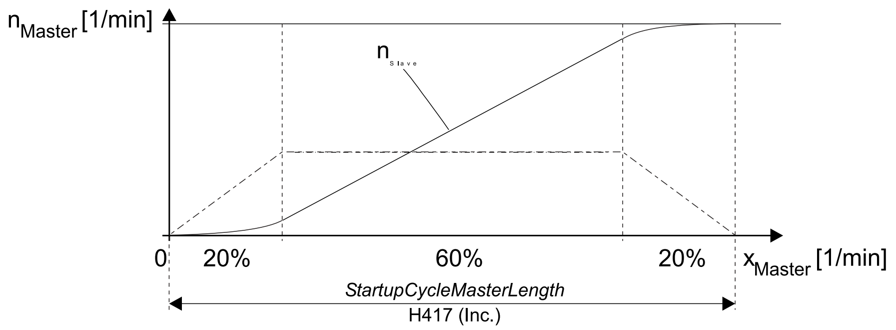



*Figure 21: Speed profile for the position-dependent synchronization process*

Synchronization takes place as follows (with reference to a constant master speed):

- in the range 0 % to 20 % and 80 % to 100 %: the slave drive moves with a quadratic ramp
- in the range 20 % to 80 %: the slave drive moves with a linear ramp

After the coupling process, the slave has covered half the distance of the master (with reference to the drive).

A control element has been added to avoid the loss of master increments during the transition from position-dependent synchronization to synchronous operation. With this element, a differential increment value (*H389 RegisterLoopOut*) in each sampling step is added to the 64-bit difference counter by a certain number of increments (*H390 RegisterLoopDXDTOut*).

The element only takes effect in main state Z3 (synchronous drive) and can be directly written by the user.

The following parameters should be set as given below to achieve exact results in position-dependent synchronization:

- *H390 RegisterLoopDXDXOut* = 2 The remaining travel distance can be reduced to zero.
- *H426 SynchronousModeControl.2* (*RegisterScale - H426*) = 1 results in multiplication with *GFSlave*.

***Startup cycle state machine***

Startup cycle mode control reacts in the main states Z0 and Z1. The startup cycle process of the slave to the master can be performed either manually, event-driven or with interrupt control.

The startup cycle mode is defined with the *H410 StartupCycleMode* system variable. Additional functions can be programmed with the *H411 StartupCycleModeControl* system variable.

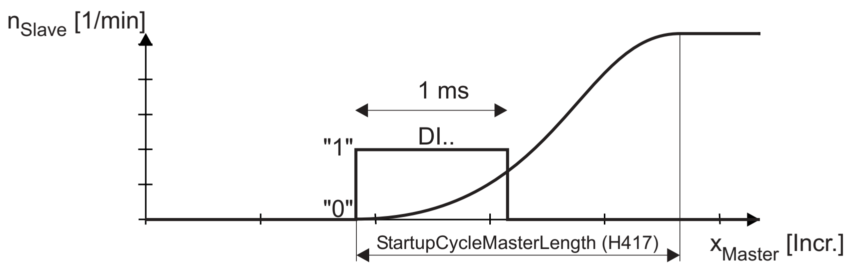
**System variable *H410 StartupCycleMode* → coupling mode:**

- **Manual coupling *(StartupCycleMode = 0)***

  The coupling process starts when the application assigns the value 2 to the *H427 SynchronousState* system variable.

- **Event-driven starting (*StartupCycleMode = 1)***

  Event-driven starting of the coupling process via binary input. The *H413 StartupCycleInputMask* system variable defines which binary input triggers the coupling process. The process is started as soon as a "1" level is present at the defined binary input. The terminal latency is 1 ms.

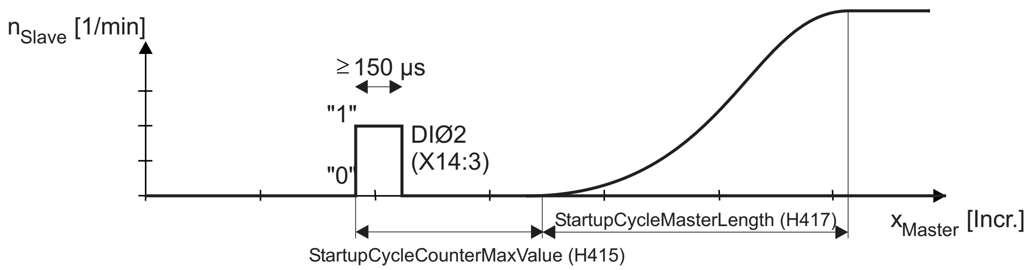



06632AEN

*Figure 22: Event-driven starting of the position-dependent coupling process*

- **Interrupt-controlled coupling *(StartupCycleMode = 2)***

  A signal edge at binary input DI02 or on the C track X14:3 triggers the coupling process (interrupt-controlled). Binary input DI02 must be programmed to "No function" for this purpose. A delay in relation to the master cycle can be defined for the coupling process start with the *H415 StartupCycleCounterMaxValue* system variable. The response time of the sensor can be taken into account with the *H416 StartupCycleDelayDI02* system variable (1 digit = 0.1 ms). This parameter is also effective for coupling with X14:3 (C track).




06633AEN

*Figure 23: Interrupt-controlled starting of the position-dependent coupling process*

**Startup cycle state machine in *H412 StartupCycleState:***

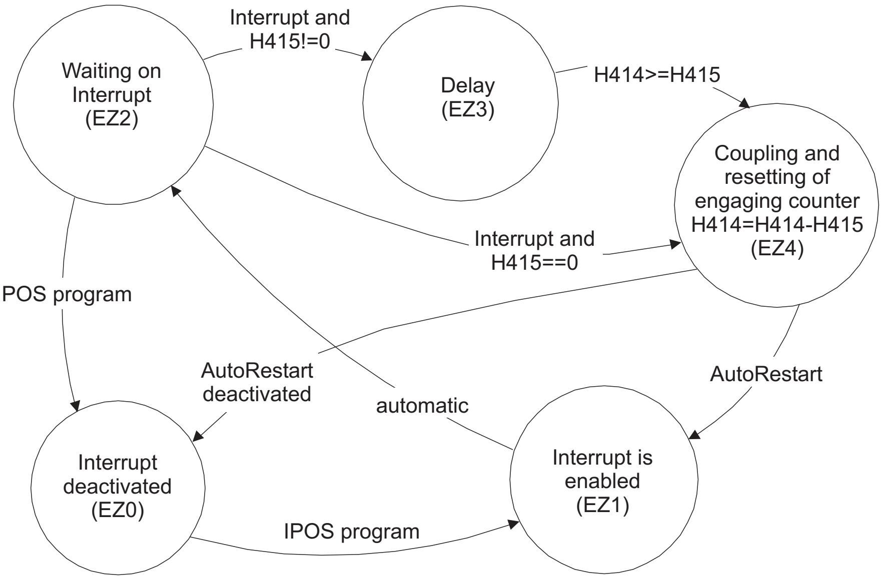



*Figure 24: Startup cycle state machine with interrupt control (startup cycle mode 2)*

*Application example*

Figure 25 shows the following application example for coupling mode 2:

The center of a moving workpiece is to be stamped by a punch. The presence of a workpiece and the starting time for synchronization are determined by a sensor [2], which scans a print mark [3] on the workpiece. Synchronizing, stamping and repositioning [7] are to be performed within one machine cycle [4].

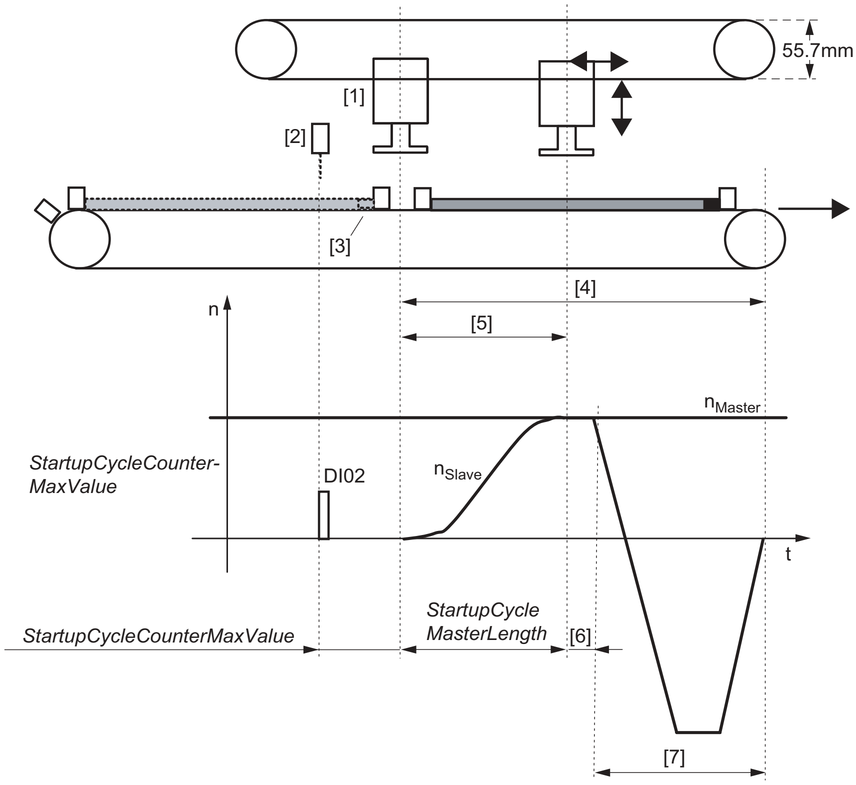
Mechanical structure:

$i_{Master}$ = 10

$i_{Slave}$ = 7

$d_{Master}$ = $d_{Slave}$ = 55.7 mm

Length of a machine cycle = 200 mm




53433AXX

*Figure 25: Application example for startup cycle mode 2*

| | | | |
|---|---|---|---|
| [1] | Start position of the punch | [5] | Half a machine cycle |
| [2] | Sensor | [6] | Synchronous operation |
| [3] | Print mark | [7] | Decoupling and repositioning |
| [4] | Machine cycle | | |

In a position-dependent startup cycle, the slave covers half the distance of the master (half the *H417 StartupCycleMasterLength*). The punch is then positioned above the center of the workpiece after exactly half a machine cycle [5] and the actual stamping process can start.

Determining *H417 StartupCycleMasterLength*:

4096 inc correspond to 1/10 x 55.7 mm x 3.14

(4096 inc x 10) / (55.7 mm x 3.14) = 234.07 inc/mm

→ *StartupCycleMasterLength* = 200 mm x 234.0749 inc/mm = 46815 inc

→ *GFMaster* = 7

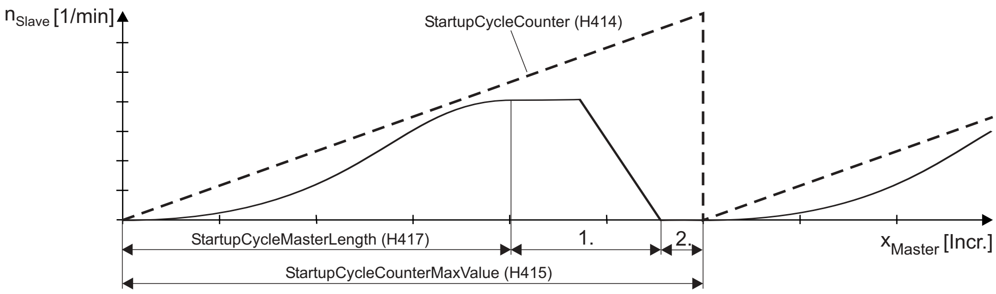
→ *GFSlave* = 10

→ *StartupCycleMode* = 2

→ *StartupCycleCounterMaxValue* = xxx (a delay for the coupling process can be programmed)

- **Position-controlled coupling *(StartupCycleMode = 3)***

  The coupling process is initiated by the *H414 StartupCycleCounter* position counter. Coupling takes place automatically if the *StartupCycleCounter* value is greater than the *H415 StartupCycleCounterMaxValue* counter overrun value. In this case, *StartupCycleCounterMaxValue* must be greater than the total number of input master encoder pulses in the startup cycle, master cycle and stop cycle.



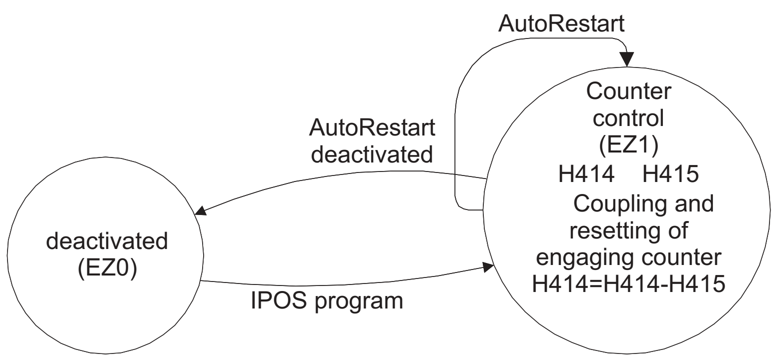

06635AEN

*Figure 26: Position-controlled starting of the position-dependent coupling process*

[1] Synchronous drive and and stop cycle time

[2] Slave is decoupled, time for repositioning to the initial position

**Startup cycle state machine in *H412 StartupCycleState:***




06636AEN

*Figure 27: Startup cycle state machine with position control (startup cycle mode 3)*

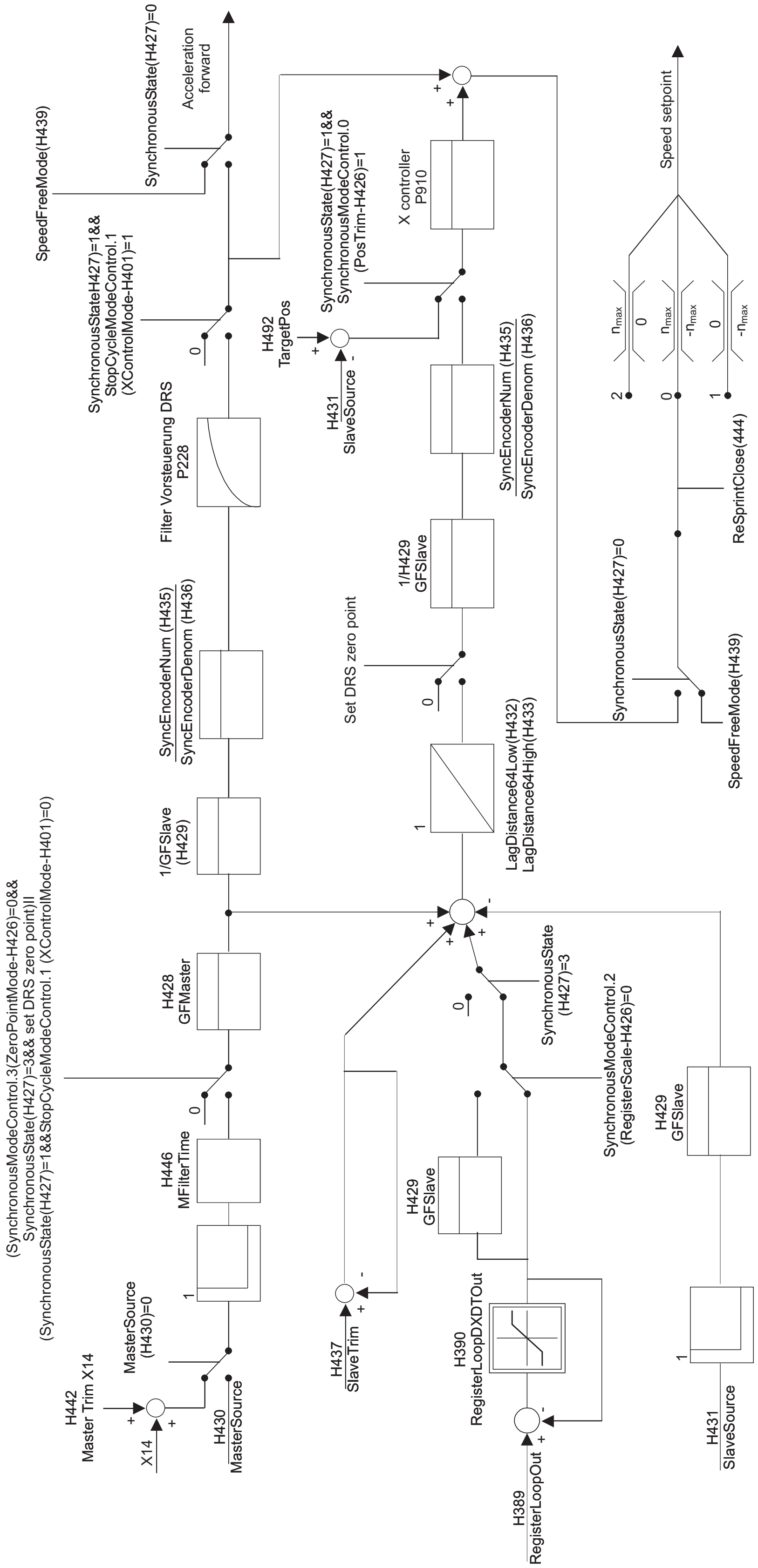
***H411 StartupCycleModeControl* system variable → Additional functions:**

| Bit | Name | Description |
|---|---|---|
| 0 | AutoRestart<br>Modes 2 and 3 | = 0: AutoRestart deactivated.<br>= 1: AutoRestart. |
| 1 | StartupDisable<br>Modes 2 and 3 | = 0: Coupling possible<br>= 1: Coupling disabled |
| 2 | InterruptSelect<br>Mode 2 | = 0: DI02.<br>= 1: X14 C track |
| 12 | StartupMode | = 0: **Time-controlled synchronization** means the existing difference of master and slave positions is eliminated by accelerating the slave drive to the synchronization speed (*P240 synchronization speed*) and the synchronization ramp (*P241 synchronization ramp*).<br>= 1: **Position-dependent synchronization process** means the slave moves synchronously to the master as soon as the master has covered the distance specified in *StartupCycleMasterLength*. **Important: The starting speed of the slave must be zero!** |

## 5.4 Synchronous operation

Control takes place with a P-controller. The master and slave pulses are evaluated with the corresponding scaling factors and added up to a 64-bit value after comparison. The P-controller together with the feedforward and subsequent limiting to the maximum speed provide the speed setpoint for the speed controller.

A control element has been added to avoid the loss of master increments during the transition from position-dependent synchronization to synchronous operation. With this element, a certain difference of increments (*H389 RegisterLoopOut*) in each sampling step is added to the 64-bit difference counter by a certain number of increments (*H390 RegisterLoopDXDTOut*). The element only takes effect in main state Z3 (synchronous drive) and can be directly written by the user.




06694AEN

*Figure 28: Block diagram for internal synchronous operation*

***Correcting function (RegisterScale / RegisterLoop)***

A correction value can be entered via *H389 RegisterLoopOut*, which is added by the difference counter. To avoid speed step changes, this correction value is not added at once but is limited by the value *H390 RegisterLoopDXDTOut* (resolution in inc/ms).

*Example*

A correction value of 10000 inc is to be added to the difference counter within 500 ms:

- 10000 inc / 500 ms = 20 inc/ms
- *H389 RegisterLoopOut* = 10000
- *H390 RegisterLoopDXDTOut* = 20

In this case, the correction value of 10000 inc is reduced at 20 inc/ms, which means the correction lasts 500 ms.

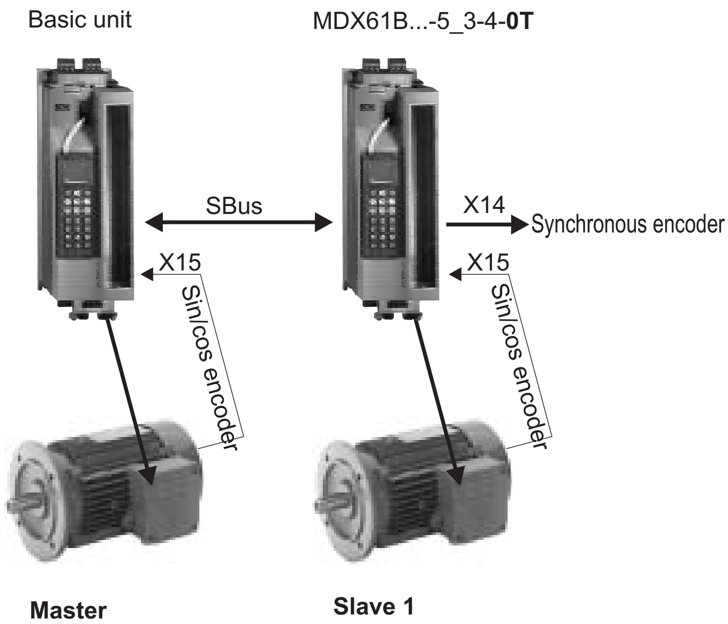
**Please note:**

- A value must be written to the limitation *H390 RegisterLoopDXDTOut*. If no value is written, a correction factor entered in *H389 RegisterLoopOut* will become invalid. The max. correction per millisecond is limited by *H390 RegisterLoopXDTOut* to values of –30000 to 30000.
- The slave scaling factor is taken into account during correction if *H426.2 RegisterScale* = 1. In this case, the correction value entered in *H389 RegisterLoopOut* directly acts on the output shaft. If the slave scaling factor is not to be taken into account, then define *H426.2 RegisterScale* = 0.
- As soon as the value of *H389 RegisterLoopOut* was added to the difference counter, *H389 RegisterLoopOut* is automatically overwritten with the value "0."

***Slave drive with slip***

If synchronous operation is required from a drive that is prone to slip, then the synchronous encoder function must be activated in MOVIDRIVE® B. The ratio between motor encoder and synchronous encoder must be specified as numerator/denominator factor in the IPOSplus® variables *H435 SyncEncoderNum* and *H436 SyncEncoderDenom*:

- Numerator factor (*H435 SyncEncoderNum*):
  - Distance per motor encoder revolution [inc / mm]
- Denominator factor (*H436 SyncEncoderDenom*):
  - Distance per synchronous encoder revolution [inc / mm]




06679AEN

*Figure 29: Hardware configuration*

The source of the slave encoder must be entered in the *H431 SlaveSource* system variable.

| Variable | Name | Value | Meaning |
|---|---|---|---|
| H431 | SlaveSource | = 0 | Source of actual position is X15 |
| | | > 0 | Pointer to variable<br>Example: H431 = 510 // Source of actual position X14 (H510 ActPos_Ext) |

The controller setting can be adjusted via *910 Gain X controller*.

## 5.5 Offset cycle type

Main state Z3 (synchronous drive) is a prerequisite for offset processing.

In this state, an offset is added to the difference counter (main state Z3) during synchronous operation. This means the slave drive receives a new synchronization point and the resulting different angle is reduced to zero by the closed-loop control. Actual synchronous operation is reestablished once the new synchronization point has been reached. The offset value can be positive or negative.

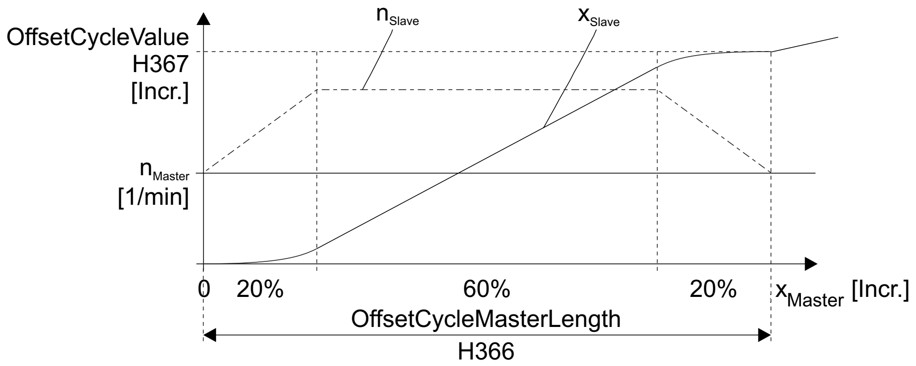
***Time-controlled offset processing***

In this state, an offset is added to the difference counter (*H367, OffsetCycleValue*). The different angle is reduced to zero by accelerating or decelerating the slave drive to the synchronization speed (P240). The slave drive has moved through an offset. The time needed for this process depends on the *synchronization speed (P240)*, the *synchronization ramp (P241)* and the master speed.

***Position-dependent offset processing***

In this state, the slave drive is subject to an offset. The offset value is entered in H367 OffsetCycleValue. The offset is being processed within a certain master distance that is stored in the *H366 OffsetCycleMasterLength* system variable. As a result, the slave is subject to an offset according to the defined master length.

The offset is modified step by step and is added to the value of the difference counter. The offset value is calculated in the following ranges with reference to a constant master speed (→ following figure).




06637AEN

*Figure 30: Speed profile for position-dependent offset processing*

In the range of 0 % to 20 % and 80 % to 100 %, the slave drive moves at variable speeds. In the range of 20 % to 80 %, the slave drive moves at constant speed.

***Offset state machine***

Offset control only reacts to required events in main state Z3 (synchronous drive). The setting is made using the *H360 OffsetCycleMode* system variable. Additional functions can be programmed with the *H361 OffsetCycleModeControl* system variable.

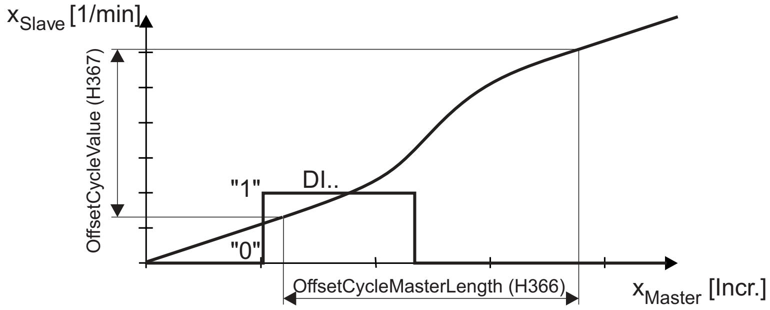
***H360 OffsetCycleMode* system variable → Offset mode:**

- ***OffsetCycleMode = 0***

  Manual offset processing using the IPOS application program by setting the *H427 SynchronousState* system variable to the value 4.

- ***OffsetCycleMode = 1***

  Offset processing using binary inputs ("1" level) with *H363 OffsetCycleInputMask* with a resolution of 1 ms.

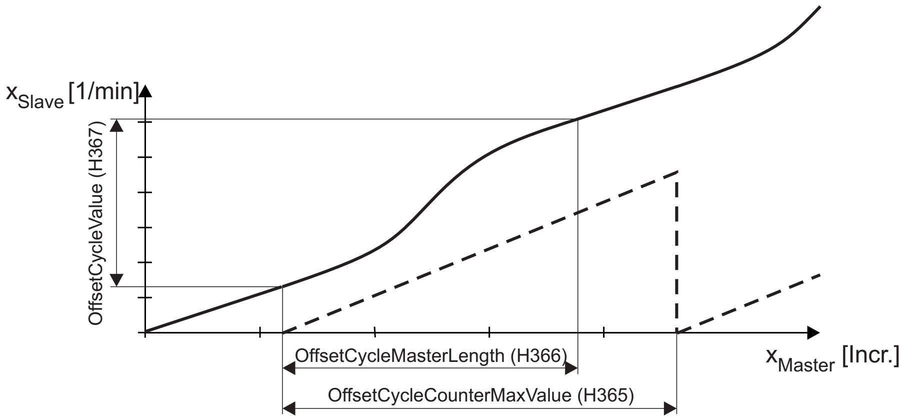



06638AEN

*Figure 31: Position-dependent offset cycle controlled by binary inputs*

- ***OffsetCycleMode = 2 → reserved***
- ***OffsetCycleMode = 3***

  Position control in conjunction with variables *H364 OffsetCycleCounter* and *H365 OffsetCycleCounterMaxValue*), with remaining distance carryover.



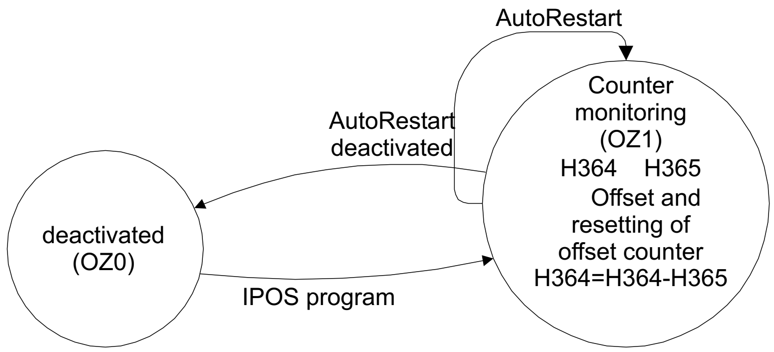

06639AEN

*Figure 32: Position-controlled and position-dependent offset cycle*

### *H360 OffsetCycleModeControl* system variable:

| Bit | Name | Description |
|---|---|---|
| 0 | AutoRestart (mode 3) | The "AutoRestart" function allows you to determine whether the offset cycle is to be triggered only once or several times.<br>= 0: AutoRestart deactivated. The offset cycle by means of position control can be run through exactly once.<br>= 1: AutoRestart activated The offset cycle by means of position control can be restarted after having been run once. |
| 1 | OffsetDisable (mode 3) | The offset cycle by means of position control can be disabled during the initialization phase of the IPOS program. A disabled offset cycle can be enabled again at any time in the main program.<br>= 0: Offset cycle by means of position control is enabled<br>= 1: Offset cycle using by means of position control is disabled |
| 12 | OffsetMode | = 0: Time-controlled offset cycle. An offset is added to the difference counter. By reducing this difference, the slave is moving at an offset (*P240 synchronous speed* and *P241 synchronous ramp* are valid)<br>= 1: Position-dependent offset cycle. The slave drive moves at an offset using the value stored in the *OffsetCycleValue* system value as soon as the master drive has covered the distance defined in the *OffsetCycleMasterLength* system variable. |

### Offset state machine in *H362 OffsetCycleState:*




06640AEN

*Figure 33: Offset state machine*

The offset cycle is triggered by the master increment counter (*H364 OffsetCycleCounter*). The offset is processed automatically if the value of the offset counter is greater than the set counter value (*H365 OffsetCycleCounterMaxValue*).

In the position-dependent offset cycle, the set counter value (*H365 OffsetCycleCounterMaxValue*) must be larger than the master length within which the offset is reduced and the new synchronization point reached. The counter start value can be used to set the master increments counter to an initial value from which this offset counter begins to count.

The offset cycle using position control is configured as a cycle (offset state machine) and can adopt different offset states (OZ0 to OZ1). The offset cycle is deactivated in offset state OZ0 (*H362 OffsetCycleState* = 0).

Value 1 must be assigned to the offset state (*H362 OffsetCycleState* = 1) to start the cycle and to enable offset counter monitoring.

If the offset counter *H364 OffsetCycleCounter* is greater than the set counter value (*H365 OffsetCycleCounterMaxValue*), then the offset is processed and the offset counter is reset. The offset state changes to OZ0 directly afterwards, or remains in OZ1.

## 5.6 Stop cycle state machine

"Decoupling" (stop cycle) means that angular synchronous operation between the slave and master drives is stopped and the slave drive enters the free running state. This means the slave drive can be moved with speed control or held in its current position with position control.

***Stop cycle mode control***

Stop cycle mode control reacts in the main states Z3 (synchronous drive) and Z4 (offset). The stop cycle process of the slave can either be performed manually or automatically. The stop cycle mode is defined in the *H400 StopCycleMode* system variable. Additional functions can be programmed with the *H401 StopCycleModeControl* system variable.

During the stop cycle, the drive changes to speed 0 using *ramp t11 (P130)* with position control; or with speed control, the drive changes to the speed defined in the *H439 SpeedFreeMode* system variable.

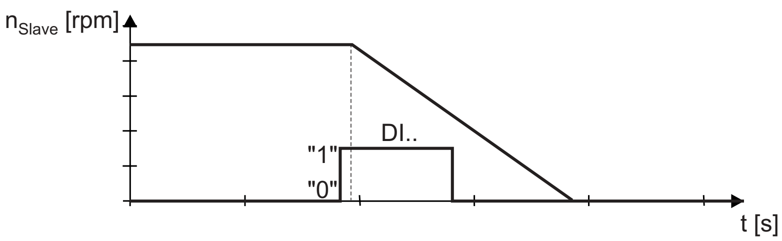
***H400 StopCycleMode* system variable → decoupling mode:**

- ***StopCycleMode = 0***

  Manual decoupling. The slave exits synchronous operation with the master when the application assigns the value 5 to the *H427 SynchronousState* system variable.

- ***StopCycleMode = 1***

  Event-driven decoupling via binary input. The *H403 StopCycleInputMask* system variable defines which binary input triggers the decoupling process. The process is started as soon as a "1" level is present at the defined binary input. The terminal latency is 1 ms.




06641AEN

*Figure 34: Event-driven decoupling*

***H401 StopCycleModeControl* system variable** → additional functions

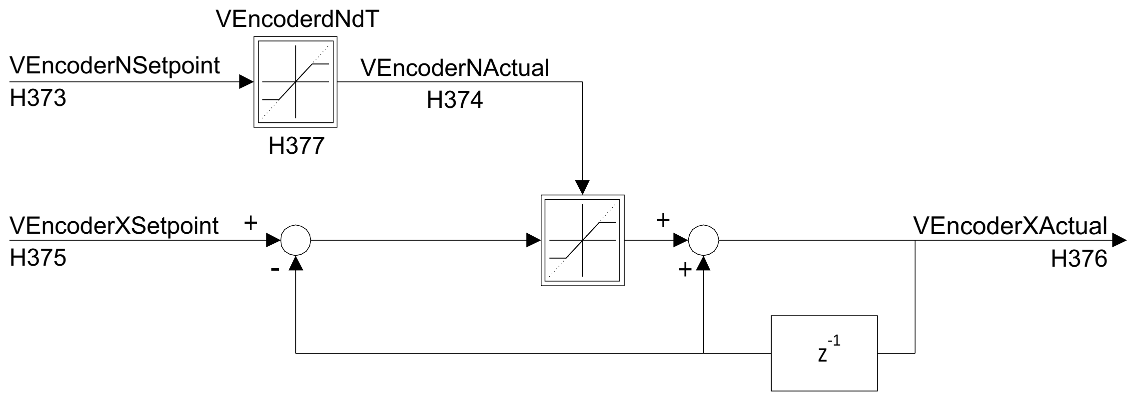
| Bit | Name | Description |
|---|---|---|
| 0 | FreeMode | = 0: Decoupling in main state 0 (speed control)<br>= 1: Decoupling in main state 1 (position control) |
| 1[1)] | XControlMode | = 0: Difference counter is reset<br>= 1: The difference counter stores the difference between master and slave |

1) This setting is only effective in the Z1 main state (position control).

## 5.7 Virtual encoder

The virtual encoder (system variables H370 to H377) is a software counter that can be used as the master encoder for synchronous operation (assignment *MasterSource H430 = H376*). A system bus connection enables the software counter reading to be transferred to other axes. SBus synchronization with the sync-ID (P885) for unit synchronization (every 5 ms) is required for this purpose.

The virtual encoder operates in a 1 ms cycle and is processed independently of the current synchronous operation state. It creates a travel profile depending on the traveling velocity (H373) and the set ramp (H377). The virtual encoder is started by assigning a value other than the actual position (H376) to the target position (H375) and its speed > 0. The virtual encoder is stopped (*VEncoderMode* = 0) when the *H374 VEncoderXActual* value reaches the *H375 VEncoderXSetpoint* value.




06642AXX

*Figure 35: Structural image of virtual encoder with ramp generator*

***Selecting the operating mode of the encoder in variable H370 (VEncoderMode)***

***VEncoderMode* = 3 → Standard positioning mode with adjustable acceleration, deceleration, speed and target position**

| | | |
|---|---|---|
| *H373VEncoderNSetpoint* | [1 inc/ms] | → Set speed |
| *H375VEncoderXSetpoint* | [1 inc] | → Target position |
| *H377VEncoderdNdT* | $[1/12^{16}$ inc/ms$^2]$ | → Acceleration/deceleration |

*H377VEncoderdNdT* is resolved with 16 bits, which means 10000 h must be set to increment by 1 inc/ms.

- Example 1:

The virtual encoder speed (*H373VEncoderNSetpoint)* = 100 inc/ms is to be reached in 15 ms.

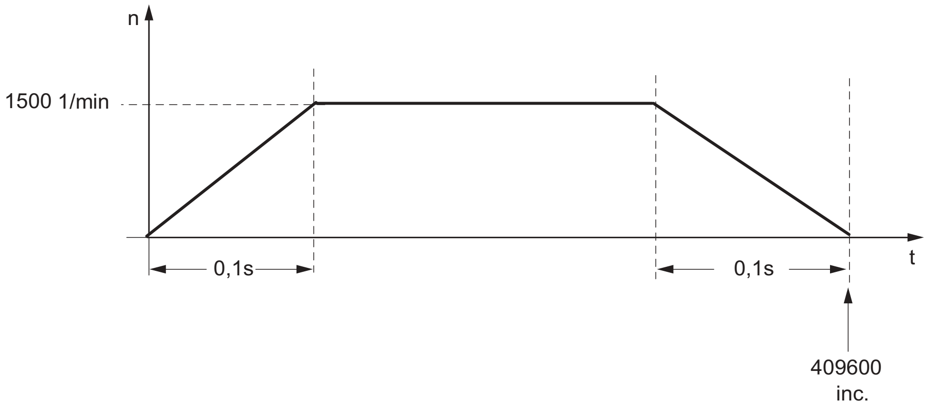
100 inc/ms : 15 ms = 6.6666667 inc/ms$^2$

6.6666667 inc/ms$^2$ x $2^{16}$ = $436906.6667_{dec}$ inc/ms$^2$ = $6AAAA_{hex}$ inc/ms$^2$

→ *H377 VEncoderNdT* = $6AAAA_{hex}$ inc/ms$^2$ (or 110 1010 1010 1010 $1010_{bin}$)

- Example 2:

An axis is to be positioned using a virtual encoder and internal synchronous operation. The speed should be n = 1500 $min^{-1}$, acceleration and deceleration ramps should be 0.1 s. The target position is 409600 inc (= 100 revolutions).




53087AXX

*H373VEncoderNSetpoint* = 1500 $min^{-1}$ x 4096 inc = 6144000 inc/min = 102 inc/ms

→ *H373 VEncoderNSetpoint* **= 102**

*H377VEncoderNdT* = (102 inc/ms : 100 ms) x $2^{16}$ = 66846.72 inc/$ms^2$ (=1051 $E_{hex}$)

→ VEncoderNdT = 66846

The scaling with $2^{16}$ is performed through the startup interface.

**For *VEncoderdNdT* , value 1.02 must be entered.**

(→ 102 inc/ms : 100 ms)

→ *H375 VEncoderXSetpoint* = 409600

→ *H428GFMaster* = *H429 GFSlave* = 1

To avoid jerks, *H377VEncoderdNdT* must be set to at least $10000_{hex}$! The maximum value that can be entered is 32768.

***VEncoderMode* = 2 → Endless counter with adjustable acceleration and speed**

| | | |
|---|---|---|
| *H373VEncoderNSetpoint* | [1 inc/ms] | → Set speed |
| *H377VEncoderdNdT* | [1 inc/ms$^2$] | → Acceleration |

To determine the values of *H373VEncoderNSetpoint* and H377*VEncoderdNdT* → *VEncoderMode = 0*.

***VEncoderMode*** = 2 → Endless counter with adjustable acceleration and speed

***VEncoderMode*** = 0 → Mode with linear, adjustable acceleration, speed, and target position

| | | |
|---|---|---|
| *H373VEncoderNSetpoint* | [1 inc/ms] | → Set speed |
| *H375VEncoderXSetpoint* | [1 inc] | → Target position |
| *H377VEncoderdNdT* | [1 inc/ms$^2$] | → Acceleration |

Even number values are entered in *H373VEncoderNSetpoint* and *H377VEncoderdNdT*. Use highest possible resolution for best results.

The following applies to determining acceleration and speed of the virtual encoder:

$$\textbf{\textit{VEncoderNSetpoint}}\ [\text{inc/ms}] = \frac{n\ [1/\text{min}] \times 4096\ \text{inc} \times \textit{GFSlave}}{6000\ [\text{inc/min}] \times \textit{GFMaster}}$$

n = motor speed

$$\textbf{\textit{VEncoderdNdT}}\ [\text{inc/ms}^2] = \frac{\textit{VEncoderNSetpoint}\ [\text{inc/ms}]}{t_A\ [\text{ms}]}$$

$t_A$ = acceleration time

- Example 1:

An axis is to move synchronously with a virtual encoder. The encoder speed corresponds to 1000 1/min. The axis should accelerate to this speed within 0.5 s. The maximum velocity of the axis should be 3000 1/min, which means it must be possible to increase the velocity to this value.

Conversion from 1/min into inc/ms:

To achieve highest possible resolution, *VEncoderNSetpoint* is set to the maximum input value of 32767 inc/ms.

3000 1/min x 4096 inc = 12288000 inc/min = 204.8 inc/ms

→ Determining *GFSlave*:

$$\textbf{\textit{GFSlave}} = \frac{\textit{VEncoderNSetpoint}\ [\text{inc/ms}]}{v_1\ [\text{inc/ms}]} = \frac{32767\ [\text{inc/ms}]}{204{,}8\ [\text{inc/ms}]} = \mathbf{160}$$

$v_1$ = required velocity

→ Determining *VEncoderNSetpoint*:

$$\textit{VEncoderNSetpoint [inc/ms]} = \frac{\text{n [1/min] x 4096 inc x } \textit{GFSlave}}{\text{60000 [inc/min] x } \textit{GFMaster}}$$

$$\textit{VEncoderNSetpoint [inc/ms]} = \frac{\text{1000 [1/min] x 4096 inc x 160}}{\text{60000 [inc/min] x 1}}$$

$$\textit{VEncoderNSetpoint [inc/ms]} = \textbf{10923 [inc/ms]}$$

→ Determining *VEncoderdNdT*:

$$\textit{VEncoderdNdT}\ \text{[inc/ms}^2\text{]} = \frac{\textit{VEncoderNSetpoint}\ \text{[inc/ms]}}{t_A\ \text{[ms]}} = \frac{\text{10923 [inc/ms]}}{\text{500 [ms]}}$$

$t_A$ = acceleration time

→ *VEncoderNdT* = 22 inc/ms$^2$

→ **Results**:

*H373 VEncoderNSetpoint* = 10923 inc/ms

*H375 VEncoderXSetpoint* = xxx

*H377 VEncoderdNdT* = 22 inc/ms$^2$

*H428 GFMaster* = 1

*H429 GFSlave* = 160

- Example 2:

A speed is to be specified with a resolution of 0.2 1/min. It should be possible to adjust the ramp in a range of 0.5 to 5 s.

The following selection is made:

*H373 VEncoderNSetpoint* = 5000 inc/ms (corresponds to 1000 1/min)

*H377 VEncoderdNdT* = 10 inc/ms$^2$ (1 = 5 s ramp, 10 = 0.5 s ramp)

*H428 GFMaster* = 25

*H429 GFSlave* = 1831

***H371 VEncoderModeControl system variable*: → Additional functions**

| Bit | Name | Description |
|---|---|---|
| 0 | AxisStop | = 0: Axis stop deactivated<br>= 1: The value of *H373 VEncoderNSetpoint* is set to 0 (stop of the virtual axis) once after a unit fault occurs. |

## 5.8 Important notes

- The possibility of specifying a signed distance in variable *H417 StartupCycleMaster-Length* and *H366 OffsetCycleMasterLength* for the master drive means it is important to check the direction of rotation of the master drive. It is also important to note that the scaling factor can also be entered as signed value in *H428 GFMaster*.
- A lag error is only triggered (P923 Lag error window) in main state Z3 (synchronous drive).
- Set zero point: The 64-bit counter can be cleared by programming an input terminal with "Set DRS zero point."
- A value other than zero should be entered in the *H390 RegisterLoopDXDTOut* system variable to achieve exact results in position-dependent synchronization and to allow that the remaining distance can be reduced. In addition, the function *RegisterScale* must be activated so the correction value is multiplied by the scaling factor of the slave.

  → Example:

  *H390 RegisterLoopDXDTOut* = 2

  *H426 SynchronousModeControl.2* (*RegisterScale* – H426) = 1

***Master/slave scaling factors***

The objective of phase-synchronous operation is to move the outputs of two or more drives and thus the line synchronously in relation to one another. The synchronous operation controller required for this purpose only processes information on increments of a master encoder and a slave encoder. This means the actual gear units and additional gear ratios of the application have to be represented by factors. This way, synchronization can be achieved within a certain proportionality ratio.

If two identical drives are used, which means same gear unit reduction ratios, same additional gears, etc., then the proportionality ratio is one-to-one.

If unequal gear unit reduction ratios occur, they are taken into account with the master drive by the scaling factor *H428 GFMaster* and with the slave drive by the scaling factor *H429 GFSlave*.

The *H428 GFMaster* scaling factor evaluates the master increments, which the synchronous operation controller obtains as setpoints. The *GFMaster* scaling factor also includes the slave gear unit reduction ratio, the slave encoder resolution, additional slave gear if present, and the master length.

→ **Calculating *H428 GFMaster***:

*GFMaster* = slave gear unit reduction ratio x additional slave gear x master length

The master length relates to the length of travel performed by the master per revolution of the output.

→ **Calculating *H429 GFSlave***:

*GFSlave* = master gear unit reduction ratio x master additional gear x slave length

The slave length relates to the length of travel performed by the slave per revolution of the output.

- **Master / slave gear unit reduction ratio**

  As a rule, the master or slave gear unit reduction ratio is indicated on the nameplate of the drive. You can either directly read the value or calculate it from the quotient of the rated speed / output speed.

  For forwards and backwards movements in a restricted travel range, it is generally sufficient to scale up the gear unit reduction ratio from the nameplate or obtained by calculation to between two and four decimal places (depending on the maximum possible resolution of the scaling factor).

STOP

**For synchronous operation is recommended that you include the individual numbers of teeth of the gear pairs in the scaling factor calculation. In other words, include the individual gear unit reduction stages in the calculation separately. Your SEW-EURODRIVE contact can tell you the numbers of teeth in the gear pairs.**

- **Master / slave additional gear ratio**

  If there is an additional gear for a further ratio reduction, this additional gear reduction ratio must be treated like an additional gear unit reduction ratio and is also included in the calculation.

- **Master / inverter length**

  The master length relates to the length of travel performed by the master / slave per revolution of the output.

  In many applications, the length of travel is described to a sufficient degree of accuracy by the calculated circumference of the drive wheel. The number can be an irrational number. In the case of endless applications, it is therefore recommended that you calculate the length of travel according to the mechanical system used.

  – Applications with a chain sprocket as transmission element:
  Travel length = Number of teeth of the chain sprocket x chain link length

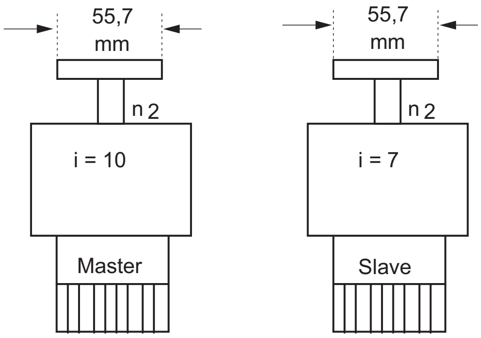
  – Applications with a gear rack as transmission element:
  Travel length = Number of teeth of the gear wheel x tooth clearance (tip-to-tip) of the gear rack

  – Applications with a toothed belt as transmission element:
  Travel length = Number of teeth of the gear wheel x tooth clearance (center-to-center) of the toothed belt

  – Applications with a spindle as transmission element:
  Travel length = Spindle pitch

*Example*

Two drives with different gear unit reduction ratios are to be moved at a synchronous angle to the master drive at the same output speed $n_2$ = 20 1/min.




53088AXX

→ Requirement: $n_{2\_Master} = n_{2\_Slave}$

$$\frac{GF_{Slave}}{GF_{Master}} = \frac{i_{Master} \text{ x Master add. gear x slave length}}{i_{Slave} \text{ x Slave add. gear x master length}}$$

$$\frac{GF_{Slave}}{GF_{Master}} = \frac{10 \text{ x } 1 \text{ x } (3{,}14 \text{ x } 55{,}7 \text{ mm})}{7 \text{ x } 1 \text{ x } (3{,}14 \text{ x } 55{,}7 \text{ mm})} = \frac{10}{7}$$

GFSlave = 10

GFMaster = 7

Two more examples with different output speeds $n_2$ of master and slave are shown in the following table.

| | | Setpoint speed n2 [1/min] | n1 [1/min] | Gear unit reduction ratio i | Scaling factor |
|---|---|---|---|---|---|
| **Example 1** | Master | 20 | 200 | 10 | 7 *GFMaster* |
| | Slave | | 140 | 7 | 10 *GFSlave* |
| **Example 2** | Master | 20 | 200 | 10 | 7 *GFMaster* |
| | Slave | 10 | 70 | 7 | 10 *GFSlave* |
| **Example 3** | Master | 20 | 200 | 10 | 7 *GFMaster* |
| | Slave | 40 | 280 | 7 | 10 *GFSlave* |

The master increments are multiplied by *H446 MFilterTime* disregarding the scaling factors *GFMaster* and *GFSlave* (default setting: *MFilterTime* =1).

# 6 Startup

## 6.1 General information

Correct project planning and installation are the prerequisites for successful startup. Refer to the MOVIDRIVE® MDX60B/61B system manual for detailed project planning instructions.

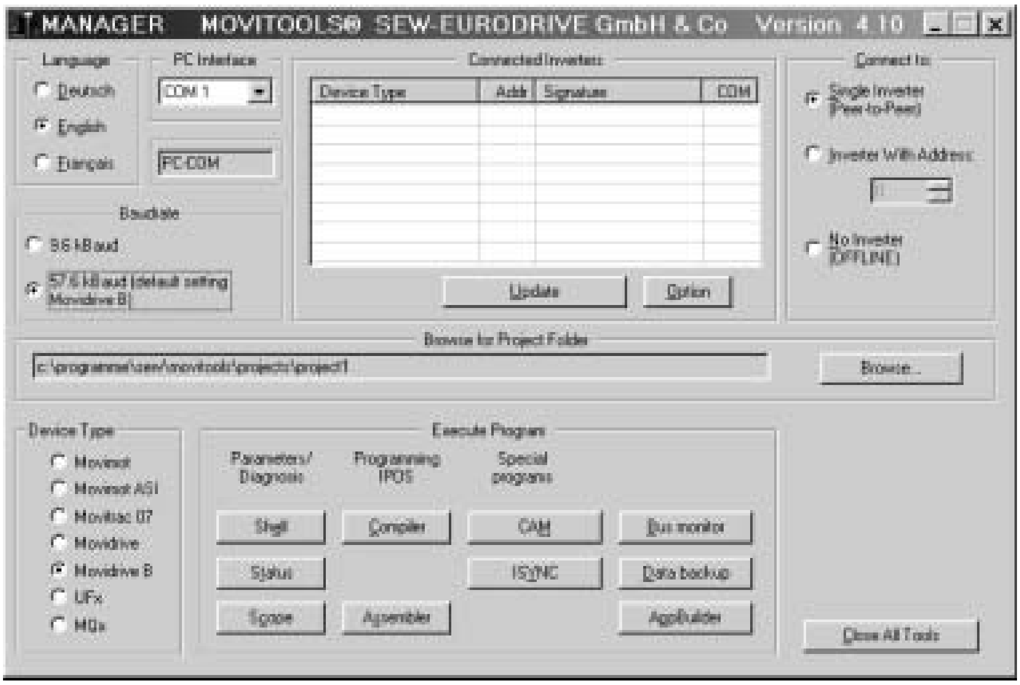
Check the installation and the encoder connection

- by following the installation instructions in the MOVIDRIVE® MDX60B/61B operating instructions
- according to the information in this manual.

## 6.2 Preliminary work

Perform the following steps before the startup of internal synchronous operation:

- Connect the inverter to the PC using the serial interface (RS-232, UWS21A on PC-COM)
- Terminal X13:1 (DIØØ "/CONTROL.INHIBIT") must receive a "0" signal!
- Start MOVITOOLS® 4.10
- Select the language in the "Language" selection field.
- From the "PC-COM" drop down menu, select the PC port (e.g. COM 1) to which the inverter is connected.
- In the "Device type" field, select "Movidrive B".
- In the "Baud rate" field, select "57.6 kBaud".
- Press the <Update> button  to display the connected inverter.



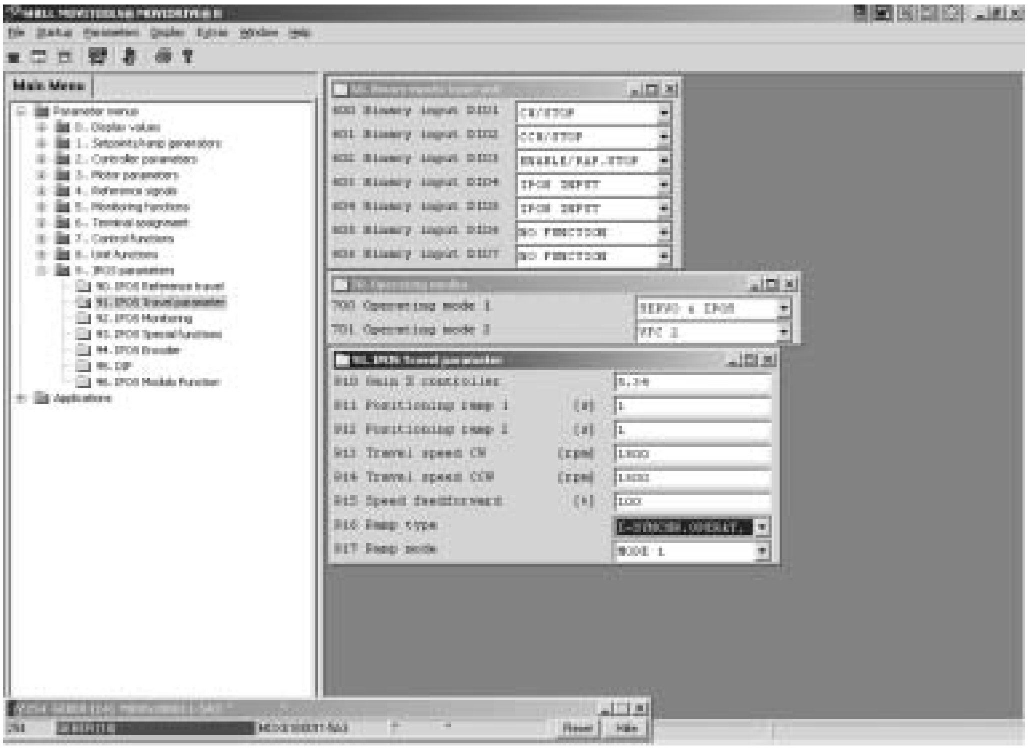

10383AEN

*Figure 36: MOVITOOLS® startup window*

## 6.3 Starting up internal synchronous operation

***General information***

- In the "Execute Program" selection field, press the <Shell> button under "Parameters/Diagnosis". The Shell program is started.
- Set the *P916 Ramp type* parameter to "I-SYNCHR. OPERAT." This setting activates internal synchronous operation. The *P916 Ramp type* parameter can also be set using the `_SetSys(SS_RAMTYPE, Hxx)` `command` in the IPOS$^{plus®}$ application program. In this case, variable *Hxx* must have the value 6.
- Set the *P700 Operating mode* parameter to the correct operating mode (CFC & IPOS / SERVO & IPOS / VFC-n-CTRL. & IPOS).
- Coupling / decoupling by interrupt: To avoid conflicting terminal function assignments, set the selected input terminal (→ following figure e.g. DI04 and DI05) in the corresponding parameter group 60x or 61x to "NO FUNCTION".
- Coupling / decoupling by binary inputs: To avoid conflicting terminal function assignments, set the selected input terminal (→ following figure e.g. DI04 and DI05) in the corresponding parameter group 60x or 61x to "IPOS INPUT".



10336AEN

*Figure 37: Parameter settings in the Shell program for starting up internal synchronous operation*

***Starting up with SBus connection***

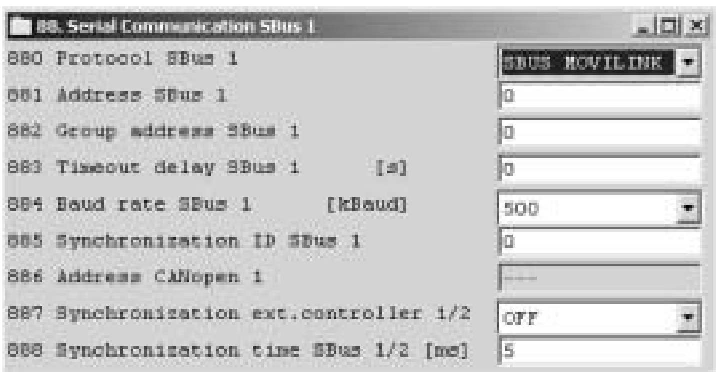
The master and slave(s) are connected via the SBus, for example in a group configuration. The master position is transmitted via this SBus. Transmitting position setpoints requires control loop synchronization between the master and slave. Comply with the following points when starting up the SBus:

**Master inverter:**

- Create the two SBus transmit objects (SCOM TRANSMIT CYCLIC) "Synchronization ID" and "Master position". Both object numbers must be greater than 1050. In addition, the object number of the synchronization ID must be lower (= with higher priority) than the object number of the master position.
- The number of the "Synchronization ID" transmit object must not be the same as its own P885 parameter value.
- The set SBus address (P881) must not be the same as the slave SBus addresses.
- The "Cycle time" in the SCOM command for the synchronization ID must be 5 ms.
- The "Cycle time" in the SCOM command for the master position must be 1 ms.

The next page shows an IPOS$^{plus®}$ sample program for cyclical data transmission via SBus and the necessary parameter settings for the master inverter. This sample program only shows the basic principles of the procedure. We cannot be held liable for incorrect program functions or parameter settings and consequences thereof.

*Necessary parameter settings*




10338AEN

*Figure 38: Necessary parameter settings for the master inverter (example)*

It is very important that the telegram with the master position in the IPOS$^{plus®}$ program is created **before** the synchronization telegram.

*IPOSplus® sample program **master inverter***

```
/*=========================================================================================
IPOS source file
=========================================================================================*/
The program sends the actual position of the master H511 to the SBus.
The following parameters must be set for this purpose:
P880 Protocol SBus 1                :SBUS MOVILINK
P881 Address SBus 1                 :e.g. "0"
P884 Baud rate SBus 1               :e.g. "500" (default setting)
P885 Synchr. ID SBus 1              :e.g. "0" (must not be the same as SBus object "Synchr.ID!)
=========================================================================================*/
#include <const.h>
#include <io.h>
SCTRCYCL Master position, Synchronization_Id;

/*=========================================================================================
Main function (IPOS initial function)
=========================================================================================*/
main ()
{

/*=========================================================================================
Initialization
=========================================================================================*/
Masterposition.ObjectNo=1100;         // Address of the object
Masterposition.CycleTime = 1;         // Time [ms]
Masterposition.Offset = 0;
Masterposition.Format = 4;            // Object length: 4 bytes
Masterposition.Dpointer = 511;        // Pointer to H511 ActPos_Mot (motor position)
Masterposition.Result = 0;

Synchronisations_Id.ObjectNo = 1090;  // Address of the synchronization identifier
Synchronisations_Id.CycleTime = 5;    // Time Synchr. telegram
Synchronisations_Id.Offset = 0;
Synchronisations_Id.Format = 0;
Synchronisations_Id.DPointer = 0;
Synchronisations_Id.Result = 0;

_SBusCommDef(SCD_TRCYCL, Masterposition);       // Create data objects
_SBusCommDef(SCD_TRCYCL, Synchronisations_Id);  // for cyclical data transmission using
_SBusCommOn();                                  // Start data transmission
/*=========================================================================================
Main program loop
=========================================================================================*/
while(1)
    {
    }
}
```

**Slave inverter:**

- Create the SBus receive object (SCOM RECEIVE) "Master position".
- The P885 parameter value must be the same as the number of the "Synchronization ID" transmit object.
- The slaves must have different SBus addresses (P881).

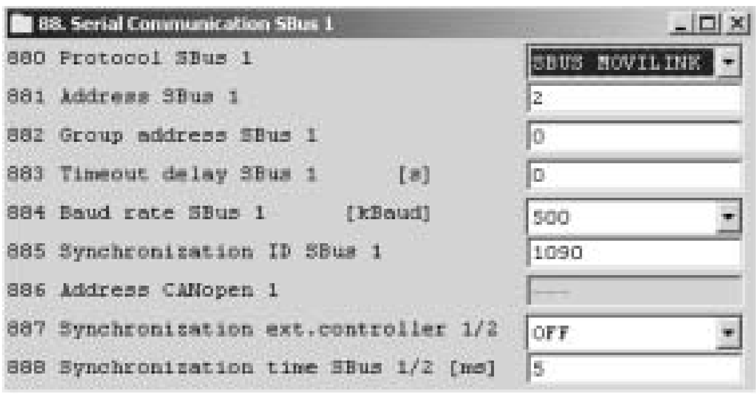
**Make sure that**

- that the entry *Variable* is selected in the "Input master encoder" input field of the "General synchronous operation parameters" startup window
- that the value of the D pointer (SCOM command structure) is selected in the "Variable used" input field, e.g. H200, according to IPOS$^{plus®}$ sample program 3 (→ Sec. " IPOS$^{plus®}$ program samples")
- the system variable *H430 MasterSource = 200* is set in the slave inverter.

Only then is the SBus active as the source for the master increments.

The next page show an IPOS$^{plus®}$ sample program for cyclical data transmission via SBus for the slave inverter. This sample program only shows the basic principles of the procedure. We cannot be held liable for incorrect program functions or parameter settings and consequences thereof.

*Necessary parameter settings*




10339AEN

*Figure 39: Necessary parameter settings for the slave inverter (example)*

*IPOS^plus®^ sample program* ***slave inverter***

```
/*=================================================================================
IPOS source file
for Synchronous Drive Control
SEW-EURODRIVE GmbH & Co KG
Ernst-Blickle-Str. 42
D76646 Bruchsal
sew@sew-eurodrive.de
http://www.SEW-EURODRIVE.de
/*=================================================================================
The following parameters have to be set for communication via the SBus:
P880 Protocol SBus 1            :SBUS MOVILINK
P881 Address SBus 1             :e.g. "2" (not the same address as the master!)
P884 Baud rate SBus 1           :e.g. "500 kBaud" (default setting)
P885 Synchr. ID SBus 1          :e.g. "1090" (see IPOS program master!)
=================================================================================*/
#pragma var 300 309
#pragma globals 310 349

#include <constb.h>
#include <iob.h>
SCREC Masterposition;
long Position_Master;           // Note: The master position is present on H313 in this case

/*=================================================================================
Main function (IPOS start function)
=================================================================================*/
main ()
{

/*=================================================================================
Startup
=================================================================================*/
InitSynchronization();

Masterposition.ObjectNo = 1100;                         // Number of the data object
Masterposition.Format = 4;                              // Length of the received object: 4 bytes
Masterposition.Dpointer = numof (Position_Master);      // Target address
_SBusCommDef(SCD_REC,Masterposition);                   // Transfer master position
_SBusCommOn();                                          // Start data transmission

/*=================================================================================
Main program loop
=================================================================================*/
while(1)
    {
    }
}
```

***Startup with slave prone to slip with absolute encoder on the travel path***

The absolute position of a connected and evaluated absolute encoder is used as the master for internal synchronous operation. The absolute encoder can be connected as follows:

- to X62 of the DIP11B option (→ only for MOVIDRIVE® MDX61B sizes 1 to 6)
- to X14 of the DEH11B (Hiperface®) or DER11B (resolver) option

**Make sure that**

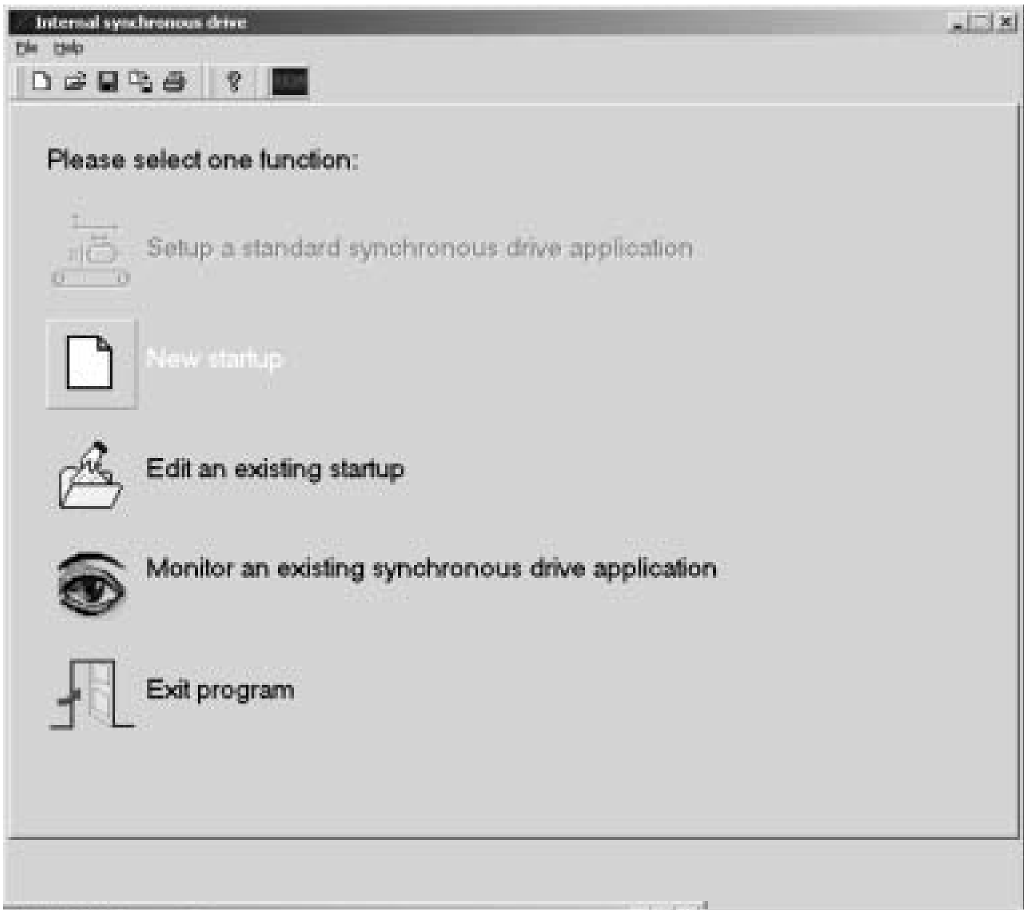
- the entry *SSI encoder* is selected in the "Input master encoder" input field of the "General synchronous operation parameters" startup window
- the slave source is set correctly in the *H431 SlaveSource* system variable:
  - H431 = 0 Source of actual position is X15
  - H431 > 0 Pointer to variable, e.g. H431 = 510 // Source actual position X14 (*H510 ActPos_Ext*)

## 6.4 Internal synchronous operation startup interface

This chapter explains the **"New startup" function of a synchronous operation application by way of an example**.

If you have questions during startup, please refer to the MOVITOOLS® **online help**.

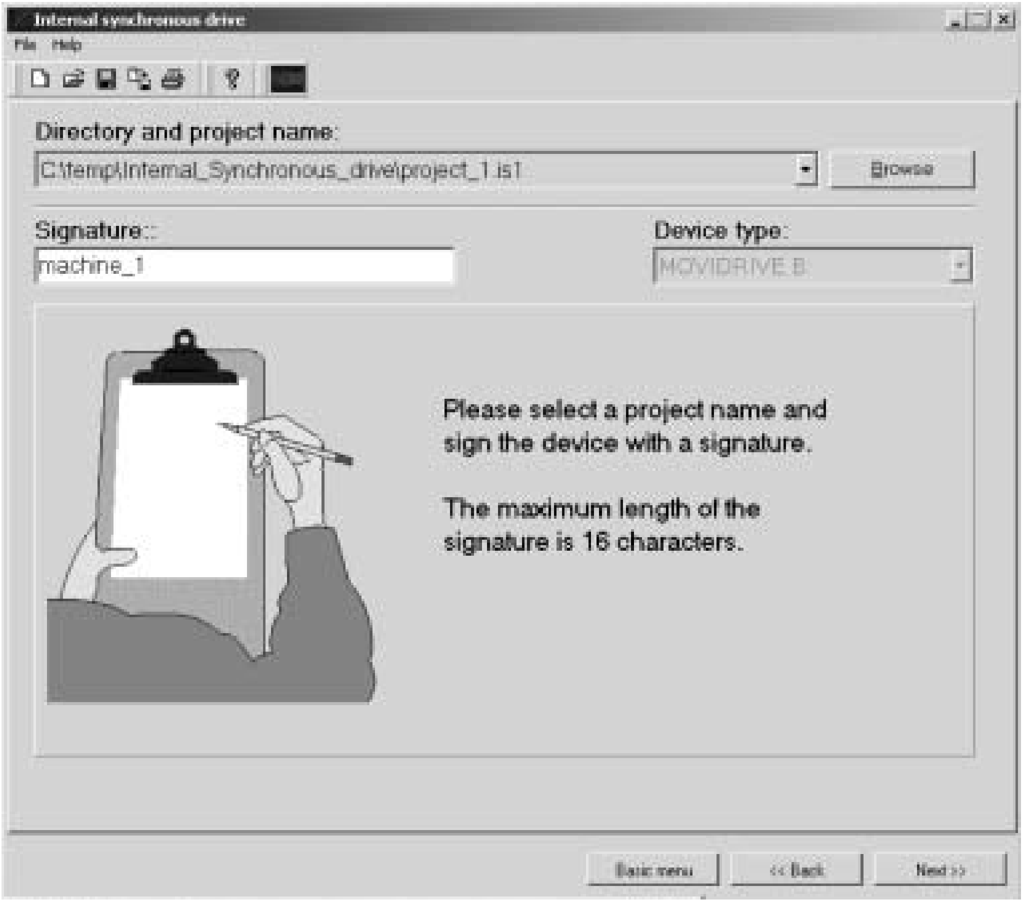
In the MOVITOOLS® manager, press the "ISYNC" button, which you find in the "Execute Program" field under "Special programs". The startup window for internal synchronous operation appears. In the following, this window is referred to as basic menu.




10323AEN

*Figure 40: Basic menu of the startup interface for internal synchronous operation*

In the basic menu, click the button next to "New startup". The "Directory and project name" window opens.

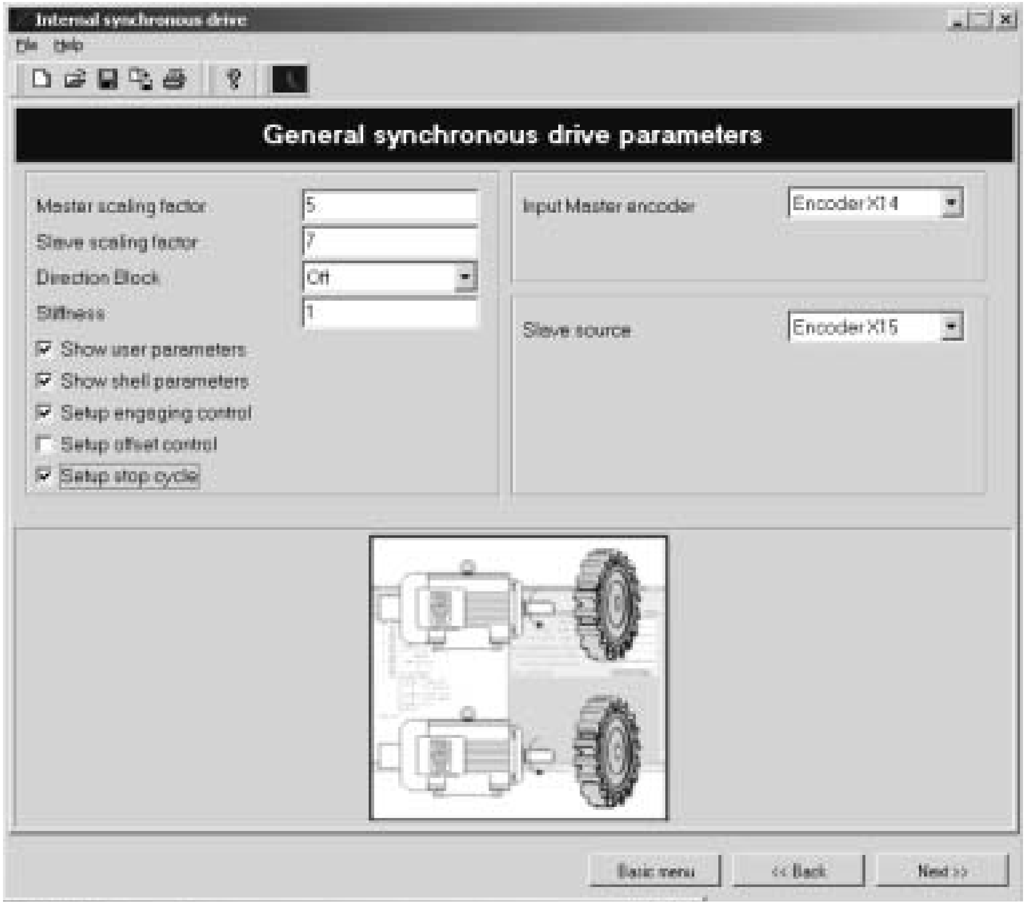



10325AEN

*Figure 41: Enter path and project name*

Select the project name and path for the new startup. In the "Signature" input field, you can sign the device with a signature.

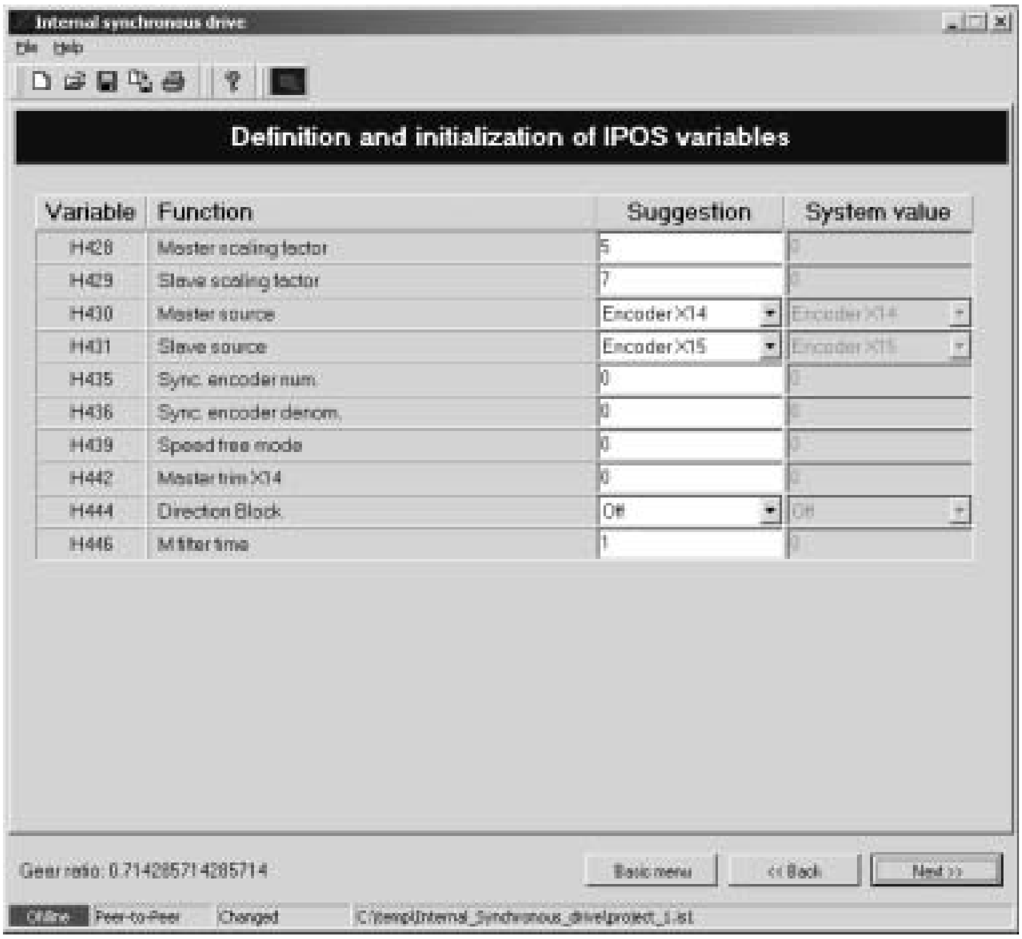
Click "Next" to continue. The "General synchronous drive parameters" window opens.




10326AEN

*Figure 42: Entering general synchronous drive parameters*

Enter the general synchronous operation parameters. Select the options required for starting up your application with internal synchronous operation.

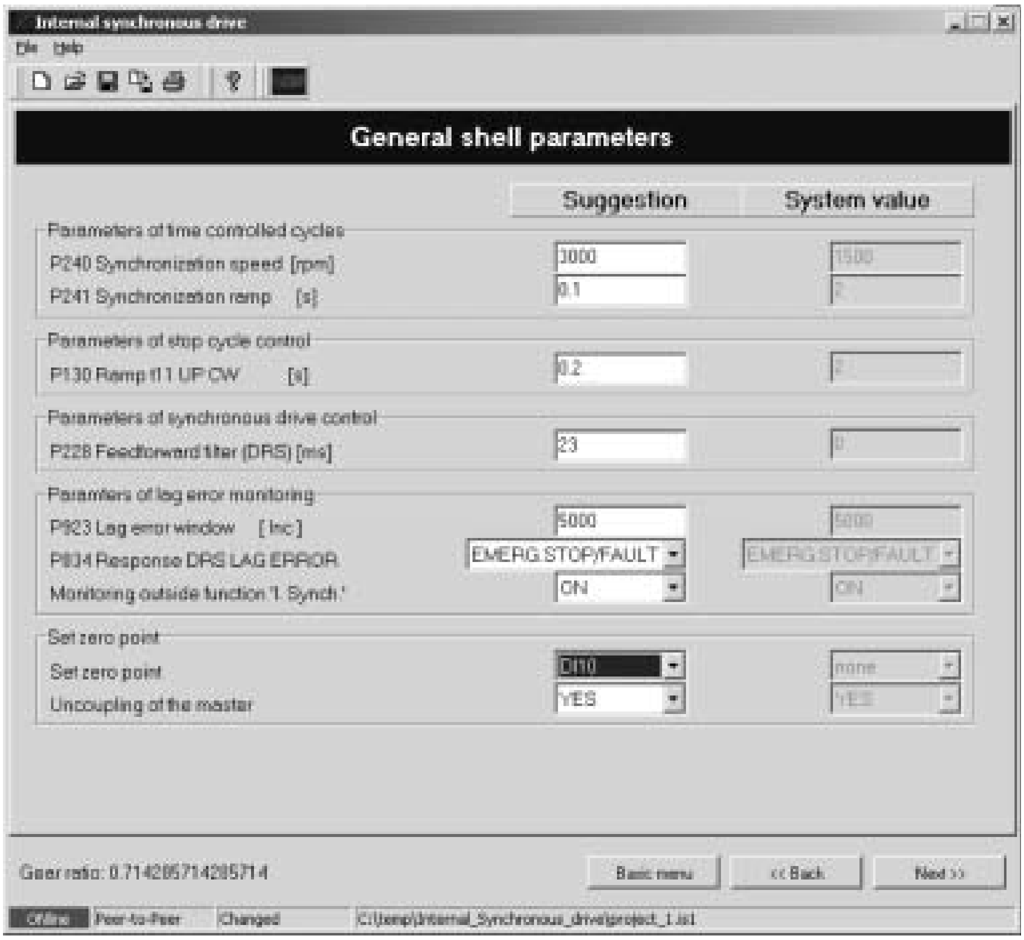



10327AEN

*Figure 43: Definition and initialization of IPOSplus® variables*

In this window, you can make additional settings for IPOSplus® variables important for internal synchronous operation, and manually modify specified values. In this case, the left text box (white background) displays a system-internal value as suggestion for a possible setting. The right text box (gray background), displays the system value of the current unit setting.

Click "Next" to continue. The "General shell parameters" window opens.

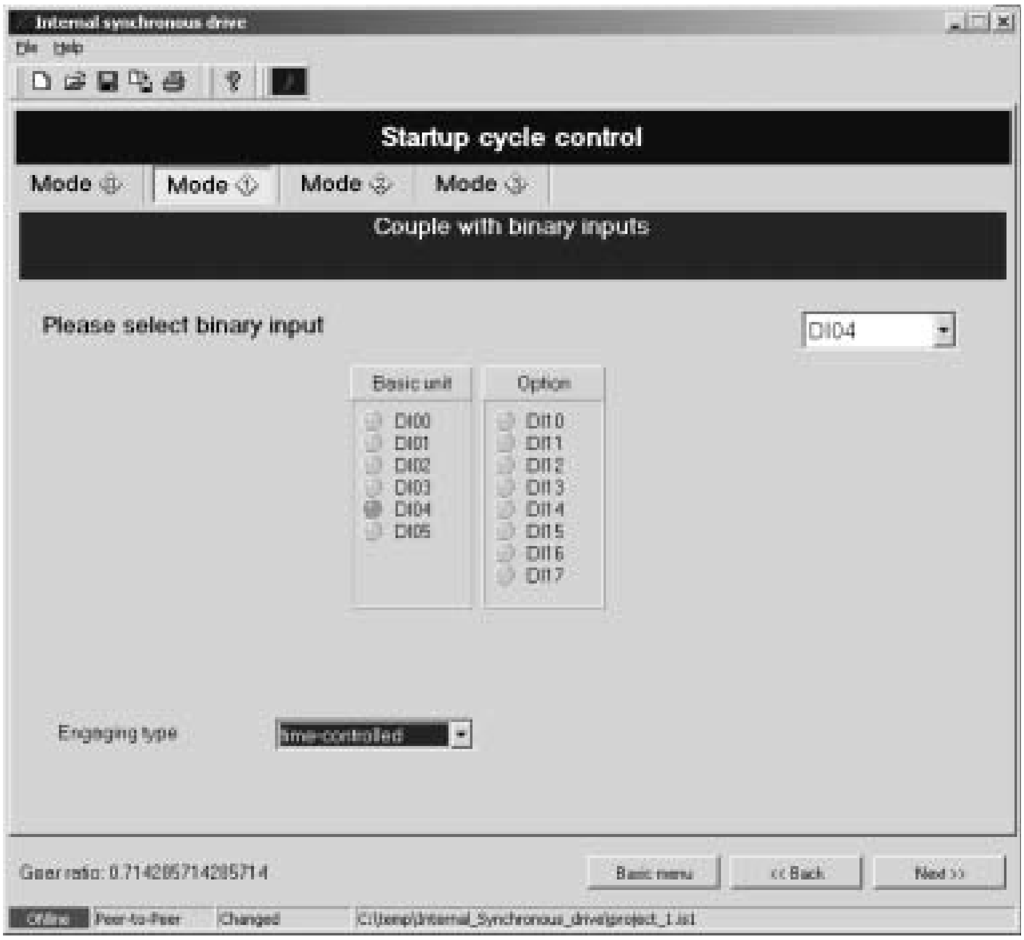



10328AEN

*Figure 44: Setting general shell parameters*

In this window, you can set the Shell parameters necessary for internal synchronous operation and configure certain additional functions. The left text box (white background) displays a suggestion for a possible setting. The right text box (gray background) displays the system value of the current unit setting.

Click "Next" to continue. The "Startup cycle control" window opens.




10329AEN

*Figure 45: Startup of time-controlled startup cycle mode control*

In the "Engaging type" input field, select whether you want the slave drive to synchronize in a time-controlled or position-dependent manner. You can select from the following engaging type modes:

- Mode 0

  Manual engaging via IPOS$^{plus®}$ program.
- Mode 1

  Engaging by binary inputs.
- Mode 2

  Interrupt and position control with DI02 ...
- Mode 3

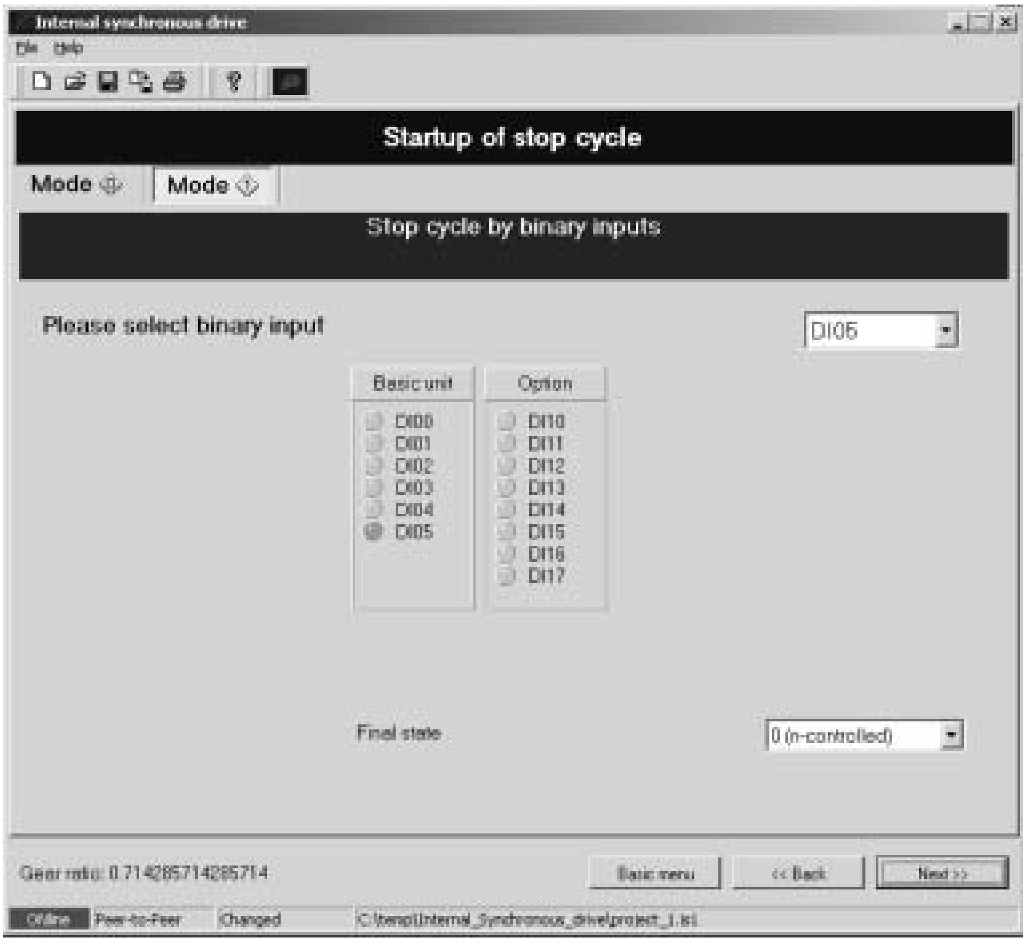
  Position control.

If the coupling process is started event-driven by a binary input (mode 1), you have to select the terminal that triggers the startup cycle process from the "Please select binary input" field. Alternatively, you can enter the terminal designation in the corresponding drop down menu (DIxx). You can use the input terminals of the basic unit (DI00 to DI05) or of the option card (DI10 to DI17) (without using the startup interface → *H403 Startup-CycleInputMask* system variable).

The startup cycle process is started as soon as a "1" level is present at the defined binary input. The terminal latency is 1 ms.

Mode 1: To avoid conflicting terminal function assignments, set the selected input terminal in the corresponding parameter group 60x or 61x to "IPOS INPUT".

Click "Next" to continue. The "Startup of stop cycle" window opens.




10330AEN

*Figure 46: Stop cycle mode control and selection of stop cycle modes*

To startup stop cycle mode control, click the appropriate button in the upper part of the interface to select the required stop cycle mode (in the example (mode 1).

- Manual decoupling by IPOS$^{plus®}$ application program (mode 0)
- Decoupling by binary inputs (mode 1)

If the stop cycle process is started event-driven by a binary input (mode 1), you have to select the terminal that triggers the stop cycle process in the "Please select binary input" field. Alternatively, you can enter the terminal designation in the corresponding drop down menu (DIxx).

You can also use the input terminals of the basic unit (DI00 to DI05) or of the option card (DI10 to DI17) (without using the startup interface → *H403 StartupCycleInputMask* system variable). The stop cycle process is started as soon as a "1" level is present at the defined binary input. The terminal latency is 1 ms.

Mode 1: To avoid conflicting terminal function assignments, set the selected input terminal in the corresponding parameter group 60x or 61x to "IPOS INPUT".

From the "Final state" drop down menu, select the free running mode, which the slave drive should adopt after the stop cycle (in the example: 0 (n-control)).

- 0 (n-control) → Free running mode with speed control

  The slave drive moves with speed control at a specified setpoint (*H439 SpeedFree-Mode*).
- 1 (x-control) → Free running mode with position control

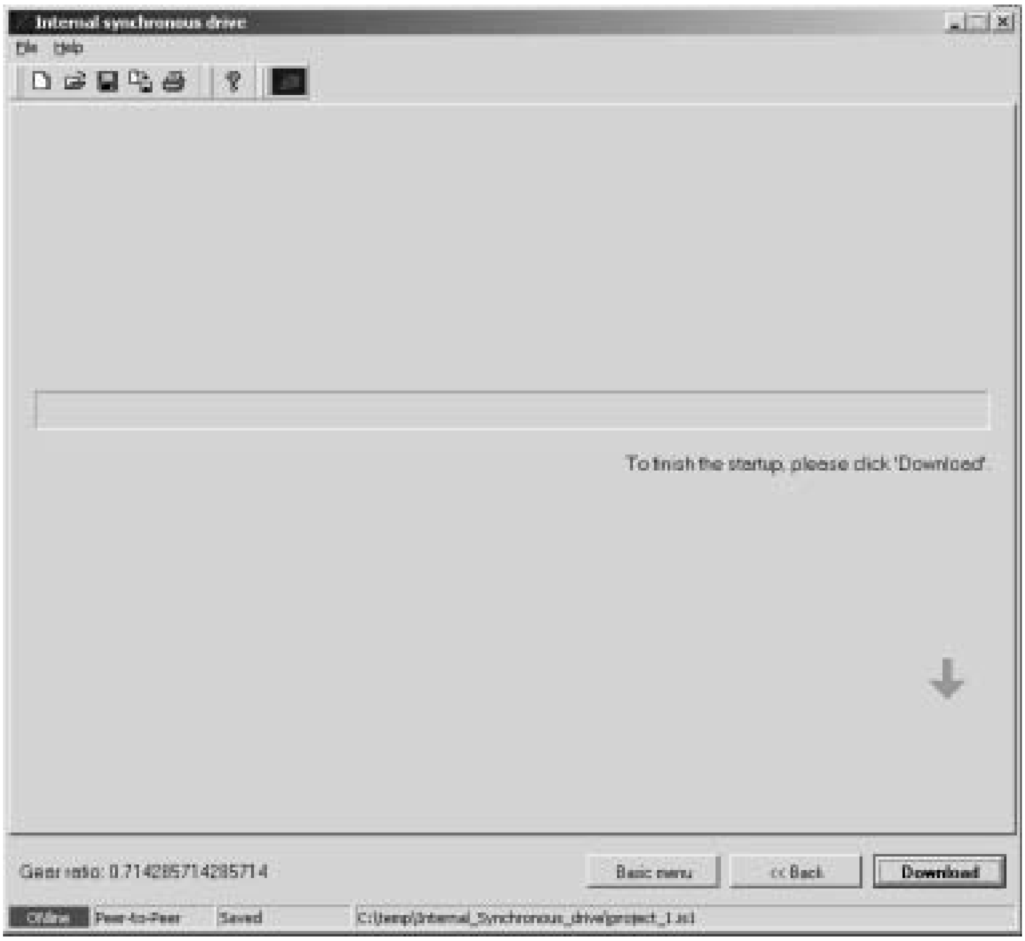
  The slave is held in its current position with position control.

Click "Next" to save the startup data.


10331AEN

*Figure 47: Saving changes*

Click "Yes" to save the data to a file (*.is1) under the project name and path you have defined at the start. This file can then be opened later if you want to edit it again.




10332AEN

*Figure 48: Finishing startup*

To finish startup of internal synchronous operation, click "Download".

The Compiler interface for IPOS$^{plus®}$ programming opens. The interface appears with a program text and with an IPOS$^{plus®}$ source file called "Projectname.ipc".


10333AEN

*Figure 49: IPOS$^{plus®}$ compiler interface*

The settings made in the startup interface are used for initializing the IPOS$^{plus®}$ variables for controlling internal synchronous operation.

This initialization involves calling a routine once from within the above mentioned IPOS$^{plus®}$ application program after it has started.

The name of the initialization routine is `InitSynchronization()` It is located in the `Projectname.h header file integrated in the` program header.

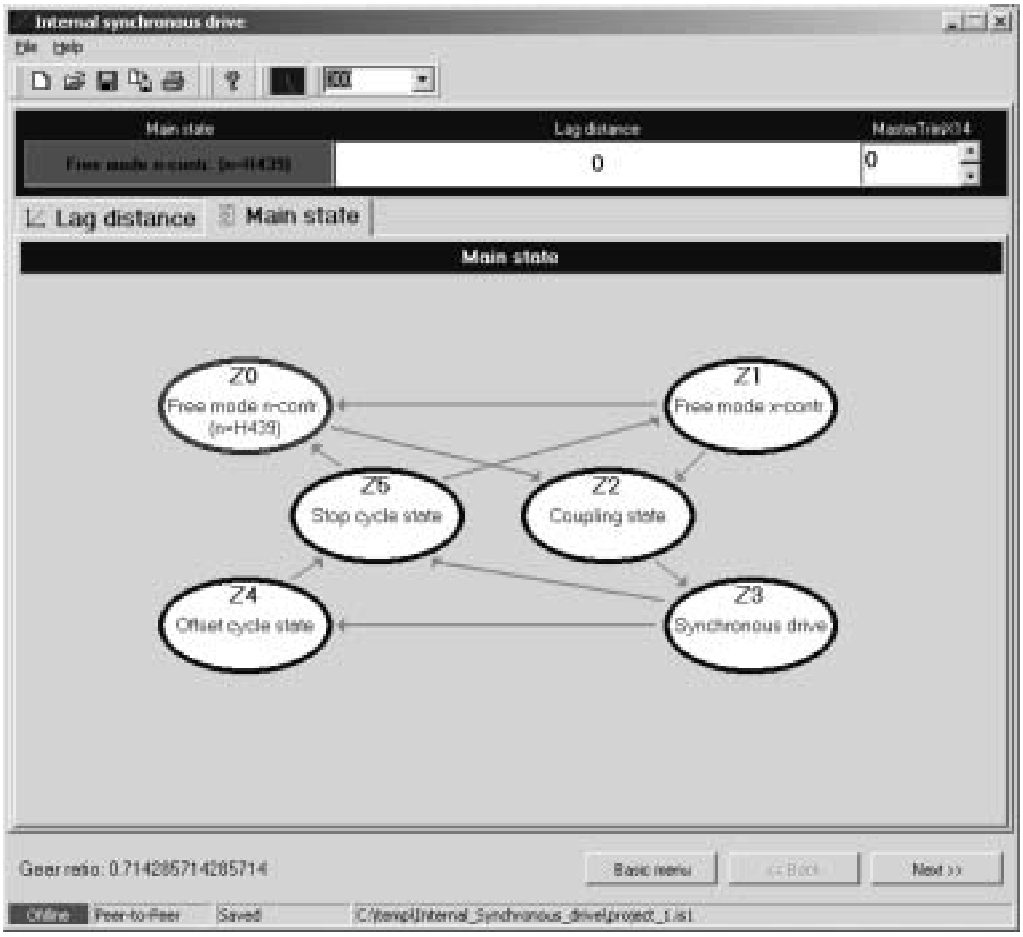
You can now create further user-specific program sections and insert them into the existing text. That way, you expand the IPOS$^{plus®}$ program that will be downloaded to the inverter and run there.

When the program text is complete, save the entire IPOS$^{plus®}$ program again. Next, compile the program into machine language. To do so, click the "Compile and load file" icon [1] to load the compiled program into the inverter. Then click the "Start program" icon [2] to start the program.

You can now start the online monitor for internal synchronous operation. You can close the compiler window in the background.


10334AEN

*Figure 50: Starting the online monitor*




10335AEN

*Figure 51: Display of the current main state*

The online monitor serves to view the current main state of the ongoing synchronous operation application or to check the lag distance.

To exit the program, click "Next". The basic menu opens. In the basic menu, click "Exit program".

# 7 System Variables

| Variable | Name and range of values | Status | Description |
|---|---|---|---|
| | | | **Offset control** |
| H360 | OffsetCyleMode<br>0 to 3 | R/W | Offset mode<br>= 0: Offset via IPOS program<br>= 1: Offset via input terminals<br>= 2: Reserved<br>= 3: Offset via position control |
| H361 | OffsetCycleModeControl | R/W | Activation of various functions<br>Bit 0: AutoRestart (in mode 3)<br>= 0: AutoRestart deactivated<br>= 1: AutoRestart activated<br>Bit 1: OffsetDisable (in mode 3)<br>= 0: Offset processing possible<br>= 1: Offset processing inhibited<br>Bit 12: OffsetMode<br>= 0: Time-controlled offset processing<br>= 1: Position-dependent offset processing |
| H362 | OffsetCycleState<br>Max. 0 to 1 (depending on Offset-CycleMode) | R/W | Control of the various modes |
| H363 | OffsetCycleInputMask | R/W | Terminal window (identical to H483 "InputLevel") |
| H364 | OffsetCycleCounter | R/W | Master counter for offset processing |
| H365 | OffsetCycleCounterMaxValue | R/W | In mode 3: Length limit for automatic offset processing |
| H366 | OffsetCycleMasterLength | R/W | Specified distance for the master drive in offset processing |
| H367 | OffsetCycleValue | R/W | Offset value for slave drive |
| | | | **Virtual encoder** |
| H370 | VEncoderMode<br>0 to 3 | R/W | Virtual encoder operating mode<br>= 0: Normal mode<br>= 1: Reserved<br>= 2: Infinite counter<br>= 3: Positioning mode |
| H371 | VEncoderModeControl | R/W | Bit 0: AxisStop<br>= 0: Deactivated<br>= 1: Axis stop in the event of a unit fault |
| H372 | VEncoderState | R/W | No function |
| H373 | VEncoderNSetpoint | R/W | Set travel speed in 1 incr./ms<br>Mode 0 and 2: –32000 to +32000<br>Mode 3: 0 to 32767 ($2^{15-1}$) |
| H374 | VEncoderNActual | R/W | Actual travel speed in 1 incr./ms |
| H375 | VEncoderXSetpoint | R/W | Target position in incr. |
| H376 | VEncoderXActual | R/W | Current position in incr. |
| H377 | VEncoderNdT | R/W | Mode 0/2: Acceleration (ramp) in 1 incr./$ms^2$<br>Mode 3: Acceleration (ramp) in $1/2^{16}$ incr./$ms^2$ |
| | | | **Control element** |
| H389 | RegisterLoopOut | R/W | The value to be reduced in connection with RegisterLoopDXDTOut |
| H390 | RegisterLoopDXDTOut<br>–30000 to 30000 | R/W | Control element limit<br>Max. addition (64-bit counter) per ms |

| Variable | Name and range of values | Status | Description |
|---|---|---|---|
| | | | **Stop cycle mode control** |
| H400 | StopCycleMode<br>0 to 1 | R/W | Stop cycle mode<br>= 0: Decoupling via IPOS program<br>= 1: Decoupling via input terminals |
| H401 | StopCycleModeControl | R/W | Activation of various functions<br>Bit 0: FreeMode<br>= 0: Decoupling in main state 0 (n-control)<br>= 1: Decoupling in main state 1 (x-control)<br>Bit 1: x-control mode<br>= 0: Difference counter is reset<br>= 1: Difference counter active, differential between master and slave is maintained |
| H402 | StopCycleState | | No function |
| H403 | StopCycleInputMask | R/W | Terminal window (identical to H483 "InputLevel") in mode 1 only |
| | | | **Startup cycle mode control** |
| H410 | StartupCycleMode<br>0 to 3 | R/W | Startup cycle mode<br>= 0: Coupling via IPOS program<br>= 1: Coupling via input terminals<br>= 2: Coupling via interrupt control<br>= 3: Coupling via position control |
| H411 | StartupCycleModeControl | R/W | Activation of various functions<br>Bit 0: Auto restart (in modes 2 and 3)<br>= 0: AutoRestart deactivated<br>= 1: AutoRestart<br>Bit 1: StartupDisable (in modes 2 and 3)<br>= 0: Startup cycle possible<br>= 1: Startup cycle inhibited<br>Bit 2: InterruptSelect (in mode 2)<br>= 0: DI02<br>= 1: X14C track<br>Bit 12: StartupMode<br>= 0: Time-controlled synchronization<br>= 1: Position-dependent synchronization |
| H412 | StartupCycleState<br>Max. 0 to 4 (depending on mode) | R/W | Control of the various modes<br>= 0: Interrupt disabled<br>= 1: Interrupt enabled<br>= 2: Waiting for interrupt<br>= 3: Delay, which means the interrupt signal was detected StartupCycleCounterMaxValue is effective<br>= 4: Coupling and resetting the startup counter<br>**Note:** If *AutoRestart* is deactivated (H411.0 = 0), the value "1" must be assigned to *StartupCycleState* (H412), else the startup cycle will not be performed. If the value "0" is assigned to *StartupCylceState*, then the counter is reset and the reference to the master is lost. |
| H413 | StartupCycleInputMask | R/W | Terminal window (identical to H483 *InputLevel*) in mode 1 only |
| H414 | StartupCycleCounter | R/W | Master counter for the startup cycle<br>The startup cycle is initiated when the value of *StartupCycleCounter* (H414) is greater than the value of *StartupCycleCounterMaxValue* (H415). |
| H415 | StartupCycleCounterMaxValue | R/W | In mode 2: Delay for startup cycle process<br>In mode 3: Length limit for automatic startup cycle |
| H416 | StartupCycleDelayDI02<br>-32768 to 32767 | R/W | Delay in units of 0.1 ms<br>Delay time of the sensor connected to touch-probe input 2 |
| H417 | StartupCycleMasterLength | R/W | Specified distance for the master drive for position-dependent coupling process. The slave has synchronized to the master within this distance. |

| Variable | Name and range of values | Status | Description |
|---|---|---|---|
| | | | **General variables** |
| H425 | Synchronous mode | | No function |
| H426 | SynchronousModeControl | R/W | Activation of various functions<br>Bit 0: PosTrim (only active in main state Z1 "x-control")<br>= 0: The drive stops position-controlled at the current position.<br>= 1: Causes the slave drive to move to *TargetPos* (H492) during position control in free running mode (main state 1) but without ramp. Therefore only use for position correction or to avoid drifting.<br>Bit 1: LagError (in state 3 → Synchronous operation)<br>= 0: Lag error monitoring<br>= 1: No lag error monitoring<br>**Note:** The "lag error monitoring" or "no lag error monitoring" setting is active when the drive is in state 3 (synchronous operation) even if the "I-SYNCHRON.OPERAT." ramp type is subsequently changed to another operating mode.<br>Bit 2: RegisterScale<br>= 0: Increments to be corrected are not scaled (one-to-one)<br>= 1: Increments are scaled with slave gear ratio factor (*GFSlave*)<br>Bit 3: Zero point mode<br>= 0: Feedforward is disabled<br>= 1: Feedforward is maintained (speed synchronization), which means the slave continues to move at the set master speed. |
| H427 | SynchronousState<br>0 to 5 | R/W | Main state machine integrated synchronous operation<br>= 0: Free mode n-control<br>= 1: Free mode x-control<br>= 2: Coupling<br>= 3: Synchronous operation<br>= 4: Offset processing<br>= 5: Decoupling |
| H428 | GFMaster<br>-2 000 000 000 to 2 000 000 000 | R/W | Scaling factor of the master increments, value = $i_{slave}$ |
| H429 | GFSlave<br>1 to 2 000 000 000 | R/W | Scaling factor of the slave increments, value = $i_{master}$ |
| H430 | MasterSource<br>0 to 1023 | R/W | Source of the master increments<br>= 0: X14 + virtual axis (H442)<br>> 0: Pointer to variable |
| H431 | SlaveSource<br>0 to 1023 | R/W | Source of the actual position<br>= 0: X15<br>> 0: Pointer to variable<br>Example: H431 = 510 // source actual position X14 (H510 *ActPos_Ext*) |
| H432 | LagDistance64Low | R/- | Low 32 bits of the 64-bit counter |
| H433 | LagDistance64High | R/- | High 32 bits of the 64-bit counter |
| H434 | LagDistance32 | R/- | 32-bit lag distance in relation to GFSlave |
| H435 | SyncEncoderNum<br>0 to 10000 | R/W | Synchronous encoder factor - numerator<br>= 0: Synchronous encoder calculation deactivated |
| H436 | SyncEncoderDenom<br>0 to 10000 | R/W | Synchronous encoder factor - denominator<br>= 0: Synchronous encoder calculation deactivated |
| H437 | SlaveTrim | R/W | The firmware automatically adds the value of H437 to the difference counter once and then zeroes the counter. |
| H438 | XMasterPos | R/- | Display value of the master counter during startup cycle process and offset cycle processing |
| H439 | SpeedFreeMode | R/W | Speed setpoint in free running n-control in 0.2 rpm |
| H440 | Reserved4 | | |
| H441 | Reserved5 | | |
| H442 | MasterTrimX14<br>-32768 to 32767 | R/W | Virtual axis<br>Pulse number 1 incr./ms |
| H443 | Reserved6 | | |
| H444 | ReSprintClose<br>0 to 2 | R/W | Direction of rotation inhibit<br>= 0: Both directions of rotation are enabled<br>= 1: Only CCW direction of rotation<br>= 2: Only CW direction of rotation |

| Variable | Name and range of values | Status | Description |
|---|---|---|---|
| H445 | Reserved7 | | |
| H446 | MFilterTime<br>1 to 30 | R/W | Interpolation time in ms<br>= 1: Without filter<br>≤ 30: Scaling up, absolute scaling factor of the master pulses<br>= GFMaster x MFilterTime |

# 8 Sample IPOS$^{plus®}$ Programs

This following sample programs only show the basic principles of the procedure. No liability can be inferred from faulty program functions and the consequences thereof!

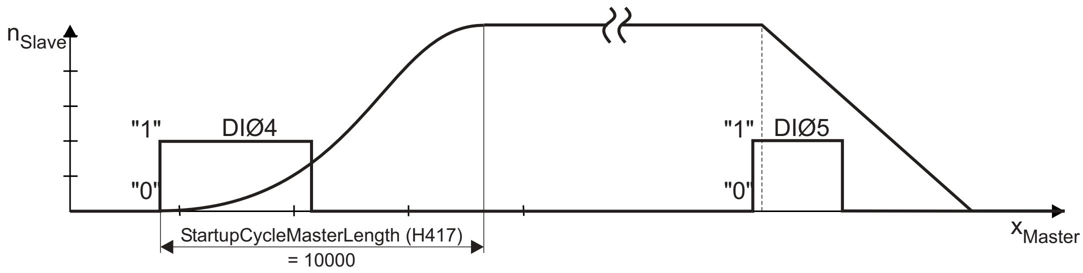
## 8.1 Example 1

***Objective***

A slave drive is to be operated at a synchronous angle to a master drive. The gear units used in this case are the same and have a gear ratio of 1:1. The master and slave inverters are connected via X14. The slave inverter is controlled via the binary inputs. Binary inputs X13:5 (DI∅4) and X13:6 (DI∅5) should be used for controlling the startup and stop cycle processes. Both binary inputs must be programmed to "NO FUNCTION."

- "1" signal on DI∅4 → starts the startup cycle process. The startup cycle process should be position-dependent and be completed after 10 000 master increments.
- "1" signal on DI∅5 → starts the stop cycle process.

The necessary IPOS$^{plus®}$ system variables are set in the initialization function.




06643AXX

*Figure 52: Event-driven coupling and decoupling*

### *IPOSplus® program*

```
/*==============================================================================
IPOS source file
for Synchronous Drive Control
SEW-EURODRIVE GmbH & Co KG
Ernst-Blickle-Str. 42
D76646 Bruchsal

sew@sew-eurodrive.de
http://www.SEW-EURODRIVE.de
==============================================================================*/
#pragma          var                300     309
#pragma          globals            310     349

#include         <const.h>
#include         <Example01.h>                        // Header file with
                                                      // variable designations
                                                      // and initialization function

*/==============================================================================
                                Main function (IPOS start function)
==============================================================================*/
main ()
       {
*/---------------------------------------------------------
                     Startup
---------------------------------------------------------*/
InitSynchronization();                                //Call up initialization function
/*---------------------------------------------------------
                    Main program loop
----------------------------------------------------------*/
while(1)
       {
       }
}
```

### *Header file with variable designation*

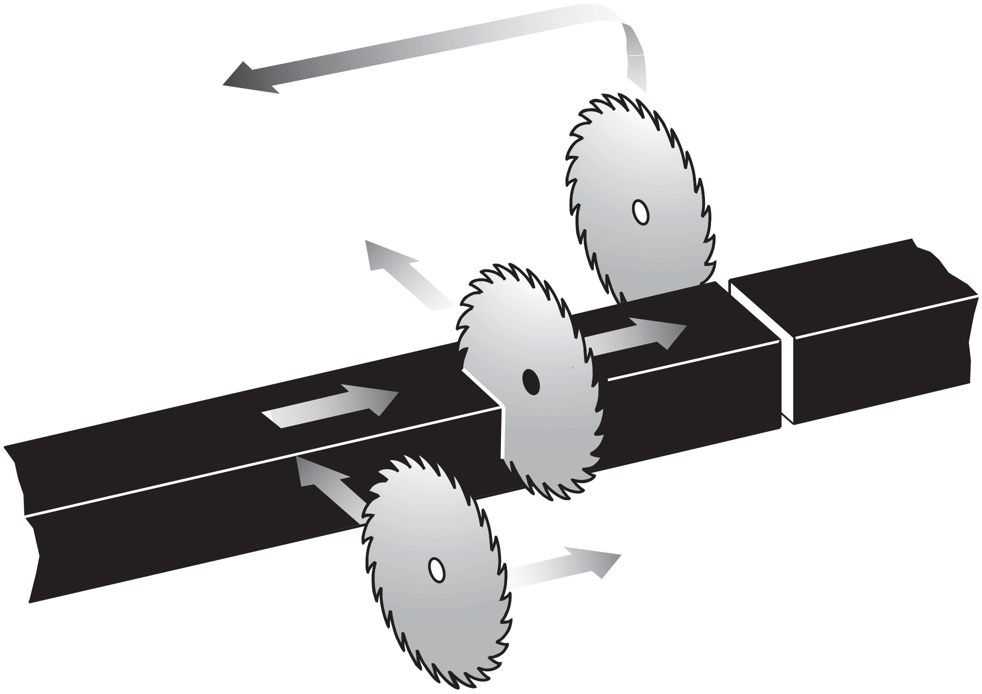
```
/******************************************************************************
Example01.h
Data and startup header file for IPOS+ Compiler.
For Startup after power on call "InitSynchronization();"
Data file Movidrive Synchronous Drive Control Version 1.0
******************************************************************************/

#define        SynchronousMode                     H425
#define        SynchronousModeControl              H426
#define        SynchronousState                    H427
#define        GFMaster                            H428
#define        GFSlave                             H429
#define        MasterSource                        H430
#define        Reserved1                           H431
#define        LagDistance64Low                    H432
#define        LagDistance64High                   H433
#define        LagDistance32                       H434
#define        Reserved2                           H435
#define        Reserved3                           H436
#define        SlaveTrim                           H437
#define        XMasterPos                          H438
#define        SpeedFreeMode                       H439
#define        Reserved4                           H440
#define        Reserved5                           H441
#define        MasterTrimX14                       H442
#define        Reserved6                           H443
#define        ReSprintClose                       H444
#define        Reserved7                           H445
#define        MFilterTime                         H446

// variables for StartupCycle, StopCycle and OffsetCycle
#define        StopCycleMode                       H400
#define        StopCycleModeControl                H401
#define        StopCycleState                      H402
#define        StopCycleInputMask                  H403

#define        StartupCycleMode                    H410
#define        StartupCycleModeControl             H411
#define        StartupCycleState                   H412
#define        StartupCycleInputMask               H413
#define        StartupCycleCounter                 H414
#define        StartupCycleCounterMaxValue         H415
#define        StartupCycleDelayDI02               H416
#define        StartupCycleMasterLength            H417
```

```
#define   OffsetCycleMode                  H360
#define   OffsetCycleModeControl           H361
#define   OffsetCycleState                 H362
#define   OffsetCycleInputMask             H363
#define   OffsetCycleCounter               H364
#define   OffsetCycleCounterMaxValue       H365
#define   OffsetCycleMasterLength          H366
#define   OffsetCycleValue                 H367

// variables to Register Control
#define   RegisterLoopOut                  H389
#define   RegisterLoopDXDTOut              H390

// variables for virtual encoder
#define   VEncoderMode                     H370
#define   VEncoderModeControl              H371
#define   VEncoderState                    H372
#define   VEncoderNSetpoint                H373
#define   VEncoderNActual                  H374
#define   VEncoderXSetpoint                H375
#define   VEncoderXActual                  H376
#define   VEncoderdNdT                     H377

//Startup data from: 08.08.2000 - 16:35:22
InitSynchronization()
      {
         for (H0=128; H0<=457; H0++)                // Reset variables greater than H128
         {
         *H0=0;
         }
         _Memorize(MEM_LDDATA);
         _Wait(100);

         GFMaster                         = 1;      // Evaluation of master increments
         GFSlave                          = 1;      // Evaluation of slave increments
         MFilterTime                      = 1;      // Processing of master increments w/o filter
         StartupCycleMode                 = 1;      // Startup cycle mode 1: Event-driven starting
                                                    of the coupling process via binary input

         StartupCycleInputMask            = 16;     // Selection of terminal DI04 for startup cycle
         StartupCycleMasterLength         = 10000;  // Length of master travel until startup cycle
                                                    finished
         _BitSet (StartupCycleModeControl, 12);     // Selection of "position-dependent startup
                                                    cycle process"
         RegisterLooppDXDTOut             = 2;      // Limiting of correction mechanism
         StopCycleMode                    = 1;      // Stop cycle mode 1: Event-driven starting of
                                                    the stop cycle process via binary input
         StopCycleInputMask               = 32;     // Selection of terminal DI05 for stop cycle
      }
```

## 8.2 Example 2

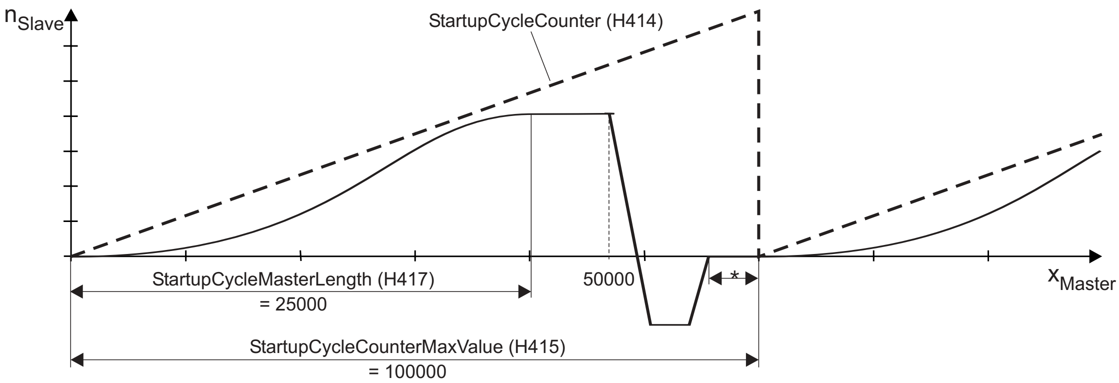
***Objective***

Extruded material is to be cut off using a flying saw. The travel increments of the extruded material are used as master increments at input X14 of the saw feed drive = slave drive. The slave drive waits in its start position. The startup cycle process is initiated with position control by the *StartupCycleCounter* position counter (H414). The extruded material is sawn during synchronous operation. The slave drive disengages after the sawing operation and moves back to its start position. The gear ratio is 1:1.



03866AXX

*Figure 53: Flying saw*




06644AXX

*Figure 54: Position-controlled starting of the startup cycle process (* slave is decoupled)*

**Important notes**:

- Reference travel type 3 (P903) is set for reference travel.
- The reference offset (P900) is set to 300 000, for example.
- The CW and CCW limit switches must have their parameters set and must be connected.

### IPOS[plus®] program

```
/*============================================================================
IPOS source file
for Synchronous Drive Control
SEW-EURODRIVE GmbH & Co KG
Ernst-Blickle-Str. 42
D76646 Bruchsal

sew@sew-eurodrive.de
http://www.SEW-EURODRIVE.de
============================================================================*/
#pragma          var               300          309
#pragma          globals           310          349

#include         <const.h>
#include         <Example01.h>                              // Header file with
                                                            // variable designations
                                                            // and initialization function

#define          LINEAR            0                        // Positioning with linear ramp
#define          Synchronous       6                        // Internal synchronous operation
                 operation
#define          Stop              _BitClear (ControlWord, 2)  // CW bit is canceled
#define          Enable            _BitSet (ControlWord, 2)    // CW bit is set

long             Ramp type, tmp;
*/===============================================================================
                                Main function (IPOS start function)
============================================================================*/
main ()
       {
*/-----------------------------------------------------------
                    Startup
-----------------------------------------------------------*/
InitSynchronization();                         // Calling up initialization function
Ramp type=LINEAR                                            // Positioning ramp
_SetSys(SS_RAMPTYPE, ramp type);

while (!DI=00);                                // Wait for high level on DI00 "/Controller inhibit"

_Go0((GO0_C_W_ZP);                             // Referencing with reference type 3 / CW limit
                                               switch
                                               // P900 "Reference offset": Move 300000 increments
_GoAbs(GO_WAIT, 0);                                         // to start position

Ramp type=SYNCHROUS OPERATION;                 // Activate internal synchronous operation
_SetSys(SS_RAMPTYPE, ramp type);

StartupCycleCounter = 0;                                    // Reset counter
```

```
StartupCycleState = 1;                                   // Activate startup cycle mode con-
                                                         trol
/*---------------------------------------------------------
                       Main program loop
---------------------------------------------------------*/
while(1)
                {
                  tmp=StartupCycleCounter;               // Save startup cycle counter in tem-
                                                         porary memory
                   if ((tmp>50000)&&(SynchronousState==3))  // Change ramp type
                                                         // if counter > 50000 master incr.
                                                         // and drive in synchronous operation
                        {
                            Halt;                        // Inhibit drive
                           SynchronousState=5;           // Decoupling ( in position control)
                             Ramp type=LINEAR;           // Positioning ramp
                              _SetSys(SS_RAMPTYPE,ramp type);
                            Enable;                      // Enable drive
                           _GoAbs(GO_WAIT, 0);           // Move to start position
                            Halt;                        // Inhibit drive
                      Ramp type=SYNCHRONOUS OPERATION;   // Activate internal synchronous
                                                         operation
                              _SetSys(SS_RAMPTYPE,ramp type);
                            Enable;                      // Enable drive
                        }
                 }
       }
```

### *Header file with variable designation*

```
/*******************************************************************************
Example01.h
Data and startup header file for IPOS+ Compiler.
For Startup after power on call "InitSynchronization();"
Data file Movidrive Synchronous Drive Control Version 1.0
*******************************************************************************/

#define         SynchronousMode                     H425
#define         SynchronousModeControl              H426
#define         SynchronousState                    H427
#define         GFMaster                            H428
#define         GFSlave                             H429
#define         MasterSource                        H430
#define         Reserved1                           H431
#define         LagDistance64Low                    H432
#define         LagDistance64High                   H433
#define         LagDistance32                       H434
#define         Reserved2                           H435
#define         Reserved3                           H436
#define         SlaveTrim                           H437
#define         XMasterPos                          H438
#define         SpeedFreeMode                       H439
#define         Reserved4                           H440
#define         Reserved5                           H441
#define         MasterTrimX14                       H442
#define         Reserved6                           H443
#define         ReSprintClose                       H444
#define         Reserved7                           H445
#define         MFilterTime                         H446

// variables for StartupCycle, StopCycle and OffsetCycle
#define         StopCycleMode                       H400
#define         StopCycleModeControl                H401
#define         StopCycleState                      H402
#define         StopCycleInputMask                  H403

#define         StartupCycleMode                    H410
#define         StartupCycleModeControl             H411
#define         StartupCycleState                   H412
#define         StartupCycleInputMask               H413
#define         StartupCycleCounter                 H414
#define         StartupCycleCounterMaxValue         H415
#define         StartupCycleDelayDI02               H416
#define         StartupCycleMasterLength            H417
```

```
#define  OffsetCycleMode                        H360
#define  OffsetCycleModeControl                 H361
#define  OffsetCycleState                       H362
#define  OffsetCycleInputMask                   H363
#define  OffsetCycleCounter                     H364
#define  OffsetCycleCounterMaxValue             H365
#define  OffsetCycleMasterLength                H366
#define  OffsetCycleValue                       H367

// variables to Register Control
#define  RegisterLoopOut                        H389
#define  RegisterLoopDXDTOut                    H390

// variables for virtual encoder
#define  VEncoderMode                           H370
#define  VEncoderModeControl                    H371
#define  VEncoderState                          H372
#define  VEncoderNSetpoint                      H373
#define  VEncoderNActual                        H374
#define  VEncoderXSetpoint                      H375
#define  VEncoderXActual                        H376
#define  VEncoderdNdT                           H377
```

```
//Startup data from: 08.08.2000 - 15:54:37
InitSynchronization()
      {
        for (H0=128; H0<=457; H0++)                        // Reset variables greater than H128
        {
        *H0=0;
        }
            _Memorize(MEM_LDDATA);
        _Wait(100);
        GFMaster                          = 1;         // Evaluation of master increments
        GFSlave                           = 1;         // Evaluation of slave increments
        MFilterTime                       = 1;         // Processing of master increments w/o
                                                       filter
        StartupCycleMode                  = 3;         // Startup cycle mode 3: Position-con-
                                                       trolled starting of the coupling process
                                                       by startup cycle counter overrun
        _BitSet (StartupCycleModeControl, 0);          // AutoRestart of coupling process acti-
                                                       vated
        StartupCycleCounterMaxValue       = 100000;    // Overrun value of the startup cycle
                                                       counter
        StartupCycleMasterLength          = 25000;     // Length of master travel until startup
                                                       cycle finished
        _BitSet (StartupCycleModeControl, 12);         // Selection of "position-dependent
                                                       startup cycle process"
        RegisterLoopDXDTOut               = 2;         // Limiting of correction mechanism
        _BitSet(StopCycleModeControl, 0);              // Decoupling in main state 1 (x-con-
                                                       trol)
        _BitSet (SynchronousModeControl, 0);           // "Movement to TargetPos (H492)" acti-
                                                       vated
        _BitSet (SynchronousModeControl, 1);           // No lag error monitoring
      }
```

## 8.3 Example 3

***Objective***

A slave drive is to be operated at a synchronous angle to a master drive. The gear units used in this case are the same and have a gear ratio of 1:1. The master and slave inverters are connected via SBus. The slave inverter is controlled via the binary inputs. Binary inputs X13:5 (DIØ4) and X13:6 (DIØ5) should be used for controlling the startup and stop cycle processes. Both binary inputs must be programmed to "NO FUNCTION."

- "1" signal on DIØ4 → starts the startup cycle process. The startup cycle process should be position-dependent and be completed after 10 000 master increments.
- "1" signal on DIØ5 → starts the stop cycle process.

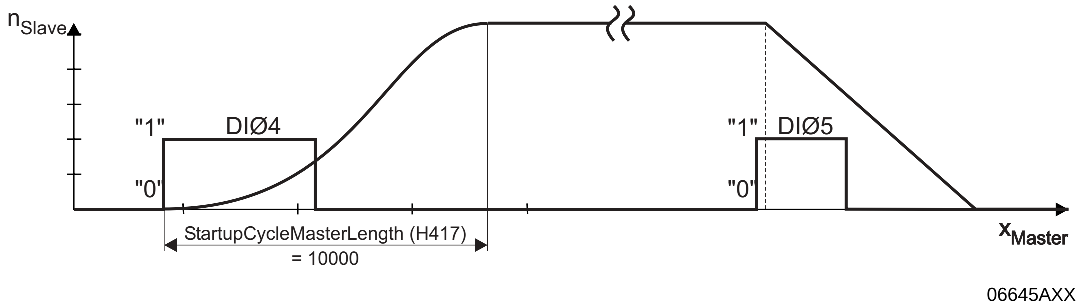



*Figure 55: Event-driven coupling and decoupling*

The necessary IPOS$^{plus®}$ system variables are set in the initialization function.

Two transmit data objects (master position H511 and synchronization ID) are set up in the main program of the master inverter and sent on the SBus when cyclical data transmission starts.

One receive data object for the master position sent on the SBus is set up in the main program of the slave inverter and cyclical data transmission is started.

The master and slave inverters must have different SBus addresses (P881).

**Note the following settings of the master inverter (→ Sec. 5.3):**

- The number of the "Synchronization ID" transmit object must not be the same as parameter value P885.
- The "Cycle time" in the SCOM command for the synchronization ID must be 5 ms.
- The "Cycle time" in the SCOM command for the master position must be 1 ms.
- The master telegram must be created **before** the synchronization telegram.

**Note the following settings of the slave inverter (→ Sec. 5.3):**

- The P885 parameter value must be the same as the number of the "Synchronization ID" transmit object.
- The *H430 MasterSource* system variable must be the same as the value of the D pointer (→ SCOM command structure).

### IPOS$^{plus®}$ program for master inverter

```
/*=============================================================================
IPOS source file Example01.h
===============================================================================*/
#include <const.h>

SCTRCYCL Position;                    // SEW standard structure for the _SbusCommDef statement
SCTRCYCL SynchID;

/*=============================================================================
Main function (IPOS initial function)
===============================================================================*/
main ()
    {

/*=============================================================================
Initialization
===============================================================================*/

Position.ObjectNo=1100;               // Describe the SEW standard structure:
Position.CycleTime=1;                 // Data object no. 1100 (32-bit master position to be sent/
                                      H511)
Position.Offset=0;                    // is sent on the SBus (cycle time 1 ms, MOTOROLA format)
Position.Format=4;
Position.DPointer=511;
Position.Result=0;

SynchID.ObjectNo=1090;                // Describe the SEW standard structure:
SynchID.CycleTime=5;                  // Data object no. 1090 (sync telegram to be sent)
SynchID.Offset=0;                     // is sent on the SBus (cycle time 5 ms)
SynchID.Format=0;
SynchID.DPointer=0;
SynchID.Result=0;

_SBusCommDef(SCD_TRCYCL, Position);   // Creating the transmit data objects
                                      // for cyclical data transmission using
_SBusCommDef(SCD_TRCYCL, SynchID);    // an SBus connection

_SBusCommOn();                        // Initialization of the send data objects and
                                      // start of cyclical data transmission via SBus

/*=============================================================================
Main program loop
===============================================================================*/
while(1)
    {
    }
}
```

### IPOS[plus®] program for slave inverter

```
/*==========================================================================
IPOS source file
for Synchronous Drive Control
SEW-EURODRIVE GmbH & Co KG
Ernst-Blickle-Str. 42
D76646 Bruchsal
sew@sew-eurodrive.de
http://www.SEW-EURODRIVE.de
============================================================================*/
#pragma      var                 300  309
#pragma      globals             310  349

#include     <const.h>
#include     <Example03.h>                 // Header file with
                                           // variable designations
                                           // and initialization function

SREC Posi-                                 // SEW standard structure for the _SbusCommDef state-
tion;                                      ment

*/==========================================================================
                                Main function (IPOS start function)
============================================================================*/
main ()
     {
*/---------------------------------------------------------
                   Startup
---------------------------------------------------------*/
InitSynchronization();                //Call up initialization function

Position.ObjectNo=1100;               // Describe the SEW standard structure:
Position.Format=4;                    // Data object no. 1100 (32-bit master position to be
                                      received)
Position.DPOinter=200;                // is sent to variable H200

_SBusCommDef(SCD_REC, Position);      // Creating a receive data object for cyclical
                                      // data transmission using an SBus connection

_SBusCommOn();                        // Initialization of the receive data object and start of
                                      // start of cyclical data transmission via SBus

/*----------------------------------------------------------
                 Main program loop
----------------------------------------------------------*/
while(1)
     {
     }
}
```

### Header file with variable designation

```
/*******************************************************************************
ISync.h
Data and startup header file for IPOS+ Compiler.
Data file Movidrive Synchronous Drive Control Version 1.0
*******************************************************************************/

#define         SynchronousMode                         H425
#define         SynchronousModeControl                  H426
#define         SynchronousState                        H427
#define         GFMaster                                H428
#define         GFSlave                                 H429
#define         MasterSource                            H430
#define         Reserved1                               H431
#define         LagDistance64Low                        H432
#define         LagDistance64High                       H433
#define         LagDistance32                           H434
#define         Reserved2                               H435
#define         Reserved3                               H436
#define         SlaveTrim                               H437
#define         XMasterPos                              H438
#define         SpeedFreeMode                           H439
#define         Reserved4                               H440
#define         Reserved5                               H441
#define         MasterTrimX14                           H442
#define         Reserved6                               H443
#define         ReSprintClose                           H444
#define         Reserved7                               H445
#define         MFilterTime                             H446

// variables for StartupCycle, StopCycle and OffsetCycle
#define         StopCycleMode                           H400
#define         StopCycleModeControl                    H401
#define         StopCycleState                          H402
#define         StopCycleInputMask                      H403

#define         StartupCycleMode                        H410
#define         StartupCycleModeControl                 H411
#define         StartupCycleState                       H412
#define         StartupCycleInputMask                   H413
#define         StartupCycleCounter                     H414
#define         StartupCycleCounterMaxValue             H415
#define         StartupCycleDelayDI02                   H416
#define         StartupCycleMasterLength                H417
#define         OffsetCycleMode                         H360
#define         OffsetCycleModeControl                  H361
#define         OffsetCycleState                        H362
```

```
#define    OffsetCycleInputMask              H363
#define    OffsetCycleCounter                H364
#define    OffsetCycleCounterMaxValue        H365
#define    OffsetCycleMasterLength           H366
#define    OffsetCycleValue                  H367

// variables to Register Control
#define    RegisterLoopOut                   H389
#define    RegisterLoopDXDTOut               H390

// variables for virtual encoder
#define    VEncoderMode                      H370
#define    VEncoderModeControl               H371
#define    VEncoderState                     H372
#define    VEncoderNSetpoint                 H373
#define    VEncoderNActual                   H374
#define    VEncoderXSetpoint                 H375
#define    VEncoderXActual                   H376
#define    VEncoderdNdT                      H377
```

```
//Startup data from: 08.08.2000 - 16:14:58
InitSynchronization()
       {
          for (H0=128; H0<=457; H0++)                // Reset variables greater than H128
          {
          *H0=0;
          }
          _Memorize(MEM_LDDATA);
          _Wait(100);

          GFMaster                         = 1;      // Evaluation of master increments
          GFSlave                          = 1;      // Evaluation of slave increments
          MasterSource                     = 200;    // Source of master increments:
                                                     // Variable H200 "Master position" (via SBus)
          MFilterTime                      = 1;      // Processing of master increments w/o filter
          StartupCycleMode                 = 1;      // Startup cycle mode 1: Event-driven starting
                                                     // of the startup cycle process via binary
                                                     input
          StartupCycleInputMask            = 16;     // Selection of terminal DI04 for startup
                                                     cycle
          StartupCycleMasterLength         = 10000;  // Overrun value of the startup cycle counter
          _BitSet (StartupCycleModeControl, 12);     // Selection of the "position-dependent star-
                                                     tup cycle process"
          RegisterLoopDXDTOut              = 2;      // Limiting of correction mechanism
          StopCycleMode                    = 1;      // Stop cycle mode 1: Event-driven starting of
                                                     the stop cycle process via binary input
          StopCycleInputMask               = 32;     // Selection of terminal DI05 for decoupling
       }
```

# 9 Index

# Address List

| Germany | | | |
|---|---|---|---|
| **Headquarters**<br>**Production**<br>**Sales**<br>**Service** | **Bruchsal** | SEW-EURODRIVE GmbH & Co KG<br>Ernst-Blickle-Straße 42<br>D-76646 Bruchsal<br>P.O. Box<br>Postfach 3023 · D-76642 Bruchsal | Tel. +49 7251 75-0<br>Fax +49 7251 75-1970<br>http://www.sew-eurodrive.de<br>sew@sew-eurodrive.de<br>Service Electronics:<br>Tel. +49 171 7210791<br>Service Gear Units and Motors:<br>Tel. +49 172 7601377 |
| **Assembly**<br>**Service** | **Garbsen**<br>(near Hannover) | SEW-EURODRIVE GmbH & Co KG<br>Alte Ricklinger Straße 40-42<br>D-30823 Garbsen<br>P.O. Box<br>Postfach 110453 · D-30804 Garbsen | Tel. +49 5137 8798-30<br>Fax +49 5137 8798-55<br>scm-garbsen@sew-eurodrive.de |
| | **Kirchheim**<br>(near München) | SEW-EURODRIVE GmbH & Co KG<br>Domagkstraße 5<br>D-85551 Kirchheim | Tel. +49 89 909552-10<br>Fax +49 89 909552-50<br>scm-kirchheim@sew-eurodrive.de |
| | **Langenfeld**<br>(near Düsseldorf) | SEW-EURODRIVE GmbH & Co KG<br>Siemensstraße 1<br>D-40764 Langenfeld | Tel. +49 2173 8507-30<br>Fax +49 2173 8507-55<br>scm-langenfeld@sew-eurodrive.de |
| | **Meerane**<br>(near Zwickau) | SEW-EURODRIVE GmbH & Co KG<br>Dänkritzer Weg 1<br>D-08393 Meerane | Tel. +49 3764 7606-0<br>Fax +49 3764 7606-30<br>scm-meerane@sew-eurodrive.de |
| | Additional addresses for service in Germany provided on request! | | |

| France | | | |
|---|---|---|---|
| **Production**<br>**Sales**<br>**Service** | **Haguenau** | SEW-USOCOME<br>48-54, route de Soufflenheim<br>B. P. 20185<br>F-67506 Haguenau Cedex | Tel. +33 3 88 73 67 00<br>Fax +33 3 88 73 66 00<br>http://www.usocome.com<br>sew@usocome.com |
| **Assembly**<br>**Sales**<br>**Service** | **Bordeaux** | SEW-USOCOME<br>Parc d'activités de Magellan<br>62, avenue de Magellan - B. P. 182<br>F-33607 Pessac Cedex | Tel. +33 5 57 26 39 00<br>Fax +33 5 57 26 39 09 |
| | **Lyon** | SEW-USOCOME<br>Parc d'Affaires Roosevelt<br>Rue Jacques Tati<br>F-69120 Vaulx en Velin | Tel. +33 4 72 15 37 00<br>Fax +33 4 72 15 37 15 |
| | **Paris** | SEW-USOCOME<br>Zone industrielle<br>2, rue Denis Papin<br>F-77390 Verneuil l'Etang | Tel. +33 1 64 42 40 80<br>Fax +33 1 64 42 40 88 |
| | Additional addresses for service in France provided on request! | | |

| Algeria | | | |
|---|---|---|---|
| **Sales** | **Alger** | Réducom<br>16, rue des Frères Zaghnoun<br>Bellevue El-Harrach<br>16200 Alger | Tel. +213 21 8222-84<br>Fax +213 21 8222-84 |

| Argentina | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Buenos Aires** | SEW EURODRIVE ARGENTINA S.A.<br>Centro Industrial Garin, Lote 35<br>Ruta Panamericana Km 37,5<br>1619 Garin | Tel. +54 3327 4572-84<br>Fax +54 3327 4572-21<br>sewar@sew-eurodrive.com.ar |

| Australia | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Melbourne** | SEW-EURODRIVE PTY. LTD.<br>27 Beverage Drive<br>Tullamarine, Victoria 3043 | Tel. +61 3 9933-1000<br>Fax +61 3 9933-1003<br>http://www.sew-eurodrive.com.au<br>enquires@sew-eurodrive.com.au |
| | **Sydney** | SEW-EURODRIVE PTY. LTD.<br>9, Sleigh Place, Wetherill Park<br>New South Wales, 2164 | Tel. +61 2 9725-9900<br>Fax +61 2 9725-9905<br>enquires@sew-eurodrive.com.au |

| Austria | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Wien** | SEW-EURODRIVE Ges.m.b.H.<br>Richard-Strauss-Strasse 24<br>A-1230 Wien | Tel. +43 1 617 55 00-0<br>Fax +43 1 617 55 00-30<br>http://sew-eurodrive.at<br>sew@sew-eurodrive.at |

| Belgium | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Brüssel** | CARON-VECTOR S.A.<br>Avenue Eiffel 5<br>B-1300 Wavre | Tel. +32 10 231-311<br>Fax +32 10 231-336<br>http://www.caron-vector.be<br>info@caron-vector.be |

| Brazil | | | |
|---|---|---|---|
| **Production**<br>**Sales**<br>**Service** | **Sao Paulo** | SEW-EURODRIVE Brasil Ltda.<br>Avenida Amâncio Gaiolli, 50<br>Caixa Postal: 201-07111-970<br>Guarulhos/SP - Cep.: 07251-250 | Tel. +55 11 6489-9133<br>Fax +55 11 6480-3328<br>http://www.sew.com.br<br>sew@sew.com.br |
| | Additional addresses for service in Brazil provided on request! | | |

| Bulgaria | | | |
|---|---|---|---|
| **Sales** | **Sofia** | BEVER-DRIVE GMBH<br>Bogdanovetz Str.1<br>BG-1606 Sofia | Tel. +359 2 9532565<br>Fax +359 2 9549345<br>bever@mbox.infotel.bg |

| Cameroon | | | |
|---|---|---|---|
| **Sales** | **Douala** | Electro-Services<br>Rue Drouot Akwa<br>B.P. 2024<br>Douala | Tel. +237 4322-99<br>Fax +237 4277-03 |

| Canada | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Toronto** | SEW-EURODRIVE CO. OF CANADA LTD.<br>210 Walker Drive<br>Bramalea, Ontario L6T3W1 | Tel. +1 905 791-1553<br>Fax +1 905 791-2999<br>http://www.sew-eurodrive.ca<br>l.reynolds@sew-eurodrive.ca |
| | **Vancouver** | SEW-EURODRIVE CO. OF CANADA LTD.<br>7188 Honeyman Street<br>Delta. B.C. V4G 1 E2 | Tel. +1 604 946-5535<br>Fax +1 604 946-2513<br>b.wake@sew-eurodrive.ca |
| | **Montreal** | SEW-EURODRIVE CO. OF CANADA LTD.<br>2555 Rue Leger Street<br>LaSalle, Quebec H8N 2V9 | Tel. +1 514 367-1124<br>Fax +1 514 367-3677<br>a.peluso@sew-eurodrive.ca |
| | Additional addresses for service in Canada provided on request! | | |

| Chile | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Santiago de Chile** | SEW-EURODRIVE CHILE LTDA.<br>Las Encinas 1295<br>Parque Industrial Valle Grande<br>LAMPA<br>RCH-Santiago de Chile<br>P.O. Box<br>Casilla 23 Correo Quilicura - Santiago - Chile | Tel. +56 2 75770-00<br>Fax +56 2 75770-01<br>sewsales@entelchile.net |

| China | | | |
|---|---|---|---|
| **Production**<br>**Assembly**<br>**Sales**<br>**Service** | **Tianjin** | SEW-EURODRIVE (Tianjin) Co., Ltd.<br>No. 46, 7th Avenue, TEDA<br>Tianjin 300457 | Tel. +86 22 25322612<br>Fax +86 22 25322611<br>http://www.sew.com.cn |
| **Assembly**<br>**Sales**<br>**Service** | **Suzhou** | SEW-EURODRIVE (Suzhou) Co., Ltd.<br>333, Suhong Middle Road<br>Suzhou Industrial Park<br>Jiangsu Province, 215021<br>P. R. China | Tel. +86 512 62581781<br>Fax +86 512 62581783<br>suzhou@sew.com.cn |

| Colombia | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Bogotá** | SEW-EURODRIVE COLOMBIA LTDA.<br>Calle 22 No. 132-60<br>Bodega 6, Manzana B<br>Santafé de Bogotá | Tel. +57 1 54750-50<br>Fax +57 1 54750-44<br>sewcol@andinet.com |

| Croatia | | | |
|---|---|---|---|
| **Sales**<br>**Service** | **Zagreb** | KOMPEKS d. o. o.<br>PIT Erdödy 4 II<br>HR 10 000 Zagreb | Tel. +385 1 4613-158<br>Fax +385 1 4613-158<br>kompeks@net.hr |

| Czech Republic | | | |
|---|---|---|---|
| **Sales** | **Praha** | SEW-EURODRIVE CZ S.R.O.<br>Business Centrum Praha<br>Luná 591<br>CZ-16000 Praha 6 - Vokovice | Tel. +420 220121234 + 220121236<br>Fax +420 220121237<br>http://www.sew-eurodrive.cz<br>sew@sew-eurodrive.cz |

| Denmark | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Kopenhagen** | SEW-EURODRIVEA/S<br>Geminivej 28-30, P.O. Box 100<br>DK-2670 Greve | Tel. +45 43 9585-00<br>Fax +45 43 9585-09<br>http://www.sew-eurodrive.dk<br>sew@sew-eurodrive.dk |

| Estonia | | | |
|---|---|---|---|
| **Sales** | **Tallin** | ALAS-KUUL AS<br>Paldiski mnt.125<br>EE 0006 Tallin | Tel. +372 6593230<br>Fax +372 6593231 |

| Finland | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Lahti** | SEW-EURODRIVE OY<br>Vesimäentie 4<br>FIN-15860 Hollola 2 | Tel. +358 3 589-300<br>Fax +358 3 7806-211<br>http://www.sew-eurodrive.fi<br>sew@sew-eurodrive.fi |

| Gabon | | | |
|---|---|---|---|
| **Sales** | **Libreville** | Electro-Services<br>B.P. 1889<br>Libreville | Tel. +241 7340-11<br>Fax +241 7340-12 |

| Great Britain | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Normanton** | SEW-EURODRIVE Ltd.<br>Beckbridge Industrial Estate<br>P.O. Box No.1<br>GB-Normanton, West- Yorkshire WF6 1QR | Tel. +44 1924 893-855<br>Fax +44 1924 893-702<br>http://www.sew-eurodrive.co.uk<br>info@sew-eurodrive.co.uk |

| Greece | | | |
|---|---|---|---|
| **Sales**<br>**Service** | **Athen** | Christ. Boznos & Son S.A.<br>12, Mavromichali Street<br>P.O. Box 80136, GR-18545 Piraeus | Tel. +30 2 1042 251-34<br>Fax +30 2 1042 251-59<br>http://www.boznos.gr<br>Boznos@otenet.gr |

| Hong Kong | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Hong Kong** | SEW-EURODRIVE LTD.<br>Unit No. 801-806, 8th Floor<br>Hong Leong Industrial Complex<br>No. 4, Wang Kwong Road<br>Kowloon, Hong Kong | Tel. +852 2 7960477 + 79604654<br>Fax +852 2 7959129<br>sew@sewhk.com |

| Hungary | | | |
|---|---|---|---|
| **Sales**<br>**Service** | **Budapest** | SEW-EURODRIVE Kft.<br>H-1037 Budapest<br>Kunigunda u. 18 | Tel. +36 1 437 06-58<br>Fax +36 1 437 06-50<br>sew-eurodrive.voros@matarnet.hu |

| | | | |
|---|---|---|---|
| **India** | | | |
| **Assembly**<br>**Sales**<br>**Service** | **Baroda** | SEW-EURODRIVE India Pvt. Ltd.<br>Plot No. 4, Gidc<br>Por Ramangamdi · Baroda - 391 243<br>Gujarat | Tel. +91 265 2831021<br>Fax +91 265 2831087<br>sew.baroda@gecsl.com |
| **Technical Offices** | **Bangalore** | SEW-EURODRIVE India Private Limited<br>308, Prestige Centre Point<br>7, Edward Road<br>Bangalore | Tel. +91 80 22266565<br>Fax +91 80 22266569<br>sewbangalore@sify.com |
| | **Mumbai** | SEW-EURODRIVE India Private Limited<br>312 A, 3rd Floor, Acme Plaza<br>Andheri Kurla Road, Andheri (E)<br>Mumbai | Tel. +91 22 28348440<br>Fax +91 22 28217858<br>sewmumbai@vsnl.net |
| **Ireland** | | | |
| **Sales**<br>**Service** | **Dublin** | Alperton Engineering Ltd.<br>48 Moyle Road<br>Dublin Industrial Estate<br>Glasnevin, Dublin 11 | Tel. +353 1 830-6277<br>Fax +353 1 830-6458 |
| **Italy** | | | |
| **Assembly**<br>**Sales**<br>**Service** | **Milano** | SEW-EURODRIVE di R. Blickle & Co.s.a.s.<br>Via Bernini,14<br>I-20020 Solaro (Milano) | Tel. +39 2 96 9801<br>Fax +39 2 96 799781<br>sewit@sew-eurodrive.it |
| **Ivory Coast** | | | |
| **Sales** | **Abidjan** | SICA<br>Ste industrielle et commerciale pour l'Afrique<br>165, Bld de Marseille<br>B.P. 2323, Abidjan 08 | Tel. +225 2579-44<br>Fax +225 2584-36 |
| **Japan** | | | |
| **Assembly**<br>**Sales**<br>**Service** | **Toyoda-cho** | SEW-EURODRIVE JAPAN CO., LTD<br>250-1, Shimoman-no,<br>Toyoda-cho, Iwata gun<br>Shizuoka prefecture, 438-0818 | Tel. +81 538 373811<br>Fax +81 538 373814<br>sewjapan@sew-eurodrive.co.jp |
| **Korea** | | | |
| **Assembly**<br>**Sales**<br>**Service** | **Ansan-City** | SEW-EURODRIVE KOREA CO., LTD.<br>B 601-4, Banweol Industrial Estate<br>Unit 1048-4, Shingil-Dong<br>Ansan 425-120 | Tel. +82 31 492-8051<br>Fax +82 31 492-8056<br>master@sew-korea.co.kr |
| **Lebanon** | | | |
| **Sales** | **Beirut** | Gabriel Acar & Fils sarl<br>B. P. 80484<br>Bourj Hammoud, Beirut | Tel. +961 1 4947-86<br>+961 1 4982-72<br>+961 3 2745-39<br>Fax +961 1 4949-71<br>gacar@beirut.com |
| **Lithuania** | | | |
| **Sales** | **Alytus** | UAB Irseva<br>Merkines g. 2A<br>LT-4580 Alytus | Tel. +370 315 79204<br>Fax +370 315 79688<br>irmantas.irseva@one.lt |
| **Luxembourg** | | | |
| **Assembly**<br>**Sales**<br>**Service** | **Brüssel** | CARON-VECTOR S.A.<br>Avenue Eiffel 5<br>B-1300 Wavre | Tel. +32 10 231-311<br>Fax +32 10 231-336<br>http://www.caron-vector.be<br>info@caron-vector.be |
| **Malaysia** | | | |
| **Assembly**<br>**Sales**<br>**Service** | **Johore** | SEW-EURODRIVE SDN BHD<br>No. 95, Jalan Seroja 39, Taman Johor Jaya<br>81000 Johor Bahru, Johor<br>West Malaysia | Tel. +60 7 3549409<br>Fax +60 7 3541404<br>kchtan@pd.jaring.my |

| Morocco | | | |
|---|---|---|---|
| **Sales** | **Casablanca** | S. R. M.<br>Société de Réalisations Mécaniques<br>5, rue Emir Abdelkader<br>05 Casablanca | Tel. +212 2 6186-69 + 6186-70 + 6186-71<br>Fax +212 2 6215-88<br>srm@marocnet.net.ma |

| Netherlands | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Rotterdam** | VECTOR Aandrijftechniek B.V.<br>Industrieweg 175<br>NL-3044 AS Rotterdam<br>Postbus 10085<br>NL-3004 AB Rotterdam | Tel. +31 10 4463-700<br>Fax +31 10 4155-552<br>http://www.vector.nu<br>info@vector.nu |

| New Zealand | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Auckland** | SEW-EURODRIVE NEW ZEALAND LTD.<br>P.O. Box 58-428<br>82 Greenmount drive<br>East Tamaki Auckland | Tel. +64 9 2745627<br>Fax +64 9 2740165<br>sales@sew-eurodrive.co.nz |
| | **Christchurch** | SEW-EURODRIVE NEW ZEALAND LTD.<br>10 Settlers Crescent, Ferrymead<br>Christchurch | Tel. +64 3 384-6251<br>Fax +64 3 384-6455<br>sales@sew-eurodrive.co.nz |

| Norway | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Moss** | SEW-EURODRIVE A/S<br>Solgaard skog 71<br>N-1599 Moss | Tel. +47 69 241-020<br>Fax +47 69 241-040<br>sew@sew-eurodrive.no |

| Peru | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Lima** | SEW DEL PERU MOTORES REDUCTORES S.A.C.<br>Los Calderos <FmSdata>[Idot ] 120-124<br>Urbanizacion Industrial Vulcano, ATE, Lima | Tel. +51 1 3495280<br>Fax +51 1 3493002<br>sewperu@terra.com.pe |

| Poland | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Lodz** | SEW-EURODRIVE Polska Sp.z.o.o.<br>ul. Techniczna 5<br>PL-92-518 Lodz | Tel. +48 42 67710-90<br>Fax +48 42 67710-99<br>http://www.sew-eurodrive.pl<br>sew@sew-eurodrive.pl |

| Portugal | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Coimbra** | SEW-EURODRIVE, LDA.<br>Apartado 15<br>P-3050-901 Mealhada | Tel. +351 231 20 9670<br>Fax +351 231 20 3685<br>http://www.sew-eurodrive.pt<br>infosew@sew-eurodrive.pt |

| Romania | | | |
|---|---|---|---|
| **Sales**<br>**Service** | **Bucuresti** | Sialco Trading SRL<br>str. Madrid nr.4<br>71222 Bucuresti | Tel. +40 21 230-1328<br>Fax +40 21 230-7170<br>sialco@sialco.ro |

| Russia | | | |
|---|---|---|---|
| **Sales** | **St. Petersburg** | ZAO SEW-EURODRIVE<br>P.O. Box 263<br>RUS-195220 St. Petersburg | Tel. +7 812 5357142 +812 5350430<br>Fax +7 812 5352287<br>sew@sew-eurodrive.ru |

| Senegal | | | |
|---|---|---|---|
| **Sales** | **Dakar** | SENEMECA<br>Mécanique Générale<br>Km 8, Route de Rufisque<br>B.P. 3251, Dakar | Tel. +221 849 47-70<br>Fax +221 849 47-71<br>senemeca@sentoo.sn |

| Serbia and Montenegro | | | |
|---|---|---|---|
| **Sales** | **Beograd** | DIPAR d.o.o.<br>Kajmakcalanska 54<br>SCG-11000 Beograd | Tel. +381 11 3046677<br>Fax +381 11 3809380<br>dipar@yubc.net |

| Singapore | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Singapore** | SEW-EURODRIVE PTE. LTD.<br>No 9, Tuas Drive 2<br>Jurong Industrial Estate<br>Singapore 638644 | Tel. +65 68621701 ... 1705<br>Fax +65 68612827<br>Telex 38 659<br>sales@sew-eurodrive.com.sg |

| Slovakia | | | |
|---|---|---|---|
| **Sales** | **Sered** | SEW-Eurodrive SK s.r.o.<br>Trnavska 920<br>SK-926 01 Sered | Tel. +421 31 7891311<br>Fax +421 31 7891312<br>sew@sew-eurodrive.sk |

| Slovenia | | | |
|---|---|---|---|
| **Sales**<br>**Service** | **Celje** | Pakman - Pogonska Tehnika d.o.o.<br>Ul. XIV. divizije 14<br>SLO – 3000 Celje | Tel. +386 3 490 83-20<br>Fax +386 3 490 83-21<br>pakman@siol.net |

| South Africa | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Johannesburg** | SEW-EURODRIVE (PROPRIETARY) LIMITED<br>Eurodrive House<br>Cnr. Adcock Ingram and Aerodrome Roads<br>Aeroton Ext. 2<br>Johannesburg 2013<br>P.O.Box 90004<br>Bertsham 2013 | Tel. +27 11 248-7000<br>Fax +27 11 494-2311<br>ljansen@sew.co.za |
| | **Capetown** | SEW-EURODRIVE (PROPRIETARY) LIMITED<br>Rainbow Park<br>Cnr. Racecourse & Omuramba Road<br>Montague Gardens<br>Cape Town<br>P.O.Box 36556<br>Chempet 7442<br>Cape Town | Tel. +27 21 552-9820<br>Fax +27 21 552-9830<br>Telex 576 062<br>dswanepoel@sew.co.za |
| | **Durban** | SEW-EURODRIVE (PROPRIETARY) LIMITED<br>2 Monaceo Place<br>Pinetown<br>Durban<br>P.O. Box 10433, Ashwood 3605 | Tel. +27 31 700-3451<br>Fax +27 31 700-3847<br>dtait@sew.co.za |

| Spain | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Bilbao** | SEW-EURODRIVE ESPAÑA, S.L.<br>Parque Tecnológico, Edificio, 302<br>E-48170 Zamudio (Vizcaya) | Tel. +34 9 4431 84-70<br>Fax +34 9 4431 84-71<br>sew.spain@sew-eurodrive.es |

| Sweden | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Jönköping** | SEW-EURODRIVE AB<br>Gnejsvägen 6-8<br>S-55303 Jönköping<br>Box 3100 S-55003 Jönköping | Tel. +46 36 3442-00<br>Fax +46 36 3442-80<br>http://www.sew-eurodrive.se<br>info@sew-eurodrive.se |

| Switzerland | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Basel** | Alfred lmhof A.G.<br>Jurastrasse 10<br>CH-4142 Münchenstein bei Basel | Tel. +41 61 41717-17<br>Fax +41 61 41717-00<br>http://www.imhof-sew.ch<br>info@imhof-sew.ch |

| Thailand | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Chon Buri** | SEW-EURODRIVE (Thailand) Ltd.<br>Bangpakong Industrial Park 2<br>700/456, Moo.7, Tambol Donhuaroh<br>Muang District<br>Chon Buri 20000 | Tel. +66 38 454281<br>Fax +66 38 454288<br>sewthailand@sew-eurodrive.co.th |

| Tunisia | | | |
|---|---|---|---|
| **Sales** | **Tunis** | T. M.S. Technic Marketing Service<br>7, rue Ibn El Heithem<br>Z.I. SMMT<br>2014 Mégrine Erriadh | Tel. +216 1 4340-64 + 1 4320-29<br>Fax +216 1 4329-76 |

| Turkey | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Istanbul** | SEW-EURODRIVE<br>Hareket Sistemleri Sirketi<br>Bagdat Cad. Koruma Cikmazi No. 3<br>TR-81540 Maltepe ISTANBUL | Tel. +90 216 4419163 + 216 4419164 + 216 3838014<br>Fax +90 216 3055867<br>sew@sew-eurodrive.com.tr |

| USA | | | |
|---|---|---|---|
| **Production**<br>**Assembly**<br>**Sales**<br>**Service** | **Greenville** | SEW-EURODRIVE INC.<br>1295 Old Spartanburg Highway<br>P.O. Box 518<br>Lyman, S.C. 29365 | Tel. +1 864 439-7537<br>Fax Sales +1 864 439-7830<br>Fax Manuf. +1 864 439-9948<br>Fax Ass. +1 864 439-0566<br>Telex 805 550<br>http://www.seweurodrive.com<br>cslyman@seweurodrive.com |
| **Assembly**<br>**Sales**<br>**Service** | **San Francisco** | SEW-EURODRIVE INC.<br>30599 San Antonio St.<br>Hayward, California 94544-7101 | Tel. +1 510 487-3560<br>Fax +1 510 487-6381<br>cshayward@seweurodrive.com |
| | **Philadelphia/PA** | SEW-EURODRIVE INC.<br>Pureland Ind. Complex<br>2107 High Hill Road, P.O. Box 481<br>Bridgeport, New Jersey 08014 | Tel. +1 856 467-2277<br>Fax +1 856 467-3792<br>csbridgeport@seweurodrive.com |
| | **Dayton** | SEW-EURODRIVE INC.<br>2001 West Main Street<br>Troy, Ohio 45373 | Tel. +1 937 335-0036<br>Fax +1 937 440-3799<br>cstroy@seweurodrive.com |
| | **Dallas** | SEW-EURODRIVE INC.<br>3950 Platinum Way<br>Dallas, Texas 75237 | Tel. +1 214 330-4824<br>Fax +1 214 330-4724<br>csdallas@seweurodrive.com |
| | Additional addresses for service in the USA provided on request! | | |

| Venezuela | | | |
|---|---|---|---|
| **Assembly**<br>**Sales**<br>**Service** | **Valencia** | SEW-EURODRIVE Venezuela S.A.<br>Av. Norte Sur No. 3, Galpon 84-319<br>Zona Industrial Municipal Norte<br>Valencia, Estado Carabobo | Tel. +58 241 832-9804<br>Fax +58 241 838-6275<br>sewventas@cantv.net<br>sewfinanzas@cantv.net |

SEW-EURODRIVE GmbH & Co KG · P.O. Box 3023 · D-76642 Bruchsal/Germany
Phone +49 7251 75-0 · Fax +49 7251 75-1970
http://www.sew-eurodrive.com · sew@sew-eurodrive.com