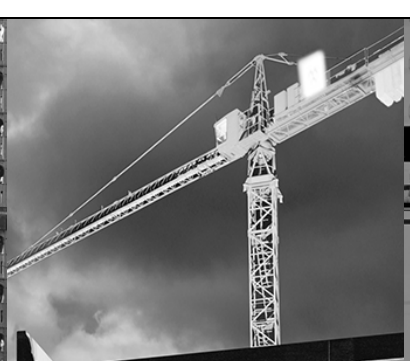
SEW EURODRIVE

Manual

MOVIDRIVE® MDX61B
Internal Synchronous Operation (ISYNC)

Edition 06/2011

19295626 / EN




KKFU 144794 3
42G1

Contents

1 Important Information

INFORMATION

- **This manual does not replace the detailed operating instructions.**
- **Only trained personnel are allowed to perform installation and startup. Adhere to all applicable accident prevention regulations and the MOVIDRIVE® operating instructions.**

1.1 Documentation

- Read through this manual carefully before you commence installation and startup of MOVIDRIVE® inverters with internal synchronous operation.
- This manual assumes that the user has access to and is familiar with the MOVIDRIVE® documentation, in particular the MOVIDRIVE® system manual.
- As a prerequisite to fault-free operation and fulfillment of warranty claims, you must adhere to the information in the documentation.

1.2 Bus systems

General safety notes for bus systems:

This communication system allows you to adapt the MOVIDRIVE® inverter to your application. As with all bus systems, there is a danger of modifications to the parameters that are not visible from outside (in relation to the inverter), which give rise to changes in the inverter behavior. This may result in unexpected (not uncontrolled) system behavior.

1.3 Operating mode "synchronous operation"

The motion controller in synchronous operation processes setpoint changes at the master drive. Set the application module in such a way that an unintended motor start is not possible.

The following measures increase operational safety:

- Deactivating the synchronous operation mode
  - when warnings or errors occur within the sync group or when the drives are not ready for operation
  - when the sync group has been stopped
- Selecting the synchronous operation mode with querying the "ready for operation" message and the "technology options" operating status of all drives.
- If an absolute position reference is required, arrange the individual drives in the positioning operation mode (no offset control).

1.4 Structure of the safety notes

1.4.1 Meaning of the signal words

The following table shows the grading and meaning of the signal words for safety notes, notes on potential risks of damage to property, and other notes.

Signal word	Meaning	Consequences if disregarded
⚠ **DANGER**	Imminent danger	Severe or fatal injuries
⚠ **WARNING**	Possible dangerous situation	Severe or fatal injuries
⚠ **CAUTION**	Possible dangerous situation	Minor injuries
NOTICE	Possible damage to property	Damage to the drive system or its environment
INFORMATION	Useful information or tip: Simplifies the handling of the drive system.	

1.4.2 Structure of the section-related safety notes

Section safety notes do not apply to a specific action, but to several actions pertaining to one subject. The used symbols indicate either a general or a specific hazard.

This is the formal structure of a section safety note:

⚠ SIGNAL WORD

Type and source of danger.

Possible consequence(s) if disregarded.

- Measure(s) to prevent the danger.

1.4.3 Structure of the embedded safety notes

Embedded safety notes are directly integrated in the instructions just before the description of the dangerous action.

This is the formal structure of an embedded safety note:

- ⚠ **SIGNAL WORD** Nature and source of hazard.

  Possible consequence(s) if disregarded.

  – Measure(s) to prevent the danger.

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 for the following industries and applications:

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

2.1.1 Advantages of internal synchronous operation

- Option of position-dependent synchronization → smooth synchronizing without overshooting
- Option of position-dependent offset
- Signed input of the master gear factor
- Option of synchronization with a virtual encoder
- Option 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.

2.2.1 Synchronizing

Time-controlled synchronizing 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 synchronizing 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.

2.2.2 Synchronous operation

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

2.2.3 Disengaging

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

2.3 State machine for 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. See also the chapter Operating principle and functions (page 16).

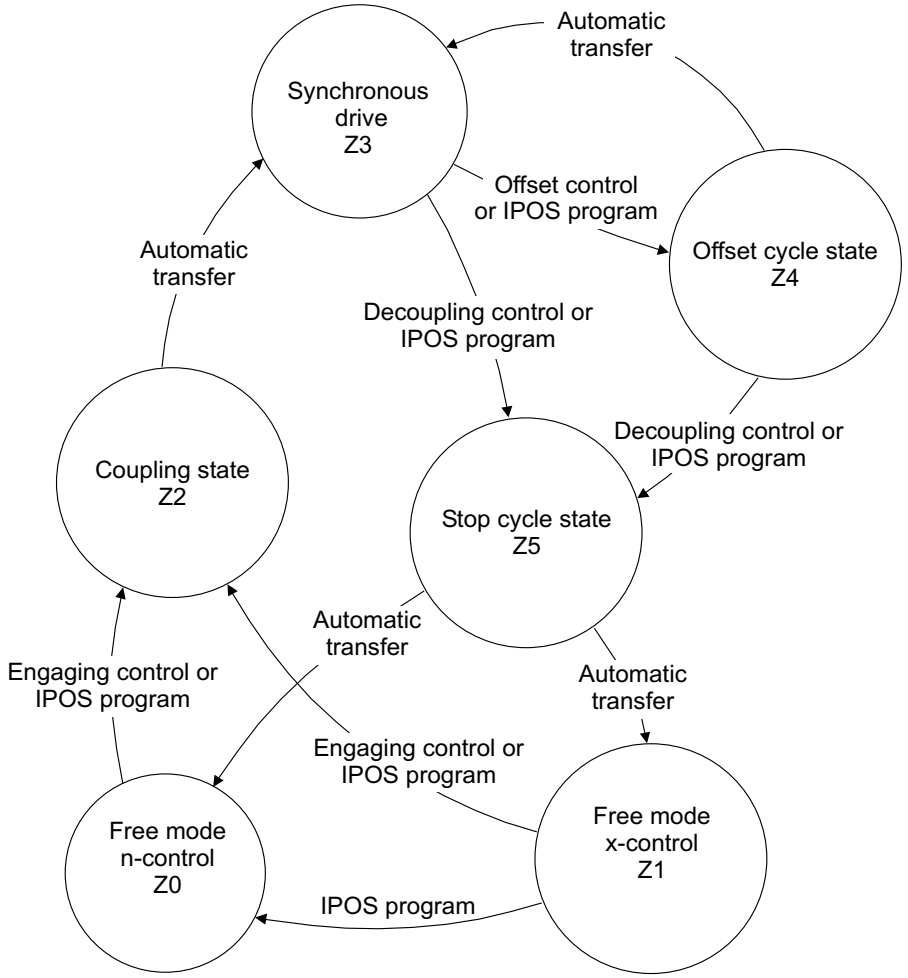


Figure 1: Overview of the state machine for internal synchronous operation

2.3.1 Six main states

The state machine distinguishes between the six states Z0 to Z5. See also the chapter Operating principle and functions (page 16).

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

  The slave drives 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 position control**

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

- **State Z2 = Engaging state**

  The slave drive is synchronized with the master drive using time or position control.

- **State Z3 = Synchronous operation**

  The slave drive moves synchronously with the master drive.

- **State Z4 = Offset**

  In synchronous operation, an offset can be set subject to time or position control.

- **State Z5 = Disengaging state**

  The slave drive exits synchronous operation.

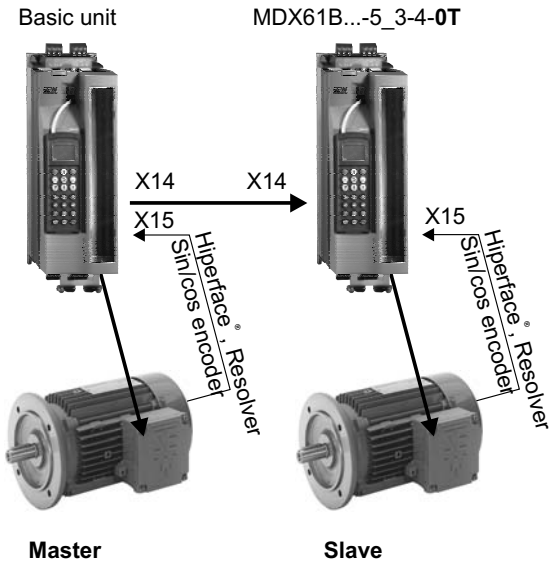
2.4 Controlling internal synchronous operation

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

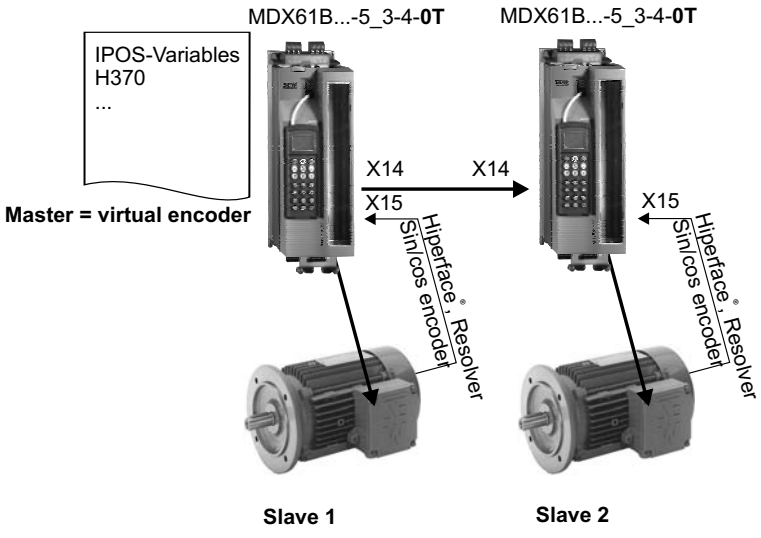
3.1.1 Master/slave operation of two drives



4040685195

Figure 2: Master/slave operation

3.1.2 Master/slave operation of two drives with virtual encoder as master



4040687883

Figure 3: Master/slave operation with virtual encoder

3.1.3 Group configuration: Master and equal-ranked slaves, e.g. multiple column hoist

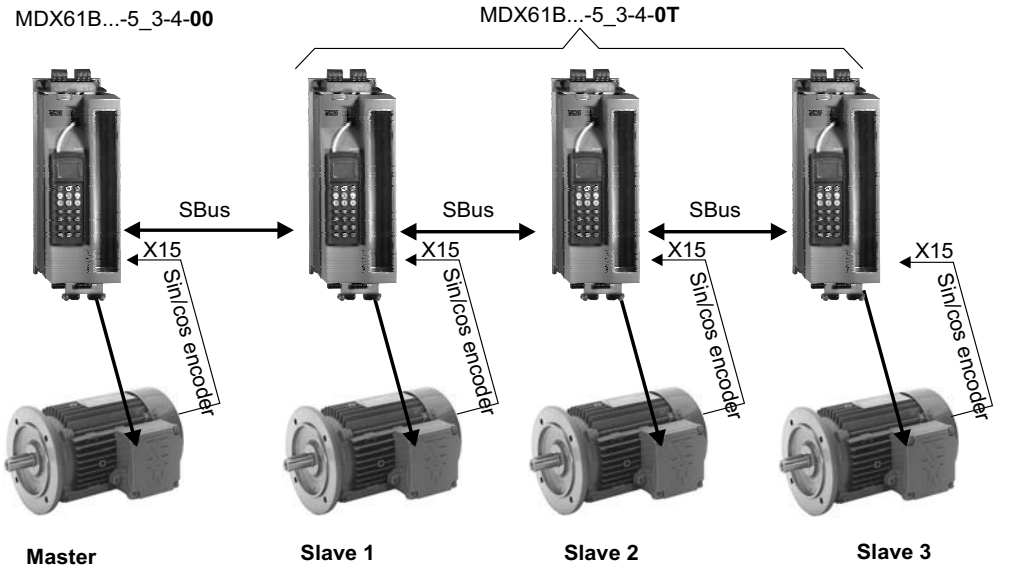


Figure 4: Group configuration

3.1.4 Group configuration with virtual encoder

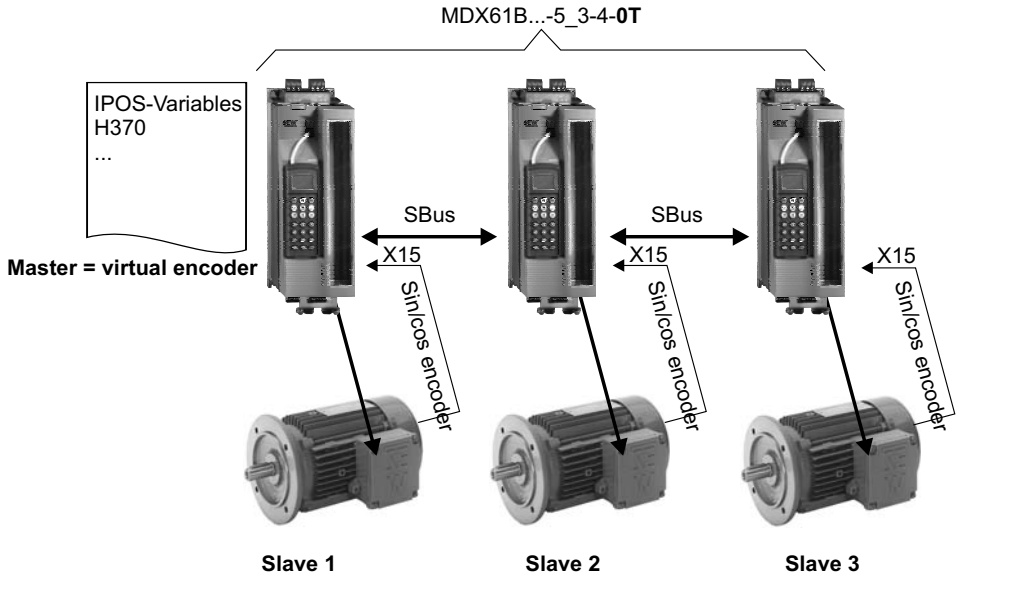


Figure 5: Group configuration with virtual master encoder

3.1.5 Slave drive subject to slip with absolute encoder

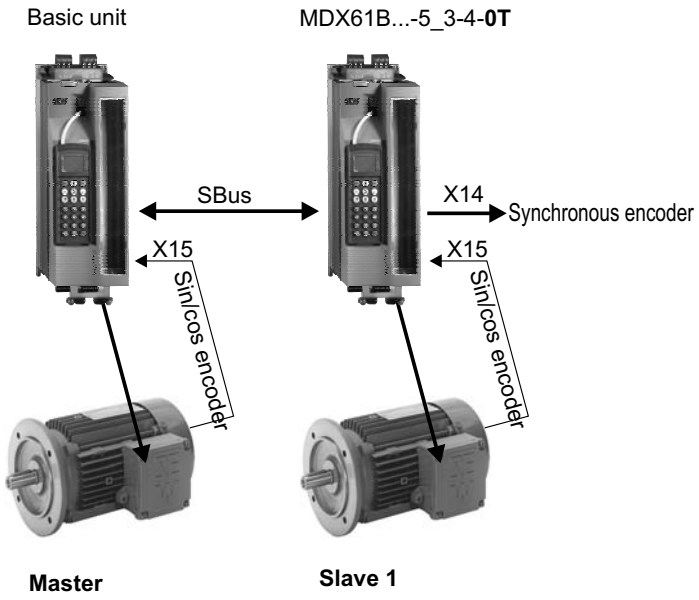


Figure 6: Slave drive subject to slip

3.2 Requirements

INFORMATION

Internal synchronous operation **is not** possible with

- MOVIDRIVE® MDX60B

3.2.1 PC and software

You need the SEW MOVITOOLS® software package version 4.10 or higher 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.

3.2.2 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.

3.2.3 Inverters

- 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
  - Operation modes "CFC", "SERVO" or "VFC-n control" 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 cannot be used.

3.2.4 Motors and encoders

- For operation on MOVIDRIVE® MDX61B:
  - CT/CV asynchronous servomotors, high-resolution sin/cos encoder installed as standard, or Hiperface® encoder.
  - DR/DT/DV series AC motors 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 information

- 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 value greater than the maximum speed of the master. When doing so, take the scaling factors of master and slave into account.
- 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), the drive that 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 highest possible resolution (max. 200 kHz).
- Operation with SBus → Setting up a **cyclical** data transfer in an IPOS$^{plus®}$ 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 proper operation.

INFORMATION

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

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

Master	Slave	Master parameters	Slave parameters	Comment
MDX61B	**MDX61B**	DOØ2 = Output stage ON	DIØ3 = Enable / rapid stop (factory setting) DIØ1 and DIØ2 = No function	Connect master binary outputs DOØ2 with slave binary input DIØ3.

NOTICE

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 Operating Principle and Functions

4.1 Controlling internal synchronous operation

Internal synchronous operation is controlled using IPOS$^{plus®}$ variables within the IPOS$^{plus®}$ 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. See also the chapter "System variables for internal synchronous operation" (page 86). All variables that are connected to internal synchronous operation have symbolic names. These variables are shown below in bold and italics.

4.2 Main state machine

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

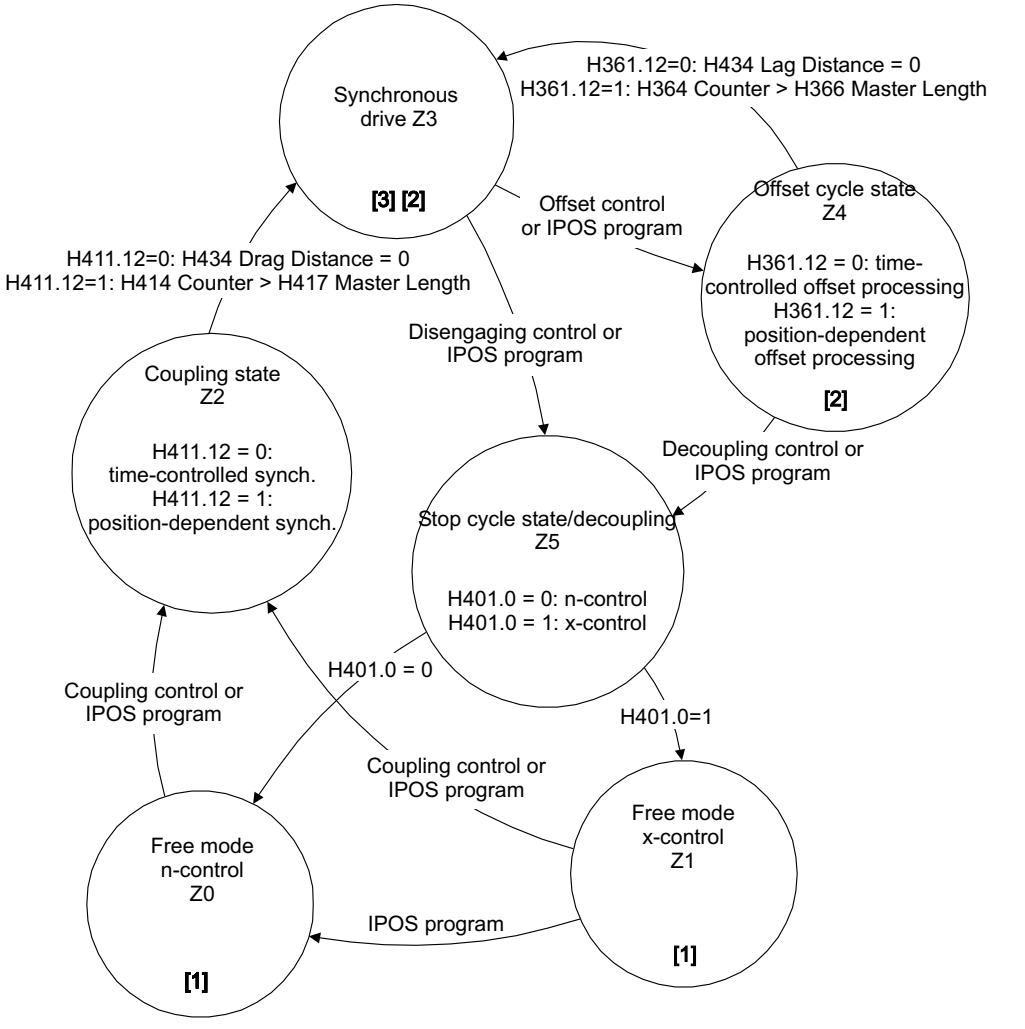


Figure 7: Main state machine of internal synchronous operation with sub-state machines

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

4.2.1 Six main states

The state machine distinguishes between six states (Z0 to Z5). These states are stored in the IPOS$^{plus®}$ variable ***H427 SynchronousState***. See also the chapter "System variables for internal synchronous operation" (page 86).

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

Additional functions with H426

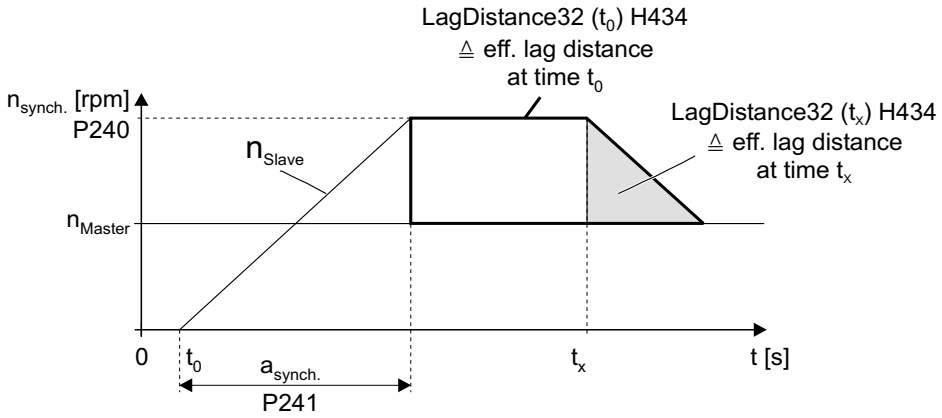
Additional functions can be selected using the bits of the ***H426 SynchronousModeControl*** IPOS$^{plus®}$ variable. See also the chapter "System variables for internal synchronous operation" (page 86).

Bit	Name	Description
0	PosTrim	= 0: Deactivated = 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. = 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 1:1. = 1: The values are scaled with *GFSlave*.
3	ZeroPointMode	= 0: Precontrol is decoupled with "Set DRS zero point". = 1: Precontrol is maintained (speed synchronization with reference to the drive), which means the slave continues to move at the same speed as the master.
4	State Change	= 0: State Change enabled = 1: State Change disabled

4.3 Startup cycle mode control

4.3.1 Time-controlled synchronizing

During time-controlled synchronizing, 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 required time 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 (for example at constant master speed).



4044006667

Figure 8: Speed profile of time-controlled synchronizing

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

Synchronizing 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 (speed synchronism).
- In the second step, any remaining angle differential (*H434 LagDistance32*) is reduced to zero by accelerating or decelerating the drive (positional synchronism).

INFORMATION

Observe the following points for setting the controller:

- n_{max_Slave} (P302) ≥ n_{sync} (P240)
  The output speed of the slave drive must be equal to or greater than the output speed of the master drive.

Observe the following during project planning:

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

4.3.2 Position-dependent synchronizing

Position-dependent synchronizing means that the slave drive only moves in sync 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.

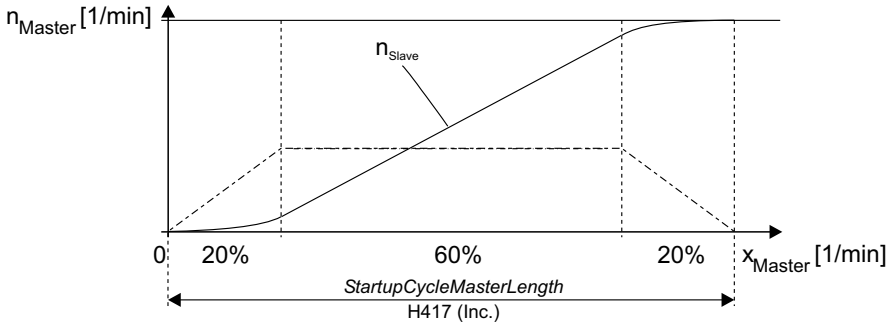


Figure 9: Speed profile for position-dependent synchronization

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

INFORMATION

A control element was added to prevent the loss of master increments during the transition from position-dependent synchronizing to synchronous operation. In this way, a specific 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 setup only takes effect in main state Z3 (synchronous operation) and can be written directly by the user.

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

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

4.3.3 Startup cycle state machine

Startup cycle control reacts in the main states Z0 and Z1. Synchronizing 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* → engaging mode:

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

  Engaging starts when the application assigns the value 2 to the *H427 SynchronousState* system variable.

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

  The startup cycle process is started event-driven via binary input. The *H413 StartupCycleInputMask* system variable defines which binary input triggers the engaging process. The process is started as soon as level "1" is present at the defined binary input. The terminal latency is 1 ms.

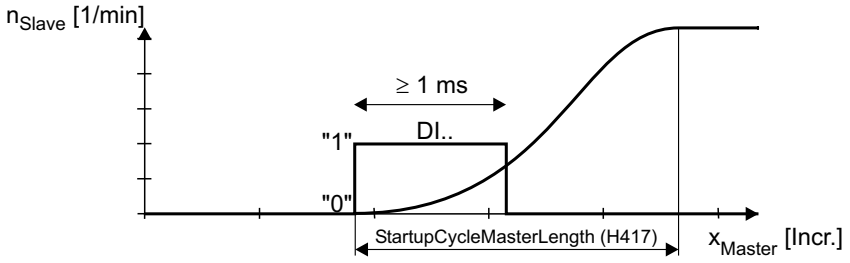


Figure 10: Event-driven starting of position-dependent synchronizing

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

  A signal edge at binary input DI02 or on the C track X14:3 triggers the startup cycle process (interrupt-controlled). For this purpose, binary input DI02 must be programmed to "No function". A delay in relation to the master cycle can be defined for the engaging 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 the startup cycle with X14:3 (C track).

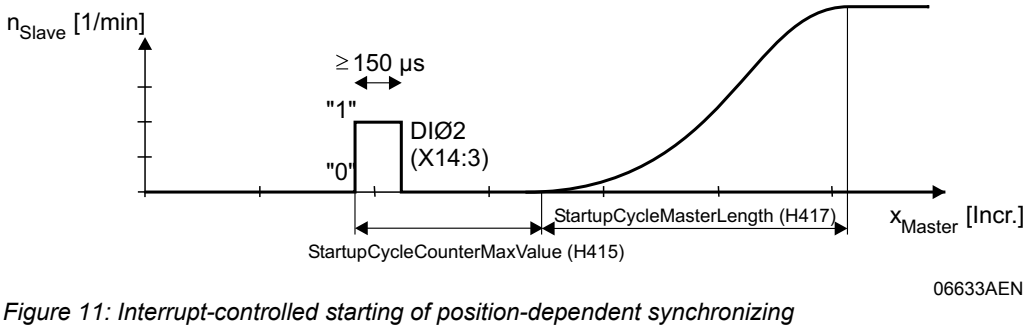


Figure 11: Interrupt-controlled starting of position-dependent synchronizing

Startup cycle state machine in *H412 StartupCycleState:*

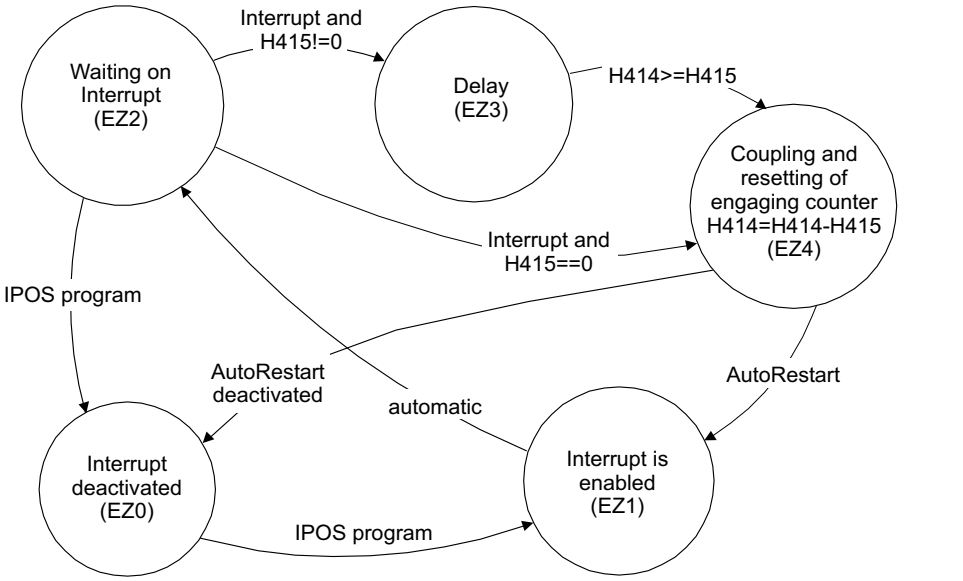


Figure 12: Startup cycle state machine with interrupt control (startup cycle mode 2)

The following application example for engaging mode 2 is illustrated in the figure below:

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 re-positioning [7] are to be performed within one machine cycle [4].

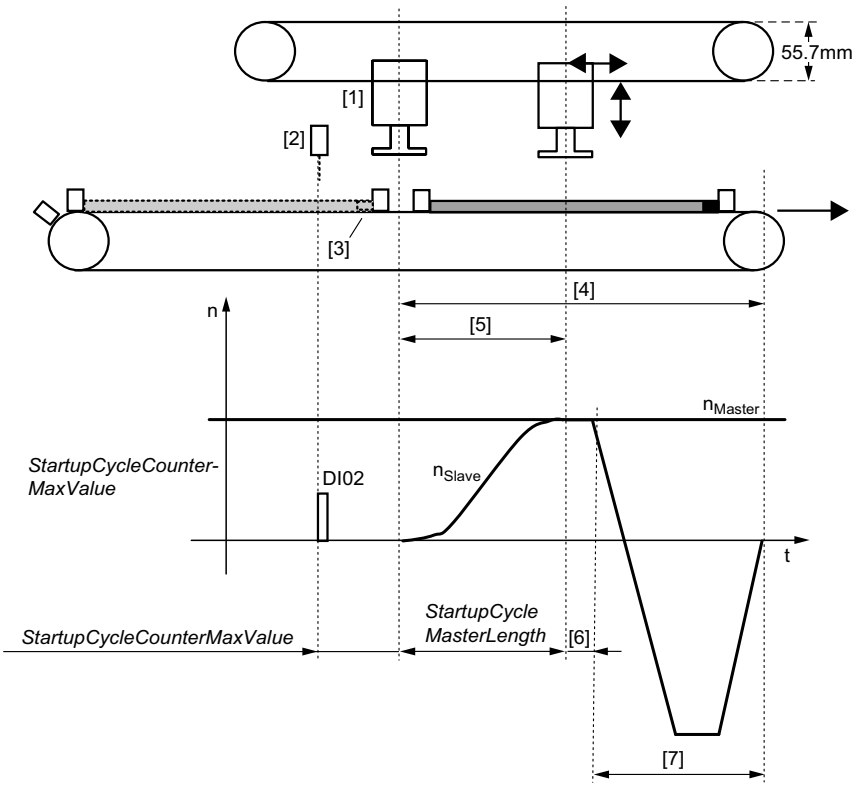
The mechanical structure is as follows:

$i_{Master} = 10$

$i_{Slave} = 7$

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

Length of a machine cycle = 200 mm



4044019723

Figure 13: Application example for startup cycle mode 2

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

In position-dependent synchronizing, 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 × 55.7 mm × 3.14

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

→ *StartupCycleMasterLength* = 200 mm × 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 engaging*(StartupCycleMode = 3)***

  Engaging is initiated by the *H414 StartupCycleCounter* position counter. Engaging 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.

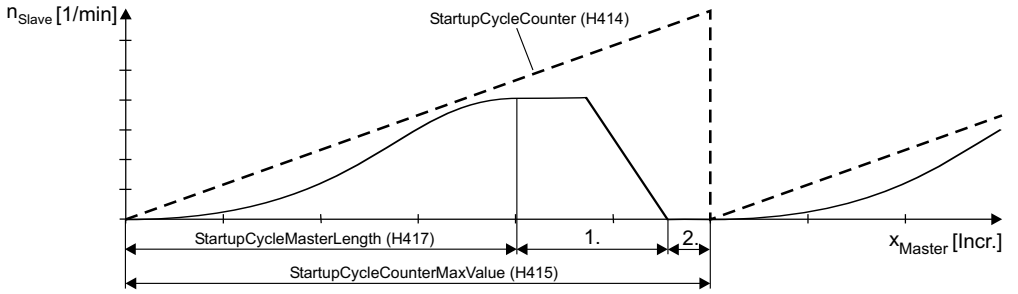
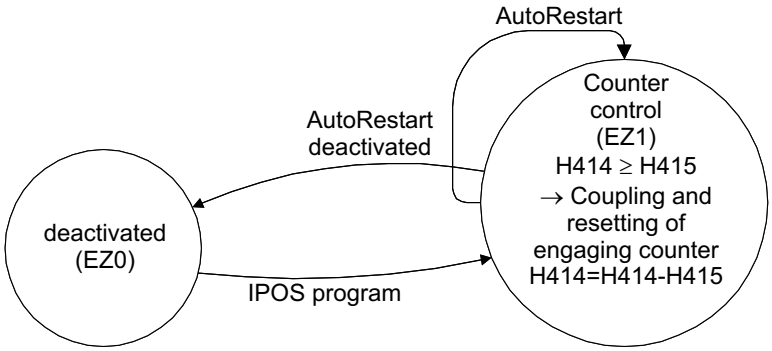


Figure 14: Position-controlled starting of position-dependent synchronizing

[1] Synchronous drive and and stop cycle time
[2] Stop cycle is performed for the slave, time for repositioning to the initial position

Startup cycle state machine in *H412 StartupCycleState:*



4044021387

Figure 15: Startup cycle state machine with position control (startup cycle mode 3)

System variable *H411 StartupCycleModeControl* → Additional functions:

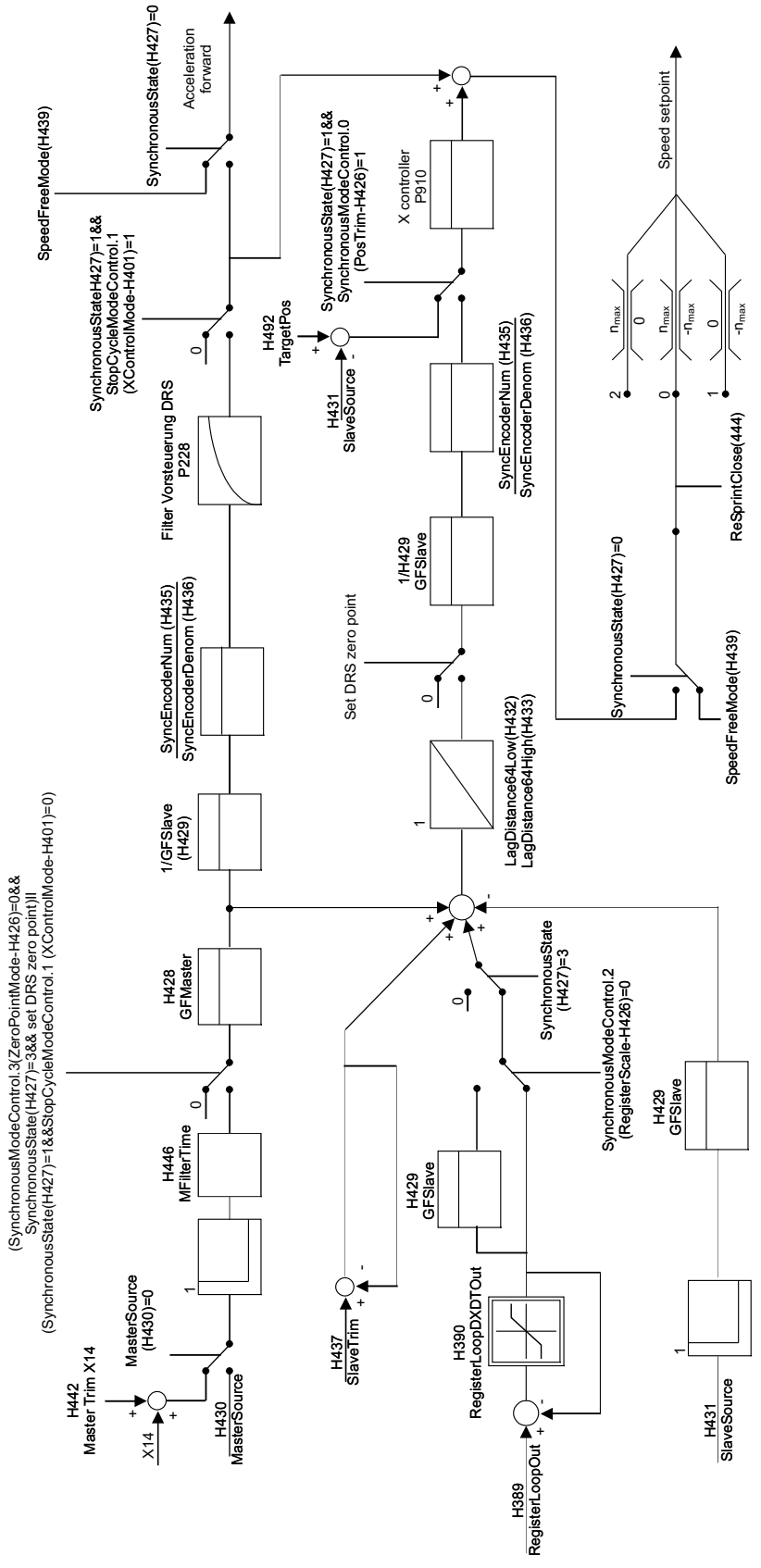
Bit	Name	Description
0	AutoRestart modes 2 and 3	= 0: AutoRestart deactivated. = 1: AutoRestart.
1	StartupDisable modes 2 and 3	= 0: Engaging possible = 1: Engaging disabled
2	InterruptSelect Mode 2	= 0: DI02. = 1: X14 C track.
3	StartupOffset (only with time-based synchronizing)	= 0: No offset during engaging = 1: The offset of variable H367 (OffsetCycleValue) is added to the difference counter and is covered in the startup cycle
4	StartupSearchMode (only with time-based synchronizing)	= 0: Slave moves first to the master speed = 1: The slave accelerates to the maximum speed depending on the angle differential before it moves at master speed.
12	StartupMode	= 0: **Time-controlled synchronizing** 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*). = 1: **Position-dependent synchronizing** 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!**

4.4 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 precontrol and subsequent limiting to the maximum speed provide the speed setpoint for the speed controller.

INFORMATION

A control element was added to prevent the loss of master increments during the transition from position-dependent synchronizing to synchronous operation. In this way, a certain increment differential in each sampling step is added to the 64-bit difference counter by a certain number of increments (*H390 RegisterLoopDXDTOut*). The setup only takes effect in main state Z3 (synchronous operation) and can be written directly by the user.



4044024075

Figure 16: Block circuit diagram for internal synchronous operation

4.4.1 Correction 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.

INFORMATION

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 be 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* acts directly on the output shaft. If the slave scaling factor is not to be taken into account, then set *H426.2 RegisterScale* = 0.
- As soon as the value of *H389 RegisterLoopOut* has been added to the difference counter, *H389 RegisterLoopOut* is automatically overwritten with the value "0".

4.4.2 Slave drive subject to slip

If synchronous operation is required from a drive that is subject to slip, the synchronous encoder function must be activated in MOVIDRIVE® B. The ratio between motor encoder and distance encoder must be specified as numerator/denominator factor in the IPOS$^{plus®}$ 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]

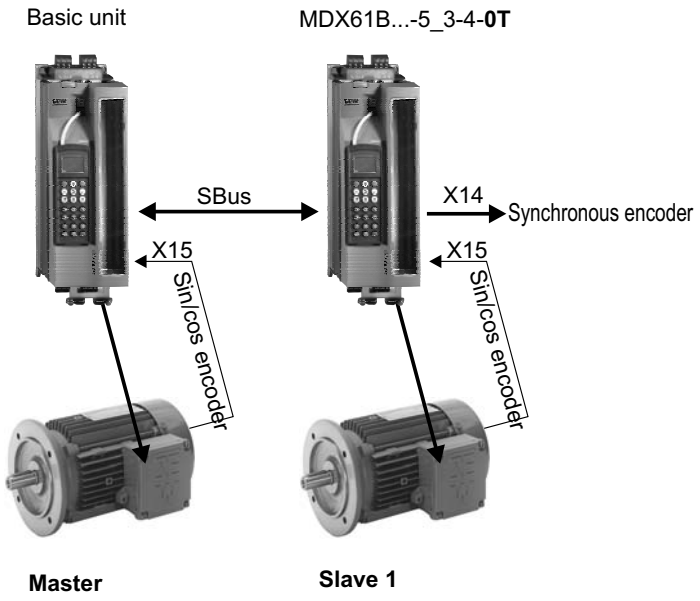


Figure 17: Hardware configuration

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 Example: H431 = 510 // Source of actual position X14 (H510 ActPos_Ext)

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

4.5 Offset cycle type

INFORMATION

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

Offset processing means that an offset is added to the difference counter (main state Z3) during synchronous operation. In this way, the slave drive obtains a new synchronization point and the resulting angle difference is reduced to zero by the control. Actual synchronous operation is reestablished once the new synchronization point has been reached. The offset value can be signed (+/–).

4.5.1 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.

4.5.2 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 has driven at 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 section by section with reference to a constant master speed (→ following figure).

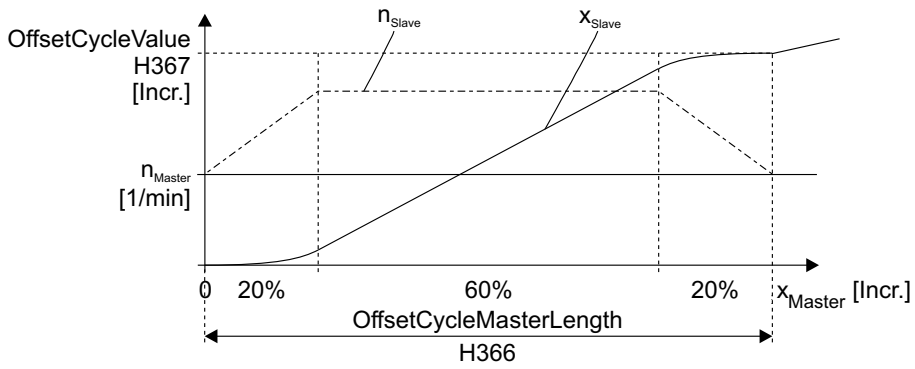


Figure 18: Speed profile for position-dependent offset processing

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.

4.5.3 Offset state machine

Offset control only reacts to specified 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.

System variable *H360 OffsetCycleMode* → Offset mode:

- ***OffsetCycleMode = 0***

  Manual offset processing using the IPOS$^{plus®}$ 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* system variable with a resolution of 1 ms.

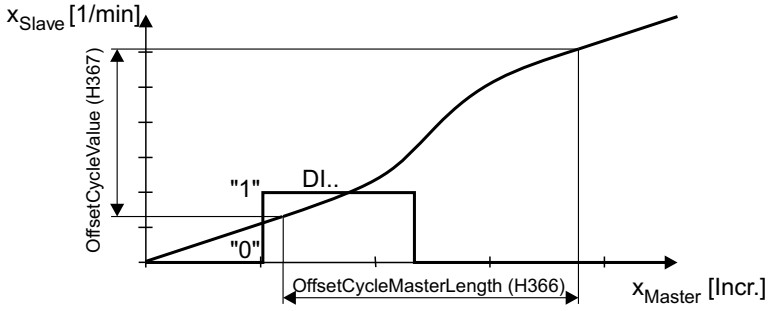


Figure 19: Position-dependent offset processing controlled by binary inputs

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

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

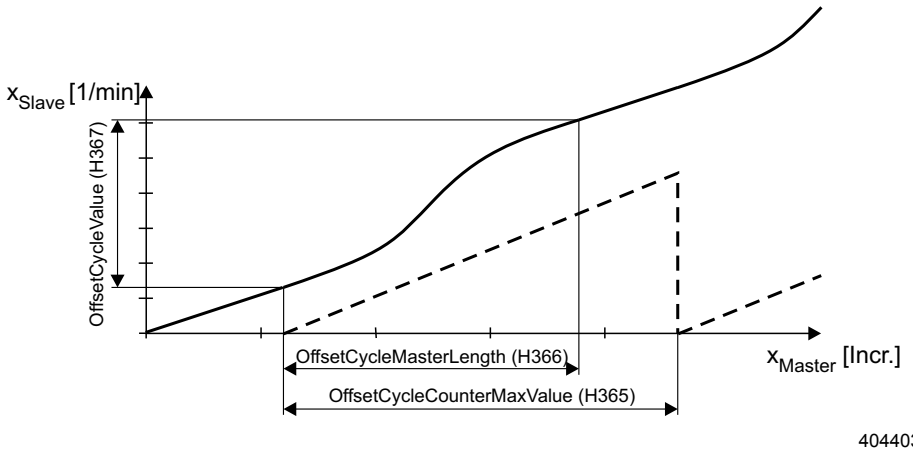
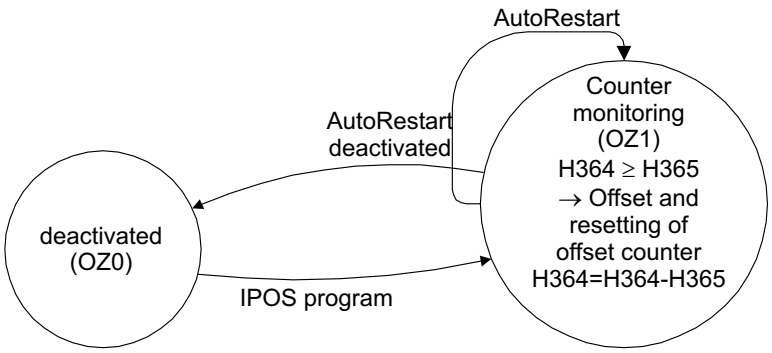


Figure 20: Position-controlled and position-dependent offset processing

System variable *H361 OffsetCycleModeControl:*

Bit	Name	Description
0	AutoRestart (mode 3)	The "AutoRestart" function enables you to determine whether the offset processing cycle should be triggered only once or several times. = 0: AutoRestart deactivated. The offset processing cycle using position control can be run through once. = 1: AutoRestart activated. The offset processing cycle using position control can be restarted after having been run through once.
1	OffsetDisable (mode 3)	Offset processing using position control can be disabled during the initialization phase of the IPOS$^{plus®}$ program. A disabled offset can be re-enabled at any time in the main program. = 0: Offset processing using position control is enabled = 1: Offset processing using position control is disabled.
12	OffsetMode	= 0: Time-controlled offset processing. 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) = 1: Position-dependent offset processing. 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:*



4044039563

Figure 21: Offset state machine

Offset processing is triggered by the master increment counter (*H364 OffsetCycle-Counter*). The offset is processed automatically if the value of the offset counter is greater than the set counter value (*H365 OffsetCycleCounterMaxValue*).

In position-dependent offset processing, the counter end value (*H365 OffsetCycle-CounterMaxValue*) must be greater 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 control using position control is configured as a cycle (offset state machine) and can adopt different offset states (OZ0 to OZ1). Offset processing 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 counter end 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.

4.5.4 Restart after terminated offset processing

NOTICE

When processing an offset during the internal synchronous operation (ISYNC), the machine/system cannot immediately be re-started under certain circumstances in the phase synchronous operation ISYNC after a termination of an offset processing (e.g. with an emergency stop). Termination results in a lag error because the slave does not run synchronously with the master during offset processing.

INFORMATION

Restarting a machine/system after an offset termination is application-dependent, which means there is no general solution. In fact, the optimal solution for the application can be programmed in the IPOS$^{plus®}$ program taking into account all system states.

In this case, the following must be differentiated:

- **Time-dependent offset processing:** Here, only the offset value H367 *OffsetCycleValue* is written to the difference counter H434 *LagDistance32* at the start of the offset process. Consequently, the reference between master and slave can be easily reestablished during restart in which the lag distance H434 LagDistance is processed.
- **Position-dependent offset processing:** Since the offset value H367 *OffsetCycleValue* is processed here position-dependent to the master length stored in H366 *OffsetCycleMasterLength*, the desired offset cannot be "abruptly" written to the difference counter. It is therefore not possible to only process the value in H434 *LagDistance32* during a restart. The restart strategy can appear differently depending on the application:
  - Reset lag distance with "set DRS zero point" and the new startup cycle process. Depending on the application, you must ensure that the original reference to the master is lost.
  - Perform a positioning process to reduce the value of the established lag distance and reestablish the reference to the master.
  - Perform a position-dependent startup cycle process with calculated distances for master and slave that take into account the offset distance already processed in order to reestablish a defined reference between master and slave.
  - Perform a time-dependent startup cycle process with writing of the difference counter with the difference between the offset distance already traveled and the originally specified offset distance.

For a positioning process, it may be very important to know how much of the original offset value was already processed. This value is shown at the moment of cancellation in the H434 *LagDistance32* variable. Note that this lag distance still changes immediately after the offset cancellation due to the stopping of the master and slave axis. For this reason, this value can be temporarily stored at the moment of cancellation using variable interrupt. For any positioning processes, we recommend that you also temporarily save the current position of master and slave with the same interrupt.

Below is a suggestion for an IPOS$^{plus®}$ program with which the values of *Lagdistance32*, the master and the slave position, can be saved at the moment of termination using variable interrupt.

Program suggestion

```
 // Insert this variable definition in the initialization part
  // Initialize variable interrupt
  Vint.Control  = 2; // Interrupt Task 3
  Vint.IntNum   = 1; // Interr. no. 1
  Vint.SrcVar   = numof(SynchronousState);// Source variable to be monitored
  Vint.CompVar  = 3; //
  Vint.Mode     = 10; // Single interrupt if "SourceVar == CompVar"
  Vint.Priority = 1; // Priority of interr. (1-10;  10 = highest priority)
  Vint.IntEvent = 0;
/*=============================================
                   Task3
=============================================*/
Task3()
  { //-------------------Variable interrupt activate -------------------------
----
    if( (SynchronousState == Offset_processing) && (Offset_active == no) )
      { Offset_active = yes;
        _SetVarInterrupt( Vint,fn_VarInterrupt );
         LagDistanceEstop = 0;
      } //-------------------------------------------------------------------
----
    if( (SynchronousState == Synchronous_operation) && (Offset_active == yes) )
      { Offset_active = no;
      }
  }
/*=============================================
                Variable interrupt
=============================================*/
fn_VarInterrupt()
  { LagDistanceEstop = -LagDistance32;
    Slave_Pos = ActPos_Mot;
    Master_Pos = *MasterSource;
  }
```

Using variable interrupt, the positions of master and slave are stored in the variables "Slave_Pos", and "Master_Pos" at the moment of cancellation. Using this data and the current master and slave position, a positioning process or a new offset process, for example, can be started application-dependent with new position values to reestablish the reference between master and slave.

4.6 Stop cycle state machine

"Stop cycle" is the name of the process where angular synchronous operation between the slave and master drives is stopped and the slave drive enters free running mode. This means the slave drive can be moved with speed control or held in its current position with position control.

4.6.1 Stop cycle mode control

Stop cycle mode control responds in the main states Z3 (synchronous operation) 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.

System variable *H400 StopCycleMode* → Stop cycle mode:

- ***StopCycleMode = 0***

  Manual stop cycle. 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 stop cycle via binary input. The *H403 StopCycleInputMask* system variable defines which binary input triggers the disengaging process. The process is started as soon as level "1" is present at the defined binary input. The terminal latency is 1 ms.

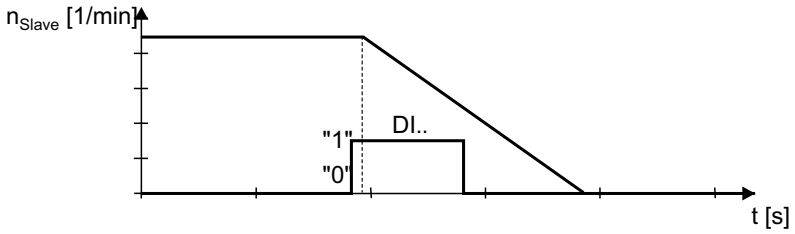


Figure 22: Event-driven stop cycle

System variable *H401 StopCycleModeControl* → additional functions

Bit	Name	Description
0	FreeMode	= 0: Stop cycle in main state 0 (speed control). = 1: Stop cycle in main state 1 (position control).
1[1)]	XControlMode	= 0: Difference counter is cleared. = 1: The difference counter stores the delay between the master and the slave.

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

4.7 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 present 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 velocity > 0. The virtual encoder is stopped (*VEncoderMode* = 0) when the *H374 VEncoderX-Actual* value reaches the *H375 VEncoderXSetpoint* value.

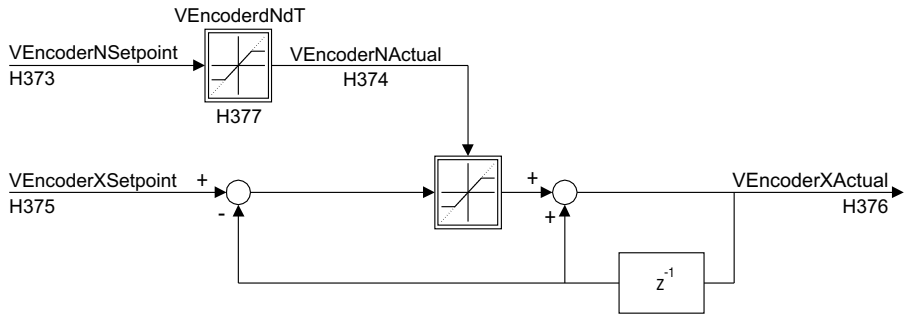


Figure 23: Structural diagram of a virtual encoder with ramp generator

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

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

H373 VEncoderNSetpoint	[1 inc/ms]	→ Setpoint velocity
H375 VEncoderXSetpoint	[1 inc]	→ Target position
H377 VEncoderdNdT	[$1/12^{16}$ inc/ms^2]	→ Acceleration and deceleration

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

- Example 1:

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

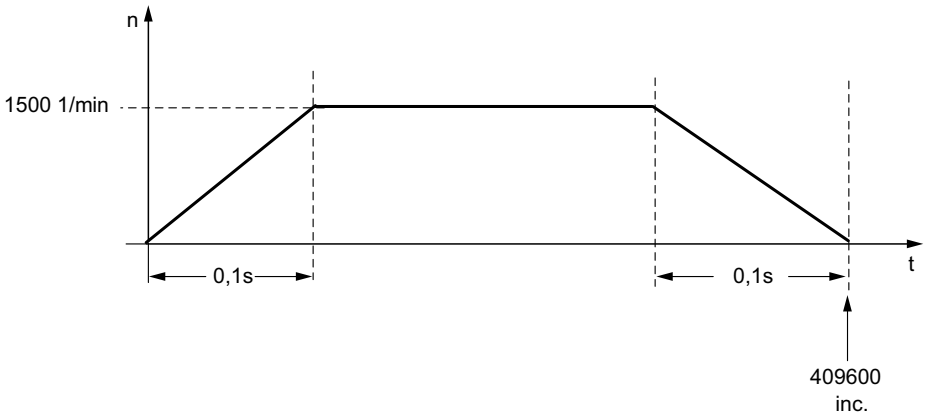
100 inc/ms: 15 ms = 6.6666667 inc/ms^2

6.6666667 inc/ms^2 × 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 rpm, acceleration and deceleration ramps should be 0.1 s. The target position is 409600 inc (= 100 revolutions).



4044051339

H373VEncoderNSetpoint = 1500 rpm × 4096 inc = 6144000 inc/min = 102 inc/ms

→ *H373 VEncoderNSetpoint* **= 102**

H377VEncoderdNdT = (102 inc/ms : 100 ms) × 2^{16} = 66846.72 inc/ms^2 (=1051 E_{hex})

→ VEncoderdNdT = 66846

INFORMATION

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

For *VEncoderdNdT* the value 1.02 must be entered.

(→ 102 inc/ms : 100 ms)

→ *H375 VEncoderXSetpoint* = 409600

→ *H428GFMaster* = *H429 GFSlave* = 1

INFORMATION

If internal synchronous operation is started using the startup interface, enter the value 1.02.

However, if the variable VEncoderdNdT is initialized by the programmer in the IPOS$^{plus®}$ program, the value 664846 must be entered here to reach the required 1500 rpm.

INFORMATION

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**

H373 VEncoderNSetpoint	[1 inc/ms]	→ Setpoint velocity
H377 VEncoderdNdT	[1 inc/ms^2]	→ Acceleration

INFORMATION

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

VEncoderMode = 2 → Endless counter with adjustable acceleration and velocity

VEncoderMode = 1 → reserved

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

H373 VEncoderNSetpoint	[1 inc/ms]	→ Setpoint velocity
H375 VEncoderXSetpoint	[1 inc]	→ Target position
H377 VEncoderdNdT	[1 inc/ms^2]	→ Acceleration

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

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

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

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 whose velocity corresponds to 1000 rpm. The axis should accelerate to this velocity within 0.5 s. The maximum velocity of the axis should be 3000 rpm, which means it must be possible to increase the velocity to this value.

Conversion from rpm to inc/ms:

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

3000 rpm 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*:

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

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

VEncoderNSetpoint [inc/ms] = **10923 [inc/ms]**

→ Determining *VEncoderdNdT*:

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

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 rpm, and it should be possible to adjust the ramp in a range of 0.5 to 5 s.

The following selection was made:

H373 VEncoderNSetpoint = 5000 inc/ms (corresponds to 1000 rpm)

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

H428 GFMaster = 25

H429 GFSlave = 1831

System variable *H371 VEncoderModeControl:* → *additional functions*

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

4.8 Important notes

- The possibility of specifying a signed distance in variables *H417 StartupCycle-MasterLength* 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 operation).
- 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 the remaining distance to 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

4.8.1 Master/slave scaling factors

The objective of phase-synchronous operation is to move two or more drives at the output – and thus the track – synchronously in relation to one another. The synchronous operation controller required for this purpose only processes incremental information of a master encoder and a slave encoder. Therefore, the actual gear unit and additional gear ratios of the application must be reproduced using factors in order to achieve synchronization in a specific proportionality ratio.

When using two identical drives (with the same gear unit reduction ratios, same additional gears, etc.), this means the proportionality ratio is 1:1.

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, any existing additional slave gear, and the master distance.

→ **Calculating *H428 GFMaster***:

GFMaster = Slave gear unit reduction ratio × slave additional gear × master distance

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

→ **Calculating *H429 GFSlave***:

GFSlave = Master gear unit reduction ratio × master additional gear × slave distance

The slave distance 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 ratios are obtained from the information 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 read from the nameplate or obtained by calculation to between two and four decimal places (depending on the maximum possible resolution of the scaling factor).

INFORMATION

For synchronous operation applications, we recommend that you include the individual numbers of teeth of the gear pairs in the scaling factor calculation; i.e. include the individual gear unit reduction stages in the calculation separately. Your SEW-EURODRIVE contact can tell you the number 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 another gear unit reduction ratio and is also included in the calculation.

- **Master/slave distance**

  The master or slave distance relates to the length of travel performed by the master/slave per revolution at 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. As the number can be an irrational number, it is recommended that you calculate the length of travel for endless applications according to the mechanical system used.

  – Applications with a chain sprocket as transmission element:

    Travel length = number of teeth of the chain sprocket × chain link length

  – Applications with a gear rack as transmission element:

    Travel length = number of teeth of the gear × tooth clearance (tip-to-tip) of the rack

  – Applications with a toothed belt as transmission element:

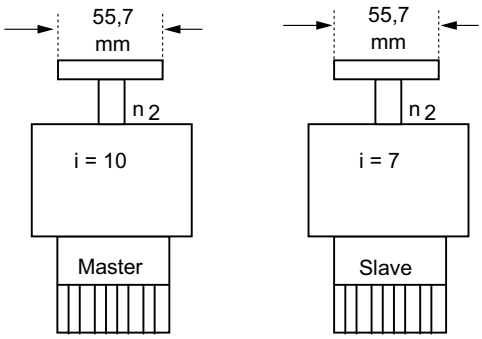
    Travel length = number of teeth of the gear wheel × tooth clearance (center-to-center) of the toothed belt

  – Applications with a spindle as the 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 rpm.



4044062731

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

GFSlave = 10

GFMaster = 7

The following table shows two further examples with different output speeds n_2 of master and slave.

		Setpoint speed n2 [rpm]	n1 [rpm]	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*

INFORMATION

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

4.9 Internal synchronous operation via SBus

4.9.1 Operating principle

A cyclical SBus telegram transfers a master position (e.g. the actual position of the master or the value of the virtual encoder) from the master to the slaves via the SBus. Sending and transferring the master position at equal intervals avoids aliasing. Therefore, the time slices of the inverter are synchronized with a synchronization telegram.

5 Startup

5.1 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.

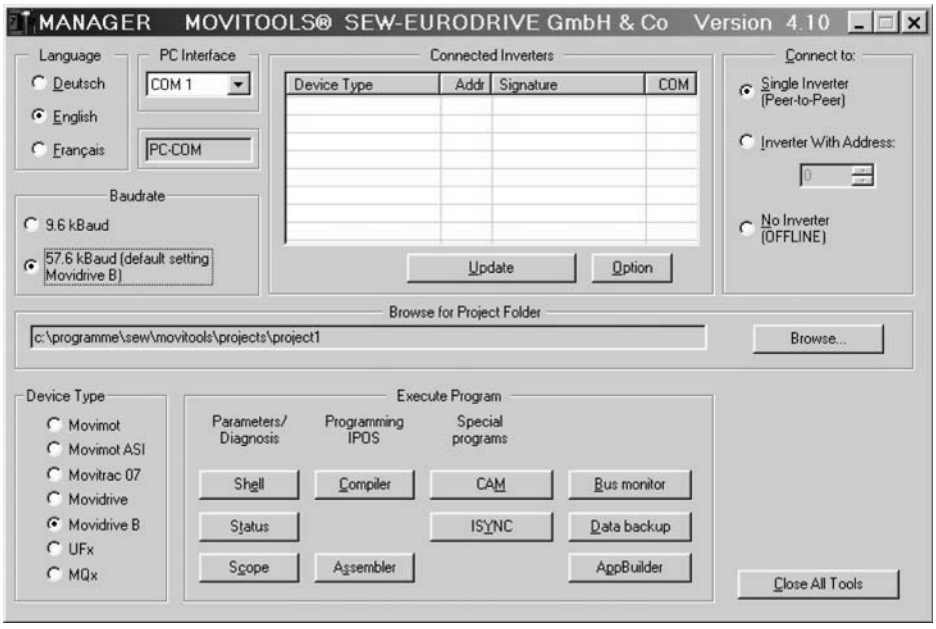
Check the installation and the encoder connection

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

5.2 Preliminary work

Perform the following steps before starting up internal synchronous operation:

- Connect the inverter to the PC via 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 required language in the "Language" group.
- 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" group, select the "Movidrive B" radio button.
- In the "Baud rate" group, select the "57.6 kbaud" radio button (default setting).
- Click [Update] to display the connected inverter.



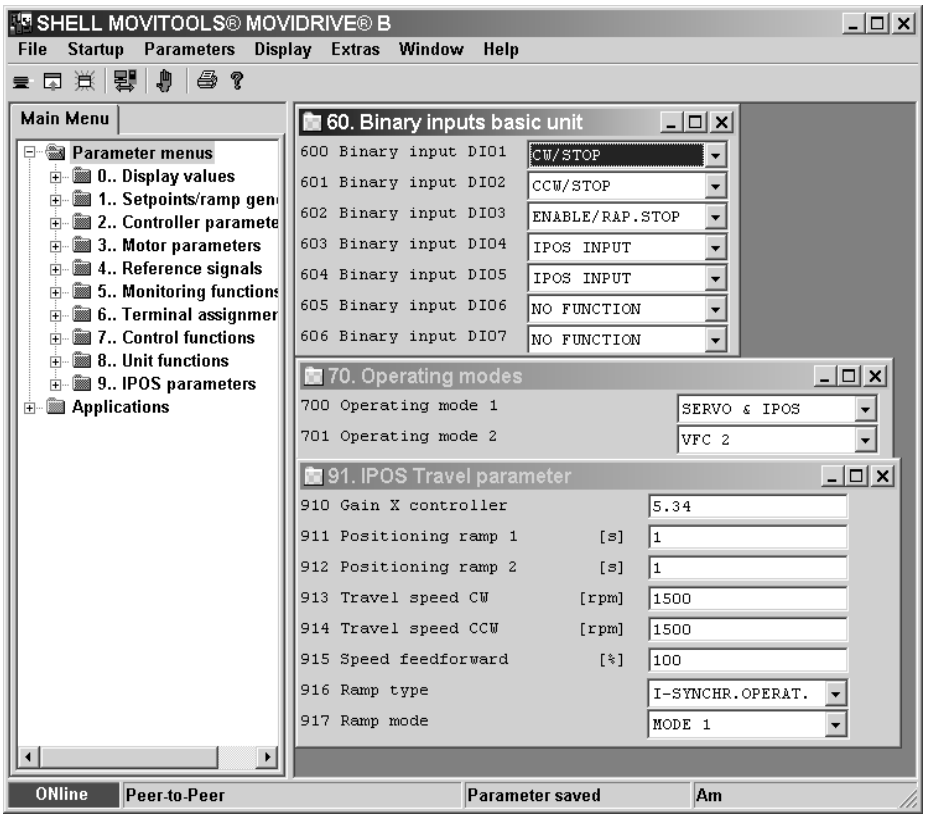
4045974667

Figure 24: MOVITOOLS® initial screen

5.3 Starting up internal synchronous operation

5.3.1 General information

- In the [Execute Program] group, click the [Shell] button under [Parameters/Diagnostics]. The SHELL program is started.
- Set the *P916 Ramp type parameter* to "I-SYNCHR. OPERATION". This setting activates internal synchronous operation. Execute the parameter *P916 Ramp type* with 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).
- Start/stop cycle using interrupt: To prevent assigning terminal functions twice, set the selected input terminal (→ following figure e.g. DI04 and DI05) in the corresponding parameter group 60x or 61x to "NO FUNCTION".
- Startup/stop cycle using binary inputs: To prevent assigning the terminal function twice, set the selected input terminal (→ following figure e.g. DI04 and DI05) in the corresponding parameter group 60x or 61x to "IPOS INPUT".



4045978379

Figure 25: Parameter settings in the Shell program for starting up internal synchronous operation

5.3.2 Startup with SBus connection

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

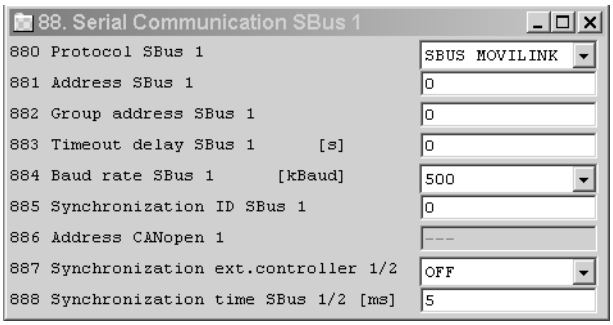
For the master inverter:

- Create the 2 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 less (= 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.

INFORMATION

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 shows the basic principles of the procedure. SEW-EURODRIVE cannot be held liable for incorrect program functions or parameter settings and the consequences thereof.

Required parameter settings



10992AEN

Figure 26: Necessary parameter settings for the master inverter (example)

INFORMATION

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

*IPOS$^{plus®}$ sample program **master inverter***

```
/*============================================================================================
============================================================================================*
/
IPOS source file
The program places the actual position of the master H511 on the SBus.

The following parameters must be set for this purpose:
P880 Protocol SBus 1                    :SBUS MOVILINK
P881 SBus 1 address                     :e.g. "0"
P884 Baud rate SBus 1                   :e.g. "500" (default setting)
P885 Synchr. ID SBus 1                  :e.g. "0" (may not be the same as SBus object "Synchr.ID")
============================================================================================*/
include <const.h>
#include <io.h>
SCTRCYCL Masterposition, Synchronisations_Id;
/*============================================================================================
Main function (IPOS initial function)
============================================================================================*
/
main()
{
/*============================================================================================
Initialization
============================================================================================*
/
Masterposition.ObjectNo=1100;           // Address of the object
Masterposition.CycleTime = 1;           // Time interval [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 interval of the synchr. telegram
Synchronisations_Id.Offset = 0;
Synchronisations_Id.Format = 0;
Synchronisations_Id.DPointer = 0;
Synchronisations_Id.Result = 0;
_SBusCommDef(SCD_TRCYCL, Masterposi-    // Create data objects
tion);                                  // for cyclical data transmission using
_SBusCommDef(SCD_TRCYCL,                // Start data transmission
Synchronizations_Id);
_SBusCommOn();
/*============================================================================================
Main program loop
============================================================================================*
/
while(1)
{
}
}
```

For the 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).

INFORMATION

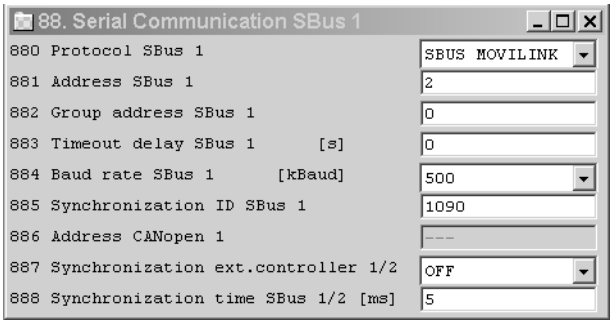
Note that

- The entry *Variable* is selected in the [Master encoder] input field of the [General synchronous operation parameters] startup window.
- The value of the D pointer (SCOM command structure) is selected in the "Variable used" input field, e.g. *H200*, according to the IPOS$^{plus®}$ sample program 3 (see chapter 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 shows the basic principles of the procedure. SEW-EURODRIVE cannot be held liable for incorrect program functions or parameter settings and the consequences thereof.

Required parameter settings



4046101003

Figure 27: Required 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
76646 Bruchsal, Germany
sew@sew-eurodrive.com
http://www.SEW-EURODRIVE.com
============================================================================================
The following parameters have to be set for communication via SBus:
P880 Protocol SBus 1                          : SBUS MOVILINK
P881 SBus 1 address                           : 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>
#include <Winkelsynchronlauf_1.h>
SCREC Masterposition;                         // Note: In this case, the master position is
long Position_Master;                         present on H313
/
*============================================================================================
Mainfunction (IPOS-Startfunction)
============================================================================================
=*/
main()
{
/
*============================================================================================
Startup
============================================================================================
=*/
InitSynchronization();
Masterposition.ObjectNo = 1100;               // Number of the data object
Masterposition.Format = 4;                    // Length of the received object: 4 bytes
Masterposition.Dpointer = numof               // Target address
(Position_Master);                            // Transfer master position
MasterSource = numof (Position_Master);       // Set up data object
_SBusCommDef(SCD_REC,Masterposition);         // Start data transmission
_SBusCommOn();
/
*============================================================================================
Mainprogram loop
============================================================================================
=*/
while(1)
  {

  }
}
```

5.3.3 Startup with slave subject to slip with absolute encoder on the slave distance

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

INFORMATION

Note that

- the entry *SSI encoder* is selected in the [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*)

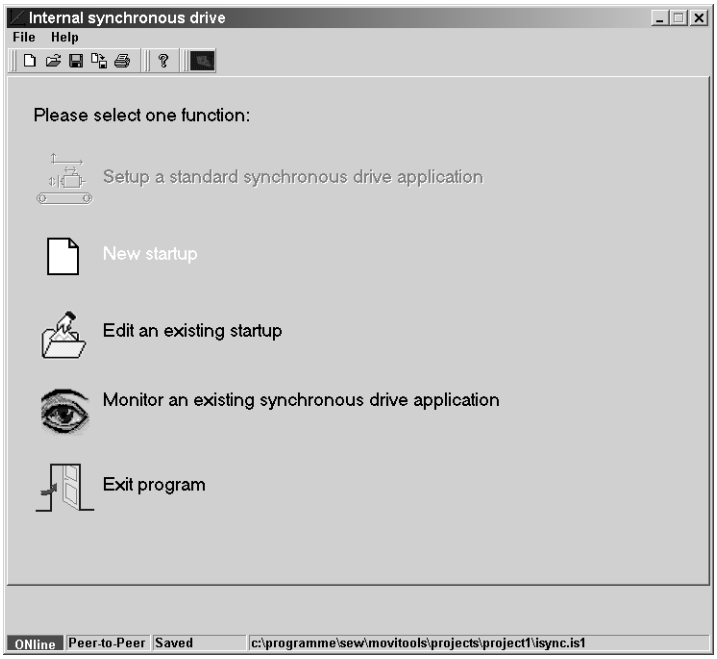
5.4 Startup interface for internal synchronous operation

INFORMATION

This chapter describes the **"New startup" function** of a synchronous operation application as an example.

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

In the MOVITOOLS® manager, select the special program "ISYNC" under [Execute Program]. The startup window for internal synchronous operation opens. In the following, this window is referred to as basic menu.



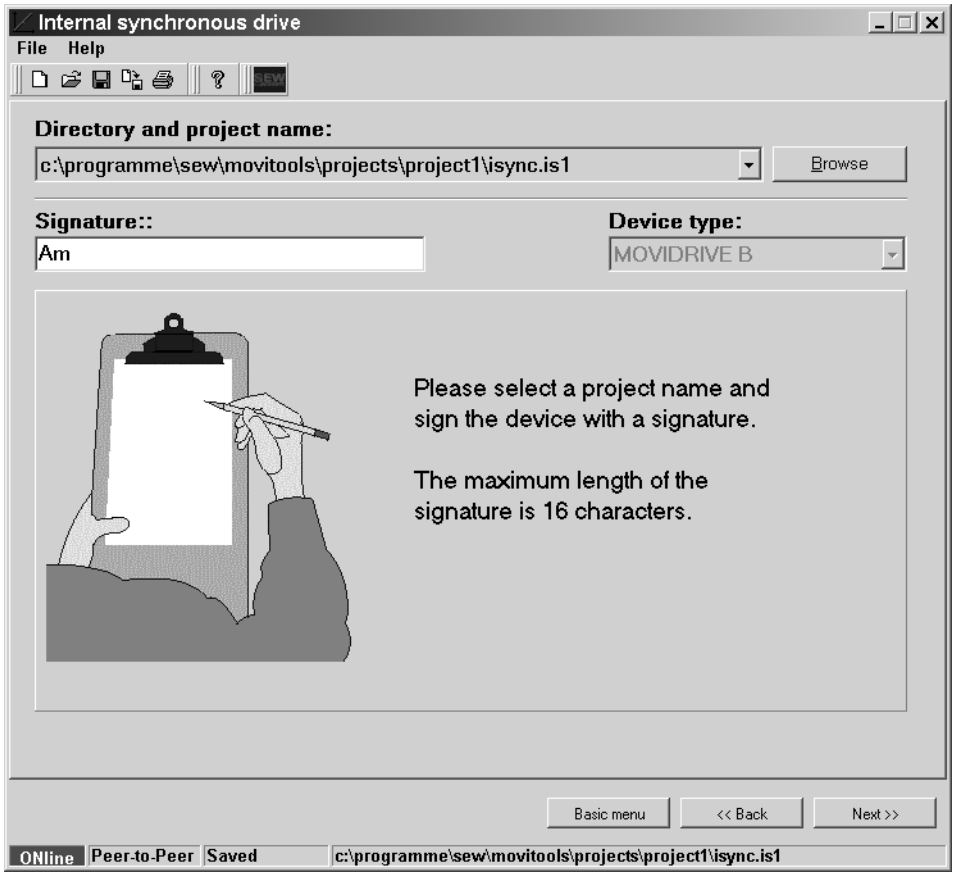
4046105739

Figure 28: Basic menu of the startup interface for internal synchronous operation

Click the function you require in the basic menu:

- New startup: To perform startup for a new project
- Edit an existing startup: To edit the files of an existing project
- Monitor an existing synchronous drive application: To monitor an existing ISYNC project during operation, e.g. diagnosing the drive state, recording the lag error.

The "Project name and path" window opens.



4046107403

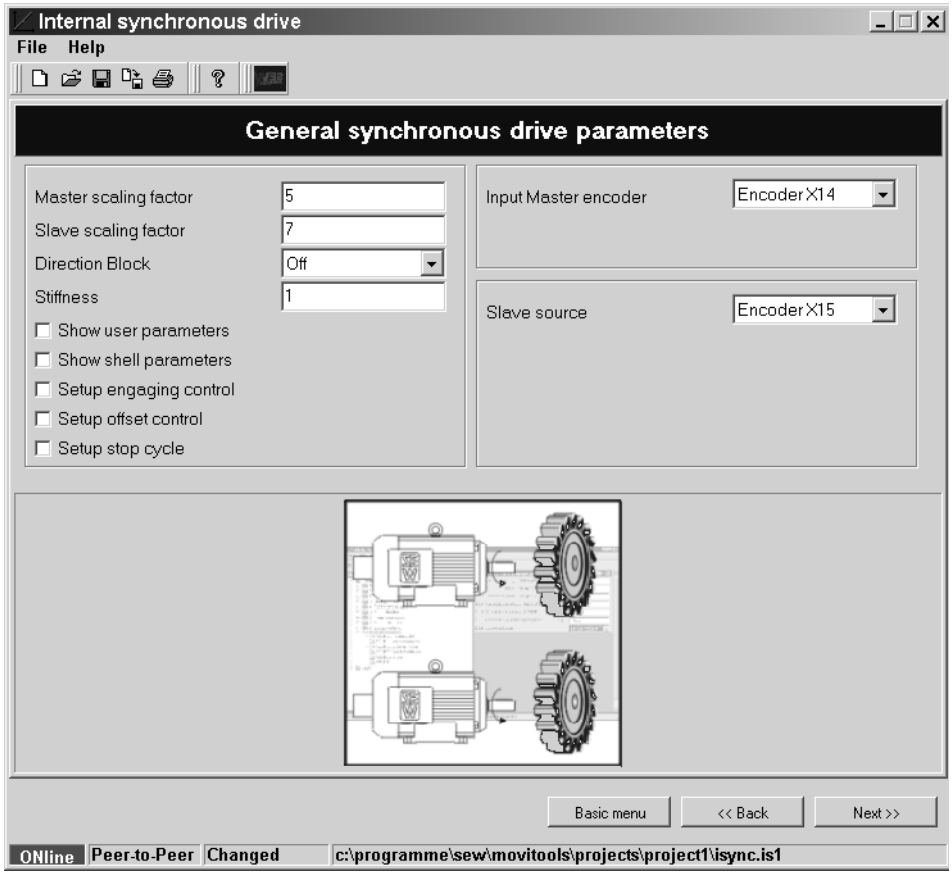
Figure 29: Enter project name and path

Select the project name and path for the new startup. In the [Signature] edit box, you can sign the device with a signature.

Click [Next >>] to continue. The [General synchronous drive parameters] window opens.

If you have selected MOVIDRIVE B as the [Device type:], you can enter the following values in addition to the slave source:

- Variable used
- Distance encoder numerator
- Distance encoder denominator



4046109067

Figure 30: Entering general synchronous drive parameters

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

5.4.1 Master scaling factor

Preliminary remark

The objective of phase-synchronous operation is to move two or more drives at the output – and thus the track – synchronously in relation to one another. The synchronous operation controller required for this purpose only processes the incremental information from a master encoder and a slave encoder. This means the actual gear units and additional gear ratios of the application must be represented by factors to achieve synchronization within a certain proportionality ratio.

If there are two identical drives, that is, with the same gear unit reduction ratios, the same additional gears, etc., the proportionality ratio is 1:1.

- Variable: H428 GFMaster
- Value range:
  - -32768 to 32767 (16-bit – up to firmware version .72)
  - -2147483648 to 2147483648 (32-bit - up to firmware version .10)
- Status: R/W
- Description: Scaling factor of the master increments

The GFMaster scaling factor evaluates the master increments entered as setpoints into the synchronous operation controller.

This GFMaster scaling factor also includes the slave gear unit reduction ratio, the slave encoder resolution, the slave additional gear, and the master distance. As a result, this GFMaster scaling factor is calculated as follows:

GFMaster = ± Slave gear unit reduction ratio × Slave additional gear

By entering a signed GFMaster scaling factor, it is possible for the master and slave to be working with opposite directions of rotation. This reverses the direction of rotation of the slave drive in the software, without swapping over encoder signal tracks.

Slave gear unit reduction ratio i_S

The slave gear unit reduction ratio can generally be obtained from the information on the nameplate of the drive. The value can either be read off directly, or it can easily be calculated by taking the quotient of the rated speed/output speed.

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

In particular for synchronous operation applications in endless mode, we recommend 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:

$$i_S = \frac{z2 \times z4 \times z6 \times \ldots zm}{z1 \times z3 \times z5 \times \ldots zn - 1}$$

n/2 = number of gear stages.

Your SEW contact can tell you the number of teeth in the gear pairs.

Slave additional gear i_{VS}

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

$$i_{VS} = \frac{z2 \times z4 \times z6 \times \ldots zm}{z1 \times z3 \times z5 \times \ldots zm - 1}$$

m/2 = number of additional gear stages.

Slave encoder resolution

In MOVIDRIVE® / MOVIDRIVE® *compact*, the incoming encoder pulses are evaluated fourfold both at terminal X15 of the motor encoder (slave) and at terminal X14 of an external encoder (master).

Slave encoder resolution = encoder nameplate information [data pulses/revolution] × 4

Important:

When connecting a motor encoder to X15 of the encoder card, the PPR count is quadrupled and scaled to 4096 increments/motor revolution for position control. This signal can be routed from the encoder card via the incremental encoder simulation output (X14) to a downstream inverter.

The incremental encoder simulation supplies the PPR count of the selected encoder. When connected to the input of the downstream inverter, the PPR count is quadrupled, **but not scaled to 4096 increments/motor revolution**. If the PPR count (multiplied by 4) deviates from the IPOS motor encoder resolution 4096 increments/motor revolution, you must adjust it via the gear unit factors of the slave inverter.

The following table lists the gear unit factors determined on the basis of the PPR count. If the mechanical design of the master and slave axes is the same, you can adopt the determined gear unit factors directly. Enter these values in the startup window "Parameters for synchronous operation (part 1)" in the "GFMaster" and "GFSlave" entry fields.

Encoder		Gear unit factor	
Type	**Resolution**	**GFMaster**	**GFSlave**
AK0H	128	8	1
AV1Y	512	2	1
ES7R	1024	1	1
AS7W	2048	1	2

Master distance

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

Master distance = Travel length [in user travel units] per output revolution

In many applications, the length of travel is described to a sufficient degree of accuracy by the calculated circumference of the drive wheel:

Master distance = D x Π, where D = diameter of the drive wheel [in user travel units].

However, the number Π = 3.14159… is an irrational number. This means there can be an infinite number of decimal places, and the number always must be rounded up or down. In the case of endless applications, we therefore recommend that you calculate the length of travel according to the mechanical system used, as follows:

Application	Travel length
Chain sprocket as transmission element	Number of teeth of the chain sprocket × chain link length
Gear rack as transmission element	Number of teeth of the gear unit × tooth clearance (tip-to-tip) of the gear rack
Toothed belt as transmission element	Number of teeth of the gear wheel × tooth clearance (center-to-center) of the toothed belt
Spindle as transmission element	Travel length = spindle pitch

If you do not know the mechanical data such as the gear unit reduction ratios, additional gear, etc., the GFMaster scaling factor can also be calculated as described below. However, this method is not recommended for endless applications because it lacks accuracy.

GFMaster = ± Slave increments × Master length [in user travel units]

In order to do this, you must either know or already have determined the travel resolution of the master and slave.

$$Position\ resolution\ master = \frac{Master-\ increments}{Master-\ distance}$$

$$Position\ resolution\ slave = \frac{Slave-\ increments}{Slave-\ distance}$$

Positive/negative GFMaster

By entering a signed GFMaster scaling factor, it is possible for the master and slave to be working with opposite directions of rotation. This reverses the direction of rotation of the slave drive in the software, without swapping over encoder signal tracks.

DRS11A

The following factors are identical in direct comparison to the DRS11A synchronous operation card:

- Slave scaling factor GFSlave = Master gear ratio factor (parameter P221)
- 1/2 Master scaling factor GFMaster 1/2 = Slave gear ratio factor (parameter P222)

5.4.2 Slave scaling factor

The objective of phase-synchronous operation is to move two or more drives at the output – and thus the track – synchronously in relation to one another. The synchronous operation controller required for this purpose only processes the incremental information from a master encoder and a slave encoder. This means the actual gear units and additional gear ratios of the application must be represented by factors to achieve synchronization within a certain proportionality ratio.

If there are two identical drives, that is, with the same gear unit reduction ratios, the same additional gears, etc., the proportionality ratio is 1:1.

- Variable: H429 GFSlave
- Value range:
  - 1 to 32767 (16-bit – up to firmware version .72)
  - 1 to 2147483648 (32-bit – from firmware version .10)
- Status: R/W
- Description: Scaling factor of the slave increments

The GFSlave scaling factor evaluates the slave increments put into the synchronous operation controller as actual values.

This GFSlave scaling factor also includes the master gear unit reduction ratio, the master encoder resolution, the master additional gear and the slave length. As a result, this GFSlave scaling factor is calculated as follows:

GFSlave = ± Master gear unit reduction ratio × Master additional gear

Master gear unit reduction ratio i_M

The master gear unit reduction ratio is generally obtained from the information on the nameplate of the drive. The value can either be read off directly, or it can easily be calculated by taking the quotient of the rated speed/output speed.

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

In particular for synchronous operation applications in endless mode, we recommend 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:

$$i_M = \frac{z2 \times z4 \times z6 \times \ldots zn}{z1 \times z3 \times z5 \times \ldots zn - 1}$$

n/2 = number of gear stages.

Your SEW contact can tell you the number of teeth in the gear pairs.

Master additional gear i_{VM}

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

$$i_{VM} = \frac{z2 \times z4 \times z6 \times \ldots zm}{z1 \times z3 \times z5 \times \ldots zm - 1}$$

m/2 = number of additional gear stages.

Master encoder resolution

In MOVIDRIVE® / MOVIDRIVE® *compact*, the incoming encoder pulses are evaluated fourfold both at terminal X15 of the motor encoder (slave) and at terminal X14 of an external encoder (master).

Master encoder resolution = Encoder nameplate information [pulses/revolution] × 4

Observe the notes in chapter "Slave encoder resolution" (page 53).

Slave distance

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

Slave distance = Travel length [in user travel units] per output revolution

In many applications, the length of travel is described to a sufficient degree of accuracy by the calculated circumference of the drive wheel:

Slave distance = D x Π, where D = diameter of the drive wheel [in user travel units].

However, the number Π = 3.14159... is an irrational number. This means there can be an infinite number of decimal places, and the number must always be rounded up or down. In the case of endless applications, we therefore recommend that you calculate the length of travel according to the mechanical system used, as follows:

Application	Travel length
Chain sprocket as transmission element	Number of teeth of the chain sprocket × chain link length
Gear rack as transmission element	Number of teeth of the gear unit × tooth clearance (tip-to-tip) of the gear rack
Toothed belt as transmission element	Number of teeth of the gear wheel × tooth clearance (center-to-center) of the toothed belt
Spindle as transmission element	Travel length = spindle pitch

The GFSlave scaling factor can also be calculated as described below if you do not know the mechanical data such as the gear unit reduction ratios, additional gear, etc. However, this method is not recommended for endless applications because it lacks accuracy.

GFSlave = Master increments × Slave length [in user travel units].

In order to do this, you must either know or already have determined the travel resolution of the master and slave.

$$Position\ resolution\ master = \frac{Master-\ increments}{Master-\ distance}$$

$$Position\ resolution\ slave = \frac{Slave-\ increments}{Slave-\ distance}$$

DRS11A

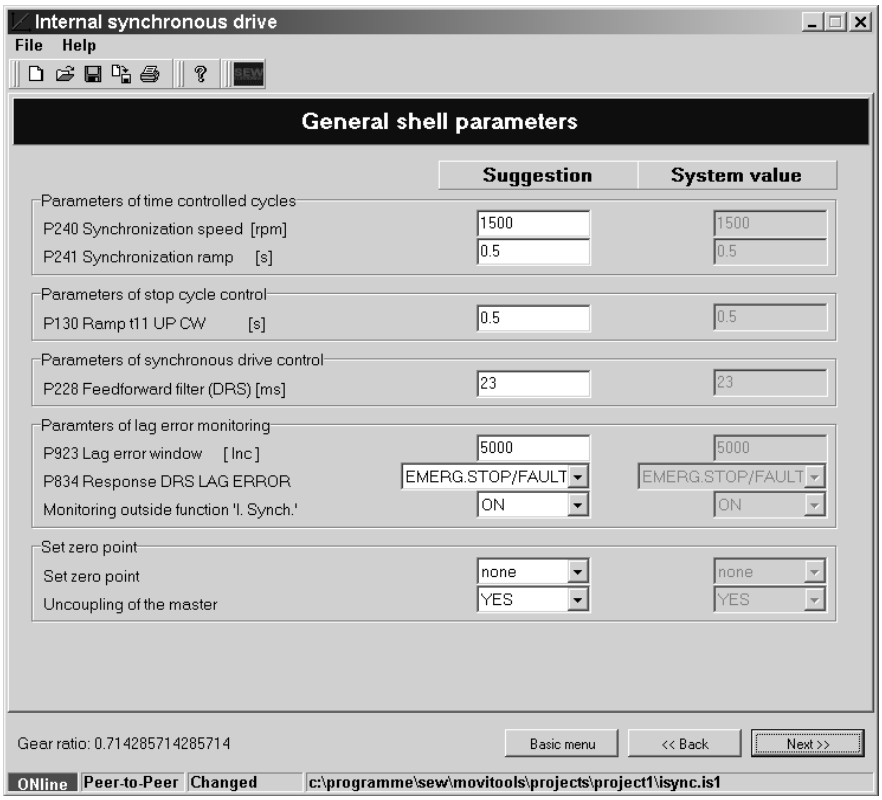
The following factors are identical in direct comparison to the DRS11A synchronous operation card:

- Slave scaling factor GFSlave = Master gear ratio factor (parameter P221)
- 1/2 Master scaling factor GFMaster 1/2 = Slave gear ratio factor (parameter P222)

5.4.3 User parameters

In order to activate additional startup interface dialog boxes to make additional entries, click the following check boxes so that a small check mark appears in them.

- Show user parameters/definition and initialization of IPOS variables
- Show shell parameters
- Start cycle mode control setup
- Offset control setup
- Stop cycle mode control setup



"Uncoupling of the master" drop-down menu: This setting influences the behavior of the slave drive in phase-synchronous operation during "Set zero point", i.e. as long as there is a "1" level at the terminal programmed for this purpose.

"YES": Normally, the slave drive remains stopped during "Set zero point"; i.e. while the difference counter is zeroed.

"NO": Master and slave drives at synchronous speed: During "Set zero point", the slave drive continues moving subject to speed control using the same value and orientation as the master speed, depending on the scaling factors of the master and slave.

5.4.4 Virtual encoder

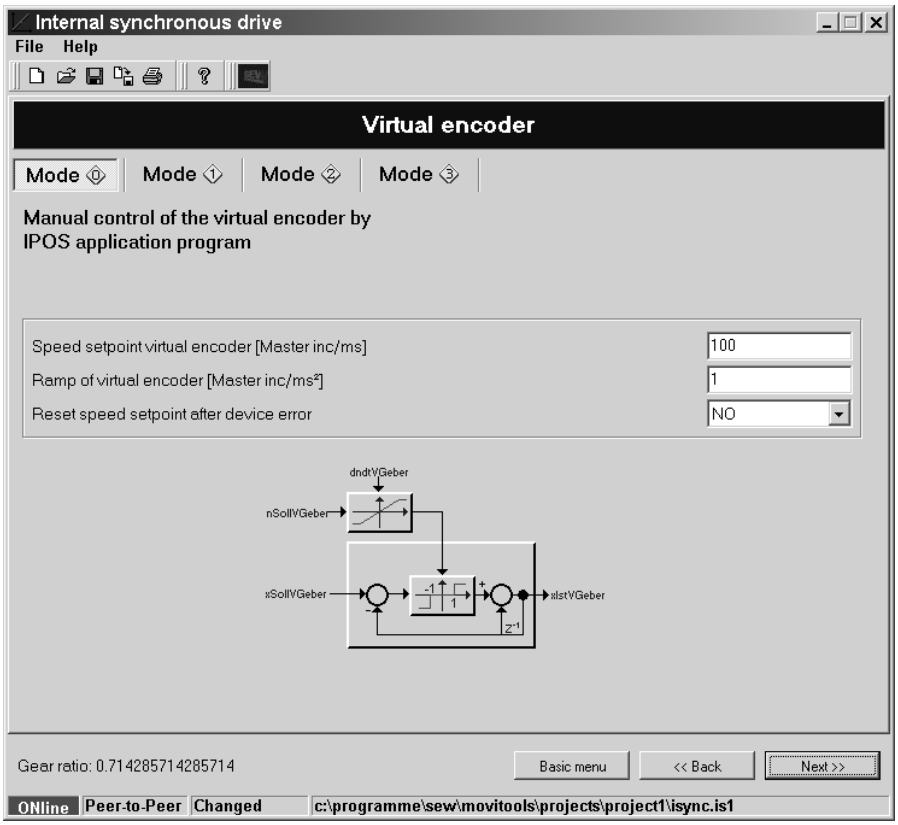
General information on virtual encoders

The virtual encoder (→ IPOS variables H370 –H377) is a software counter that can be used as the master encoder for synchronous operation without using the startup interface (→ assignment MasterSource H430 = H376). Using a system bus connection (SCOM command), you can transfer this software counter from the "generator" of the virtual encoder to other MOVIDRIVE® inverters. To do this, it is necessary to have SBus synchronization with the sync-ID (P817) for unit synchronization (every 5 ms).

The virtual encoder operates in a 1 ms cycle and is processed independently of the present 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). The virtual encoder is stopped (VEncoderMode = 0) when the VEncoderXActual value (H374) reaches the VEncoderXSetpoint value (H375).

Virtual encoder mode 0

In mode 0, the virtual encoder works with linear, adjustable acceleration, adjustable velocity, and target position. The virtual encoder is controlled and monitored using variables H370 to H377:



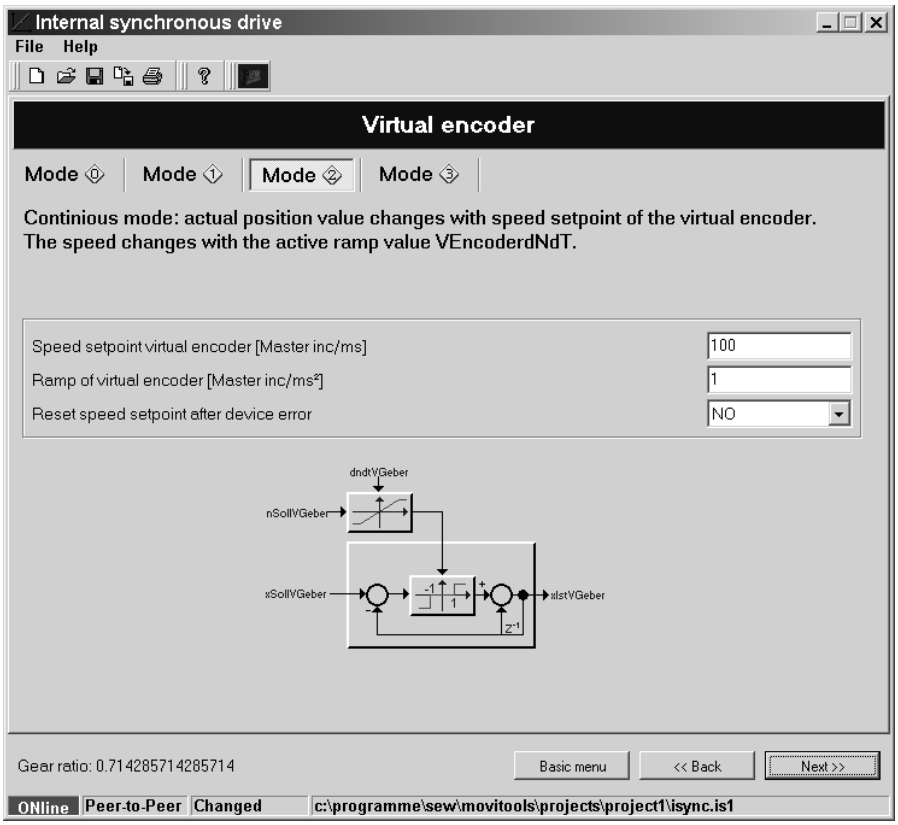
Variable	Meaning
H370 VEncoderMode = 0	Selection of encoder operating mode
H371 VEncoderModeControl	Reset speed setpoint after device error: 1 = Axis stop activated in the event of a unit fault 0 = Axis stop deactivated in the event of a unit fault NO = Velocity of the virtual encoder is not reset YES = Velocity of the virtual encoder is reset once
H372 VEncoderState	No function
H373 VEncoderNSetpoint [1 inc/ms]	Setpoint velocity (setting range: 0,1 – 1000 – 10000)
H374 VEncoderNActual [1 inc/ms]	Actual velocity
H375 VEncoderXSetpoint [1 inc]	Target position
H376 VEncoderXActual [inc]	Actual position
H377 VEncoderdNdT [1 inc/ms^2]	Acceleration (setting range: 0,1 – 10 – 100)

Virtual encoder mode 1

This mode is reserved and cannot be used.

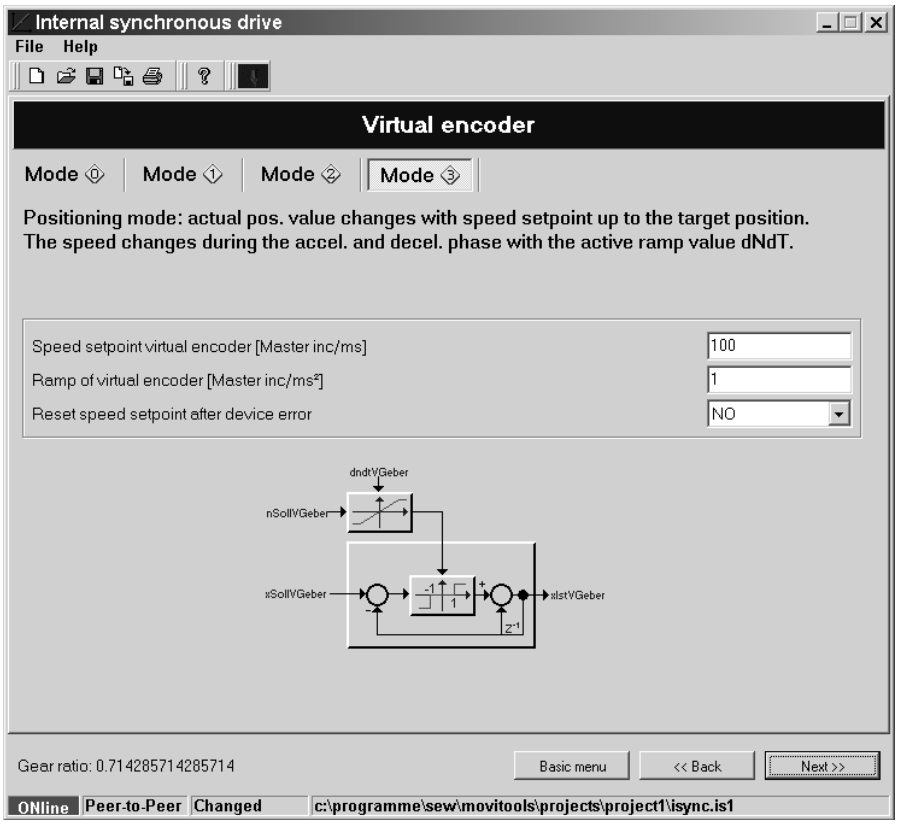
Virtual encoder mode 2

In mode 2, the virtual encoder works as an endless counter with linear, adjustable acceleration and adjustable velocity. The virtual encoder is controlled and monitored using variables H370 to H377:



Variable	Meaning
H370 VEncoderMode = 2	Select the operating mode: Endless counter
H371 VEncoderModeControl	Reset speed setpoint after device error: 1 = Axis stop activated in the event of a unit fault 0 = Axis stop deactivated in the event of a unit fault NO = Velocity of the virtual encoder is not reset YES = Velocity of the virtual encoder is reset once
H372 VEncoderState	No function
H373 VEncoderNSetpoint [1 inc/ms]	Setpoint velocity (setting range: 0,1 – 1000 – 10000)
H374 VEncoderNActual [1 inc/ms]	Actual velocity
H375 VEncoderXSetpoint [1 inc]	No function
H376 VEncoderXActual [inc]	Actual position
H377 VEncoderdNdT [1 inc/ms^2]	Acceleration (setting range: 0,1 – 10 – 100)

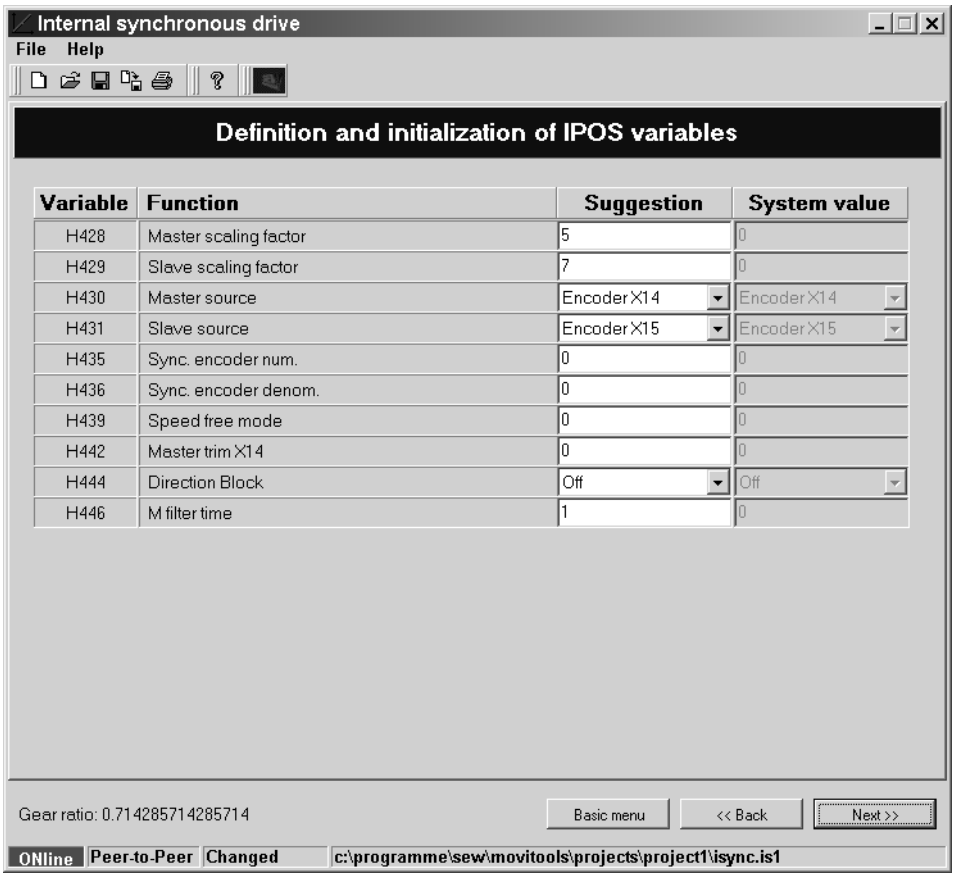
Virtual encoder mode 3



In mode 3, the virtual encoder works with linear, adjustable acceleration and deceleration as well as with adjustable speed and target position. The virtual encoder is controlled and monitored using variables H370 to H377:

Variable	Meaning
H370 VEncoderMode = 3	Select the operating mode: Positioning mode with acceleration and deceleration ramps
H371 VEncoderModeControl	No function
H372 VEncoderState	No function
H373 VEncoderNSetpoint [1 inc/ms]	Setpoint velocity (setting range: 0,1 – 1000 – 10000)
H374 VEncoderNActual [1 inc/ms]	Actual velocity
H375 VEncoderXSetpoint [1 inc]	Target position
H376 VEncoderXActual [inc]	Actual position
H377 VEncoderdNdT [1 inc/ms^2]	Acceleration/deceleration (setting range: 0,1 – 10000 – 32768)

5.4.5 Definition and initialization of IPOS$^{plus®}$ variables



In the "Speed setpoint for the stop cycle state" input field, enter the required setpoint with the resolution [0.2 rpm]. Ensure that the required speed corresponds to the maximum speed (parameter 302/312) because otherwise this speed cannot be achieved.

In the "Master Trim Function X14" input field, enter the number of increments that X14 should be added to every ms.

In the "Backstop" radio button, you can choose one of the following options:

- "Off": Both directions of rotation are enabled
- "Block CW": Only CCW direction of rotation
- "Block CCW": Only CW direction of rotation

In the "M Filter" input field, enter the corresponding value for the calculation of the absolute master scaling factor. The default setting is 1.

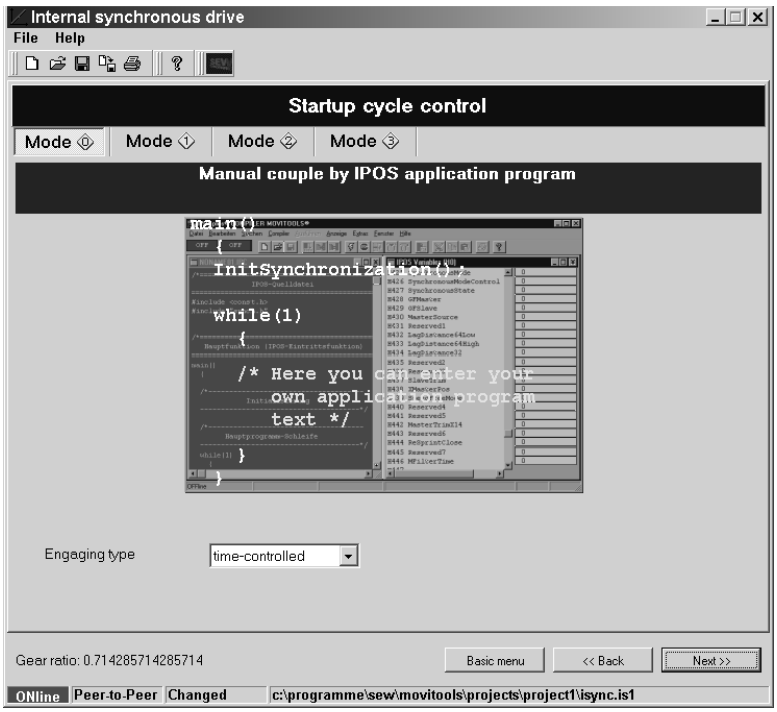
5.4.6 Startup cycle type

Startup cycle type

The ISYNC technology function provides different modes such as synchronizing travel to the master and triggering this synchronization process. For more information, see "Startup cycle type" and "Mode 1" to "Mode 4".

Startup cycle type mode 0

Manual engaging using the IPOS$^{plus®}$ program.



- Variable: H410 StartupCycleMode
- Value range: 0 = Startup cycle via IPOS program
- Status: R/W
- Description: Startup cycle mode

StartupCycleMode = 0

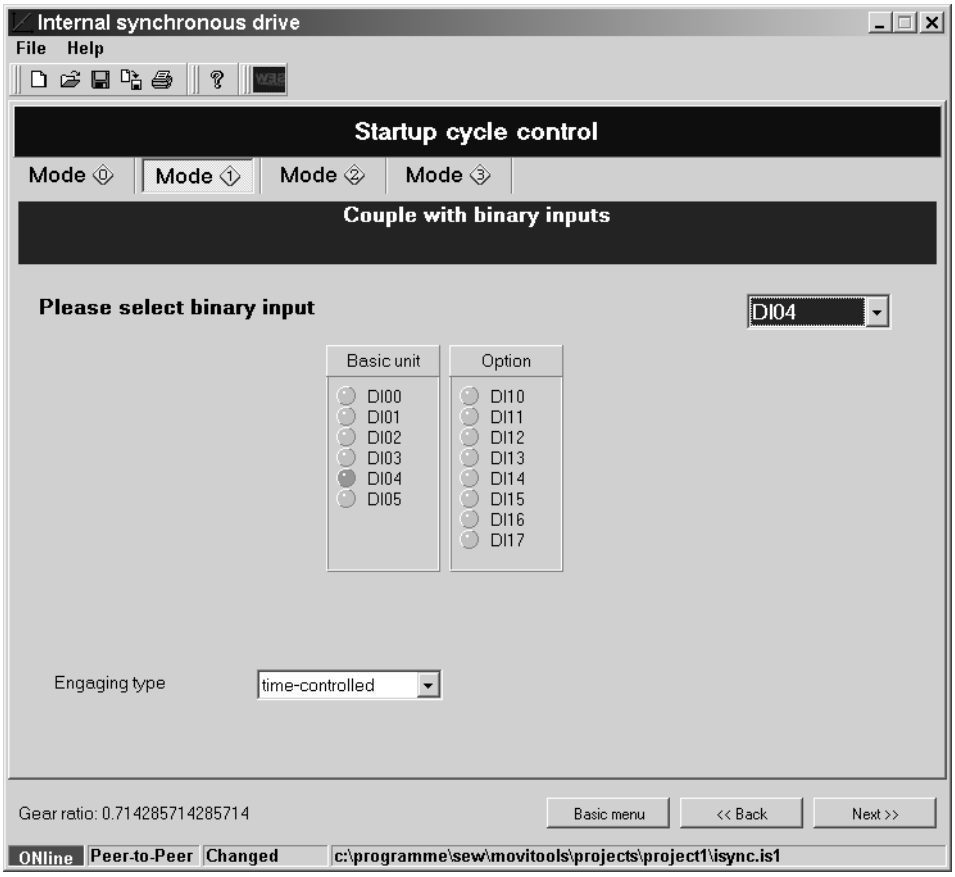
If the startup cycle is performed manually using an IPOS application program, the value 2 must be assigned to the SynchronousState (H427) system variable in the program sequence.

SynchronousState = 2;

The startup cycle process is started immediately afterwards.

Startup cycle type mode 1

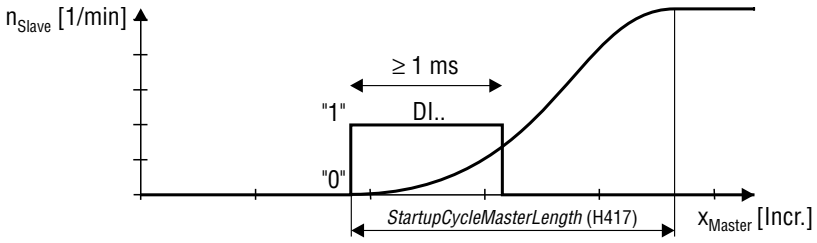
Startup cycle using binary inputs.



StartupCycleMode = 1

If the startup cycle process is started in event-driven mode using a binary input, you must select the terminal in the input field "Please select binary input" that triggers the startup cycle process (e.g. DI04). Alternatively, you can enter the terminal designation in the corresponding drop-down menu (DIxx). When doing this, you can use the input terminals of the basic unit (DI00 – DI05) or of the option pcb (DI10 – DI17). If you do not use the startup interface, the required binary input can be selected via the H403 StartupCycleInputMask 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.

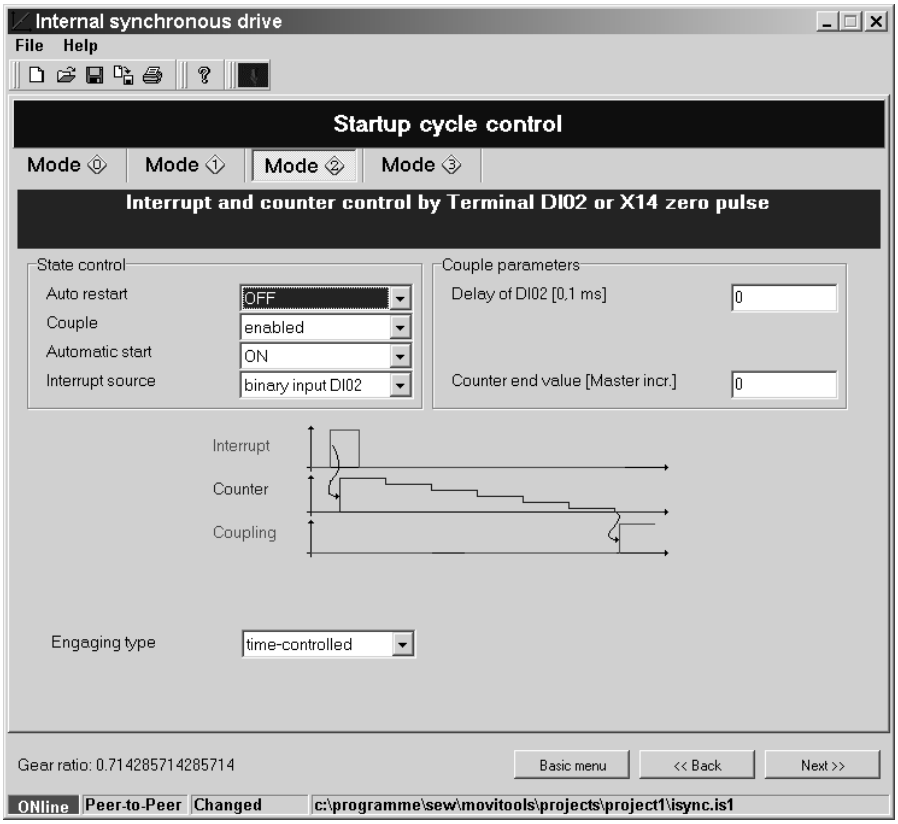


INFORMATION

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

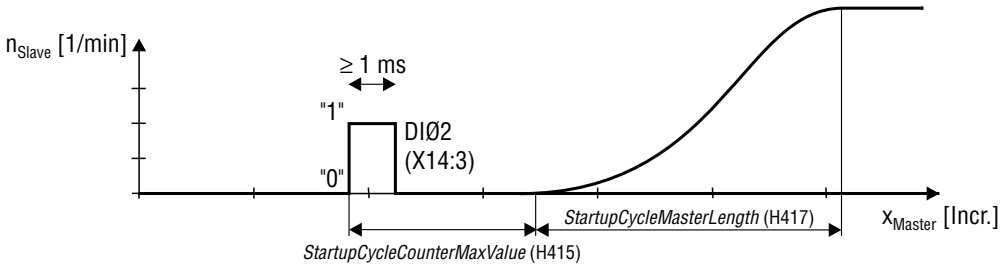
Startup cycle type mode 2

Interrupt and position control with DI02 or C track (X14).



StartupCycleMode = 2

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



Interrupt source

- Variable: H411 StartupCycleModeControl
  - 0 = DI02
  - 1 = X14 C track
- Status: R/W
- Description: Bit 2: InterruptSelect (in mode 2)

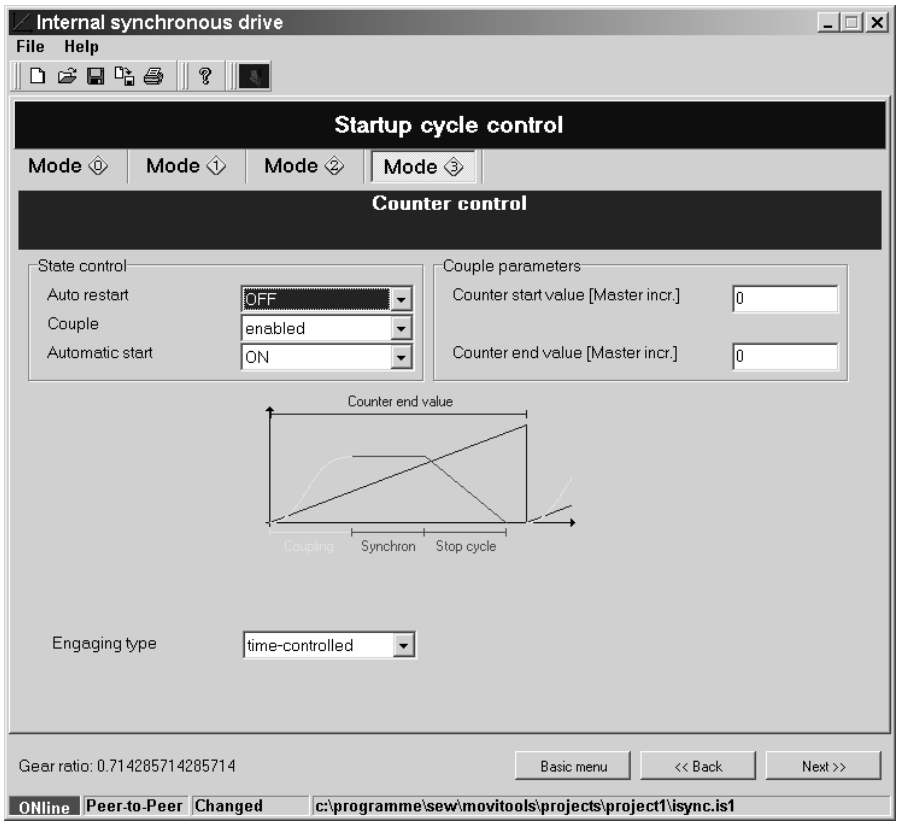
The interrupt-controlled start of the startup cycle process is triggered either by a rising edge (level change - "0" > "1") at binary input DI02; e.g., caused by a sensor signal or a rising edge (level change "0" > "1") of the X14:3 track. When using the rising edge at the binary input DI02, set the parameter of this input to "No function".

Please make the appropriate selection in the "Interrupt source" input field:

- "Binary input DI02": A rising edge (level change "0" -> "1") at binary input DI02 triggers the startup cycle process.
- "C track X14:3": A rising edge (level change "0" -> "1") of the C track X14:3 triggers the startup cycle process.

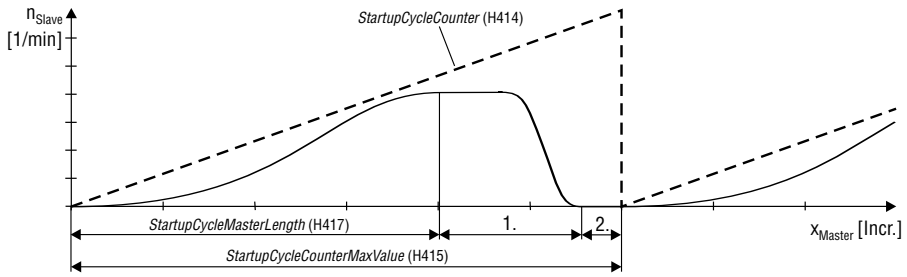
Startup cycle type mode 3

Position control.



StartupCycleMode = 3

The startup cycle process is initiated by the StartupCycleCounter position counter (H414). If the StartupCycleCounter value is greater than the StartupCycleCounter-MaxValue (H415) counter overrun value, the startup cycle takes place automatically. 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.



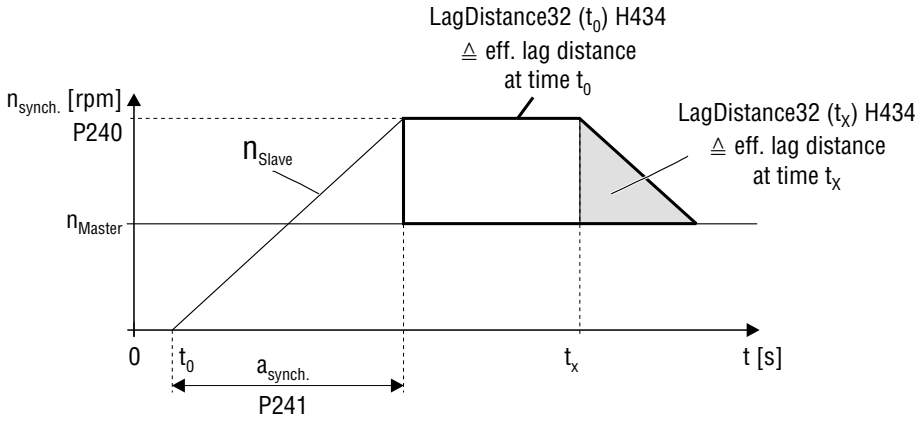
1. Synchronous operation and stop cycle
2. The slave is disengaged, time for repositioning to the initial position

Startup cycle type

In the "Startup cycle 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 startup cycle modes:

Time-controlled synchronizing

Time-controlled synchronization means the existing position differential between the master and slave drive (64-bit counter) is equalized by accelerating or decelerating to the synchronization speed. The time required depends on the catch-up speed, the catch-up 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.



- Variable: H411 StartupCycleModeControl
  - 0 = Time-controlled sychronizing
  - 1 = Position-dependent synchronizing
- Status: R/W
- Description: Bit 12: StartupMode

In this edit box, select whether the slave drive should be synchronized in a time-controlled or position-dependent manner.

Startup cycle or synchronizing means that the slave drive reaches the synchronization point specified by the master. In the following actual synchronous operation (main state Z3), the control adopts this synchronization point and attempts to maintain it.

During the time-controlled startup cycle, the existing or resulting angle differential between the master and slave drive (64-bit counter) is eliminated by accelerating or decelerating to the synchronization speed (P240). The time needed for this depends on the synchronization speed (P240), the synchronization ramp (P241) and the lag distance (H434, LagDistance32).

The lag distance depends on the master speed value so that the master speed also influences the time-controlled startup cycle.

Position-dependent engaging

With this synchronizing mechanism, the slave drive moves in sync with the master drive once the master drive has covered the specified distance.

The specified distance is entered in the "Master distance" edit field in master increments. If the startup interface is not used -> System variable StartupCycleMasterLength H417. The restriction is that the slave drive starts with speed zero.

The accelerating ramp is automatically calculated by the internal synchronous operation firmware depending on the specified distance.

The slave drive reaches the synchronization point after the specified travel length irrespective of time. This means the position-dependent startup cycle is independent of the master speed and produces "smooth" synchronization.

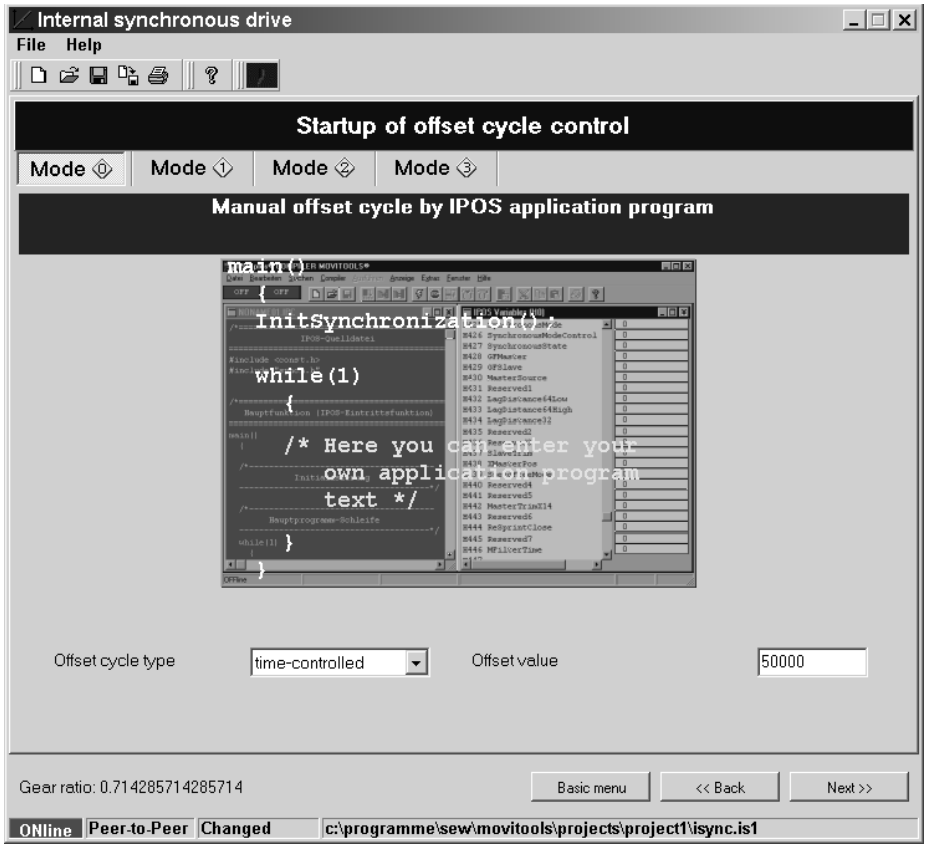
Master distance startup cycle

- Variable: H417 StartupCycleMasterLength
- Value range: 0 – 7FFFFFFFhex master incr.
- Status: R/W
- Description: Specified distance for the master drive for position-dependent startup cycle.

The master length is the distance within which the slave synchronizes with the master. The reference for this type of startup cycle is the length of travel covered by the master drive. It is entered in master increments.

5.4.7 Offset control

Offset control mode 0



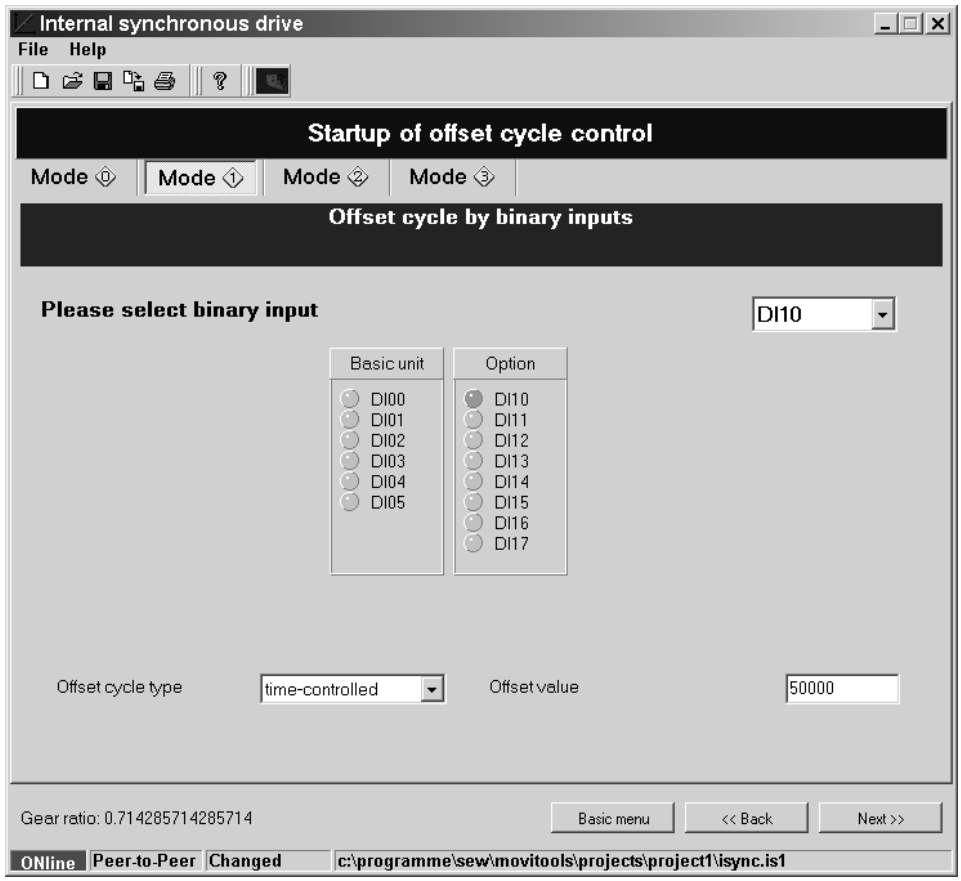
H360 OffsetCycleMode = 0 = Offset via IPOS program

If the offset cycle is initiated manually using an IPOS application program, the value 4 must be assigned to the SynchronousState system variable (H427) in the program sequence.

The offset cycle is started immediately afterwards.

Offset control mode 1

H360 OffsetCycleMode = 1 = Offset via input terminals.



If the offset cycle is started using event-control via a binary input, you must enter the terminal in the "Select binary input" edit box that triggers the offset cycle. If the startup interface is not used -> System variable OffsetCycleInputMask H363. The process is started as soon as a "1" level is present at the defined binary input. The terminal latency is 1 ms.

Which binary input should be selected?

Select the required binary input: the offset cycle will be started when there is a rising edge at this input (level change: "0" -> "1"). To do this, either directly click the dot to the left of the corresponding terminal name or enter the terminal name "DIxx" directly in the text box provided for this purpose. When doing this, you can use the input terminals of the basic unit (DI00 – DI05) or of the option pcb (DI10 – DI17).

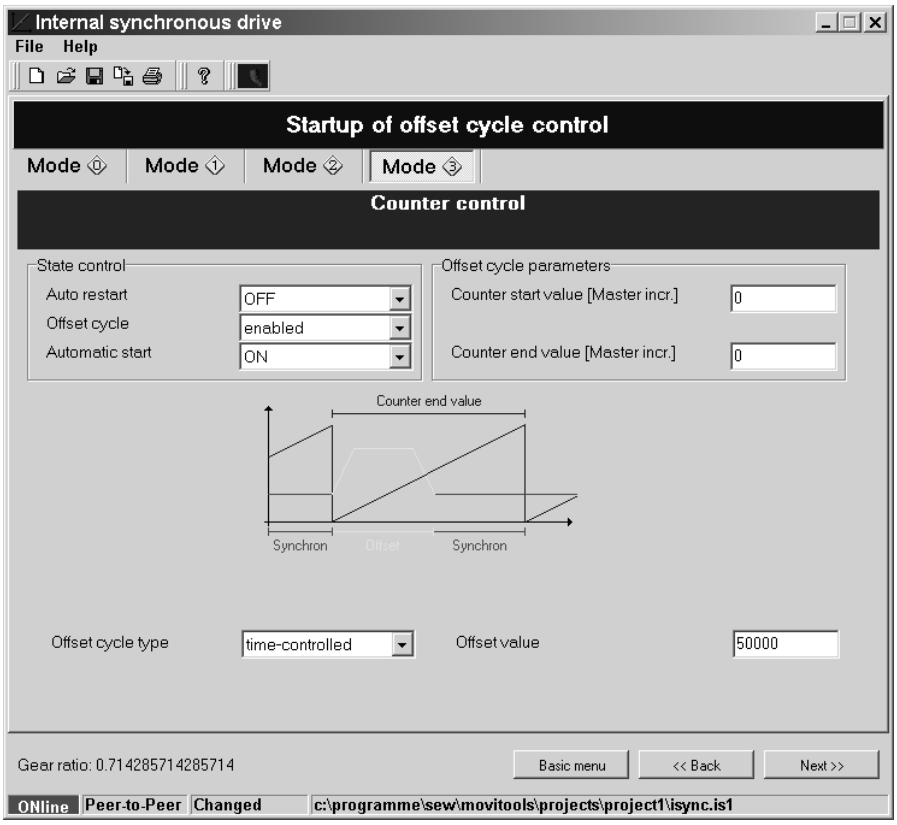
To avoid conflicting terminal function assignments, the parameter of the selected input terminal must be set to "NO FUNCTION" or "IPOS INPUT" in the corresponding parameter group 60x or 61x.

Offset control mode 2

H360 OffsetCycleMode = 2 = reserved

This mode of the offset cycle is currently reserved and cannot be used.

Offset control mode 3



H360 OffsetCycleMode = 3 = Offset via position control

With internal synchronous operation, the offset cycle can be started automatically. In this case, the start signal for the offset cycle is generated internally by the overflow of the offset counter that reached the specified master distance. This can be performed cyclically.

The offset cycle is initiated by the master increments counter (OffsetCycleCounter H364). The offset is processed automatically if the value of the offset counter exceeds that of the maximum counter value. If the startup interface is not used -> System variable OffsetCycleCounterMaxValue H365.

In the position-dependent offset cycle, the counter maximum value (OffsetCycle-CounterMaxValue H365) must be greater than the master distance within which the offset must be eliminated and the new synchronization point is 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 cyclically executed (offset machine) and can have varying offset states (OZ0 – OZ1).

The offset cycle is deactivated in offset state OZ0 (OffsetCycleState H362 = 0).

The value 1 must be assigned to the offset state (OffsetCycleState H362 = 1) in order to start running through the cycle and to enable offset counter monitoring. If the offset counter (OffsetCycleCounter H364) is greater than the counter maximum value (Offset-CycleCounterMaxValue H365), the offset is processed and the offset counter is reset. The offset state changes to OZ0 immediately afterwards, or remains in OZ1.

AutoRestart offset

- Variable: H361 OffsetCycleModeControl
  - 0 = AutoRestart deactivated
  - 1 = AutoRestart activated
- Status: R/W
- Description: Bit 0: AutoRestart (in mode 3)

The "AutoRestart" function allows you to determine whether the offset cycle is to be triggered "only" once or several times. Either a single or a continuous enable is set for running through the offset cycle. Make the appropriate selection in the "AutoRestart" edit box:

- "OFF": When AutoRestart is deactivated, the offset cycle can be run through exactly once using position control.
- "ON": When AutoRestart is activated, the offset cycle can be restarted using position control after having been run through first.

Offset cycle

- Variable: H361 OffsetCycleModeControl
  - 0 = Offset cycle possible
  - 1 = Offset cycle disabled
- Status: R/W
- Description: Bit 1 OffsetDisable (in mode 3)

It is possible to set a general disable on the offset cycle using position control. "General" means during the initialization phase of the IPOS program; a disable which has been set can again be revoked at any time in the main program. Make the appropriate selection in the "Offset cycle" edit box:

- "Activated": General enable for the offset cycle using position control.
- "Deactivated": General disable for the offset cycle using position control.

Automatic start offset

The value 1 must be assigned to the offset state (OffsetCycleState H362 = 1) in order to trigger the offset cycle using position control. Make the appropriate selection in the "Automatic start" edit box:

- "ON": The change from offset state OZ0 to OZ1 takes place automatically after the program starts.
- "OFF": The change from offset state OZ0 to OZ1 must be programmed manually in the IPOS application program.

Counter start value offset

- Variable: H364 OffsetCycleCounter
- Value range: 0 – 7FFFFFFFhex master incr.
- Status: R/W
- Description: Master counter for offset cycle

The "Counter start value" input field enables the offset counter to be set to any initial value from which it begins to count the master increments. The reference for this input value is the travel length covered by the master drive. It is entered in master increments. Enter the required start value for the offset counter (master incr.) in the "Counter start value" edit box.

Maximum counter value offset

- Variable: H365 OffsetCycleCounterMaxValue
- Value range: 0 – 7FFFFFFFhex master incr.
- Status: R/W
- Description: In mode 3: Length limit for the automatic offset cycle

The "Counter maximum value" edit box makes it possible to set the master distance at which the offset counter overruns and the start of the offset cycle is triggered internally. It is also the length of the cycle after which successive automatic offset cycles can take place. The reference for this input value is the travel length covered by the master drive. It is entered in master increments. Enter the required end value for the offset counter (master incr.) in the "Counter maximum value" input field.

Offset process

- Variable: H361 OffsetCycleModeControl
  - 0 = Time-controlled offset cycle
  - 1 = Position-dependent offset cycle
- Status: R/W
- Description: Bit 12: OffsetMode

In this edit box, select whether the offset cycle should be time-controlled or position-dependent. Offset cycle means that an offset is added to the difference counter (main state Z3) during synchronous operation. In this way, the slave drive obtains a new synchronization point and the resulting angle difference is reduced to zero by the control. Actual synchronous operation (main state Z3) is reestablished once the new synchronization point has been reached. The offset value can be signed (+/–). Main state 3 (synchronous operation) is a prerequisite for offset cycle processing.

Time-controlled offset cycle

In this state, an offset is added to the difference counter. If the startup interface is not used -> System variable OffsetCycleValue H367. The angle differential is reduced to zero by accelerating or decelerating to the synchronization speed (P240), and the slave drive moves through an offset. The time needed for this depends on the synchronization speed (P240), the synchronization ramp (P241) and the master speed. (see the "Startup cycle mode control" subsection, additional notes -> time-controlled synchronizing.)

The offset value must be entered in the edit box of the same name. If the startup interface is not used -> System variable OffsetCycleValue H367.

Position-dependent offset cycle

In this state, the slave drive is offset after the master drive has covered a specified distance. The specified distance is entered in master increments in the "Master distance" edit box. If the startup interface is not used -> System variable OffsetCycle-MasterLength H366.

The acceleration ramp is automatically calculated by the internal synchronous operation firmware depending on the specified distance.

The slave drive reaches the new synchronization point after the specified travel length – irrespective of time. This means the position-dependent offset cycle is independent of the master speed and results in "smooth" processing of the offset as well as a "smooth" startup cycle to the new synchronization point.

The offset value must be entered in the edit box of the same name. If the startup interface is not used -> System variable OffsetCycleValue H367.

Offset value

- Variable: H367 OffsetCycleValue
- Value range: 0 – 7FFFFFFFhex slave incr.
- Status: R/W
- Description: Offset value for slave drive

Position-depending offset cycle: Specified distance for the master drive in offset processing.

OffsetCycleSlaveLength is the distance that is traveled within the offset cycle. The reference for this input value is the travel length covered by the slave drive. It is entered in slave increments. Enter the required slave travel length (slave increments) here.

5.4.8 Stop cycle mode control

Stop cycle mode control

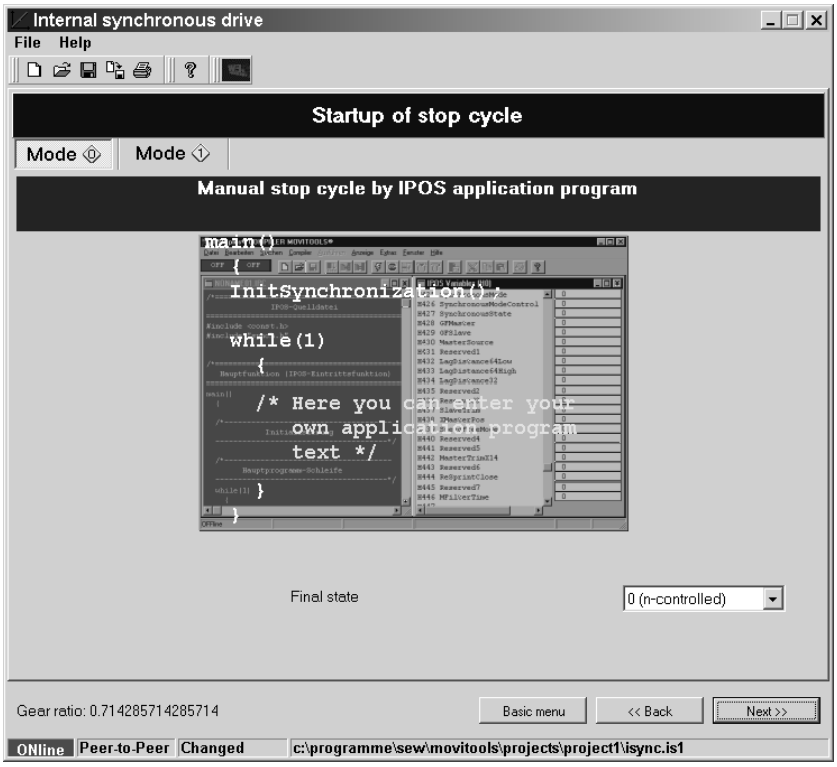
You can put stop cycle mode control into operation using this dialog box.

To do this, click the appropriate button in the top part of the interface to select the required stop cycle mode.

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

During the stop cycle, the drive changes to speed 0 along ramp t11 (P130) with x-control; alternatively, with n-control, the drive changes to the speed defined in the SpeedFreeMode system variable (H439).

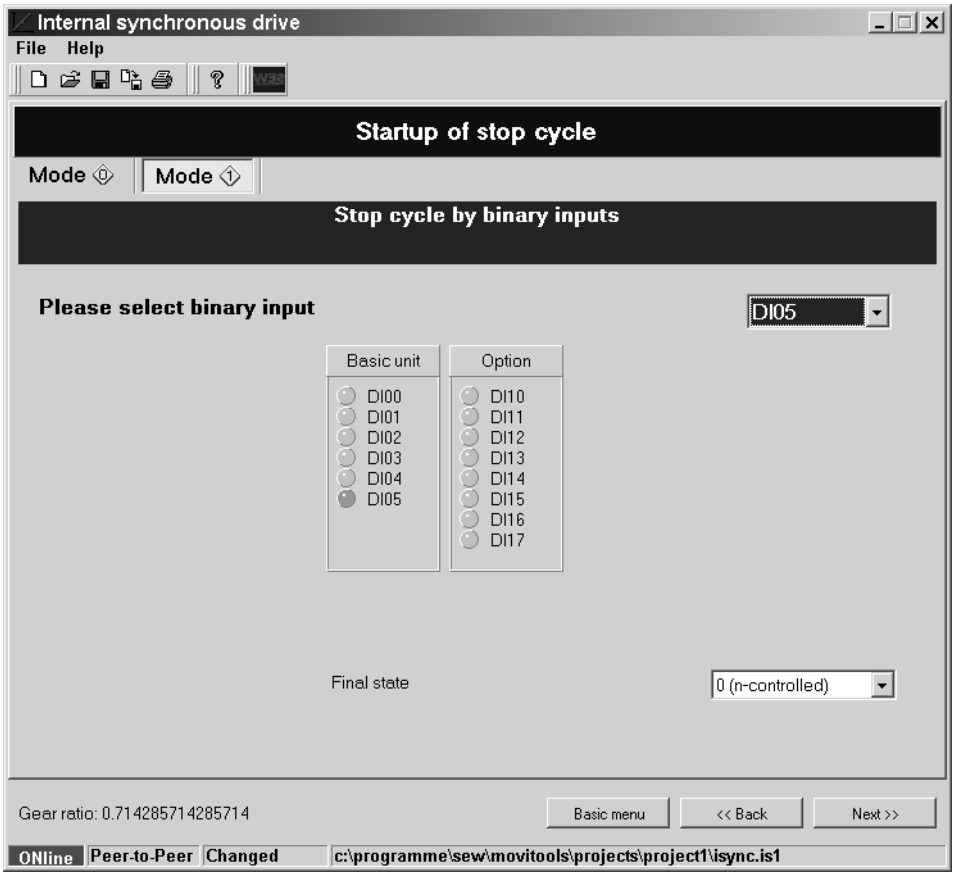
Mode 0 Manual stop cycle by IPOS$^{plus®}$ program



StopCycleMode = 0

The slave ceases synchronous operation with the master when the application assigns the value 5 to the SynchronousState system variable (H427).

Mode 1: Stop cycle via binary inputs



StopCycleMode = 1

Event-driven stop cycle via binary input. The StopCycleInputMask variable (H403) defines which binary input triggers the stop cycle process. The process is started as soon as level "1" is present at the defined binary input. The terminal latency is 1 ms.

Which binary input should be selected?

Select the required binary input. The stop cycle will be started when there is a rising edge at this input (level change "0" -> "1"). To do this, either directly click the dot to the left of the corresponding terminal name or enter the terminal name "DIxx" directly in the text box provided for this purpose. When doing this, you can use the input terminals of the basic unit (DI00 – DI05) or of the option pcb (DI10 – DI17).

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

State after stop cycle

- Variable: H401 StopCycleModeControl
  - 0 = Stop cycle in main state 0 (n-controlled)
  - 1 = Stop cycle in main state 1 (x-controlled)
- Status: R/W
- Description: Bit 0: FreeMode

Internal synchronous operation enables you to distinguish between two free-running states.

The slave drive can either be moved with speed control using a specified setpoint (SpeedFree Mode H439) or held at its current position with position control.

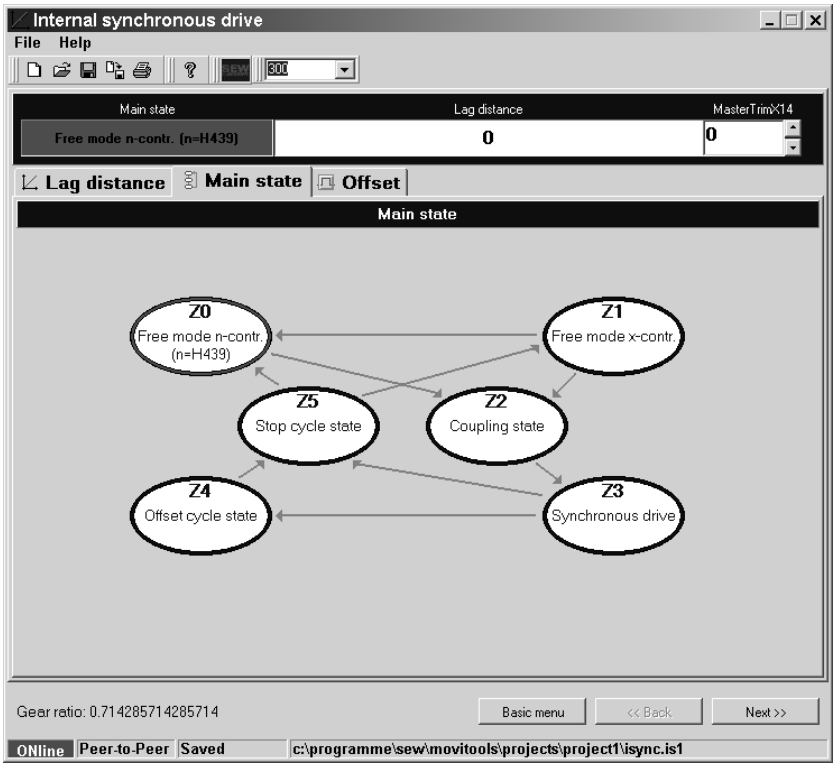
In free running n-control, a 64-bit difference counter stores the delay.

You can use this edit box to select which of these two free-running states the slave drive should adopt after the stop cycle. Make the corresponding selection in the "Final state" edit box:

- "0 (n-control)": Stop cycle into main state 0 -> Free-running n-control
- "1 (x-control)": Stop cycle into main state 1 -> Free-running x-control

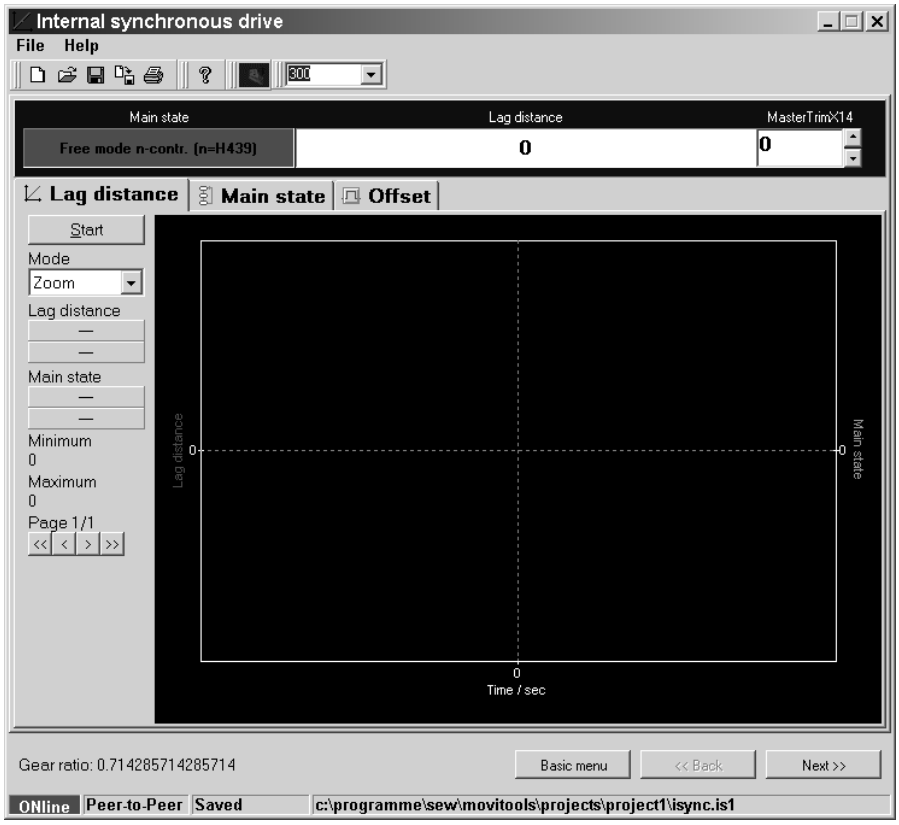
5.4.9 Online monitor

Online monitor main state



You can use the online monitor to observe the current main state of the synchronous operation application. To exit the program, click [Next]. The basic menu opens. In the basic menu, click [Exit program].

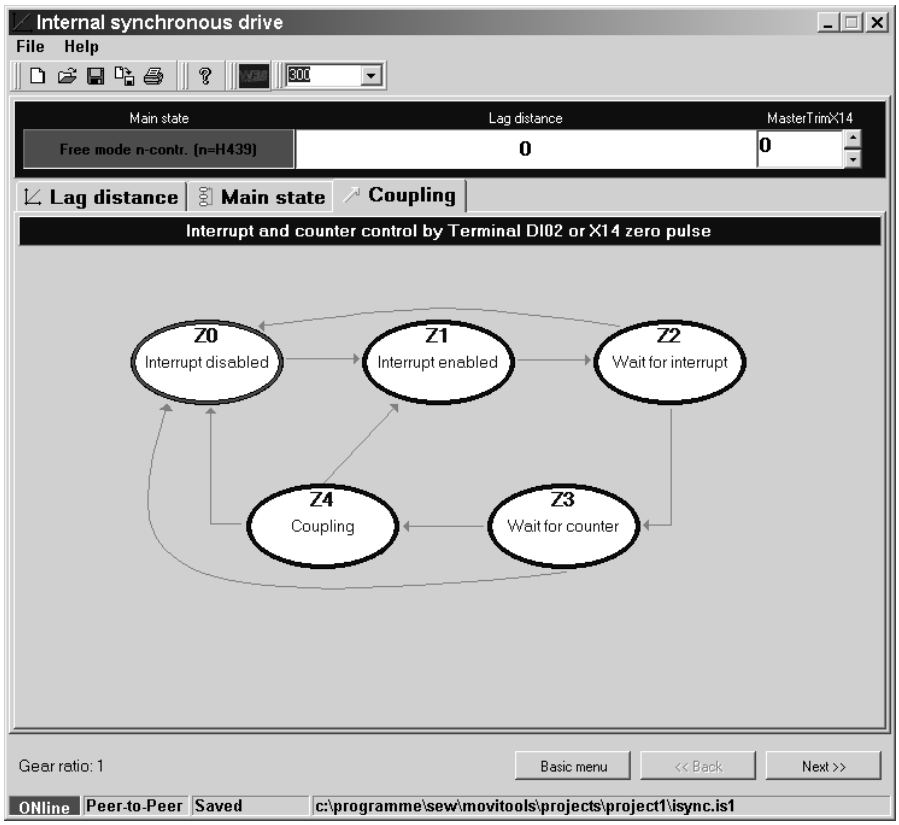
Online monitor lag distance



In the "Lag distance" menu, the current lag distance of the active application can be recorded and analyzed more closely using the "Zoom" function. To do so, switch between "Measure" and "Zoom" for the required function using the pulldown menu on the left side of the screen. The scope functionality for the lag distance is particularly helpful for system optimization and fault diagnostics to obtain information on the course of the lag error. The lag error is also displayed in digital form at the top of the screen.

The current main state of the synchronous operation application and the value of the IPOS variable MasterTrimX14 is also displayed at the top of the screen.

Online monitor startup cycle (only in startup cycle mode 2)

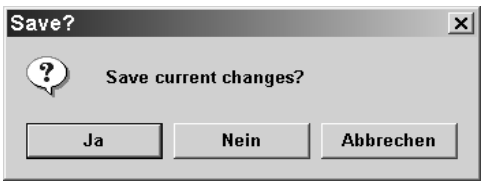


In the "Startup cycle" menu, the current state of the startup cycle machine is displayed. Here, you can see which startup cycle event the application is waiting for.

5.4.10 Save/download/compile

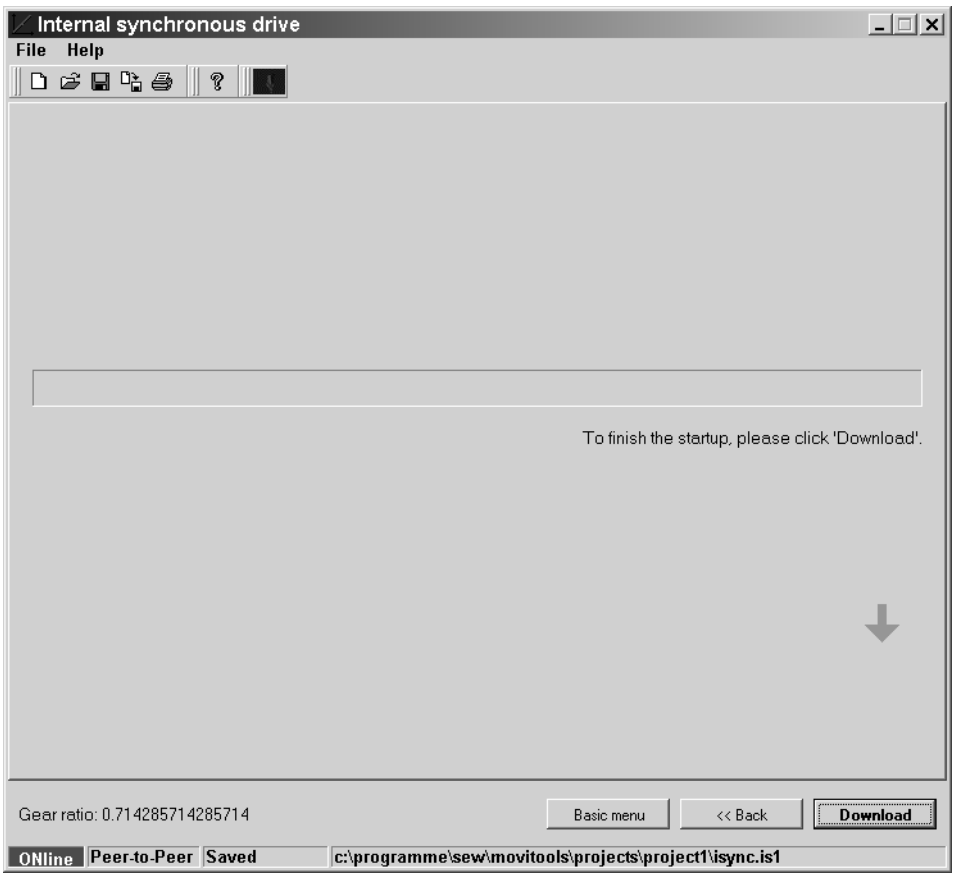
Save

Click "Yes" to save the data to a file (*.is1) using 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.



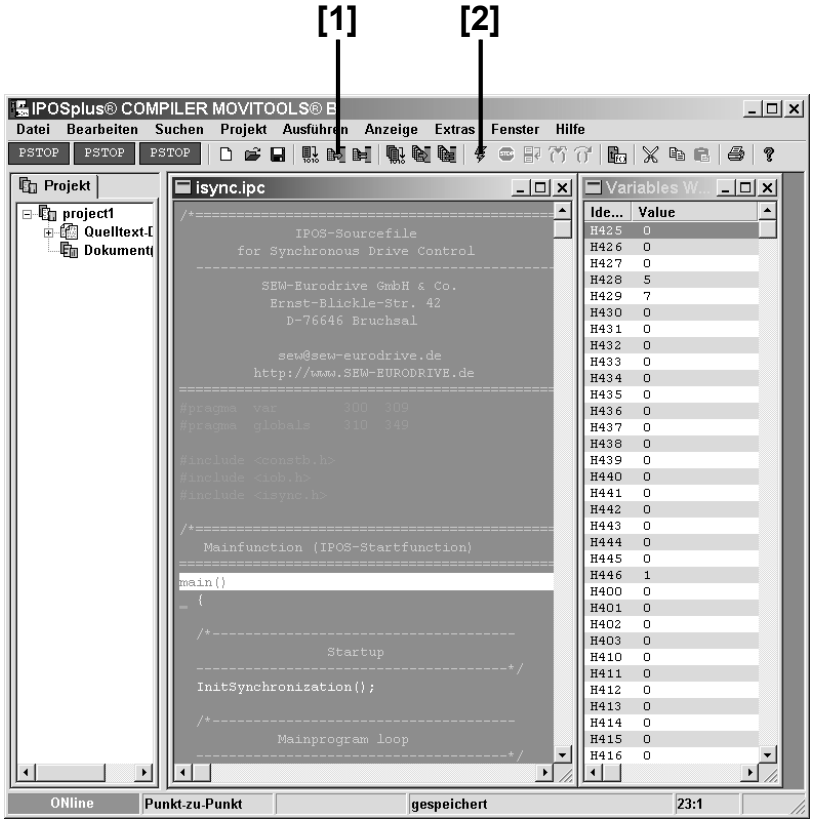
Download

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



Compiler

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".



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 `Projektname.h` header file which is integrated in the header file.

You can now create further user-specific program sections and insert them into the existing text. This 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 startup interface's online monitor for internal synchronous operation. The compiler interface is still in the background and can be closed.

5.4.11 Parameters of internal synchronous operation

Stiffness setpoint

- Value range: 0.1 – 1 – 2

The stiffness value influences how "hard" the synchronous operation control loop is. Basically, this refers to the speed with which the synchronous operation controller responds to compensate for differentials at its input.

The user must optimize this value depending on the situation on site. If the lag error limit is reached, the value is too small or the control loop too soft. If the slave drive vibrates, jerks and an audible buzzing noise is heard, the value is too large or the control loop too hard.

The factory setting of 1 is recommended as the initial value.

Set zero point (terminal programming)

Here, you can assign the "DRS SET ZERO PT." function to any programmable binary input. This sets the difference counter to zero and deletes any possible angular misalignment:

- "1" signal = Counter is set to zero
- "1" -> 0 new reference point for synchronous operation

This function is needed during startup if the master and the slave need to be calibrated with one another.

P130 Ramp t11 to the right [s]

- Parameter: P130 Ramp t11 to the right [s]
- Value range: 0 – 0.5 – 2000 s

This ramp is used during the stop cycle.

The ramp times refer to a setpoint step change of Δn = 3000 rpm. The ramp takes effect when the speed setpoint is changed and the enable is revoked via the CW/CCW terminal.

Monitoring when not in the "Internal synchronous operation" function (parameters for lag error monitoring)

- "On": If internal synchronous operation is deactivated during the "Synchronous operation" mode (Z3) by changing the ramp type P916, it is possible to continue lag error monitoring for internal synchronous operation.
- "Off": However, to avoid lag errors, e.g. during positioning with IPOS$^{plus®}$, it is advisable either to switch off monitoring when not in the "Internal synchronous operation" function, or to switch to the stop cycle state (Z5) before deactivating internal synchronous operation.

P228 Precontrol filter

- Parameter: P228 Precontrol filter
- Value range: 0 – 23 – 100 s

Setpoint filter for synchronous operation precontrol.

P240 Synchr. speed [rpm] (parameter for time-controlled process)

- Parameter: P240 Synchronization speed [rpm]
- Value range: 0 – 1500 – 5500 rpm

This parameter indicates the duration of the synchronization procedure. Make sure that the synchronization speed (catch-up speed) is faster than the maximum value for the master speed during operation.

The synchronization speed is limited by the maximum speed (P302).

P241 Synchr. ramp [s] (Parameter for time-controlled operation)

- Parameter: P241 Synchronization ramp [s]
- Value range: 0 – 0.5 – 50 s

Value of the acceleration ramp for synchronizing the slave with the master. The value 0 should be entered if the slave is to synchronize with the master using maximum possible acceleration.

P834 LAG ERROR response

- Parameter: P834 LAG ERROR response
- Value range: Programmable responses (see MOVIDRIVE® system manual, chapter "Parameters"). Factory setting: EMERG.STOP/FAULT

P834 programs the fault response which is triggered by the lag error monitoring of synchronous operation and of positioning mode with IPOS$^{plus®}$.

P923 Lag error window

- Parameter: P923 Lag error window
- Value range: 0 – 5000 – 7FFFFFFFhex inc.

The lag error window defines a permitted difference between the setpoint and actual position value. A lag error message or response is triggered if the limit is exceeded. This response can be adjusted using P834 "Lag error response".

Deactivation: Lag error monitoring is deactivated when the value is set to 0.

6 System Variables for Internal Synchronous Operation

Variable	Name and value range	Status	Description
Offset control			
H360	OffsetCyleMode 0 to 3	R/W	The OffsetCycleMode variable is used to set how an offset cycle is to be started. The following offset modes can be selected: Offset mode = 0: Offset via IPOS$^{plus®}$ program = 1: Offset via input terminals = 2: Reserved = 3: Offset via position control
H361	OffsetCycleModeControl	R/W	Activates different functions
			Bit 0: AutoRestart (in mode 3) The "AutoRestart" function allows you to determine whether the offset cycle is to be triggered "only" once or several times. = 0: AutoRestart deactivated When AutoRestart is deactivated, the offset processing cycle can be run through exactly once using counter control. = 1: AutoRestart activated When AutoRestart is activated, the offset cycle can be restarted using position control after having been run through first.
			Bit 1: OffsetDisable (in mode 3) It is possible to set a general inhibit for the offset cycle using counter control. "General" means during the initialization phase of the IPOS program; a disable which has been set can again be revoked at any time in the main program. = 0: Offset cycle possible = 1: Offset cycle disabled
			Bit 12: OffsetMode = 0: Time-controlled offset cycle In this state, an offset is added to the difference counter. The angle differential is reduced to zero by accelerating or decelerating to P240 synchronization speed, and the slave drive moves through an offset. The time needed for this depends on the synchronization speed (P240), the synchronization ramp (P241) and the master speed. = 1: Position-dependent offset cycle In this state, the slave drive is offset after the master drive has covered a specified distance. The specified distance is entered in the "Master length" input field in master increments. The acceleration ramp is automatically calculated by the internal synchronous operation firmware depending on the specified distance. The slave drive reaches the new synchronization point after the specified travel length irrespective of time. This means the position-dependent offset cycle is independent of the master speed and produces "smooth" processing of the offset as well as a "smooth" startup cycle to the new synchronization point.
H362	OffsetCycleState Max. 0 to 1 (depending on OffsetCycleMode)	R/W	Controlling the various modes Offset control can have different states depending on the OffsetCycleMode H360 selected: =0: Offset cycle is deactivated in offset state 0 (with position control). =1: Monitoring of the offset counter is enabled (with position control). The value 1 must be assigned to the offset state (OffsetCycleState H362 = 1) in order to start running through the cycle and to enable offset counter monitoring.

Variable	Name and value range	Status	Description
H363	OffsetCycleInputMask	R/W	Selects the binary input with which the offset cycle will be started (OffsetCycleMode 1) when there is a rising edge at this input (level change "0" to "1"). Terminal mask (identical to H483 "InputLevel") Bit Binary input Unit 0 DI00 Basic unit (with fixed assignment "Controller inhibit") 1 DI01 Basic unit 2 DI02 Basic unit 3 DI03 Basic unit 4 DI04 Basic unit 5 DI05 Basic unit 6 DI10 Option DIO11B 7 DI11 Option DIO11B 8 DI12 Option DIO11B 9 DI13 Option DIO11B 10 DI14 Option DIO11B 11 DI15 Option DIO11B 12 DI16 Option DIO11B 13 DI17 Option DIO11B To avoid conflicting terminal function assignments, the parameter of the selected input terminal must be set to "No function" or "IPOS input" in the corresponding parameter groups 60x or 61x.
H364	OffsetCycleCounter 0 to 7FFFFFFFhex	R/W	Master counter for offset cycle The reference for this input value is the travel length covered by the master drive. It is entered in master increments. Enter the required start value for the offset counter (master increments) here.
H365	OffsetCycleCounterMaxValue 0 to 7FFFFFFFhex	R/W	In mode 3: Length limit for the automatic offset cycle OffsetCycleCounterMaxValue can be used to set the master length at which the offset counter overruns and the start of the offset cycle is triggered internally. It is also the length of the cycle after which successive automatic offset cycles can take place. The reference for this input value is the travel length covered by the master drive. It is entered in master increments. Enter the required end value for the offset counter (master increments) here.
H366	OffsetCycleMasterLength 0 to 7FFFFFFFhex	R/W	Position-depending offset cycle: Specified distance for the master drive during offset cycle. OffsetCycleMasterLength is the distance that is traveled within the offset cycle. The reference for this input value is the travel length covered by the master drive. It is entered in master increments. Enter the required master travel length (master increments) here.
H367	OffsetCycleValue 0 to 7FFFFFFFhex	R/W	Position-depending offset cycle: Specified distance for the master drive in offset processing. OffsetCycleValue is the distance that is traveled within the offset cycle. The reference for this input value is the travel length covered by the slave drive. It is entered in slave increments. Enter the required slave travel length (slave increments) here.
Virtual encoder			
H370	VEncoderMode 0 to 3	R/W	Virtual encoder operating mode = 0: In mode 0, the virtual encoder works with linear, adjustable acceleration, adjustable velocity, and target position. = 1: Reserved = 2: Endless counter with linear, adjustable acceleration and speed. = 3: Standard mode (positioning mode): In mode 3, the virtual encoder works with linear, adjustable acceleration and deceleration as well as with adjustable velocity and target position.
H371	VEncoderModeControl	R/W	**Bit 0**: AxisStop = 0: Deactivated (Speed of the virtual encoder is not reset) = 1: Axis stop in the event of a unit fault (Speed of the virtual encoder is reset once) When a unit fault occurs, the speed of the virtual encoder is set once to zero (VEncoderNSetpoint H373 = 0). This setting stops the virtual axis.

Variable	Name and value range	Status	Description
H372	VEncoderState	R/W	No function
H373	VEncoderNSetpoint 0 to 32767	R/W	Setpoint speed of the virtual encoder in 1 incr./ms
H374	VEncoderNActual	R/W	Actual speed of the virtual encoder in 1 incr./ms
H375	VEncoderXSetpoint	R/W	Target position of the virtual encoder in incr.
H376	VEncoderXActual	R/W	Current position of the virtual encoder in incr.
H377	VEncoderNdT	R/W	Acceleration ramp of the virtual encoder The resolution is: Mode 0 + 2: Ramp in [1 incr./ms^2] Mode 3: Ramp in [1/2^{16} incr./ms^2] In mode 3, the specified value is scaled by 2^{16} via the startup interface. On the other hand, if the variable VEncderdNdT is written by the programmer in the IPOS$^{plus®}$ program, scaling by 2^{16} must be taken into account in mode 3 (see chapter "Virtual encoder"). NOTE: To avoid jerks, the variable VEncoderdNdT in mode 3 must be set at least to the value 65536 (65536 = 1 incr./ms).
Control element			
H389	RegisterLoopOut	R/W	Value to be reduced in connection with RegisterLoopDXDTOut To make the start cycle even more accurate, a correction value in [incr.] can be added to the difference counter via RegisterLoopOut. However, this correction value is only added once by the firmware, i.e. RegisterLoopOut H389 is overwritten automatically with zero once the correction has been made. You can use bit 2 from SynchronousModeControl to set whether this correction value should be scaled using the slave scaling factor.
H390	RegisterLoopDXDTOut −30000 to 30000	R/W	Control element limitation with resolution [incr./ms] Max. addition (64-bit counter) per ms RegisterLoopDXDTOut is used to specify how a possible correction value output through RegisterLoopOut is to be added to the difference counter. That is, the correction value is not added at once, but in x [incr./ms]. Example: A correction value of 2000 incr. is specified via RegisterLoopOut. RegisterLoopDXDTOut is set to 50 → 50 incr./ms are added to the difference counter. This means that the correction value is processed in 40 ms.

Variable	Name and value range	Status	Description
Stop cycle mode control			
H400	StopCycleMode 0 to 1	R/W	Stop cycle mode This variable is used to specify how the stop cycle mode functions: = 0: Stop cycle via IPOS$^{plus®}$ program = 1: Stop cycle via input terminals
H401	StopCycleModeControl	R/W	Activates different functions **Bit 0**: FreeMode = 0: Stop cycle in main state 0 (n-control) → speed control = 1: Stop cycle in main state 1 (x-control) → position control **Bit 1**: x-control mode When switching to x-control after the stop cycle, you can select whether the difference counter should be zeroed, i.e. the reference between master and slave is then lost. = 0: Difference counter is zeroed The reference between master and slave is lost. = 1: Difference counter active, delay between master and slave is maintained. Difference counter is not deleted, i.e. reference between master and slave is maintained.
H402	StopCycleState		No function
H403	StopCycleInputMask	R/W	Selects the binary input with which the stop cycle will be initiated (StopCycle-Mode 1) when there is a rising edge at this input (level change "0" → "1"). Terminal window (identical to H483 "InputLevel") in mode 1 only Bit Binary input Unit 0 DI00 Basic unit (with fixed assignment "Controller inhibit") 1 DI01 Basic unit 2 DI02 Basic unit 3 DI03 Basic unit 4 DI04 Basic unit 5 DI05 Basic unit 6 DI10 Option DIO11B 7 DI11 Option DIO11B 8 DI12 Option DIO11B 9 DI13 Option DIO11B 10 DI14 Option DIO11B 11 DI15 Option DIO11B 12 DI16 Option DIO11B 13 DI17 Option DIO11B To avoid conflicting terminal function assignments, the parameter of the selected input terminal must be set to "No function" or "IPOS input" in the corresponding parameter groups 60x or 61x.
Startup cycle mode control			
H410	StartupCycleMode 0 to 3	R/W	Startup cycle mode This variable is used to specify how the startup cycle mode functions: = 0: Start cycle via IPOS$^{plus®}$ program = 1: Startup cycle via input terminals = 2: Startup cycle using interrupt control = 3: Startup cycle via position control

Variable	Name and value range	Status	Description
H411	StartupCycleModeControl	R/W	Activates different functions
			Bit 0: Auto restart (in modes 2 and 3) The "AutoRestart" function allows you to determine whether the startup cycle is to be triggered "only" once or several times. Either a single or a continuous enable is set for running through the startup cycle process. = 0: AutoRestart deactivated When AutoRestart is deactivated, the startup cycle process can be run through exactly once by interrupt control or position control. = 1: AutoRestart activated When AutoRestart is activated, the startup cycle process can be restarted by interrupt control or position control after having been run through first.
			Bit 1: StartupDisable (in modes 2 and 3) = 0: Startup cycle possible = 1: Startup cycle disabled It is possible to set a general disable for the startup cycle using interrupt control or position control. "General" means during the initialization phase of the IPOS program; a disable which has been set can again be revoked at any time in the main program.
			Bit 2: InterruptSelect (in mode 2) The interrupt-controlled start of the startup cycle process is triggered either by a rising edge (level change - "0" -> "1") at binary input DI02; e.g., caused by a sensor signal or a rising edge (level change "0" -> "1") of the X14:3 track. When using the rising edge at the binary input DI02, set the parameter of this input to "No function". = 0: DI02 A rising edge (level change "0" -> "1") at binary input DI02 triggers the startup cycle process. = 1: X14C track A rising edge (level change "0" -> "1") of the C track X14:3 triggers the startup cycle process.
			Bit 3: StartupOffset (only with time-based synchronizing) = 0: No offset during startup cycle = 1: The offset from variable H367 (OffsetCycleValue) is added to the difference counter and is run through in the startup cycle
			Bit 4: StartupSearchMode (only with time-based synchronizing) If the slave drive moves faster than the master during the startup cycle (for example because of a high velocity setpoint of SpeedFreeMode), then you can use StartupSearchMode to specify whether the slave moves first at the speed of the master before it accelerates to possibly maximum speed depending on the angle difference. = 0: Slave moves first at master speed = 1: The slave accelerates to the maximum speed depending on the angle differential before it moves at master speed.
			Bit 12: StartupMode In StartupMode, you can choose between two startup cycle types: = 0: Time-controlled synchronizing The existing or arising angle differential between the master and slave drive (64-bit counter) is eliminated by accelerating or decelerating to the synchronization speed (P240). The time needed for this depends on the synchronization speed (P240), the synchronization ramp (P241) and the lag distance (H434, LagDistance32). = 1: Position-dependent synchronizing With this synchronizing mechanism, the slave drive moves in sync with the master drive once the master drive has covered the specified distance. The specified distance is entered in master increments in the "Master distance" edit box. The restriction is that the slave drive starts with speed zero. The acceleration ramp is automatically calculated by the internal synchronous operation firmware depending on the specified distance. The slave drive reaches the synchronization point after the specified travel length – regardless of the time. This means the position-dependent startup cycle is independent of the master speed and produces "smooth" synchronization.

Variable	Name and value range	Status	Description
H412	StartupCycleState Max. 0 to 4 (depending on mode)	R/W	Controlling the various modes The value 1 must be assigned to the startup cycle state (StartupCycleState H412 = 1) in order to initiate running through the startup cycle process using interrupt control or position control. = 0: Interrupt deactivated (the change from startup cycle state EZ0 to EZ1 must be programmed manually in the IPOS application program). = 1: Interrupt activated (the change from startup cycle state EZ0 to EZ1 takes place automatically after the program starts). = 2: The drive waits for the interrupt signal. = 3: Delay, i.e. the interrupt signal was detected. StartupCycleCounterMax-Value is active. = 4: Startup cycle and resetting the startup counter. **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 *StartupCycleState*, then the counter is reset and the reference to the master is lost.
H413	StartupCycleInputMask	R/W	Selects the binary input with which the start cycle will be initiated (StopCycle-Mode 1) when there is a rising edge at this input (level change "0" -> "1"). Terminal window (identical to H483 *InputLevel*) in mode 1 only Bit Binary input Unit 0 DI00 Basic unit (with fixed assignment "Controller inhibit") 1 DI01 Basic unit 2 DI02 Basic unit 3 DI03 Basic unit 4 DI04 Basic unit 5 DI05 Basic unit 6 DI10 Option DIO11B 7 DI11 Option DIO11B 8 DI12 Option DIO11B 9 DI13 Option DIO11B 10 DI14 Option DIO11B 11 DI15 Option DIO11B 12 DI16 Option DIO11B 13 DI17 Option DIO11B To avoid conflicting terminal function assignments, the parameter of the selected input terminal must be set to "No function" or "IPOS input" in the corresponding parameter groups 60x or 61x.
H414	StartupCycleCounter 0 – 7FFFFFFFhex master incr.	R/W	Master counter for the startup cycle The startup cycle is initiated when the value of *StartupCycleCounter* (H414) is greater than the value of *StartupCycleCounterMaxValue* (H415). The "Counter start value" input field enables the startup cycle counter to be set to any initial value from which it begins to count the master increments. The reference for this input value is the travel length covered by the master drive. It is entered in master increments.
H415	StartupCycleCounterMaxValue 0 – 7FFFFFFFhex master incr.	R/W	In mode 2: Delay for startup cycle process The "Counter end value" edit box lets you delay the interrupt-controlled start of the startup cycle process. The reference for this input value is the travel length covered by the master drive. It is entered in master increments. In mode 3: Length limit for automatic startup cycle The "Counter end value" edit box makes it possible to set the master distance at which the startup cycle counter overruns and the start of the startup cycle process is triggered internally. It is also the length of the cycle after which a successive automatic startup cycle can take place. The reference for this input value is the travel length covered by the master drive. It is entered in master increments.
H416	StartupCycleDelayDI02 −32768 to 32767	R/W	Delay in units of 0.1 ms Delay time of the sensor connected to touch probe input 2. StartupCycleDelay can be used to take into account a delay from DI02 for the interrupt-controlled start of the startup cycle process. The delay of the sensor connected at DI02 is entered in [0.1 ms].

Variable	Name and value range	Status	Description
H417	StartupCycleMasterLength	R/W	Specified distance for the master drive for position-dependent startup cycle. The slave has synchronized to the master within this distance. The master length is the specified distance for the master drive in the position-dependent startup cycle, within which the slave synchronizes with the master. The master distance is entered here in [1 incr.].

Variable	Name and value range	Status	Description
General variables			
H425	Synchronous mode		No function
H426	SynchronousModeControl	R/W	Activates different functions
			Bit 0: PosTrim (only active in main state Z1 "x-control") = 0: The drive remains in the current position subject to position control. = 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. During the stop cycle into main state Z1 "x-control", the system stores the current position as the setpoint for internal position control when the speed ZERO is reached. If the "Slave trim in free-running mode" function is deactivated, the setpoint is overwritten by the value of TargetPos H492 and position control is based on this setpoint. This function is not suitable for positioning tasks because movement takes place to the TargetPos H492 at maximum speed and with the minimum ramp (zero ramp) when the function is deactivated. Make the appropriate selection in the "Slave trim in free-running mode" edit box: OFF: PosTrim active, i.e. the position setpoint corresponds to the current position when speed ZERO is reached. ON: PosTrim deactivated, i.e. position setpoint is the TargetPos H492.
			Bit 1: LagError (in state 3 → Synchronous operation) = 0: Lag error monitoring If internal synchronous operation is deactivated during the "Synchronous operation" operating mode (Z3) by changing the ramp type P916, it is possible to continue lag error monitoring for internal synchronous operation. = 1: No lag error monitoring However, to avoid lag errors, e.g. during positioning with IPOS$^{plus®}$, it is advisable either to switch off monitoring when not in the "Internal synchronous operation" function or to switch to the stop cycle state (Z5) before deactivating internal synchronous operation. **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.
			Bit 2: RegisterScale = 0: Increments to be corrected are not scaled (1:1). = 1: Increments are scaled with slave gear ratio factor (*GFSlave*). RegisterScale can be used to select whether the correction value set in H389 RegisterLoopOut is to be scaled with the slave gear ratio factor or not (see also H389 RegisterLoopOut).
			Bit 3: Zero PointMode (uncoupling of the master) This setting influences the behavior of the slave drive in angular-synchronous operation during "Set zero point", i.e. as long as there is a 1 level at the terminal programmed for this. = 0: Precontrol / master is uncoupled The slave drive remains at a standstill during Set zero point, i.e. while the difference counter is zeroed. = 1: Precontrol / master is maintained (speed synchronization), i.e. the slave continues to move at the same speed as the master. During "Set zero point", the slave drive continues moving subject to speed control using the same value and orientation as the master speed, depending on the scaling factors of the master and slave.

Variable	Name and value range	Status	Description
H426	SynchronousModeControl	R/W	**Bit 4:** State Change Bit 4 can be used to disable automatic state change of variable H427 SynchronousState. Automatic state change is carried out during startup and stop cycles or offset processing when controller inhibit is active or the operating mode is changed. In this case, the state is automatically changed to "SychronousState == 3" (hard synchronous operation). By disabling automatic state change, you can stop an interrupted startup or stop cycle or offset processing after renewed enable. However, with distance-related engaging or position-dependent offset processing, it is important that the positions between master and slave are synchronous at restart, else a jerk will occur. =0: State Change enabled Automatic change to state 3 (hard synchronous operation) is enabled after activated controller inhibit or change of operating mode during a startup cycle, or after offset processing in a start or stop cycle, or offset processing, and is performed by the firmware. =1: State Change disabled Automatic change to state 3 (hard sychronous operation) is disabled after activated controller inhibit or change of operating mode during a startup cycle, or offset processing during startup or stop cycle or offset cycle and is therefore not performed by the firmware.
H427	SynchronousState 0 to 5	R/W	Main machine integrated synchronous operation = 0: Free-running mode n-control = 1: Free-running mode x-control = 2: Start cycle = 3: Synchronous operation = 4: Offset cycle = 5: Stop cycle The main state machine distinguishes between the above mentioned states (Z0 to Z5) that are stored in the IPOS$^{plus®}$ variable H427 SynchronousState. The state can be changed by writing the SynchronousState in the IPOS$^{plus®}$ program. Ensure however that the correct transitions in accordance with the main state machine are observed (see also chapter "Main state machine").
H428	GFMaster −2 000 000 000 to 2 000 000 000	R/W	Scaling factor of the master increments, value = i_{slave}. The GFMaster scaling factor evaluates the master increments entered as setpoints into the synchronous operation controller. This GFMaster scaling factor also includes the slave gear unit reduction ratio, the slave encoder resolution, the slave additional gear and the master length (see also the "Important Notes" chapter).
H429	GFSlave 1 to 2 000 000 000	R/W	Scaling factor of the slave increments, value = i_{master}. The GFSlave scaling factor evaluates the slave increments entered into the synchronous operation controller as actual values. This GFSlave scaling factor includes the master gear unit reduction ratio, the master encoder resolution, the master additional gear and the slave length (see also the "Important Notes" chapter).
H430	MasterSource 0 to 1023	R/W	Source of the master increments = 0: X14 + virtual axis (H442) > 0: Pointer to variable The source of the master increments must be specified for the setpoint of the synchronous operation controller. The following sources for the master increments are available: • "EncoderX14": External encoder on terminal X14 • "SSIEncoder": Absolute encoder on terminal X62 (DIP11A) • "Variable": IPOS variable as source of master increments • "Virtual encoder": Virtual encoder If you select "Variable", a list box called "Variables used" appears. In this box, you must select the variable number of the IPOS variable where the master increments are stored. If you select "Virtual encoder", you must make additional entries in an extra dialog box "Virtual encoder mode selection". It is important that the velocity of the master does not exceed the value 32767 / MFilterTime inc/ms, e.g. MFilterTime = 5 → max. velocity = 6553 inc/ms

Variable	Name and value range	Status	Description
H431	SlaveSource 0 to 1023	R/W	Actual position source = 0: X15 > 0: Pointer to variable Example: H431 = 510 // Actual position source X14 (H510 *ActPos_Ext*) Specify the source of the slave increments for the actual value of the slave axis. Note: If you select "X14" or "Variable" you must also enter the ratio of the motor encoder [incr./mm] to the synchronous encoder [incr./mm] (see also H435 and H436). The following sources for slave increments are available in the "Slave source" selection field: • "Encoder X15: Motor encoder on terminal X15 • "Encoder X14: External encoder on terminal X14 (IPOS variable H510 ActPos_Ext) • "SSIEncoder: Absolute encoder on terminal X62 with DIP11B (IPOS variable H509 ActPos_Abs) • "Variable": IPOS variable as source of master increments If you select a variable as the source for the source of the slave information, you must enter the number of the variable here (H1 – H1023). It is important that the velocity of the slave does not exceed the value 32767 / MFilterTime inc/ms.
H432	LagDistance64Low	R/-	Lower 32 bits of the 64-bit counter
H433	LagDistance64High	R/-	Upper 32 bits of the 64-bit counter
H434	LagDistance32	R/-	32 bit lag distance in relation to GFSlave
H435	SyncEncoderNum 0 to 10000	R/W	Distance encoder factor numerator (slave motor encoder revolution [incr./mm]) = 0: Distance encoder calculation deactivated If synchronous operation is required from a drive that is subject to slip, the distance encoder function must be activated. To do so, the ratio between the motor encoder and distance encoder must be entered as the numerator/denominator factor.
H436	SyncEncoderDenom 0 to 10000	R/W	Distance encoder factor denominator (slave distance encoder revolution [incr./mm]) = 0: Distance encoder calculation deactivated If synchronous operation is required from a drive that is subject to slip, the distance encoder function must be activated. To do so, the ratio between the motor encoder and distance encoder must be entered as the numerator/denominator factor.
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 In the main state "Free running n-control" (SynchronousState = 0), the slave drive can be operated with speed control using a speed setpoint (SpeedFreeMode<>0). A 64-bit difference counter stores the resulting angular misalignment. Example: The axis is to move at a speed of 1500 rpm after disengaging. $\frac{1500\frac{1}{\text{min}}}{0.2\frac{1}{\text{min}}} = 7500$ A setpoint of 7500 entered in variable H439 corresponds to a speed of 1500 rpm.
H440	Reserved4		
H441	Reserved5		

Variable	Name and value range	Status	Description
H442	MasterTrimX14 -32768 to 32767	R/W	Virtual axis Pulse number 1 incr./ms The MasterTrimX14 IPOS variable (H442) represents the simplest variant of a virtual encoder without ramp generator. If the physical encoder on terminal X14 is activated by the startup interface (if not using the startup interface → assign H430 = 0), the assignment MasterTrimX14 = k means that pulses are physically added to the external encoder on terminal X14 every ms. The number of pulses added is defined by k and the pulses are added with observance of the sign (+/- character). This means it is possible to apply an offset to terminal X14 as a physical source of master increments, or even to dispense completely with a physical encoder. This means that assigning MasterTrimX14 = k activates an endless counter as a virtual source of master increments. Important: Terminal X14 can no longer be used as an encoder simulation for X15 if you are using the virtual encoder.
H443	Reserved6		
H444	ReSprintClose 0 to 2	R/W	Direction of rotation inhibit = 0: Both directions of rotation are enabled = 1: Only CCW direction of rotation = 2: Only CW direction of rotation In some applications, the slave is never allowed to follow the master if the master changes direction. This means it is possible to prevent the slave drive from changing its direction of rotation in spite of the fact that phase-synchronous operation is active. The lag error limit may be reached depending on its setting, in which case a lag error is generated. The synchronization point is not lost until the lag error limit is reached. This means the slave drive restarts in the permitted direction of rotation once the master has returned, including the synchronization point.
H445	Reserved7		
H446	MFilterTime 1 to 30	R/W	Interpolation time in ms = 1: without filter ≤ 30: Scaling up, absolute scaling factor of the master pulses = GFMaster × MFilterTime The MFilterTime variable (H446) acts for interpolation of the incoming master pulses. Increasing it causes the scaling of the master pulses to be changed: Absolute master gear ratio factor = GFMaster (H428) × MFilterTime (H446). Make sure the result does not exceed 2147483648.

7 IPOS^plus® Sample Programs

INFORMATION

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

7.1 Example 1

7.1.1 Task

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

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

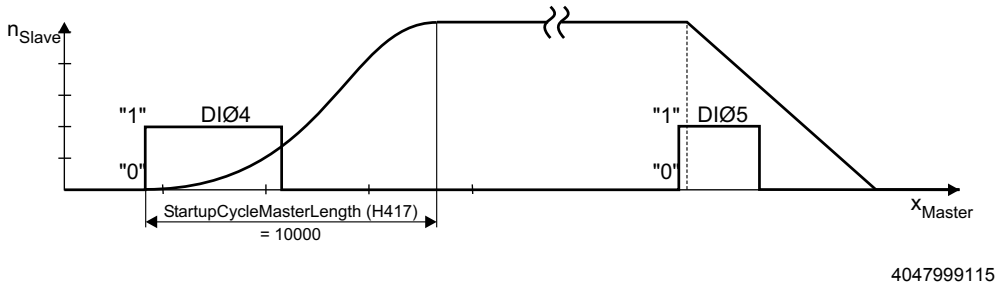


Figure 31: Event-driven start and stop cycles

7.1.2 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
*/============================================================================
Mainfunction (IPOS-Startfunction)
============================================================================*/
main ()
{
*/-----------------------------------------------------------
Startup
-----------------------------------------------------------*/
InitSynchronization(); // Call the initialization function

/*-----------------------------------------------------------
Mainprogram loop
-----------------------------------------------------------
*/
while (1)
{
}
}
```

7.1.3 Header file with variable designation

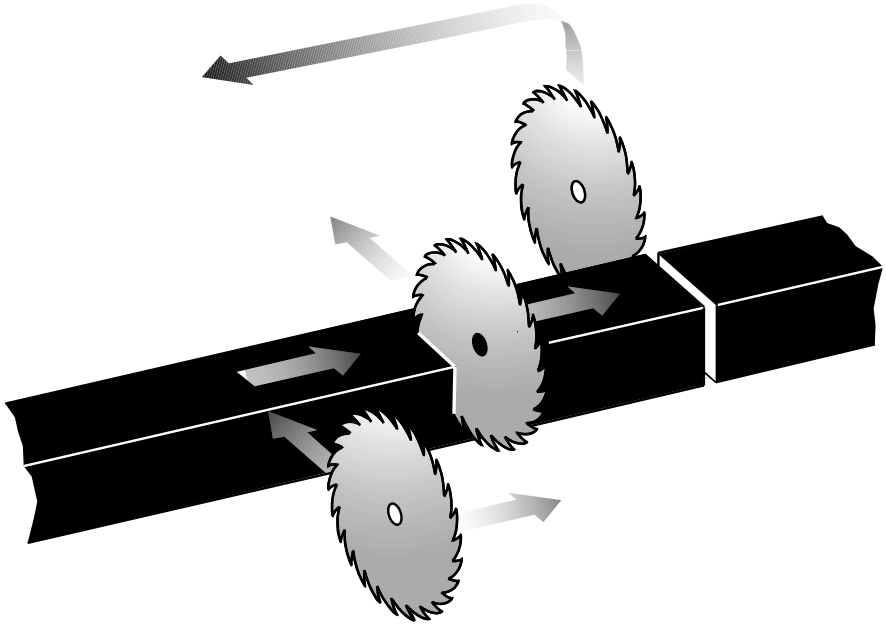
```
/*****************************************************************************
Example01.h
Data and startup header file for IPOS+ Compiler.
For Startup after power on call "InitSynchronization();"
Data file Movidrive Synchronous Drive Control Version 1.0
*****************************************************************************/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

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

7.2 Example 2

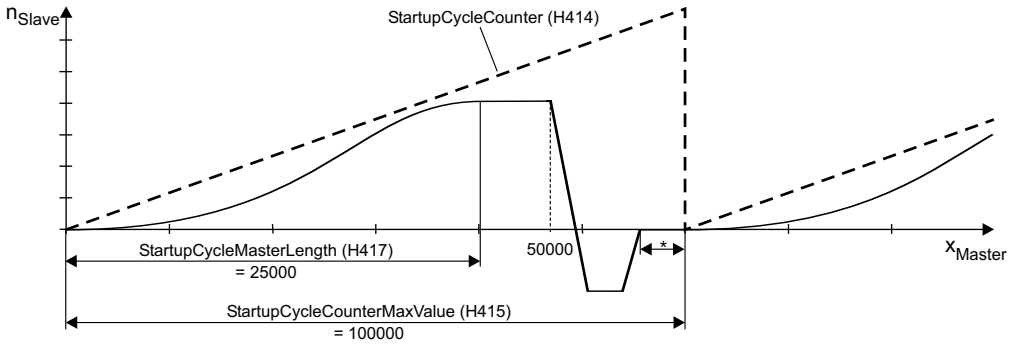
7.2.1 Task

Extruded material is to be cut with 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.



41692939

Figure 32: Flying saw



06644AXX

Figure 33: Position-controlled initiation of the startup cycle process (slave is disengaged)*

INFORMATION

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

7.2.2 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 SYNCHRONLAUF 6                      // Internal synchronous operation
#define HALT _BitClear (ControlWord, 2)     // Right bit is cleared
#define FREIGABE _BitSet (ControlWord, 2)   // Right bit is set
long Rampenform, tmp;
*/============================================================================
Mainfunction (IPOS start function)
============================================================================*/
main()
{
*/-----------------------------------------------------
Startup
-----------------------------------------------------*/
InitSynchronization();
Rampenform = LINEAR; // Positioning ramp
_SetSys(SS_RAMPTYPE, Ramp type);
while (!DI=00); // Wait for high (1) level on DI00 "/Controller inhibit"
_Go0((GO0_C_W_ZP);                          // Referencing with reference type 3 / right limit
_GoAbs(GO_WAIT, 0);                            switch
                                            // P900 "Reference offset": 300000 incr.
                                            // Move to start position
Rampenform=SYNCHRONLAUF;                    // Activate internal synchronous operation
_SetSys(SS_RAMPTYPE, Ramp type);
StartupCycleCounter = 0;                    // Reset counter
StartupCycleState = 1;                      // Activate startup cycle control
/*-----------------------------------------------------
Mainprogram loop
-----------------------------------------------------*/
while (1)
{                                           // Save startup cycle counter in temporary memory
tmp = StartupCycleCounter;                  // Change ramp type
if ((tmp>50000)&&(SynchronousState==3))     // if counter > 50000 master incr.
{                                           // and drive in synchronous operation
Halt;                                       // Inhibit drive
SynchronousState=5;                         // Disengage (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
}
}
}
```

7.2.3 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
```

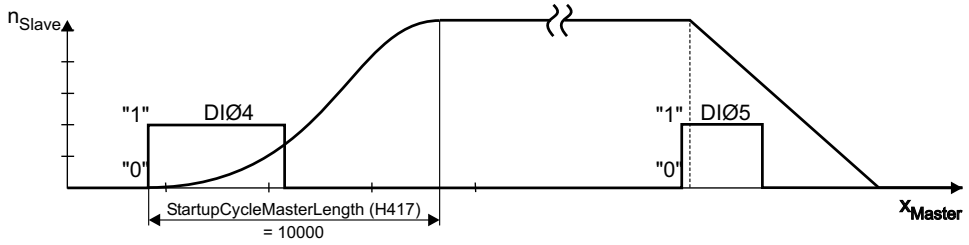
```
#define VEncoderMode                H370
#define VEncoderModeControl         H371
#define VEncoderState               H372
#define VEncoderNSetpoint           H373
#define VEncoderNActual             H374
#define VEncoderXSetpoint           H375
#define VEncoderXActual             H376
#define VEncoderdNdT                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-controlled starting of the
_BitSet(StartupCycleModeControl, 0);      startup cycle
StartupCycleCounterMaxValue= 100000;   // AutoRestart of startup cycle activated
StartupCycleMasterLength= 25000;       // Overrun value of the startup cycle counter
_BitSet (StartupCycleModeControl, 12); // Length of master travel until startup cycle is finished
RegisterLoopDXDTOut= 2;                // Selection of "Position-dependent startup cycle process"
_BitSet(StopCycleModeControl, 0);      // Limiting of correction mechanism
_BitSet(SynchronousModeControl, 0);    // Stop cycle in main state 1 (x-control)
_BitSet (SynchronousModeControl, 1);   // "Movement to TargetPos (H492)" activated
}                                      // No lag error monitoring
```

7.3 Example 3

7.3.1 Task

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



06645AXX

Figure 34: Event-driven start and stop cycles

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).

INFORMATION

Note the following settings of the master inverter:

- 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.

INFORMATION

Note the following settings of the slave inverter:

- 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).

7.4 IPOS^plus® program 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;             // SEW standard structure is written:
Position.CycleTime=1;               // Data object no. 1100 (32-bit master position to be sent/
Position.Offset=0;                     H511)
Position.Format=4;                  // is sent on the SBus (cycle time 1 ms, MOTOROLA format)
Position.DPointer=511;
Position.Result=0;
SynchID.ObjectNo=1090;              // SEW standard structure is written:
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
_SBusCommDef(SCD_TRCYCL, SynchID);  // for cyclical data transmission using
                                    // an SBus connection
_SBusCommOn();                      // Initialization of the send data objects and start of the
                                    // cyclical data transmission via SBus
/*============================================================================
Main program loop
============================================================================*/
while(1)
{
}
}
```

7.5 IPOS^plus® 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
=============================================================================*/
#pragma var 300 309
#pragma globals 310 349
#include <const.h>
#include<Example03.h> // Header file with variable designations and initialization function
SREC Position;                    // SEW standard structure for the statement
_SBusCommDef
*/=============================================================================
Mainfunction (IPOS start function)

=============================================================================*/
main()
{
*/-----------------------------------------------------------
Startup
-----------------------------------------------------------*/
InitSynchronization(); // Call the initialization function
Position.ObjectNo = 1100;         // SEW standard structure is written:
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
                                  // cyclical data transmission via SBus
/*-----------------------------------------------------------
Mainprogram loop
-----------------------------------------------------------
*/
while(1)
{
}
}
```

7.6 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                 H361
#define OffsetCycleModeControl          H360
#define OffsetCycleState                H362

#define OffsetCycleInputMask            H363
#define OffsetCycleCounterMaxValue      H364
#define OffsetCycleMasterLength         H365
#define OffsetCycleValue                H366
                                        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 start
                                                  // 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 "Position-dependent startup
                                                     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 stop cycle
        }
```

Index

T

V

X




SEW-EURODRIVE
Driving the world

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

➞ **www.sew-eurodrive.com**