• This information does not replace the detailed operating instructions!
• Installation and startup only by trained personnel observing applicable accident prevention regulations and the MOVIDRIVE® operating instructions!

• Read through this manual carefully before you commence installation and startup of MOVIDRIVE® drive inverters with internal synchronous operation.
  This manual assumes that the user has access to and is familiar with the documentation on the MOVIDRIVE® system, in particular the MOVIDRIVE® system manual.

• Safety notes:
  Always follow the safety and warning instructions contained in this manual!
  Safety notes are marked as follows:

  Electrical hazard, e.g. during live working.

  Mechanical hazard, e.g. when working on hoists.

  Important instructions for safe and fault-free operation of the driven machine/system, e.g. pre-setting before startup.

• In this manual, cross references are marked with a ▸, e.g.: (→ Sec. X.X) means that further information can be found in section X.X of this manual.

• A requirement of fault-free operation and fulfillment of any rights to claim under guarantee is that this information is observed.
# Contents

1 **System Description** ........................................................................................................... 4  
   1.1 Application fields ............................................................................................................. 4  
   1.2 Functional description ..................................................................................................... 4  
   1.3 State machine of internal synchronous operation ......................................................... 5  
   1.4 Controlling internal synchronous operation ................................................................. 5  

2 **Project Planning** ................................................................................................................. 6  
   2.1 Application examples ....................................................................................................... 6  
   2.2 Pre-requisites ................................................................................................................... 8  
      2.2.1 PC and software ......................................................................................................... 8  
      2.2.2 IPOS\textsuperscript{\textregistered} Compiler ............................................................. 8  
      2.2.3 Inverter ..................................................................................................................... 8  
      2.2.4 Motors and encoders ............................................................................................... 9  
   2.3 Project planning notes ..................................................................................................... 10  
   2.4 Synchronous start/stop .................................................................................................... 11  

3 **Installation** .......................................................................................................................... 12  
   3.1 Software .......................................................................................................................... 12  
   3.2 Connecting the incremental encoder master to MOVIDRIVE\textsuperscript{\textregistered} slave ... 13  
   3.3 Connecting MOVIDRIVE\textsuperscript{\textregistered} master to MOVIDRIVE\textsuperscript{\textregistered} slave 14  
   3.4 SBus connection of master/slave(s) ................................................................................ 15  

4 **Startup** ............................................................................................................................... 16  
   4.1 General information ......................................................................................................... 16  
   4.2 Preliminary work .............................................................................................................. 16  
   4.3 Starting up internal synchronous operation ..................................................................... 16  
      4.3.1 General information ................................................................................................. 16  
      4.3.2 Starting up with X14 – X14 connection .................................................................... 17  
      4.3.3 Starting up with SBus connection ............................................................................. 17  

5 **Working Method and Functions** ......................................................................................... 18  
   5.1 Controlling internal synchronous operation .................................................................. 18  
   5.2 Main state machine ......................................................................................................... 18  
   5.3 Startup cycle mode control ............................................................................................ 20  
      5.3.1 Time-controlled synchronization process ............................................................... 20  
      5.3.2 Travel-dependent synchronization process .............................................................. 20  
      5.3.3 Startup cycle state machine ..................................................................................... 21  
   5.4 Stop cycle state machine ............................................................................................... 24  
   5.5 Offset control ................................................................................................................ 25  
      5.5.1 Time-controlled offset processing ............................................................................ 25  
      5.5.2 Travel-dependent offset processing ....................................................................... 25  
      5.5.3 Offset state machine ............................................................................................... 25  
   5.6 Synchronous operation .................................................................................................. 27  
   5.7 Virtual encoder .............................................................................................................. 28  
      5.7.1 Virtual encoder without ramp generator ................................................................. 28  
      5.7.2 Virtual encoder with ramp generator ...................................................................... 28  
   5.8 Important notes .............................................................................................................. 29  

6 **System Variables of Internal Synchronous Operation** ...................................................... 30  

7 **Sample IPOS Programs** ..................................................................................................... 33  
   7.1 Example 1: .................................................................................................................... 33  
   7.2 Example 2: .................................................................................................................... 36  
   7.3 Example 3: .................................................................................................................... 40
1 System Description

1.1 Application fields
The internal synchronous operation function enables a group of motors to be operated at a synchronous angle in relation to one another or with an adjustable proportional relationship (electronic gear).

Internal synchronous operation is particularly suited to the following sectors and applications:

- **Beverage industry**
  - Filling stations

- **Multiple column hoist**

- **Synchronous material transport**

- **Extruder applications, cutting to length material off the roll**
  - Flying saw
  - Rotating knife

Internal synchronous operation offers the following advantages in these applications:

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

1.2 Functional description
The internal synchronous operation function takes the form of a special firmware package which only expects increments from a master. The master can either be the X14 input (physical master drive) or any IPOS variable (virtual master drive), for example in conjunction with the SBus or a virtual encoder.

- **Synchronization**
  The time-controlled synchronization mechanism has been implemented. An angular differential in the slave drive resulting from free running is reduced to zero.

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

- **Synchronous operation**
  Various functions are included in synchronous operation. For example, it is possible to operate with a specified offset after a specific travel distance. The offset between the master and slave drive comes into effect after a specified number of master increments.

- **Disengaging**
  The slave disengages from synchronous operation using the stop cycle process. This process can be started manually by setting a system variable or it may be event-driven via a binary input.
1.3 State machine of internal synchronous operation

The individual functions of internal synchronous operation are controlled using something referred to as a state machine. This state machine is divided into five main states.

- **State Z0 = Free running speed control:** The slave drive moves in free running mode with speed control. The reference to the master drive is stored in a difference counter.
- **State Z1 = Free running position control:** The slave drive stops with position control and therefore does not drift out of position. The reference to the master drive is not stored.
- **State Z2 = Engaging:** The slave drive is synchronized with the master drive either under time control or travel control.
- **State Z3 = Synchronous operation:** The slave drive moves in synchronicity with the master drive.
- **State Z4 = Offset:** An offset can be set in the synchronous operation.
- **State Z5 = Disengaging:** The slave drive exits synchronous operation.

1.4 Controlling internal synchronous operation

Internal synchronous operation is controlled using IPOSplus® variables within the IPOSplus® application program. All states can be viewed and set in a variable range from H360 to H446 which is reserved for internal synchronous operation (→ System variables section).
2 Project Planning

2.1 Application examples

a) Master/slave mode of two drives

b) Master/slave mode of two drives with virtual encoder as master
c) Group configuration: Master and equivalent slaves, e.g. multiple column hoist

![Group configuration diagram](image_url)

**Fig. 4: Group configuration**

03558AEN

---

d) Group configuration with virtual master encoder:

![Group configuration with virtual master encoder diagram](image_url)

**Fig. 5: Group configuration with virtual master encoder**

03561AEN
2.2 Pre-requisites

2.2.1 PC and software

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

2.2.2 IPOSplus® Compiler

The user program for internal synchronous operation must be created using IPOSplus® compiler. Do not use the assembler (on-screen programming) for this purpose.

IPOSplus® variables H360 to H450 are defined for internal synchronous operation.

2.2.3 Inverter

- The MOVIDRIVE® MD_60A...-5_3-4-06 special version contains a special firmware package for internal synchronous operation.

- Internal synchronous operation can only be implemented with MOVIDRIVE® MDV60A in the CFC operating modes or with MOVIDRIVE® MDS60A (SERVO). Internal synchronous operation cannot be implemented with MOVIDRIVE® MDV60A in the VFC operating modes or with MOVIDRIVE® MDF60A.

- Only 1 parameter set is available; parameter set 2 cannot be used.

- The following options are not supported and therefore may not be used: “Single-axis positioning control type DPI11A” and “Synchronous operation board type DRS11A.”

The special version for internal synchronous operation has the following part numbers:

<table>
<thead>
<tr>
<th>MOVIDRIVE® MDV60A...</th>
<th>Part number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0015-5A3-4-06</td>
<td>826 994 7</td>
</tr>
<tr>
<td>0022-5A3-4-06</td>
<td>826 995 5</td>
</tr>
<tr>
<td>0030-5A3-4-06</td>
<td>826 996 3</td>
</tr>
<tr>
<td>0040-5A3-4-06</td>
<td>826 997 1</td>
</tr>
<tr>
<td>0055-5A3-4-06</td>
<td>826 998 X</td>
</tr>
<tr>
<td>0075-5A3-4-06</td>
<td>826 999 8</td>
</tr>
<tr>
<td>0110-5A3-4-06</td>
<td>827 000 7</td>
</tr>
<tr>
<td>0150-503-4-06</td>
<td>827 001 5</td>
</tr>
<tr>
<td>0220-503-4-06</td>
<td>827 002 3</td>
</tr>
<tr>
<td>0300-503-4-06</td>
<td>827 003 1</td>
</tr>
<tr>
<td>0370-503-4-06</td>
<td>827 004 X</td>
</tr>
<tr>
<td>0450-503-4-06</td>
<td>827 005 8</td>
</tr>
<tr>
<td>0550-503-4-06</td>
<td>827 006 6</td>
</tr>
<tr>
<td>0750-503-4-06</td>
<td>827 007 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MOVIDRIVE® MDS60A...</th>
<th>Part number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0015-5A3-4-06</td>
<td>827 008 2</td>
</tr>
<tr>
<td>0022-5A3-4-06</td>
<td>827 009 0</td>
</tr>
<tr>
<td>0030-5A3-4-06</td>
<td>827 010 4</td>
</tr>
<tr>
<td>0040-5A3-4-06</td>
<td>827 011 2</td>
</tr>
<tr>
<td>0055-5A3-4-06</td>
<td>827 012 0</td>
</tr>
<tr>
<td>0075-5A3-4-06</td>
<td>827 013 9</td>
</tr>
<tr>
<td>0110-5A3-4-06</td>
<td>827 014 7</td>
</tr>
<tr>
<td>0150-503-4-06</td>
<td>827 015 5</td>
</tr>
<tr>
<td>0220-503-4-06</td>
<td>827 016 3</td>
</tr>
<tr>
<td>0300-503-4-06</td>
<td>827 017 1</td>
</tr>
<tr>
<td>0370-503-4-06</td>
<td>827 018 X</td>
</tr>
</tbody>
</table>
2.2.4 Motors and encoders

• For operation on MOVIDRIVE® MDV60A:
  – Asynchronous servomotors CT/CV, high-resolution sin/cos encoder installed as standard.
  – AC motors DT/DV/D with incremental encoder option, preferably high-resolution sin/cos encoder.

• For operation on MOVIDRIVE® MDS60A:
  – Synchronous servomotors DS/DY, resolver installed as standard.

High-resolution speed detection is required for optimum operation of internal synchronous operation. The encoders installed as standard on CT/CV and DS/DY motors fulfill these requirements. SEW recommends using high-resolution sin/cos encoders ES1S, ES2S or EV1S as incremental encoders if DT/DV/D motors are used.
2.3 Project planning notes

- Do not use internal synchronous operation with systems that have a rigid mechanical connection.

- Fit slave inverters with a braking resistor.

- During project planning for the synchronous operation application, bear in mind that the slave must be able to reduce the angle differential between itself and the master to zero at any time. For this reason, set the maximum speed (P302) of the slave to a greater value than the maximum speed of the master.

- During the time-controlled synchronization process, the synchronization speed of the slave drive must be faster than the maximum speed of the master drive.

- If possible, always use the same type of drives for internal synchronous operation.

- In the case of multiple column hoists, always use the same motors and the same gear units (identical ratios).

- When drives of the same type are operating as a synchronized group (e.g. multiple column hoist), then the drive which carries the highest proportion of the load during operation must be selected as the master.

- Connect the slave motor encoder to X15 (ENCODER IN) and the master incremental encoder to X14 (ENCODER IN/OUT) → MOVIDRIVE® operating instructions.

- Master is incremental encoder on X14: use an incremental encoder with the maximum possible resolution, however no more than 200 kHz.

- It is impossible to evaluate signals from any encoders (e.g. an external encoder on the distance) other than the master incremental encoder on X14, unless an additional option card is fitted. Exception: Signals from an absolute encoder can be evaluated as from an external encoder if the slave inverter is equipped with the ‘absolute encoder interface type DIP11A’ option.

- Only slave drives with an interlocking (= slip-free) connection between the motor shaft and the driven machine are allowed to be used.

- Operation with SBus → Setting up a cyclical data transfer in an IPOS 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.
  Important: Delays may occur as a result of the SBus transfer.

- Synchronous start/stop (→ Sec. 2.4)

- Direct cable-break monitoring (X14-X14 connection, encoder connection) is not possible. Indirect cable-break monitoring is possible during operation with SBus by way of the SBus timeout response (P836).
2.4 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 in synchronicity. This is a prerequisite for correct operation. As a result, combinations in which the master is more dynamic than the slave are not permitted.

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

<table>
<thead>
<tr>
<th>Master</th>
<th>Slave</th>
<th>Master parameter</th>
<th>Slave parameter</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDV</td>
<td>MDV</td>
<td>DOØ2 = Output stage on</td>
<td>DIØ3 = Enable/rapid stop (factory setting) DIØ1 and DIØ2 = No function</td>
<td>Connect master binary output DOØ2 to slave binary input DIØ3.</td>
</tr>
<tr>
<td>MDV</td>
<td>MDS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDS</td>
<td>MDS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Essential note:**

- The brake function must be active in the master and the slave (P730 "Brake function 1" = ON).
- The brake release time (P731) of the master must be increased by the premagnetizing time (P323) of the slave drive.
- The free running function of the slave is only possible if slave binary input X13:4 (DIØ3) gets a “1” signal from elsewhere.
3 Installation

3.1 Software

Proceed as follows to install MOVITOOLS® on your computer:

1. Insert the MOVITOOLS® CD into the CD ROM drive of your PC.
2. Select “Start/Run...”.
3. Type “{Drive letter of your CD drive}:setup” and press the Enter key.
4. The MOVITOOLS® setup menu appears. Follow the instructions of the installation wizard.

You can now use the Program Manager to start MOVITOOLS®, if a MOVIDRIVE® unit is connected to your PC, select the correct port (PC COM port) and set point-to-point connection. Select <Update> to display the inverter in the “Connected Units” window.

Fig. 6: MOVITOOLS® window
3.2 Connecting the incremental encoder master to MOVIDRIVE® slave

Connect the master incremental encoder (TTL sensor with 24 V supply, for example ES1R, ES2R or EV1R) as follows:

![Diagram of encoder connection](image)

**Important note:**
- The maximum permitted line length of the incremental encoder/motor encoder is 100 m (330 ft).
- Only use shielded encoder cables with twisted pair conductors (A and \(A\), B and \(B\), C and \(C\)). Connect the shield at both ends.
- Route the encoder cable separately from the power cables.
- SEW offers pre-fabricated cables for a simple and fault-free encoder connection.
3.3 Connecting MOVIDRIVE® master to MOVIDRIVE® slave

**X14 – X14 connection:**

1. Necessary connection for synchronous start/stop (→ 'Synchronous start/stop' on page 11).
2. It is essential to comply with Fig. 10 and the following instructions for the X14 – X14 connection!

*Fig. 9: Connecting MOVIDRIVE® master to MOVIDRIVE® slave*

*Fig. 10: X14 – X14 master/slave connection*

**Essential note:**
- With MOVIDRIVE® master: Jumper X14:4 with X14:5.
- Do not connect X14:4 and X14:9.
- SEW offers a prefabricated cable for a straightforward and trouble-free X14 – X14 connection. You can order this cable from SEW by quoting part number 815 355 8.
3.4 SBus connection of master/slave(s)

The "System Bus" manual contains detailed information about the system bus (SBus). This manual can be obtained from SEW, publication number 0918 0915.

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

**Special note:**
- Use a 2-core twisted and shielded copper cable (data transmission cable with shield comprising copper braid). Connect the shield at either end to the electronics shield clamp of MOVIDRIVE® and ensure the shield is connected over a large area. Also connect the ends of the shield to DGND.

The cable must meet the following specifications:
- Conductor cross section 0.75 mm² (AWG18)
- Cable resistance 120 Ω at 1 MHz
- Capacitance per unit length ≤ 40 pF/m (12 pF/ft) at 1 kHz

Suitable cables are CAN bus or DeviceNet cables, for example.

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

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

- There must not be any potential displacement between the units which are connected together using the SBus. Take suitable measures to avoid a potential displacement, e.g. by connecting the unit ground connectors using a separate lead.
4  Startup

4.1  General information
Correct project planning and installation are the pre-requisites for successful startup. Refer to the MOVIDRIVE® system manual for detailed project planning instructions. The system manual forms part of the MOVIDRIVE® documentation package (publication number 0919 3219).

Check the installation, including the encoder connection, by following the installation instructions in the MOVIDRIVE® MD_60A operating instructions and in this manual (Sec. 3, page 12).

4.2  Preliminary work
Perform the following steps before the startup of "internal synchronous operation":

- Connect the inverter to the PC using the serial port (RS-232, USS21A on PC-COM).
- Install MOVITOOLS® on the PC (Sec. 3.1, page 12) and start the program.
- '0' signal at terminal X13:1 (DIØØ, /Controller inhibit).
- Start up the inverter using <Shgl>.
  - With MOVIDRIVE® MDV60A and DT/DV/D or CT/CV motors, in CFC & IPOS operating mode.
  - With MOVIDRIVE® MDS60A and DS/DY motors, in SERVO & IPOS operating mode.

4.3  Starting up internal synchronous operation

4.3.1  General information

- Start <Shell>.
- Set parameter P916 "Ramp type" to "I-SYNCHR.OPERAT.," thereby activating internal synchronous operation.

![Fig. 12: Activating internal synchronous operation with P916](image)

This setting can also be made with the "_SetSys(SS_RAMTYPE, Hxx)" command in the IPOS application program. In this case, variable Hxx must be given the value 6.
4.3.2 Starting up with X14 – X14 connection

The incremental encoder simulation from X14 of a MOVIDRIVE® master inverter is used as the master for internal synchronous operation. Make sure that system variable H430 MasterSource = 0 is set in the slave inverter. Only then is X14 active as the source for the master increments.

4.3.3 Starting up with SBus connection

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

With the master inverter:

• Create two SBus transmit objects (SCOM TRANSMIT CYCLIC), namely "Synchronization ID" and "Master position". Both object numbers must be greater than 1050. In addition, the object number of the synchronization ID must be lower (= with higher priority) than the object number of the master position.
• The number of the "Synchronization ID" transmit object must not be the same as its own P817 parameter value.
• The set SBus address (P813) 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.

With the slave inverter:

• Create an SBus receive object (SCOM RECEIVE), namely "Master position".
• The P817 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).
• The slaves must have different SBus addresses (P813).
5  Operating Principle and Functions

5.1 Controlling internal synchronous operation

Internal synchronous operation is controlled using IPOSplus® variables within the IPOSPlus® program, referred to below as the "application." All states of internal synchronous operation can be viewed and set in a variable range from H360 to H446 which is reserved for internal synchronous operation (see the section on system variables). All variables which are connected to internal synchronous operation have symbolic names. These variables are shown below in bold and italics.

5.2 Main state machine

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

Sub-state machines:
- Startup cycle state machine → Sec. 5.3.3, page 21
- Stop cycle state machine → Sec. 5.4, page 24
- Offset state machine → Sec. 5.5.3, page 25

Fig. 13: Main state machine of internal synchronous operation with sub-state machines
The state machine differentiates between six (6) states saved in the \textit{SynchronousState} variable (H427).

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>- SynchronousState = 0</td>
<td>Free running n-control → The slave drive can be moved with speed control using H439 (SpeedFreeMode), a 64-bit difference counter saves the distortion.</td>
</tr>
<tr>
<td>- SynchronousState = 1</td>
<td>Free running x-control → The slave drive is held in its current position.</td>
</tr>
<tr>
<td>- SynchronousState = 2</td>
<td>Engaging phase → Synchronization takes place depending on bit 12 in H411 (StartupCycleModeControl).</td>
</tr>
<tr>
<td>- SynchronousState = 3</td>
<td>“Hard” synchronous operation → The slave drive follows the master drive with angular accuracy.</td>
</tr>
<tr>
<td>- SynchronousState = 4</td>
<td>Offset → The offset is set depending on bit 12 in H361 (OffsetCycleModeControl).</td>
</tr>
<tr>
<td>- SynchronousState = 5</td>
<td>Disengaging phase → The slave drive is disengaged with the \textit{t11 ramp} (P130).</td>
</tr>
</tbody>
</table>

Additional functions can be selected using the bits of the \textit{SynchronousModeControl} variable (H426):

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0   | PosTrim   | = 0: Deactivated  
|     |           | = 1: During position control in free running mode (main state 1), causes the slave drive to move to \textit{TargetPos} (H492) |
| 1   | Lag Error | = 0: Lag error monitoring  
|     |           | = 1: No lag error monitoring |
5.3 Startup cycle mode control

5.3.1 Time-controlled synchronization process

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

The synchronization speed \( n_{\text{synch}} \) and the synchronization ramp \( a_{\text{synch}} \) are set using parameters P240 “Synchronization speed” and P241 “Synchronization ramp.” These two parameters only have any effect in the time-controlled synchronization process and in time-controlled offset processing.

5.3.2 Travel-dependent synchronization process

In this synchronization mechanism, the slave drive moves in synchronicity with the master drive once the master drive has covered the specified distance. The specified distance must be saved in the StartupCycleMasterLength system variable (H417), with the value given in increments in relation to the master. The restriction is that the slave drive starts with speed zero.

Fig. 14: Speed profile of the time-controlled synchronization process

Fig. 15: Speed profile for the travel-dependent synchronization process
5.3.3 Startup cycle state machine

Startup cycle mode control reacts in the main states Z0 and Z1 (→ Fig. 13). The startup cycle process of the slave to the master can be performed either manually, event-driven or with interrupt control. The startup cycle mode is defined with the \textit{StartupCycleMode} system variable (H410). Additional functions can be programmed with the \textit{StartupCycleModeControl} system variable (H411).

Variable H410 (\textit{StartupCycleMode}) → engaging mode:

- \textit{StartupCycleMode} = 0: Manual engaging. The startup cycle process starts when the application assigns the value 2 to the \textit{SynchronousState} system variable (H427).
- \textit{StartupCycleMode} = 1: Event-driven starting of the startup cycle process via binary input. The \textit{StartupCycleInputMask} variable (H413) defines which binary input triggers the startup cycle process. The process is started as soon as a “1” level is present at the defined binary input. The terminal latency is 1 ms.

\begin{center}
\textbf{Fig. 16: Event-driven starting of the travel-dependent startup cycle process}
\end{center}

- \textit{StartupCycleMode} = 2: A signal edge at binary input DI02 or on the C track X14:3 triggers the startup cycle process (interrupt-controlled). To do this, 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 \textit{StartupCycleCounterMaxValue} (H415) variable. The response time of the sensor can be taken into account using the \textit{StartupCycleDelayDI02} variable (H416) (1 digit = 0.1 ms). This parameter is also effective for engaging with X14:3 (C track).

\begin{center}
\textbf{Fig. 17: Interrupt-controlled starting of the travel-dependent startup cycle process}
\end{center}
Operating Principle and Functions

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

- **StartupCycleMode = 3**: The startup cycle process is initiated by the **StartupCycleCounter** position counter (H414). Engaging takes place automatically if the **StartupCycleCounter** value is greater than the **StartupCycleCounterMaxValue** counter overrun value (H415). In this case, **StartupCycleCounterMaxValue** must be greater than the total number of master encoder pulses in the startup cycle, master cycle and stop cycle.

**Fig. 19: Position-controlled starting of the travel-dependent startup cycle process**

1. Synchronous operation and stop cycle
2. Slave is disengaged, time for positioning back to the initial position

**Fig. 18: Startup cycle state machine with interrupt control (engaging mode 2)**
Startup cycle state machine in (H412) **StartupCycleState:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0   | AutoRestart (StartupCycleMode 2 and 3) | = 0: AutoRestart deactivated  
= 1: AutoRestart activated |
| 1   | StartupDisable (StartupCycleMode 2 and 3)  | = 0: Engaging possible  
= 1: Engaging inhibited |
| 2   | InterruptSelect (StartupCycleMode 2)       | = 0: DI02  
= 1: X14:3 (C track) |
| 12  | StartupMode                  | = 0: Time-controlled synchronization  
= 1: Travel-dependent synchronization |
5.4 Stop cycle state machine

Stop cycle mode control reacts in the main states Z3 and Z4 (→ Fig. 13). The stop cycle process of the slave can either 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 disengaging, 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).

Variable H400 (StopCycleMode) → disengaging mode:

- \( \text{StopCycleMode} = 0 \): Manual disengaging. The slave ceases synchronous operation with the master when the application assigns the value 5 to the SynchronousState system variable (H427).
- \( \text{StopCycleMode} = 1 \): Event-driven disengaging via binary input. The StartupCycleInputMask variable (H413) defines which binary input triggers the stop cycle process. The process is started as soon as a "1" level is present at the defined binary input. The terminal latency is 1 ms.

Variable H401 (StopCycleModeControl) → Additional functions:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FreeMode</td>
<td>= 0: Disengaging in main state 0 (n-control)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1: Disengaging in main state 1 (x-control)</td>
</tr>
</tbody>
</table>

Fig. 21: Event-driven disengaging

03855AXX
5.5 Offset control

5.5.1 Time-controlled offset processing

In this state, an offset is added to the difference counter (H367, OffsetCycleValue). The slave drive moves an offset by the reduction in the angle differential to zero (time-controlled synchronization → Sec. 5.3.1).

5.5.2 Travel-dependent offset processing

The slave drive is subject to an offset in this state after the master drive has covered the value entered in the OffsetCycleMasterLength system variable (H366). The offset value must be stored in the OffsetCycleValue system variable (H367).

State 3 (synchronous operation) is a prerequisite for this process.

5.5.3 Offset state machine

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

Variable H360 (OffsetCycleMode) → Offset mode:

- **OffsetCycleMode = 0**: Manual offset processing using the IPOS program by setting H427 (SynchronousState) to the value 4.
- **OffsetCycleMode = 1**: Offset processing using binary inputs ("1" level) with H363 (OffsetCycleInputMask) with a resolution of 1 ms.

Fig. 22: Speed profile for travel-dependent offset processing

![Speed profile for travel-dependent offset processing](03412AXX)

The slave drive is subject to an offset in this state after the master drive has covered the value entered in the OffsetCycleMasterLength system variable (H366). The offset value must be stored in the OffsetCycleValue system variable (H367).

State 3 (synchronous operation) is a prerequisite for this process.

Fig. 23: Travel-dependent offset processing controlled by binary inputs

![Travel-dependent offset processing controlled by binary inputs](03791AXX)
Operating Principle and Functions

- **OffsetCycleMode = 2**: Reserved
- **OffsetCycleMode = 3**: Position control in conjunction with variables H364 (OffsetCycleCounter) and H365 (OffsetCycleCounterMaxValue), with remaining distance carryover.

![Diagram](image)

*Fig. 24: Position-controlled, travel-dependent offset processing*

**Variable H361 (OffsetCycleModeControl) → Additional functions:**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AutoRestart (OffsetCycleMode 3)</td>
<td>= 0: AutoRestart deactivated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1: AutoRestart activated</td>
</tr>
<tr>
<td>1</td>
<td>OffsetDisable (OffsetCycleMode 3)</td>
<td>= 0: Offset processing possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1: Offset processing inhibited</td>
</tr>
<tr>
<td>12</td>
<td>OffsetMode</td>
<td>= 0: Time-controlled offset processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1: Travel-dependent offset processing</td>
</tr>
</tbody>
</table>

**Offset state machine in H362 (OffsetCycleState):**

![Diagram](image)

*Fig. 25: Offset state machine*
5.6 Synchronous operation

Control takes place with a P-controller (P910 “Gain X controller”). The master and slave pulses are evaluated with the corresponding weighting factors and added to a 64-bit value after comparison. The P-controller together with the feedforward (P228 “Feedforward filter”) and subsequent limiting to the maximum speed forms the speed setpoint for the speed controller.

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

Block diagram for internal synchronous operation:
5.7 Virtual encoder

5.7.1 Virtual encoder without ramp generator

The MasterTrimX14 IPOS variable (H442) represents the most simple variant of a virtual encoder. If the physical encoder is activated (assignment H430 = 0), then k pulses are physically added to the master encoder every millisecond, observing the correct sign, by assigning MasterTrimX14 = k.

Important: X14 can no longer be used as an encoder simulation if you are using the virtual encoder.

5.7.2 Virtual encoder with ramp generator

The virtual encoder (→ IPOS variables H370 – H377) is a software counter which can be used as the master encoder for synchronous operation. (Assignment of MasterSource H430 = 376.) A system bus connection (SCOM command) enables this software counter to be transferred to other MOVIDRIVE® axes by the “generator” of the virtual encoder. To do this, it is necessary to have SBus synchronization with the synch-ID (P817) for unit synchronization (every 5 ms).

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

H375 = [VEncoderXSetpoint] = 1 *master pulse* Target position
H376 = [VEncoderXActual] = 1 *master pulse* Actual position
H373 = [VEncoderNSetpoint] = 1 *master pulse'/ms Set speed
H374 = [VEncoderNActual] = 1 *master pulse'/ms Actual speed
H377 = [VEncoderdNdT] = 1 *master pulse'/ms² Acceleration

Selecting the operating mode of the encoder in variable H370 (VEncoderMode):  
- VEncoderMode = 0: Normal mode (specified by the travel profile programmed using variables H373, H375 and H377)
- VEncoderMode = 1: Reserved
- VEncoderMode = 2: Endless counter with travel speed VEncoderNSetpoint and set ramp VEncoderdNdT

![Structural image of virtual encoder with ramp generator](image-url)
Variable H371 (VEncoderModeControl):

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value 0</th>
<th>Value 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AxisStop</td>
<td>Axis stop deactivated</td>
<td>The value of VEncoderNSetpoint (H373) is set to 0 (stop of the virtual axis) once after a unit fault occurs.</td>
</tr>
</tbody>
</table>

5.8 Important notes

- The possibility of specifying a signed distance in variable H417 (StartupCycleMasterLength) or H366 (OffsetCycleMasterLength) for the master drive means it is important to check the direction of rotation of the master drive. In addition, the gear factor in variable H428 (GFMaster) can also be entered as a signed value.

- A lag error is only triggered (P923 "Lag error window") in main state Z3 (synchronous operation).

- The 64-bit counter can be cleared by programming the terminals with "Set DRS zero point". This step disconnects the master branch if the drive is in main state Z3 (synchronous operation).

- A value other than zero should be entered in variable H390 (RegisterLoopDXDTOut) in order to achieve exact results in travel-dependent synchronization. This is so the remaining travel can be reduced to zero.

- P910 "Gain X controller": This parameter is set to its optimum value during startup with MOVITOOLS®.

- P228 "Feedforward filter (DRS)": Setpoint filter for feedforward of internal synchronous operation control. Factory setting = 0 = Filter has no effect. Recommended setting = 10 ms.

- The MFilterTime variable (H446) acts for interpolation of the incoming master pulses. Increasing it causes the weighting of the master pulses to be changed. A correction can be made by multiplying the GFSlave variable (H429) with the MFilterTime variable (H446), for example.

- Absolute master gear factor = GFMaster (H428) × MFilterTime (H446)
  Make sure the result does not exceed 32767.
## System Variables of Internal Synchronous Operation

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name and range of values</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
</table>
| H360     | OffsetCycleMode 0 to 3   | R/W    | Offset mode  
  = 0: Offset via IPOS program  
  = 1: Offset via input terminals  
  = 2: Reserved  
  = 3: Offset via position control |
| H361     | OffsetCycleModeControl   | R/W    | Activation of various functions  
  Bit 0: AutoRestart (in mode 3)  
  = 0: AutoRestart deactivated  
  = 1: AutoRestart activated  
  Bit 1: OffsetDisable (in mode 3)  
  = 0: Offset processing possible  
  = 1: Offset processing inhibited  
  Bit 12: OffsetMode  
  = 0: Time-controlled offset processing  
  = 1: Travel-dependent offset processing |
| H362     | OffsetCycleState Max. 0 to 1 (depending on OffsetCycleMode) | R/W | Control of the various modes |
| H363     | OffsetCycleInputMask     | R/W    | Terminal window (identical to H483 "InputLevel") |
| H364     | OffsetCycleCounter       | R/W    | Master counter for offset processing |
| H365     | OffsetCycleCounterMaxValue | R/W | In mode 3: Length limit for automatic offset processing |
| H366     | OffsetCycleMasterLength  | R/W    | Specified distance for the master drive in offset processing |
| H367     | OffsetCycleValue         | R/W    | Offset value for slave drive |
| H370     | VEncoderMode 0 to 2      | R/W    | Virtual encoder operating mode  
  = 0: Normal mode  
  = 1: Reserved  
  = 2: Infinite counter |
| H371     | VEncoderModeControl      | R/W    | Bit 0: AxisStop  
  = 0: Deactivated  
  = 1: Axis stop on unit fault |
| H372     | VEncoderState            | R/W    | No function |
| H373     | VEncoderNSetpoint        | R/W    | Set travel speed in 1 incr./ms |
| H374     | VEncoderNActual          | R/W    | Actual travel speed in 1 incr./ms |
| H375     | VEncoderXSetpoint        | R/W    | Target position in incr. |
| H376     | VEncoderXActual          | R/W    | Current position in incr. |
| H377     | VEncoderNdT              | R/W    | Acceleration (ramp) in 1 incr./ms² |

### Virtual encoder

### Control element

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name and range of values</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H389</td>
<td>RegisterLoopOut</td>
<td>R/W</td>
<td>The value to be reduced in connection with RegisterLoopDXDTOut</td>
</tr>
</tbody>
</table>
| H390     | RegisterLoopDXDTOut      | R/W    | Control element limit  
  Max. addition (64-bit counter) per ms |
<table>
<thead>
<tr>
<th>Variable</th>
<th>Name and range of values</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
</table>
| H400     | StopCycleMode 0 to 1    | R/W    | Stop cycle mode  
  = 0: Disengaging via IPOS program  
  = 1: Disengaging via input terminals |
| H401     | StopCycleModeControl    | R/W    | Activation of various functions  
  Bit 0: FreeMode  
  = 0: Disengaging in main state 0 (n-control)  
  = 1: Disengaging in main state 1 (x-control) |
| H402     | StopCycleState          |        | No function |
| H403     | StopCycleInputMask      | R/W    | Terminal window (identical to H483 "InputLevel") |
| H410     | StartupCycleMode 0 to 3 | R/W    | Startup cycle mode  
  = 0: Engaging via IPOS program  
  = 1: Engaging via input terminals  
  = 2: Engaging via interrupt control  
  = 3: Engaging via position control |
| H411     | StartupCycleModeControl | R/W    | Activation of various functions  
  Bit 0: AutoRestart (in mode 2 and 3)  
  = 0: AutoRestart deactivated  
  = 1: AutoRestart  
  Bit 1: StartupDisable (in mode 2 and 3)  
  = 0: Engaging possible  
  = 1: Engaging inhibited  
  Bit 2: InterruptSelect (in mode 2)  
  = 0: DI02  
  = 1: X14C track  
  Bit 12: StartupMode  
  = 0: Time-controlled synchronization  
  = 1: Travel-dependent synchronization |
| H412     | StartupCycleState       | R/W    | Control of the various modes |
| H413     | StartupCycleInputMask   | R/W    | Terminal window (identical to H483 "InputLevel") |
| H414     | StartupCycleCounter     | R/W    | Master counter for engaging |
| H415     | StartupCycleCounterMaxValue | R/W | In mode 2: Delay for startup cycle process  
  In mode 3: Length limit for automatic engaging |
| H416     | StartupCycleDelayDI02   | R/W    | Delay in units of 0.1 ms  
  Delay time of the sensor connected to touch-probe input 2 |
| H417     | StartupCycleMasterLength| R/W    | Specified distance for the master drive in travel-dependent engaging |
### System Variables of Internal Synchronous

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name and range of values</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H425</td>
<td>SynchronousMode</td>
<td>No function</td>
<td>Activation of various functions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 0: PosTrim (only active in main state Z1 &quot;X-control&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0: Activated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1: Movement to TargetPos (H492)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 1: LagError (in state 3 → Synchronous operation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0: Lag error monitoring</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1: No lag error monitoring</td>
</tr>
<tr>
<td>H426</td>
<td>SynchronousModeControl</td>
<td>R/W</td>
<td>Main state machine integrated synchronous operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0: Free running n-control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1: Free running x-control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 2: Engaging</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 3: Synchronous operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 4: Offset processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 5: Disengaging</td>
</tr>
<tr>
<td>H427</td>
<td>SynchronousState</td>
<td>0 to 5</td>
<td>Source of the master increments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0: X14 + Virtual axis (H442)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; 0: Pointer to variable</td>
</tr>
<tr>
<td>H428</td>
<td>GFMaster</td>
<td>-32768 to 32767</td>
<td>Weighting factor of the master increments, value = l_S</td>
</tr>
<tr>
<td>H429</td>
<td>GFSlave</td>
<td>1 to 32767</td>
<td>Weighting factor of the slave increments, value = l_M</td>
</tr>
<tr>
<td>H430</td>
<td>MasterSource</td>
<td>0 to 511</td>
<td>Source of the master increments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0: X14 + Virtual axis (H442)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt; 0: Pointer to variable</td>
</tr>
<tr>
<td>H431</td>
<td>Reserved1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H432</td>
<td>LagDistance64Low</td>
<td>R/-</td>
<td>Low 32 bits of the 64-bit counter</td>
</tr>
<tr>
<td>H433</td>
<td>LagDistance64High</td>
<td>R/-</td>
<td>High 32 bits of the 64-bit counter</td>
</tr>
<tr>
<td>H434</td>
<td>LagDistance32</td>
<td>R/-</td>
<td>32-bit lag distance in relation to GFSlave</td>
</tr>
<tr>
<td>H435</td>
<td>Reserved2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H436</td>
<td>Reserved3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H437</td>
<td>SlaveTrim</td>
<td>R/W</td>
<td>Current position setpoint of the slave drive</td>
</tr>
<tr>
<td>H438</td>
<td>XMasterPos</td>
<td>R/-</td>
<td>Display value of the master counter during startup cycle process and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>during offset processing</td>
</tr>
<tr>
<td>H439</td>
<td>SpeedFreeMode</td>
<td>R/W</td>
<td>Speed setpoint in free running n-control in 0.2 rpm</td>
</tr>
<tr>
<td>H440</td>
<td>Reserved4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H441</td>
<td>Reserved5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H442</td>
<td>MasterTrimX14</td>
<td>R/W</td>
<td>Virtual axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pulse number 1 incr./ms</td>
</tr>
<tr>
<td>H443</td>
<td>Reserved6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H444</td>
<td>ReSprintClose</td>
<td>R/W</td>
<td>Direction of rotation inhibit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 0: Enable both directions of rotation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1: Only CCW direction of rotation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 2: Only CW direction of rotation</td>
</tr>
<tr>
<td>H445</td>
<td>Reserved7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H446</td>
<td>MFiterTime</td>
<td>1 to 30</td>
<td>Interpolation time in ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= 1: Without filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>≤ 30: Scaling up, absolute weighting factor of the master pulses</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= GFMaster × MFiterTime</td>
</tr>
</tbody>
</table>
7 Sample IPOS Programs

7.1 Example 1:

Objective:
A slave drive is to be operated at a synchronous angle to a master drive. The gear units used in this case are the same. The gear ratio is 1:1. The master and slave inverters are connected via X14. Control of the slave inverter is via the binary inputs. Binary inputs X13:5 (DI04) and X13:6 (DI05) should be used for controlling the startup and stop cycle processes. Both binary inputs must be programmed to "NO FUNCTION."

- "1" signal on DI04 → The startup cycle process is started. The startup cycle process should be travel-dependent and completed after 10,000 master increments.
- "1" signal on DI05 → The stop cycle process is started.

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

---

Fig. 28: Event-driven engaging and disengaging

\[ \text{StartupCycleMasterLength (H417)} = 10000 \]
# IPOS program:

```c
/*===============================================================================
IPOS source file
for Synchronous Drive Control
===============================================================================*/
SEW-Eurodrive GmbH & Co.
Ernst-Blickle-Str. 42
D-76646 Bruchsal

sew@sew-eurodrive.de
http://www.SEW-EURODRIVE.de

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

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

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

/*===============================================================================
Header file with variable designation:
===============================================================================*/
Example01.h
Data and startup header file for IPOS+ Compiler.
For startup after power on call "InitSynchronization();"
Datafile 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  MasterTrimX14              H438
#define  SpeedFreeMode              H439
#define  Reserved4                  H440
#define  Reserved5                  H441
#define  MasterTrimX14              H442
#define  Reserved6                  H443
```
```c
#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 incr. w/o filter
    StartupCycleMode = 1;  // Startup cycle mode 1: Event-driven starting
    StartupCycleInputMask = 16;  // Selection of terminal DI04 for engaging
    StartupCycleMasterLength = 10000;  // Length of master travel until engag.finished
    _BitSet(StartupCycleModeControl, 12);  // Sel. of "travel-dep. startup cycle process"
    RegisterLoopDXDTOut = 2;  // Limiting of correction mechanism
    StopCycleMode = 1;  // Stop cycle mode 1: Event-driven starting
    StopCycleInputMask = 32;  // Selection of terminal DI05 for disengaging
    StartupCycleMasterLength = 10000;  // Length of master travel until engag.finished
    _BitSet(StartupCycleModeControl, 12);  // Sel. of "travel-dep. startup cycle process"
    RegisterLoopDXDTOut = 2;  // Limiting of correction mechanism
    StopCycleMode = 1;  // Stop cycle mode 1: Event-driven starting
    StopCycleInputMask = 32;  // Selection of terminal DI05 for disengaging
    }```
7.2 Example 2:

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

Fig. 29: Flying saw

Fig. 30: Position-controlled starting of the startup cycle process

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

```c
#include <const.h>
#include <io.h>
#include <Example02.h>                        // Header file with
// variable designations
// and initialization function

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

long Rampenform, tmp;

// Main function (IPOS start function)
main()
{
    /*-------------------------------------
    Startup
    --------------------------------------*/
    InitSynchronization();                      // Call up initialization function
    Rampenform=LINEAR;                          // Positioning ramp
    _SetSys(SS_RAMPTYPE, Rampenform);
    while (!DI00);                        // Wait for high level on DI00 /*Controller inhibit*/
        _Go0(G00_C_W_ZP);                     // Referencing with ref. type 3 / CW limit switch
        _GoAbs(GO_WAIT, 0);                     // Move to start position
    Rampenform=SYNCHRONLAUF;
    _SetSys(SS_RAMPTYPE, Rampenform);
    StartupCycleCounter = 0;                    // Reset counter
    StartupCycleState = 1;                      // Activate startup cycle mode control
    /*-------------------------------------
    Main program loop
    --------------------------------------*/
    while(1)
    {
        tmp=StartupCycleCounter;                 // Save engagement counter in temp. memory
        if ((tmp>50000)&&(SynchronousState==3))  // Switchover ramp function,
            // if counter > 50,000 master incr.
            // and drive in synchronous operation
            Halt;                                 // Inhibit drive
            SynchronousState=5;                   // Disengage (in position control)
            Rampenform=LINEAR;                    // Positioning ramp
            _SetSys(SS_RAMPTYPE, Rampenform);
```
Sample IPOS Programs

Header file with variable designation:

    /*********************************************************************/
    /* Example02.h */
    /*********************************************************************/

    #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 incr. w/o filter
    StartupCycleMode = 3;                  // Startup cycle mode 3: Position-controlled
                                          // starting of
    _BitSet(StartupCycleModeControl, 0);   // AutoRestart of startup cycle proc. activated
    StartupCycleMasterLength = 25000;      // Overrun value of the engaging counter
    RegisterLoopDXDTOut = 2;               // Limiting of correction mechanism
    _BitSet(StopCycleModeControl, 0);      // Disengaging in main state 1 (x-control)
    _BitSet(SynchronousModeControl, 0);    // "Movement to TargetPos (H492)" activated
    _BitSet(SynchronousModeControl, 1);    // No lag error monitoring
}
7.3 Example 3:

Objective:
A slave drive is to be operated at a synchronous angle to a master drive. The gear units used in this case are the same. The gear ratio is 1:1. The master and slave inverters are connected via SBus. Control of the slave inverter is 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 travel-dependent and completed after 10,000 master increments.
- "1" signal on DIØ5 → The stop cycle process is started.

![Diagram showing event-driven engaging and disengaging](image)

The necessary IPOS 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 (P813).

**Note the following settings on the master inverter:**
- The number of the "Synchronization ID" transmit object must not be the same as parameter value P817.
- 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.

**Note the following settings on the slave inverter:**
- The P817 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).

In contrast to the X14 – X14 connection (example 1), it is possible to implement cable-break monitoring with the SBus connection (→ Timeout error).
IPOS program master inverter:

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

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

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

/*-------------------------------
Initialization
-------------------------------*/

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

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

_SBusCommDef(SCD_TRCYCL, SynchID); // Setting up the transmit data objects
_SBusCommDef(SCD_TRCYCL, Position); // for cyclical data transmission using an SBus connection
_SBusCommOn(); // Initialization of the send data objects and start of cyclical data transmission via SBUS

/*-------------------------------
Main program loop
-------------------------------*/
while(1)
{
}
}
Sample IPOS Programs

IPOS program slave inverter:

/*======================================================================
 IPOS source file
 for Synchronous Drive Control
 ========================================================================
 SEW-Eurodrive GmbH & Co.
 Ernst-Blickle-Str. 42
 D-76646 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
SCREC Position;      // SEW standard structure for the _SBusCommDef
                      // statement

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

    Position.ObjectNo=1100;       // Describe the SEW standard structure:
    Position.Format=4;            // Data obj.no. 1100 (32-bit master pos. to be recvd.)
    Position.DPointer=200;        // is sent to variable H200

    _SBusCommDef(SCD_REC, Position); // Setting up 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

    /*-------------------------------------
     Main program loop
     --------------------------------------*/
    while(1)
    {
        
    }
}
MOVIDRIVE® Internal Synchronous Operation

Sample IPOS Programs

7

Header file with variable designation:

/*******************************************************************************/
Example03.h
Data and startup header file for IPOS+ Compiler.
For startup after power on call "InitSynchronization();"
Datafile Movidrive Synchronous Drive Control Version 1.0
*******************************************************************************/

#define SynchronousMode            H425
#define SynchronousModeControl     H426
#define SynchronousState           H427
#define GFMaster                   H428
#define GPSlave                    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
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 incr. w/o filter
StartupCycleMode = 1;  // Startup cycle mode 1: Event-driven starting
                      // of the startup cycle process via binary input
StartupCycleInputMask = 16;  // Selection of terminal DI04 for engaging
StartupCycleMasterLength = 10000;  // Length of master travel until eng. finished
_bit_set(StartupCycleModeControl, 12);  // Sel. of "travel-dep. startup cycle process"
RegisterLoopDXDTOut = 2;  // Limiting of correction mechanism
StopCycleMode = 1;  // Stop cycle mode 1: Event-driven starting
StopCycleInputMask = 32;  // Selection of terminal DI05 for disengaging
}
SEW-EURODRIVE right around the globe is your competent partner in matters of power transmission with manufacturing and assembly plants in most major industrial countries.

We are available, wherever you need us. Worldwide.