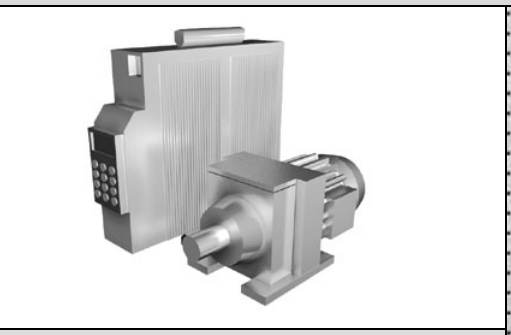
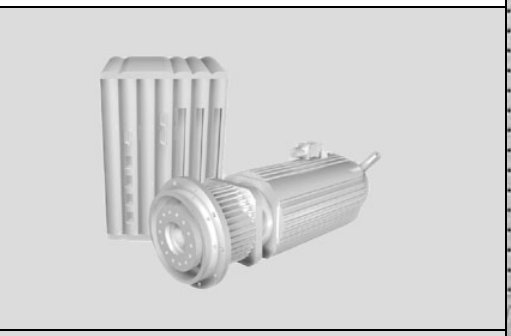
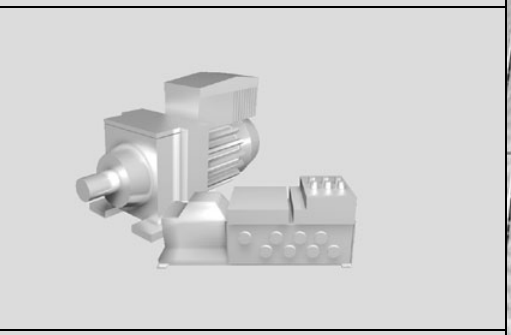
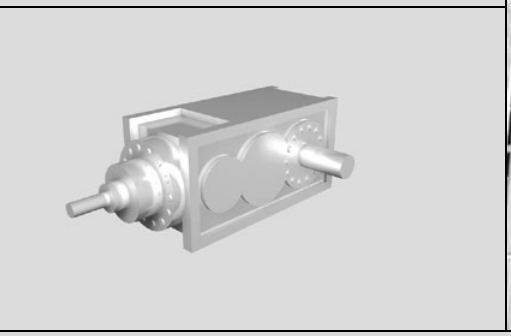
SEW EURODRIVE









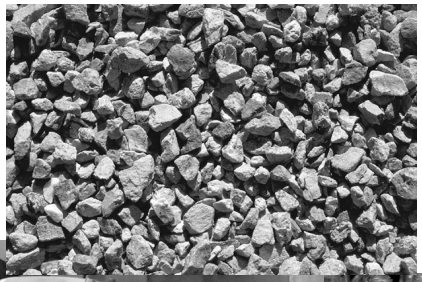
Electronic Cam

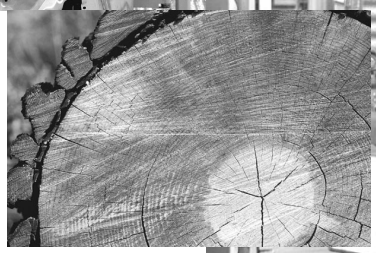
A5.J51

Edition 05/2004

11220228 / EN

Manual





SEW
EURODRIVE

1 System Description

1.1 General information

The process involved in designing the electronic cam differs significantly from the mechanical solution.

In contrast to the mechanical solution, the electronic cam allows for optimizing the operation of the machine in terms of

- smooth running
- maximum acceleration
- tendency to vibrate

A particular motion profile is used for generating a curve point table in which the interrelation between master movement and slave movement is stored. The main expertise of the machine designer is concerned with generating this curve point table.

The master movement is normally represented as a machine angle between 0 and 360 degrees. A number of curve points is defined with reference to this machine angle. These curve points specify the position of the particular slave drive with reference to the master. The number of curve points and the interpolation between these points determine the accuracy of the electronic cam.

1.2 Application areas

The electronic cam is particularly suited for the following sectors and applications:

- Packaging industry
- Hoists
- Rail vehicles
- Wood processing industry
- Storage and retrieval systems
- Transverse carriages
- Automated material handling
- Multi-axis systems
- Printing machine technology

1.3 Advantages of the electronic cam

- User-friendly user interface
- Different cam profiles are possible without having to modify the machine resulting in optimized operation of the machine and accommodation of different product types.
- Guided project planning and startup of the curve.
- The machine can be optimized with regard to
  - smooth running
  - maximum acceleration
  - tendency to vibrate
- Monitor mode for optimum diagnostics.
- Virtual encoder (software counter) possible as master encoder.
- Master encoder changeover possible using the synchronized system bus (SBus).

1.4 Functional description

The term electronic cam refers to a definite assignment of positions between a master drive and a slave drive. The master drive can either be a physical drive or a virtual master encoder. The relationship between the positions of the master drive and the slave drive is often specified in a 2-dimensional graph. The position of the master drive is entered along the horizontal axis and the position of the slave drive along the vertical axis. The range of positions along the horizontal axis is referred to as the master cycle, the range of positions along the vertical axis as the slave cycle.

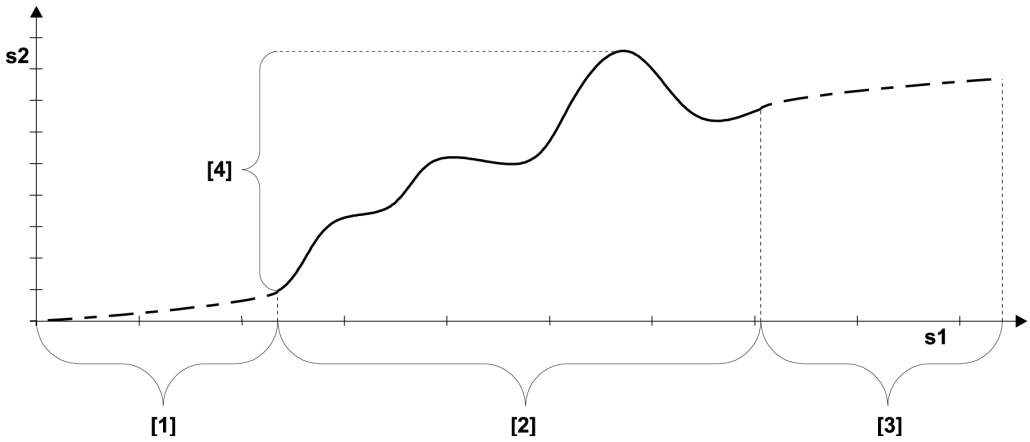


Figure 1: Electronic cam profile

[1] Startup cycle, is run through once
[2] Master cycle, is repeated cyclically
[3] Stop cycle, is run through once
[4] Slave cycle
s1 Distance master
s2 Distance slave

The graph generally comprises three sections:

Startup cycle with startup cycle curve

The startup cycle curve permits the slave drive to be smoothly synchronized with the position of the master drive. The startup cycle curve consists of up to 256 curve points and is run through once.

Master cycle with main curve

In the synchronization phase, the slave drive follows the master drive in accordance with the position setpoints entered in the main curve. The slave drive passes through the main curve cyclically until it ceases synchronization with the master drive. This process is referred to as disengaging.

The main curve consists of up to 256 curve points. Without startup cycle curve, you can increase the number of curve points to 512.

Stop cycle with stop cycle curve

The stop cycle curve makes it possible to quit the main curve at a particular position. After completed stop cycle, the slave drive loses its position reference to the master drive and ceases to follow it.

1.5 Controlling the electronic cam

The electronic cam is controlled using IPOS$^{plus®}$ variables within the IPOS$^{plus®}$ application program. All states of the electronic cam can be viewed and set in a variable range from H370 to H452 which is reserved for electronic cam control (see the section on system variables). All variables concerning the electronic cam have symbolic names. These variables are shown in **italics** in this online help.

2 Project Planning

2.1 PC and software

The technology function "Electronic cam" is part of the MOVITOOLS® software from SEW-EURODRIVE. To use MOVITOOLS®, you need a PC with one of the following operating systems:

- Windows® 95
- Windows® 98
- Windows® 2000
- Windows® NT 4.0
- Windows® XP

2.2 IPOS$^{plus®}$ compiler

The application program for the electronic cam must be created with the IPOS$^{plus®}$ compiler. Do not use the IPOS$^{plus®}$ assembler for this purpose.

- The IPOS$^{plus®}$ variables **H00 ... H15** are protected from loss in case of a power failure in the following units:
  - MDS
  - MDV
  - MCV
  - MCS
- In MOVIDRIVE® MCH and MDX60/61B, the IPOS$^{plus®}$ variables **H00 ... H127** are protected from loss in case of a power failure.
- If the electronic cam technology function is enabled, IPOS$^{plus®}$ variables **H370 ... H452** are reserved for electronic cam control.

2.3 Inverters

- A MOVIDRIVE® or MOVIDRIVE® *compact* unit with **encoder feedback** in **application version** is required for the electronic cam function. For more information, please refer to the MOVIDRIVE® and MOVIDRIVE® *compact* documentation.
- The electronic cam can only be implemented in the CFC or SERVO operating modes. The electronic cam cannot be implemented in the VFC operating mode.
- Only parameter set 1 is available; parameter set 2 cannot be used.
- "Single-Axis positioning control type DPI11A/DPA11A" and "Synchronous operation card type DRS11A" are not supported and therefore may not be used.
- All fieldbus interfaces are supported.

2.4 Motors and encoders

High-resolution speed detection is required for optimum operation of the electronic cam. The encoders installed as standard on SEW motors meet these requirements.

- CT/CV asynchronous servomotors: High-resolution sin/cos encoder installed as standard. Hiperface® encoders are possible as option.
- DT/DV/D asynchronous AC motors with incremental encoder option: Preferably high-resolution sin/cos encoder or Hiperface®.
- DS/DY synchronous servomotors: Resolver installed as standard.
- CM synchronous servomotors: Resolver installed as standard. Hiperface® encoders are possible as option.

3 Installation

3.1 Software

MOVITOOLS® 3.0. is required for startup of the electronic cam. For MOVIDIRVE® MDX60/61B, you require MOVITOOLS® 4.0. The functions of the electronic cam are then implemented as an IPOS$^{plus®}$ application program. This application program is written using the IPOS$^{plus®}$ compiler in MOVITOOLS®. Install the programs in the following order:

- Insert the MOVITOOLS® CD into the CD-ROM drive of your PC.
- Select "Start\Run..."
- Type "{drive letter of your CD drive}:setup" and press Enter.
- The MOVITOOLS® setup menu appears. Follow the instructions of the installation wizard.

You can now use the Program Manager to start MOVITOOLS®. Connect a MOVIDRIVE® unit to your PC, select the correct port (PC COM) and select peer-to-peer connection. Click on <Update> to display the inverter in the "Connected Inverters" window. To start the curve editor, click the <CAM> button in the "Execute Program" field.

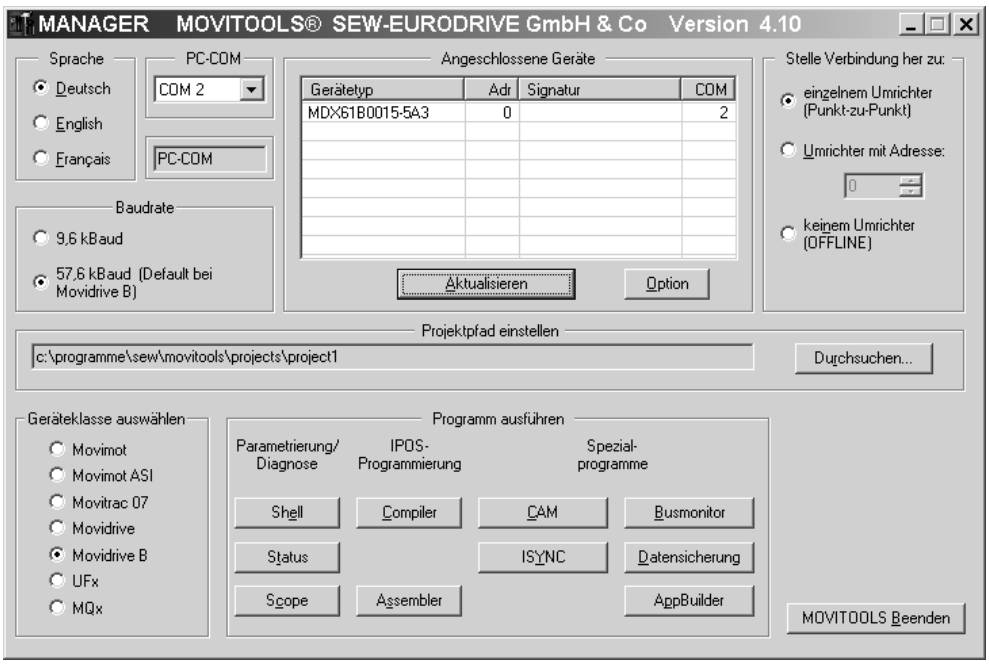


Figure 2: MOVITOOLS® 3.00 Manager with curve editor

3.2 Connecting the incremental encoder master to the MOVIDRIVE® slave

MOVIDRIVE® version A

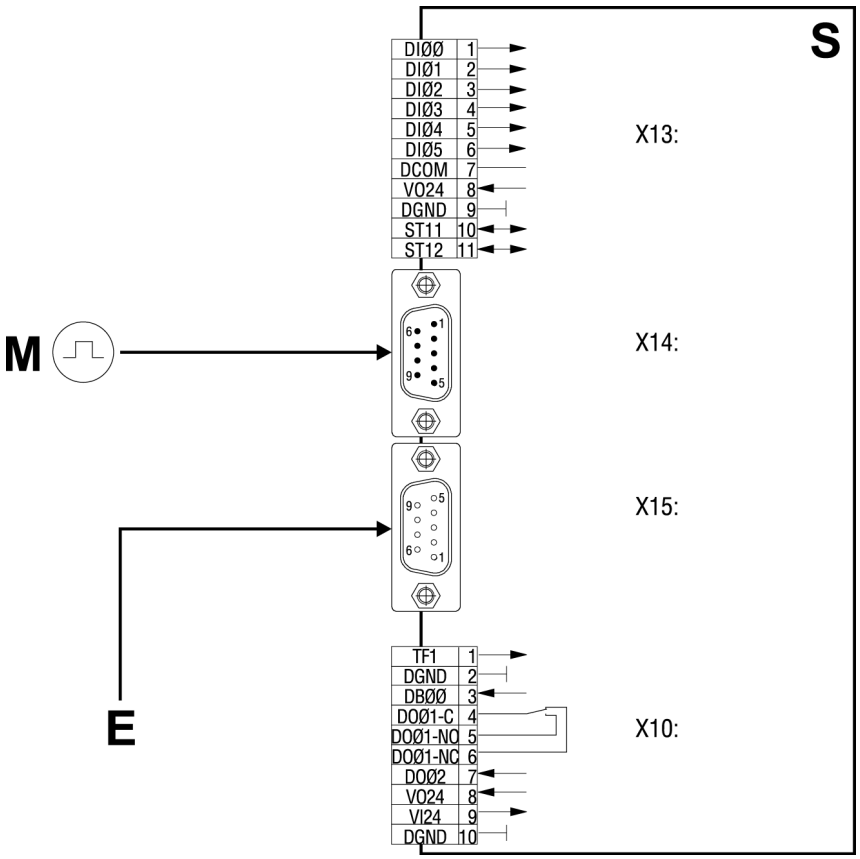


Figure 3: Connecting the incremental encoder master to MOVIDRIVE® slave

S = MOVIDRIVE® slave
M = Master: Incremental encoder TTL/RS-422 with DC 24 V supply, I_{max} = 180 mA

X14: "External encoder" input

X13:1 DI00 = /Controller inhibit
X13:2 DI01 = No function
X13:3 DI02 = No function
X13:4 DI03 = Enable / rapid stop (factory setting)
X13:5 DI04 = IPOS input
X13:6 DI05 = IPOS input
X13:7 DCOM = Reference DI00 ... DI05
X13:8 V024 = + 24 V output
X13:9 DGND = Reference potential for binary signals
X13:10 ST11 = RS-485 +
X13:11 ST12 = RS-485 –

E = Motor encoder: high-resolution sin/cos encoder in MDV or resolver in MDS

X15: Motor encoder: Incremental encoder (MDV) or resolver (MDS)

X10:1 TF1 = TF input
X10:2 DGND = Reference potential for binary signals
X10:3 DB00 = /Brake
X10:4 DO01-C = Relay contact /Ready (factory setting)
X10:5 DO01-NO = Normally open contact relay
X10:6 DO01-NC = Normally closed contact relay
X10:7 DO02 = /Fault (factory setting)
X10:8 VO24 = + 24 V output
X10:9 VI24 = + 24 V input
X10:10 DGND = Reference potential for binary signals

If the incremental encoder simulation of a MOVIDRIVE® inverter is used as master, you have to connect X14 master to X14 slave as shown in the following figure:

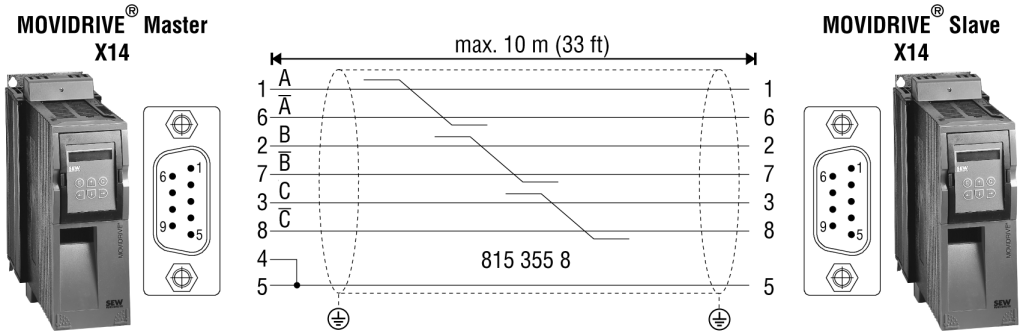


Figure 4: X14-X14 master/slave connection

It is essential to adhere to the following instructions:

- 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. To order this cable from SEW-EURODRIVE, please quote part number 815 355 8.

MOVIDRIVE® version B

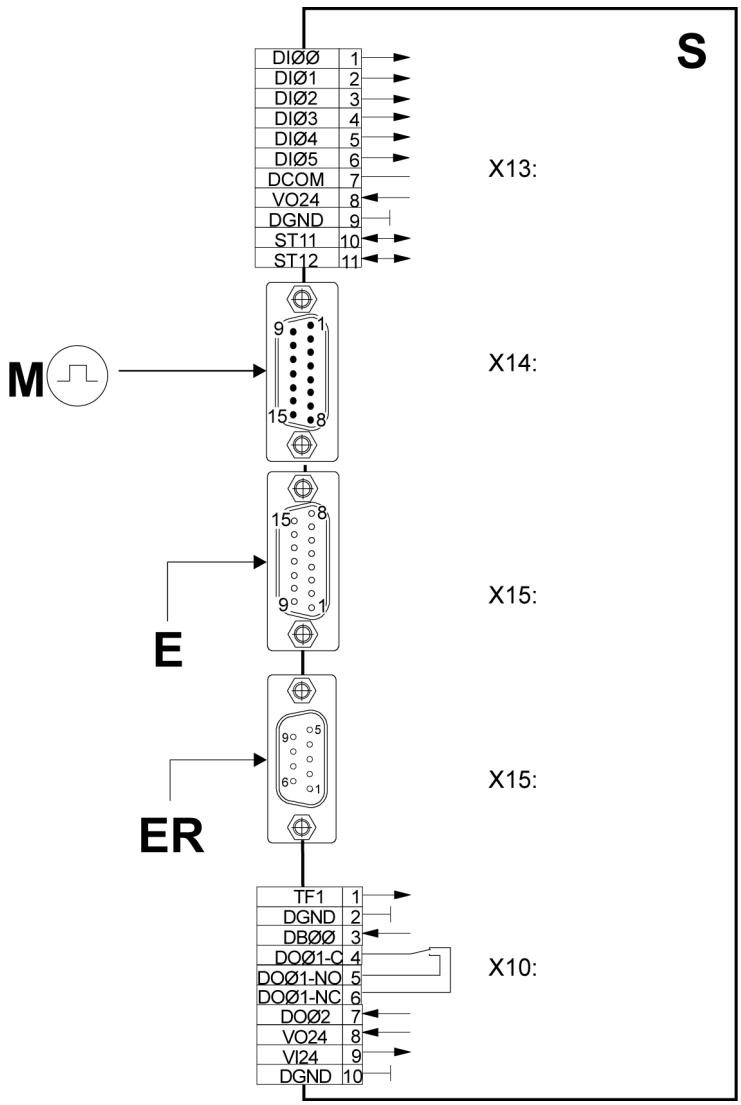


Figure 5: Connecting the incremental encoder master to the MOVIDRIVE® slave

S = MOVIDRIVE® slave
M = Master: Incremental encoder TTL/RS-422 with DC 24 V supply, I_{max} = 180 mA

X14: "External encoder" input

X13:1 DI00 = /Controller inhibit
X13:2 DI01 = No function
X13:3 DI02 = No function
X13:4 DI03 = Enable / rapid stop (factory setting)
X13:5 DI04 = IPOS input
X13:6 DI05 = IPOS input
X13:7 DCOM = Reference DI00 ... DI05
X13:8 V024 = + 24 V output
X13:9 DGND = Reference potential for binary signals
X13:10 ST11 = RS-485 +
X13:11 ST12 = RS-485 –

E = Motor encoder: High-resolution sin/cos encoder in MDV
ER = Resolver in MDS

X15: Motor encoder: Incremental encoder (MDV) or resolver (MDS)

X10:1 TF1 = TF input
X10:2 DGND = Reference potential for binary signals
X10:3 DB00 = /Brake
X10:4 DO01-C = Relay contact /Ready (factory setting)
X10:5 DO01-NO = Normally open contact relay
X10:6 DO01-NC = Normally closed contact relay
X10:7 DO02 = /Fault (factory setting)
X10:8 VO24 = + 24 V output
X10:9 VI24 = + 24 V input
X10:10 DGND = Reference potential for binary signals

If the incremental encoder simulation of a MOVIDRIVE® inverter is used as master, you have to connect X14 master to X14 slave as shown in the following figure:

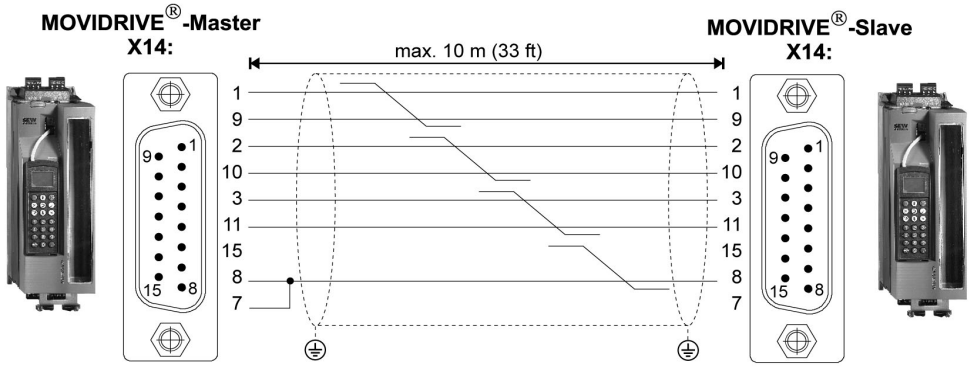


Figure 6: X14-X14 master/slave connection

It is essential to adhere to the following instructions:

- 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. To order this cable from SEW-EURODRIVE, quote part number 815 355 8.

3.3 System bus (SBus)

For more information on the system bus (SBus), please refer to the "System Bus" manual which you can obtain from SEW-EURODRIVE.

The system bus (SBus) allows you to interconnect up to 64 MOVIDRIVE® units. For example, these units can be MOVIDRIVE® units for master/slave operation, or 63 MOVIDRIVE® units and a CAN bus master control. The SBus supports transmission technology compliant with ISO 11898.

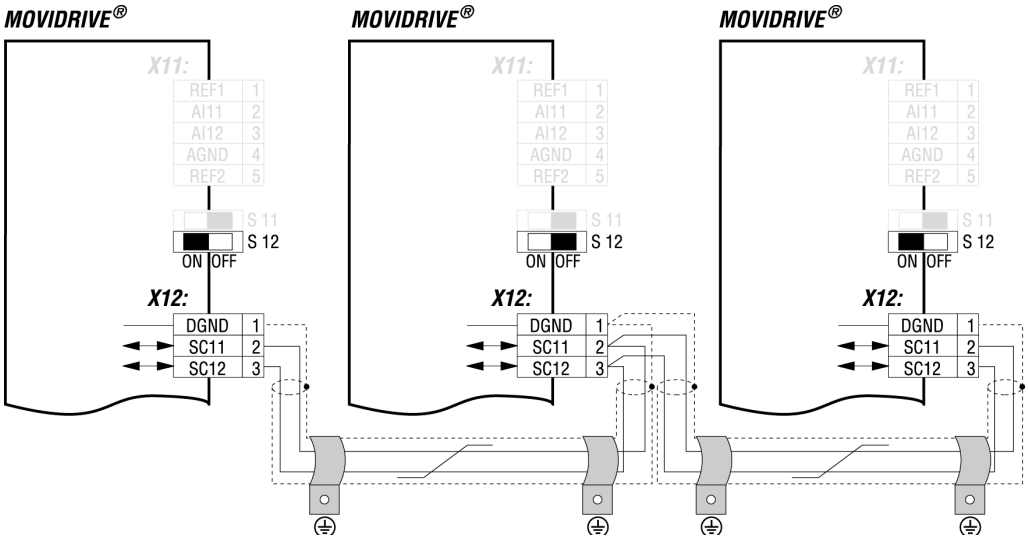


Figure 7: SBus connection

SC11 = System bus High
SC12 = System bus Low

DGND = System bus reference
S12 = System bus terminating resistor

What else you need to observe:

- Use a shielded, twisted-pair copper cable (data transmission cable with braided copper shield). Connect the shield to the electronics shield clamp and make sure it is connected over a wide area at both ends.
- The cable must be specified as CAN bus or DeviceNet cable.
- In practice, cables such as Unitronic BUS CAN 2 × 26nbsp;× 0.22 from Lapp have proven effective even under harsh conditions. The CAN signals are carried along one wire pair; the other wire pair is used for CAN GND: Yellow = CAN High / Green = CAN Low / Brown = CAN GND. This offers the advantage that the necessary compensating current of the bus drivers does not have to be carried along the shield. As a result, no magnetic EMC interference can occur in the shield and the electronics.
- The permitted total cable length depends on the baud rate setting of the SBus:
  - 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 end of the system bus connection.
- There must not be any difference of potential between the units connected using the SBus. Take suitable measures to avoid a difference of potential, such as connecting the unit ground connectors using a separate line.
- MOVIDRIVE® units of the type MCH4_A must only be operated in conjunction with MCH4_A units with 1000 kBaud. You have to set a lower transmission rate in other MOVIDRIVE® units.

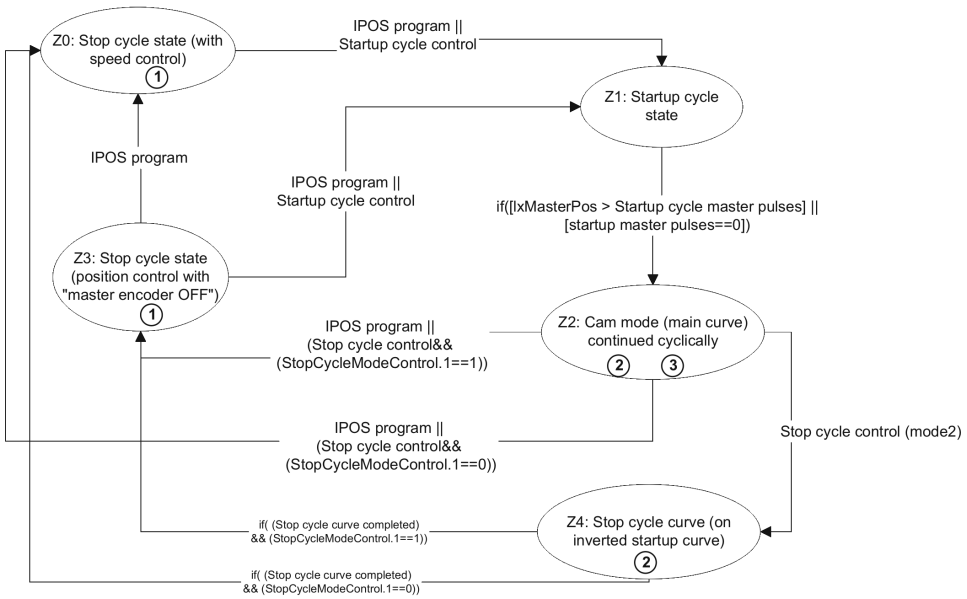
4 Cam Operation

4.1 Electronic cam state machine

The respective state of the electronic cam state machine is mapped in the CamState variable H436 (read and write).

This means the startup cycle process can be started when CamState = 1 even if the startup cycle conditions are not met.

The main curve condition must only be reached from startup cycle state Z1, even if no startup cycle curve exists. CamState must also be set to 1 if engaging is to take place using IPOS and no startup cycle curve exists. The firmware then switches to state 2 immediately.



① Startup cycle control (state machine)
② Stop cycle control (state machine)
③ Register loop control (state machine)

06581AEN

Figure 8: Electronic cam state machine

4.2 Startup cycle state machine

The startup cycle state machine (H410... H416) controls the startup cycle process. The startup cycle curve is defined if a value ≠ zero is entered in the "Startup Cycle" field of the curve editor. The startup cycle curve permits the slave to be smoothly synchronized with the position of the master.

You can perform the startup cycle process of the slave with the master as follows:

- Manually (IPOS$^{plus®}$ program)
- Event-driven (for example triggered by a counter)
- Interrupt controlled

The startup cycle consists of up to 256 curve points and is run through exactly once. The startup cycle mode is defined using the **StartupCycleMode** system variable H410:

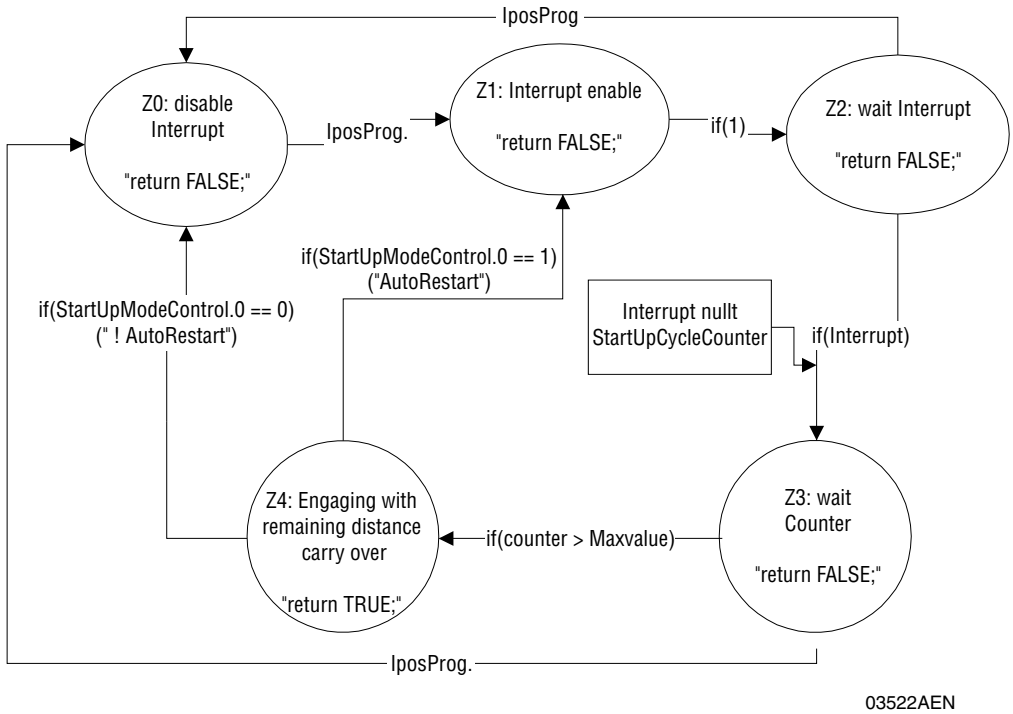


Figure 9: Startup cycle state machine

StartupCycle-Mode = 0 Manual engaging (IPOS$^{plus®}$ program). The startup cycle process starts when the application assigns the value 1 to the **CamState** system variable (H436).

StartupCycle-Mode = 1

The startup cycle process is started event-driven via binary input. The variable **StartupCycleInputMask** H413 defines which binary input triggers the startup cycle process. This binary input must be programmed to the "IPOS input" function. The inverter starts the process as soon as a "1" level is present at the defined binary input.

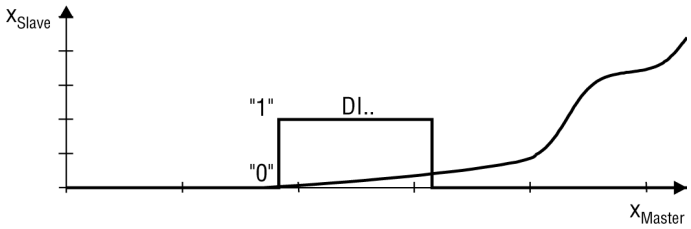


Figure 10: Event-driven starting of the startup cycle process

StartupCycle-Mode = 2

A signal edge on binary input DIØ2 or a C track pulse on X14 triggers the startup cycle process (interrupt-controlled). Binary input DIØ2 must be set to "No function" for this purpose. In conjunction with the **StartupCycleCounterMaxValue** variable H415, it is possible to define an offset for the distance in relation to the master cycle from where the actual startup cycle process begins. In state Z4, bit 0 of **StartupCycleModeControl** H411 (AutoRestart) makes it possible to decide whether the application should go to state Z0 (interrupt disabled) or Z1 (interrupt enabled). The **StartupCycleCounter** counter H414 is set to zero during state transition from Z2 to Z3. Even if AutoRestart (**StartupCycleModeControl.0** = 1) is enabled, the state of **StartupCycleState** must be set to 1 during the initialization of the application program to start interrupt processing. The interrupt source (edge on DI02 or C track pulse) is selected using H411 **StartupCycleModeControl.2**.

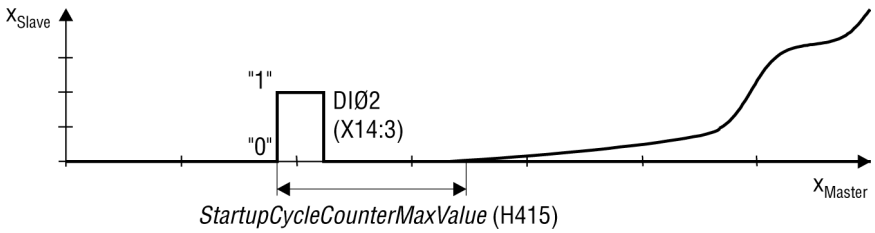


Figure 11: Interrupt-controlled starting of the startup cycle process

StartupCycle-Mode = 3

The startup cycle process is initiated by the H414 **StartupCycleCounter** position counter. Engaging takes place automatically if the **StartupCycleCounter** value is higher 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. Else, the slave remains in disengaged state.

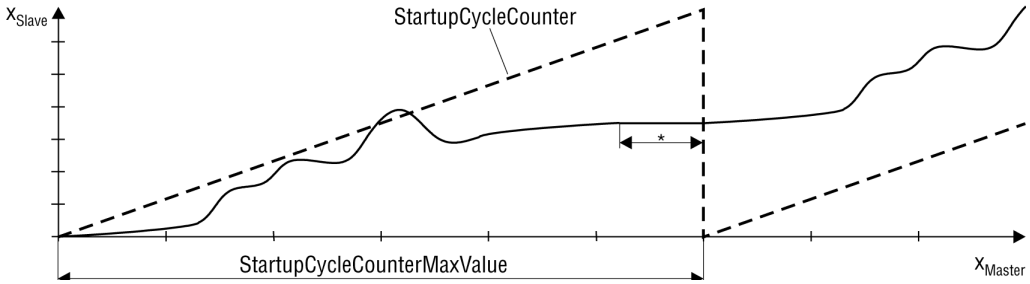


Figure 12: Counter-controlled starting of the startup cycle process

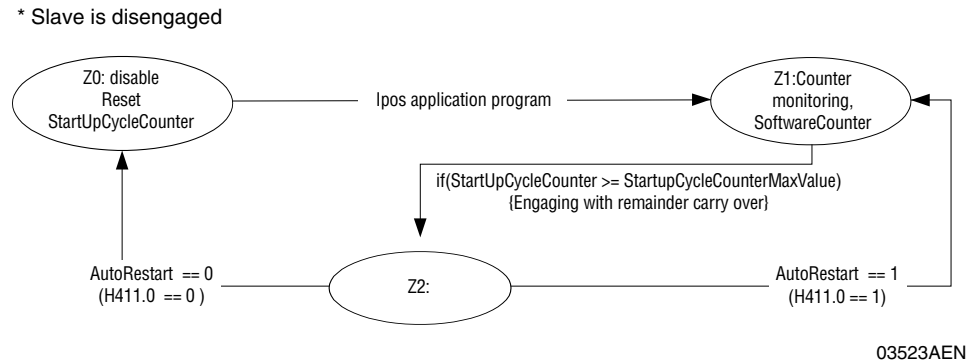


Figure 13: Startup cycle state machine: StartupCycleMode = 3

The startup cycle state machine is only active if the following conditions are met:

- The startup cycle state machine is in state Z0 or Z3.
- The application assigns the value 1 to the **StartupCycleState** variable H412.

In all cases, the value 2 is automatically assigned to the **CamState** variable H436 after the startup cycle has been run through.

4.3 Stop cycle state machine

The stop cycle state machine (H400...H406) controls the stop cycle process. The slave no longer follows the master once the stop cycle process has been completed. The slave then moves with speed or position control depending on the definition in the **StopCycleModeControl** system variable H401. The inverse of the startup cycle curve is defined as the stop cycle curve if a value unequal zero is entered in the "Startup Cycle" field of the curve editor. A separate stop cycle curve cannot be generated. The stop cycle curve makes it possible to quit the main curve at a particular position.

You can perform the stop cycle of the slave manually (IPOS$^{plus®}$ program) or automatically. The stop cycle mode is defined by the **StopCycleMode** system variable H400.

StopCycleMode = 0

Manual disengaging (IPOS$^{plus®}$ program). The slave stops synchronization with the master curve when the application assigns the value 0 to the **CamState** system variable H436.

StopCycleMode = 1

Automatic disengaging. The slave stops synchronization with the master curve after it has run through the number of master cycles defined in the **StopCycleCounter** variable H404. Each time the master cycle is completed, this counter is decremented or incremented by one depending on the direction. The value of the **StopCycleReload** variable H403 is reassigned to the counter after the stop cycle process. Disengaging takes place according to the **StopCycleModeControl** stop cycle mode H411.

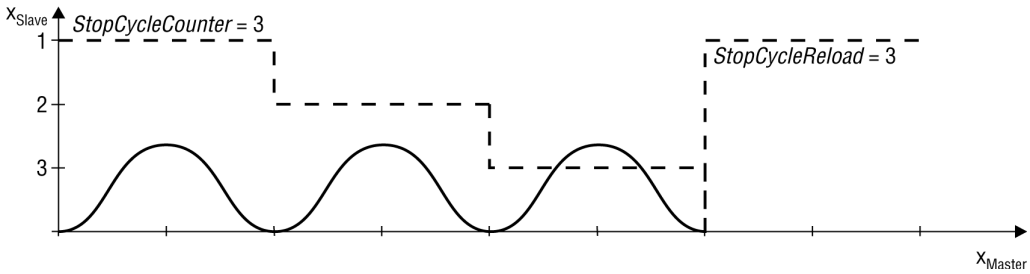


Figure 14: Example of main curve without startup cycle curve, StopCycleCounter = StopCycleReload = 3

StopCycleMode = 2

Same as **StopCycleMode** = 1 but the mirror image of the startup cycle curve is run through as the stop cycle curve. Disengaging takes place by "inversely" running through the "mirror image" of the startup cycle curve. This means the application runs backwards through a curve mirrored along the horizontal plane.

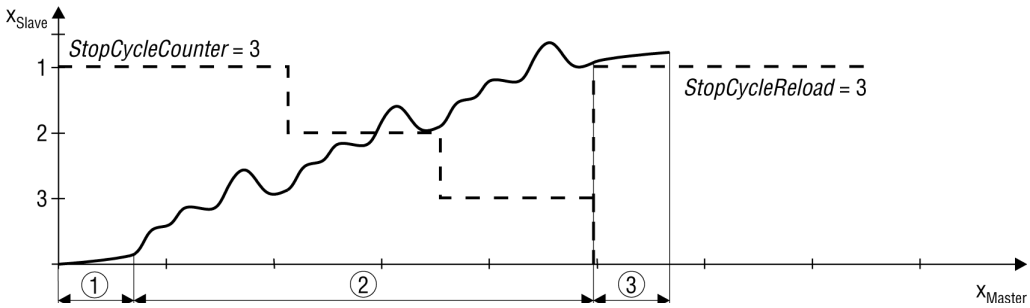


Figure 15: Main curve with startup cycle curve, StopCycleCounter = StopCycleReload = 3

[1] Startup cycle curve
[2] Main curve is run through three times
[3] Stop cycle curve

StopCycleMode = 3

Same as **StopCycleMode** = 2. Disengaging is triggered by a touch probe on DIØ3 or on the C track pulse of the slave motor. In contrast to mode 2, the application quits the main curve immediately after the remaining distance specified in **RemainCounter** H405. Disengaging takes place by "inversely" running through the "mirror image" of the startup cycle curve. This means the application runs backwards through a curve mirrored along the horizontal plane. A slave range (only within the main curve) is defined by the **RegisterLimitLeft** H383 and **RegisterLimitRight** H384 window limits. All signal edges of the DIØ3 interrupt or the C track pulse are enabled in this range. The entire main curve range is considered activated when both limits are set to zero.

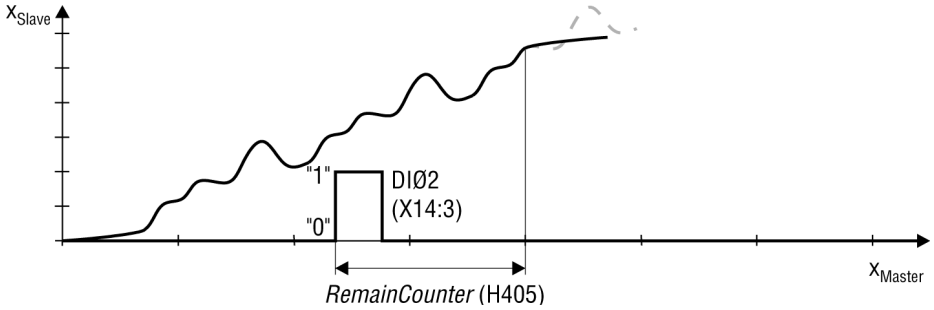


Figure 16: Sample main curve with startup cycle curve

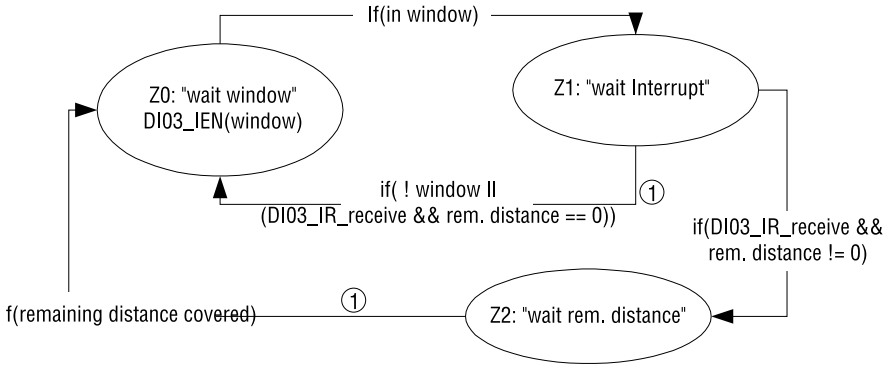


Figure 17: Stop cycle state machine: StartupCycleMode = 3

The register loop controller and stop cycle mode 3 cannot be used at the same time. In mode 3, the control element (software actuator) is used for correcting the remaining distance calculation due to the interrupt latency. Cam parameter **RegisterDXDTOut** H390 is used for this purpose. Correction is disabled when **RegisterDXDTOut** = 0. In contrast to the function of the register loop controller, the control element also operates in register mode 0 (previously only 2). The control element is then also effective in the main curve state Z4. **RegisterMode** = 3 must be selected if position correction should only be applied to the stop cycle curve and not to the main curve. The correction must then take place correspondingly faster. In this case, **you have to set RegisterDXDTOut** to a higher value.

Operating modes of stop cycle mode

The **StopCycleModeControl** variable H401 defines the operating mode of the slave in disengaged state. The following different modes are possible:

- **StopCycleModeControl.0** = 0: The slave is in speed control mode. The speed is defined by the **SpeedFreeMode** variable H439
- **StopCycleModeControl.0** = 1: The slave is in position control mode
- **StopCycleModeControl.1** = 0: Interrupt enabled on DI03
- **StopCycleModeControl.1** = 1: Signal edge change of zero track on X15 as startup cycle source

5 Ring Buffer Management

Packaging machines often require a function that allows for taking account of gaps in the flow of products during processing (e.g. labeling). If a gap is detected, the startup cycle process must be prevented. Depending on the sensor location, the startup cycle process must only be prevented once after a certain number of cycles. Ring buffer management is used for this purpose.

The ring buffer function acts on bit 1 of H411 (**StartupCycleModeControl.1**). When this bit (startup disable) is in logical state "1", engaging is prevented even if the other engaging conditions have been met.

The ring buffer function is available in the following "StartupCycleModes":

- **StartupCycleMode** == 2 (interrupt position control).
- **StartupCycleMode** == 3 (counter control)

Parameters of the ring buffer function

- **StartupCycleModeControl.3** H411.3 = Ring buffer function is enabled.
- **StartupCycleInputMask** H413 = Mask that is used to define the input for the product signal. Its assignment is identical to H483 (IPOS$^{plus®}$).
- **StartupCycleBuffer** H417 = 32-bit ring buffer which is shifted left and loaded with the current product signal from the LSB (least significant bit).
- **StartupCycleBufferLength** H418 = Effective buffer length. This parameter specifies which bit of the buffer is transferred to the "Disable Bit" H411.1. The level changes with each transfer. When the product signal = "0", disable bit H411.1 = "1". Input value: 0 ... 31.

The product signal is evaluated in all main curve states. The product signal must be active for at least 1 ms. The shifting and the "disable transfer" are triggered by an event.

Shift result

The following events can be used for shifting the ring buffer:

- Interrupt on X14 zero pulse (if **StartupCycleMode** H410 = 2) and **StartupCycleModeControl**.2 (H411) =1 (X14 C track select = ON).
- Interrupt on DIØ2 (if **StartupCycleMode** H410 = 2 and **StartupCycleModeControl.2** H411 =0 (X14 C track select = OFF).
- Counter overrun of the **StartupCycleCounter** H414 ≥ **StartupCycleCounterMaxValue** H415 if **StartupCycleMode** H410 = 3.

6 Register Loop Control

In register loop control (H380 ... H396), the actual/setpoint difference of master or slave drives subject to slip is corrected using a register recognition function. The register loop controller operates in a fixed relationship with binary input DIØ3. This input must be programmed to "No function."

Binary input DIØ3 is scanned for the first signal edge change within a position window in order to calculate the setpoint/actual difference (register loop controller). Bit 1 of the **RegisterModeControl** variable H381 is used to set whether a rising or a falling edge is detected. If there is a signal edge change within the defined position window, the curve is used for determining the position of the slave drive and the position is stored in the **RegisterActValue** system variable H386. The position window is defined by the **RegisterLimitRight** H384 and **RegisterLimitLeft** H383 variables. A register is not recognized as being valid unless it corresponds to the length defined in **RegisterLengthMin** H392 and **RegisterLengthMax** H391. A current **RegisterActValue** H386 is not generated until the validity check has been completed. Setpoint/actual value control is then switched over to the control element. The calculated setpoint/actual value difference is weighted using the **RegisterLoopPGain** variable H387 and limited by **RegisterLoopMaxOut** H388. The difference is then stored in the **RegisterLoopOut** variable H389.

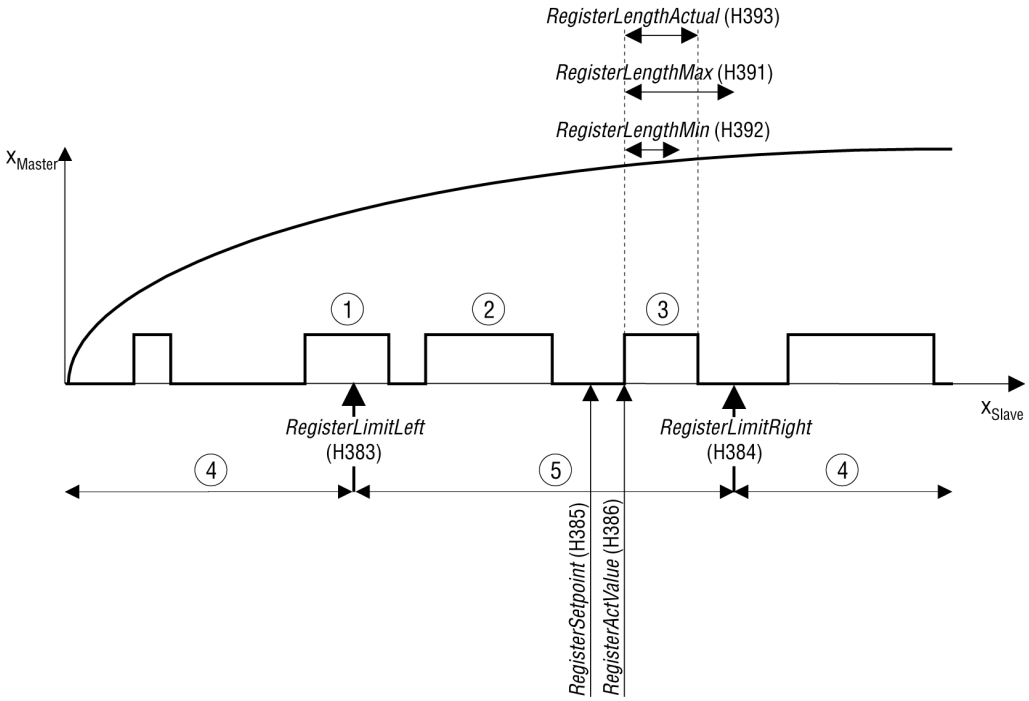


Figure 18: Register recognition diagram

[1] Register start outside the window
[2] Register is too "long"
[3] Detected as valid register
[4] Interrupt disabled
[5] Interrupt enabled

The **RegisterLoopPGain** KP factor H387 must be defined as positive or negative depending on the slope of the slave curve, otherwise positive feedback will occur. If the slave curve runs as shown in the above figure, a positive value must be selected for **RegisterLoopPGain**.
Register loop control operates in two stages. The first stage (register loop controller) detects a valid register and calculates a setpoint/actual value difference. The second stage (control element) activates the controller for adjusting the measured setpoint/actual value difference. The **RegisterMode** variable H380 defines which stage of register loop control is active or whether the calculated setpoint/actual value difference is applied to the control element. The control element and the register loop controller are only active in main state **CamState** == 2.

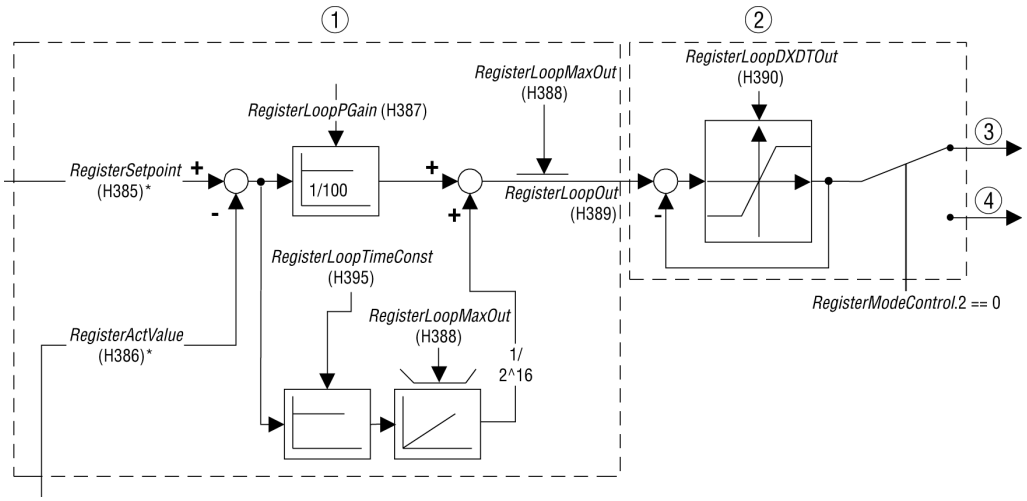


Figure 19: Register loop controller in mode 1

[1] Register loop controller
[2] Control element
* Slave pulse unit (main curve)
[3] Offset for SlaveTrim (H437)
[4] Offset for XMasterPos (H442)

Dynamic lag error correction

Dynamic lag error correction (enabled by H397 RegisterLagPGain ≠ 0)

"Dynamic lag error correction" serves to take account of the dependency of speed of the lag error within register loop control. The current lag error **LagDistance** H495 is weighted with the value of **RegisterLagPGain** H397 for this purpose. The value range for **RegisterLagPGain** is –30000 ... 0 ... 30000. To avoid positive feedback, the KP factor **RegisterLagPGain** must be defined as positive or negative depending on the slope of the slave curve. If the slope of the slave curve is positive at the time of the register signal, the KP factor must also be positive. The weighted lag error is then processed as actual value in the closed-loop control system together with **Register-ActValue** H386.

Operating modes of the register state machine

RegisterMode = 0 The register state machine is disabled.

RegisterMode = 1 Register loop control with an interrupt on touch probe terminal 2 (always DIØ3) and monitoring of register position, level and window. Register loop control is automatically enabled and disabled within the **RegisterLimitLeft** H483 and **RegisterLimitRight** H484 register window. Register loop control is deactivated outside this window. Auto restart **RegisterModeControl.0** H381 causes automatic restart to be activated again after a valid signal on the touchprobe terminal. The setpoint/actual value difference is added to the control element. Value 1 must be assigned to **RegisterState** H382 before this can happen. The control element is called up in millisecond cycles and adds the value specified in the **RegisterLoopDXDTOut variable H390 to the SlaveTrim** variable H437 observing the signs until the control difference has been eliminated. If **RegisterModeControl.**2 = 0, the control difference is added to **SlaveTrim** H437. If **RegisterModeControl.2** = 1, the control difference is added to **XMasterPos** H438. Bit 0 in **RegisterModeControl** H381 decides whether the register loop controller remains enabled when state Z2 or Z3 is exited (transition to Z1 if bit 0 = 0) or if state Z0 should be entered (bit 0 = 1).

RegisterMode = 2 Only the control element is active. Calculation of the setpoint/actual value difference is disabled. In this case, it is possible to make a manual correction by writing the **RegisterLoopOut** variable H398.

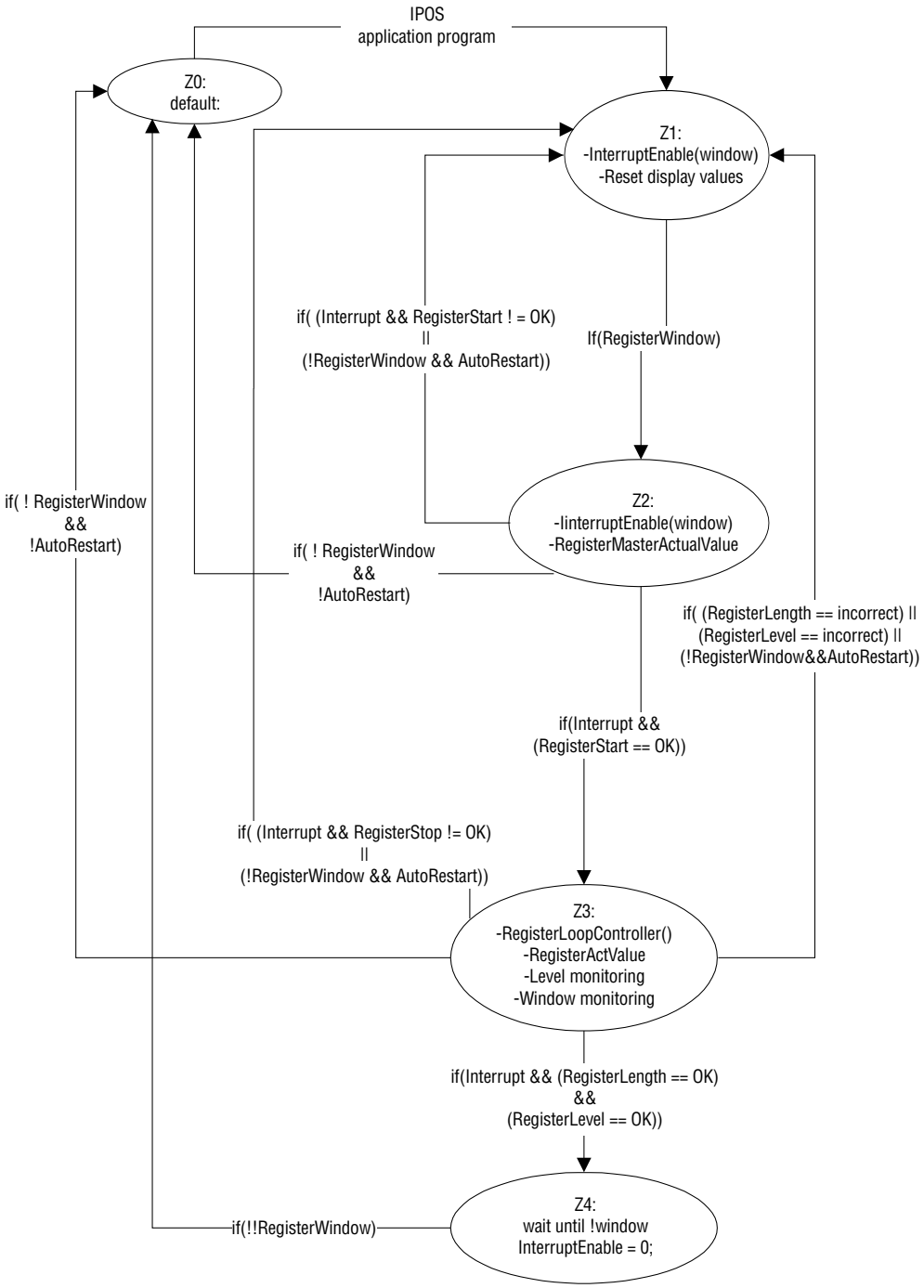


Figure 20: Register loop controller in mode 1

7 Virtual Encoder

7.1 Virtual encoder without ramp generator

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

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

7.2 Virtual encoder with ramp generator

The virtual encoder is a software counter (IPOS$^{plus®}$ variable) which can be used by the electronic cam software as a master encoder (assignment **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. For this purpose, it is necessary to have SBus synchronization with the sync ID P817 for synchronizing the units.

The virtual encoder operates in a 1 ms cycle and is processed independently of the current curve 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 when the **VEncoderXActual** value H374 reaches the **VEncoderXSetpoint** value H375. Wheter a deceleration ramp is used or not depends on H371 **VEncoderMode**.

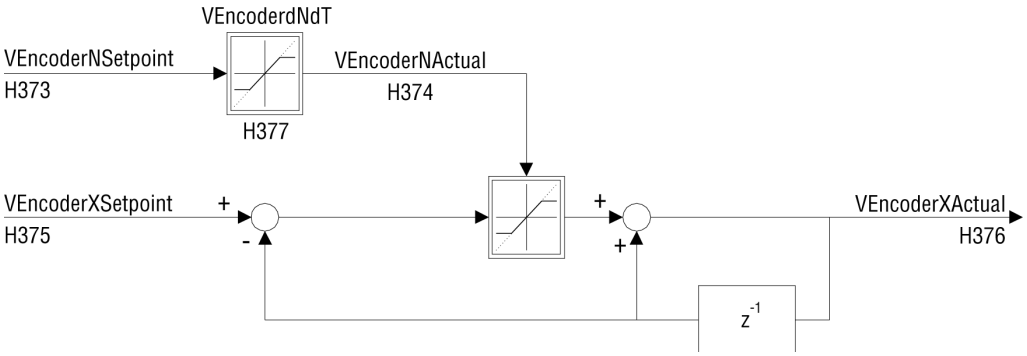


Figure 21: Virtual encoder with ramp generator

H375 = [VEncoderXSetpoint] = 1 "master pulse"
H376 = [VEncoderXActual] = 1 "master pulse"
H373 = [VEncoderNSetpoint] = 1 "master pulse"/ms
H374 = [VEncoderNActual] = 1 "master pulse" /ms
H377 = [VEncoderdNdT] = 1 "master pulse" /ms^2 in VEncoderMode = 2
1/2^{16} "master pulse" /ms^2 in VEncoderMode = 3

Selecting the operating mode of the virtual encoder in variable H370

- **VEncoderMode** = 0: Ramp generator with acceleration ramp. Ramp and speed can be set with
  - H373 **VEncoderNSetpoint** [1 inc/ms]
  - H377 **VEncoderdNdT** [1 inc/ms^2]

  There is no deceleration ramp in this operating mode. The counter stops once the target position has been reached. Values of the **VEncoderNSetpoint** variable H373 are always changed with the **VEncoderNdT** ramp H377.
- **VEncoderMode** = 1: Reserved
- **VEncoderMode** = 2: Endless counter with speed, can be set with
  - H373 **VEncoderNSetpoint** [1 inc/ms]

  Values of the **VEncoderNSetpoint** variable H373 are always changed with the **VEncoderNdT** ramp H377.
- **VEncoderMode** = 3: Ramp generator with acceleration and deceleration ramp. Ramp and speed can be set with VEncoderMode = 0. The scaling is different in this operating mode to allow for better resolution of ramp H377 VEncoderdNdT.
  - H373 **VEncoderdNdT** [1/2^{16} inc/ms^2]

  Values < 2^{16} are not suitable because such values will cause jerks.

VEncoderMode = 3 is possible with the following firmware:

MCx40/41/42A	firmware version xxx.14 and later
MDx60A	firmware version xxx.14 and later
MCH	all firmware versions

VEncoderMode = 3 is the preferred mode for positioning with virtual encoder because this is the only mode with deceleration ramp.

VEncoderModeControl.0 (H371.0 == 1) "FaultStopFunction" of the virtual encoder

The value of VEncoderNSetpoint is set to 0 once after a unit fault. This causes the virtual axis to stop (in modes 0, 2, and 3).

8 Scaling the Curve Point Table

The **TableScaleControlState** parameter H448 allows for altering the scale of the curve point table stored in the EEPROM (startup cycle curve + main curve or only main curve points / index range 20000 ... 20513). The scaling factors are stored as numerator / denominator factors in the **TableScaleNominator** variable H449, value range $-(2^{15}-1)$ ... 0 ... $+(2^{15}-1)$ and **TableScaleDenominator** H450, value range 1 ... $+(2^{31}-1)$. The scaled values can be uploaded from the RAM.

The **TableScaleControlState** variable is used as a control and state variable for converting the curve and is coded as follows:

- **TableScaleControlState** = 0: No scaling or scaling is canceled if already started (e.g. denominator == 0).

  High word (bits 16 ... 31) specifies the curve you want to scale.

  Example: TableScaleControlState = 1 | 3 << 16;

  This command in IPOS$^{plus®}$ starts the scaling of curve 4. Please observe the number of available curves mentioned in the "Startup / Generating a new curve" section.
- **TableScaleControlState** = 1: Start curve scaling (control); always the entire data range is scaled.
- **TableScaleControlState** = 2: Scaling now in progress (state message).
- **TableScaleControlState** = 3: Scaling completed successfully (state message).

The scaling function is calculated with a 48-bit interim result. It is important, however, that the result data are in the 32-bit signed range (H440 **ActPosition-Scale**). This is neither detected nor limited by the "conversion." Curve scaling takes about 50 ms computation time.

Scaling the control point table always results in a change of slave direction with the same number of master increments. But scaling can also take place in direction of the master by means of **MasterCycleScale** H432. This means the slave performs its slave cycle with a different number of master increments.

The **MasterNorm** variable H432 must be taken into account for this purpose. See Sec. "Curve variables."
Scaling takes place in the volatile memory only. The original curve is reloaded after a fault reset.

9 Multi-Cam Operation

Changing curves in disengaged state (CamState = 0)

Please refer to the "Startup /Generating a new curve" section for the number of available curves.

You can select the required curves via the **TableSelect.0** variable H451.0.

Variable: **TableSelect** H451:

- 0 = curve "0" selected
- 1 = curve "1" selected
- 2 = curve "2" selected
- ...

You can influence the variable in the IPOS$^{plus®}$ application program.

Curves can be changed in disengaged state without further adaptation (**CamState** H436 = 0).

Changing curves in engaged state (CamState = 2)

Change-over between two different curves in engaged state may cause a setpoint step change as shown in the following example.

Example

The relative slave positions of the two curves differ in the change-over point.

The setpoint position of the slave changes quickly when changing over from curve 1 to curve 2. If the drive was now enabled with active curve control, the control would attempt to compensate the deviation between setpoint and actual value. To avoid this, the deviation of setpoint position and actual position must be entered in the **SlaveTrim** variable H437. The value for **SlaveTrim** H437 must be calculated for this purpose. In this case you also have to disable the filter elements that are active for setpoint processing. This step is necessary because some filter elements (in particular P203) cause a lag error or jerk when the drive is enabled immediately afterwards. Setting **CamModeControl.1** H429.1 to "1" starts initialization. After the reset, bit H429.1 automatically changes to state "0." This means immediate enable is possible without wait commands.

In addition, the corresponding **setCAMParameter()** function must be called each time before changing over to another curve in order to alter the curve variable accordingly.

The following programming example shows a change-over between the two curves via DI02 as well as the calculation of **SlaveTrim**H437:

```
/*--------------------------------------
          Main program loop
--------------------------------------*/
while(1)
  {
  // Inverter in "No enable" state during change-over
  if(DI02 == 0 && Flag_2==0)         // Change-over to curve 0 selected
    {
     setCAMParameter();
    TableSelect = 0;                 // Change-over to curve 0
    _BitSet( CamModeControl, 1 );    // Filter reset
     // Adjustment of setpoint and actual position values
    SlaveTrim += (ActPos_Mot << ActPositionScale)-XSlaveSetpointScale;
     _BitSet( CamModeControl, 1 );   // Filter reset
    Flag_2 = 1;
    Flag_1 = 0;
    }
  // Inverter in "No enable" state during change-over
    if(DI02 && Flag_1==0) // Change-over to curve 1 selected
    {
     setCAMParameter2();
    TableSelect = 1;                 // Change-over to curve 1
    _BitSet( CamModeControl, 1 );    // Filter reset
    // Adjustment of setpoint and actual position values
    SlaveTrim += (ActPos_Mot << ActPositionScale)-XSlaveSetpointScale;
    _BitSet( CamModeControl, 1 );    // Filter reset
    Flag_1 = 1;
    Flag_2 = 0;
  }
```

10 Startup

10.1 General information

Correct project planning and installation are the prerequisites for successful startup. For detailed project planning instructions, refer to the MOVIDRIVE® system manual, which forms part of the MOVIDRIVE® or MOVIDRIVE® *compact* documentation package.

Check the encoder connection and installation of option cards by following the installation instructions in the system manual.

If you use an absolute encoder as external encoder (connection to DIP11A X62; not MOVIDRIVE® *compact*), please also refer to the installation and startup instructions in the "Positioning with Absolute Encoder and Absolute Encoder Interface DIP11A" manual.

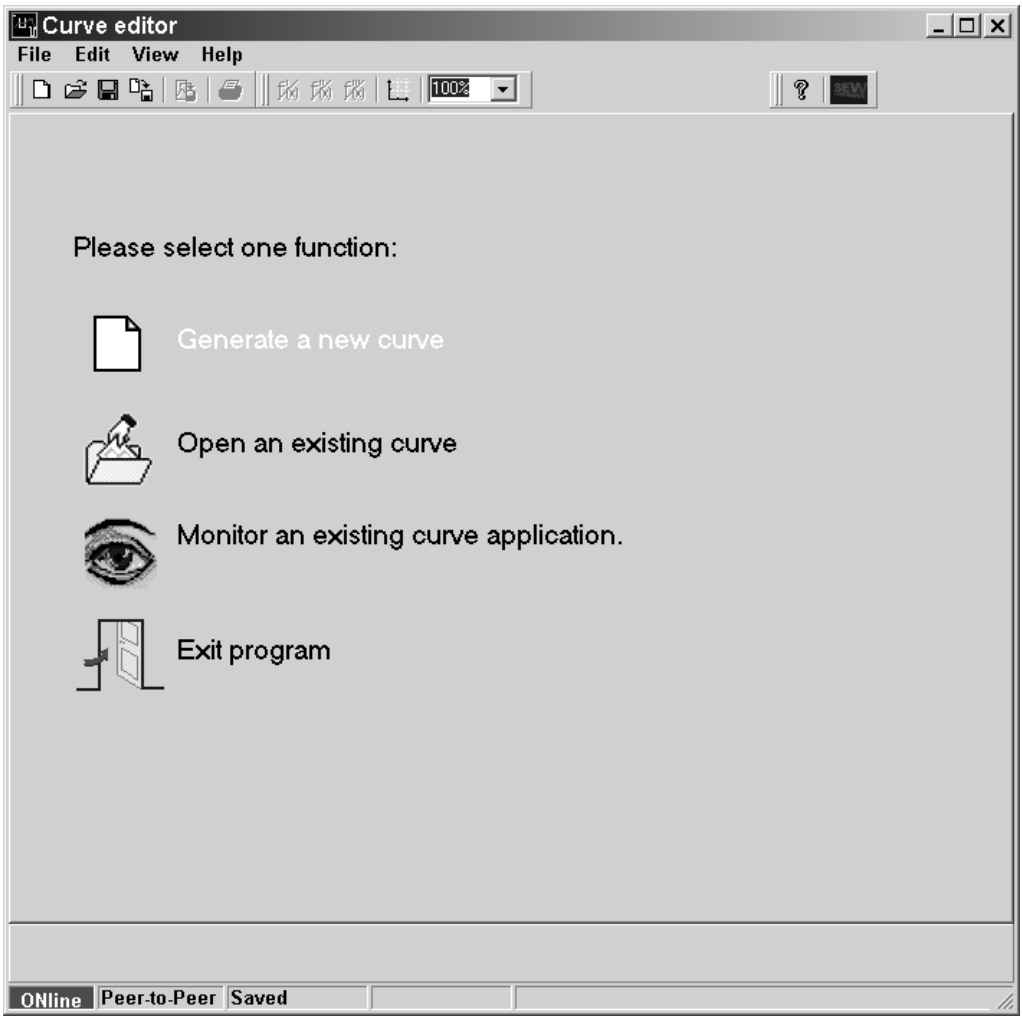
10.2 Preliminary work

Perform the following steps before startup of the electronic cam:

- Connect the inverter to the PC using the serial port (RS-232, USS21A on PC-COM).
- Start MOVITOOLS®. Start up the inverter using <Shell>. Asynchronous machines must be started up in CFC & IPOS operating mode, and synchronous machines in the servo & IPOS operating mode.
- "0" signal at terminal X13:1 (DIØØ, /Controller inhibit). From the Startup menu, select the technology function electronic cam.
- Set parameter P916 "Ramp type" to "Electronic cam." The parameter can also be set using the IPOS$^{plus®}$ command **_SetSys(SS_RAMTYPE,Hxx)**. In this case, variable Hxx must be set to the value 5.

10.3 Generating a new curve

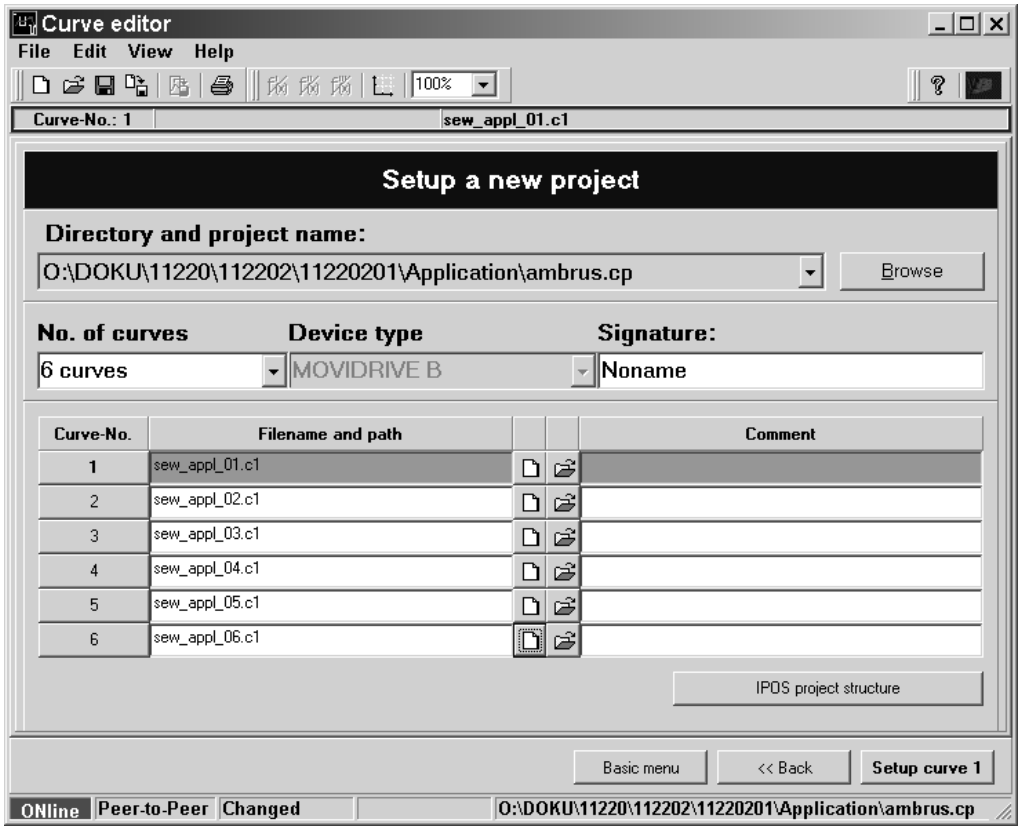
In the MOVITOOLS® manager, click <CAM>. The startup screen opens.



10730AEN

Figure 22: Startup screen of the curve editor

To create a new project, click “Generate a new curve“. The next window opens.



10731AEN

Figure 23: Curve editor

Enter project name and project path, number of curves, signature as well as file name and file path.

Six different curves are only possible when the following unit requirements are met:
- MOVIDRIVE® *compact* MCH and MDX60/61B from firmware version 15 and later. This means only motors with Hiperface® encoder, sin/cos encoder and TTL sensor.
- MOVITOOLS® version 3.0 and later

For **all other MOVIDRIVE® motor combinations, you can only select one curve.** The curves have different designations in the user interface and in the later IPOS code. Curve 1 in the user interface = curve 0 in IPOS.

The following units allow for storing two curves in the MOVIDRIVE® by using a copy function:

MCX40/41A	Firmware version xxx.12 and later
MCH / MDX60/61B	All firmware versions

Functional description

Downloading and copying curves

Curves are downloaded using the curve editor. When doing so, the curve is always transferred to the non-volatile memory. After transfer of the first curve, you must copy the loaded curve into the second memory area (volatile memory), which can neither be accessed externally nor via IPOS$^{plus®}$. Next, the second curve can be transferred using the curve editor. The second curve is then stored in the non-volatile memory.

The curve is copied by setting bit 0 of the **TableCopy** variable H452 to "1", for example via IPOS$^{plus®}$. After completed copying process, this bit changes from state "1" back to state "0." For selecting the various curves, please refer to the multi-curve operation section.

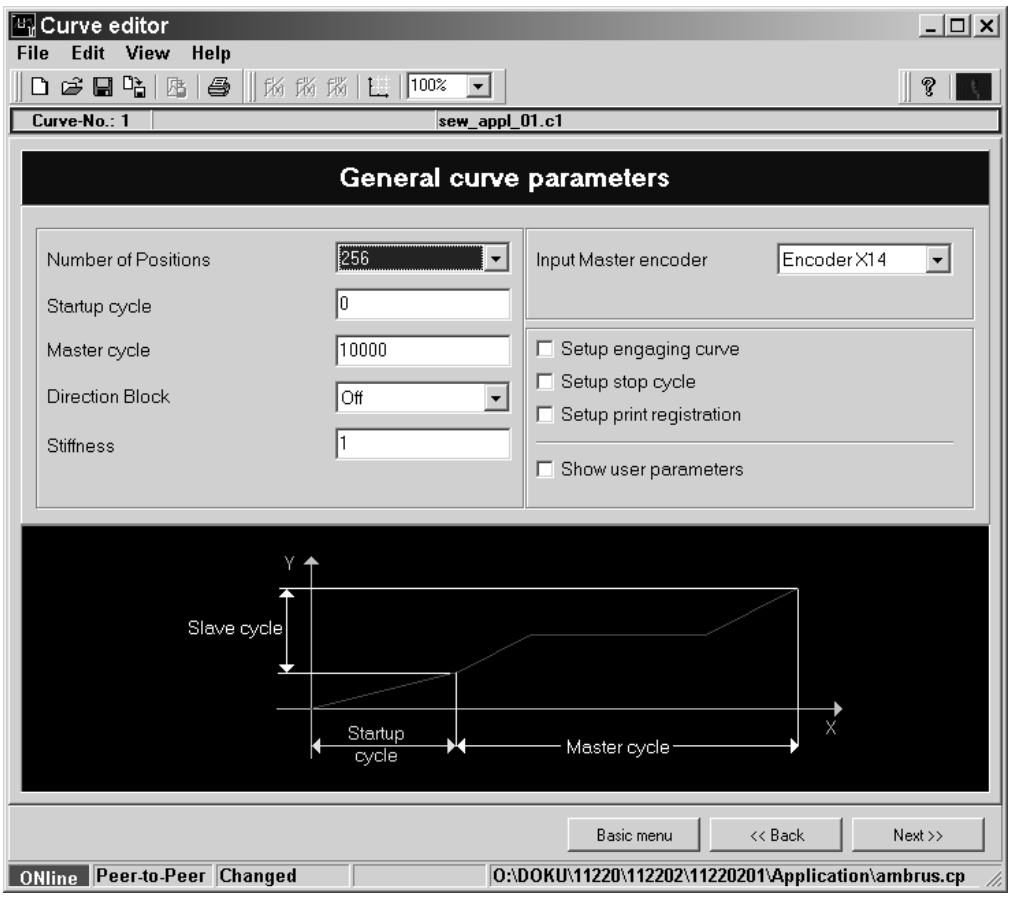
Variable: **TableCopy.0** H452.0:	0 = No copying or requested copying completed successfully
	1 = Copying requested (curve set 0 -> curve set 1)

Make sure the electronic cam is in disengaged state (**CamState** H436 = 0) when downloading or copying a curve.

The copy function can also be used with MCH and MOVIDRIVE B but will not be necessary anymore in 6-cam operation. Curve 1 is overwritten.

Setup curve 1

Clicking "Setup curve 1" opens a window for entering startup data. You are prompted to close all open compiler windows.



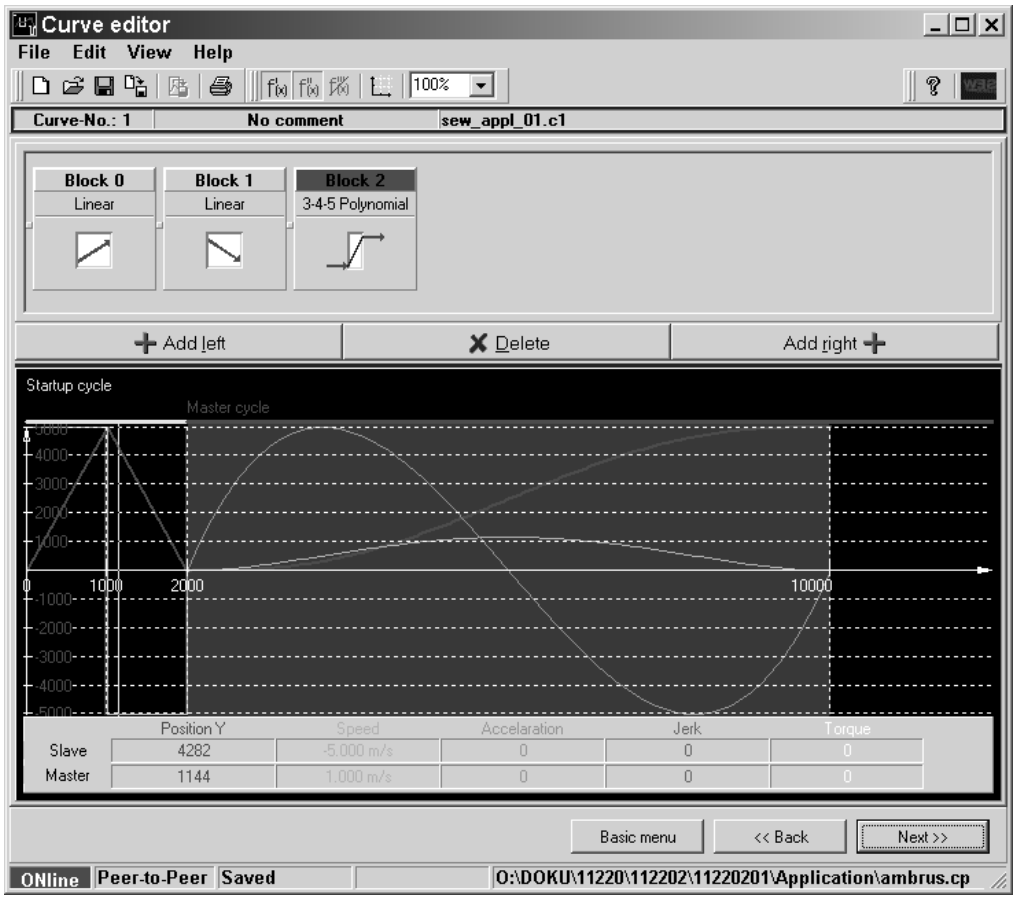
10732AEN

Figure 24: Entering startup data

Enter the following data:

- **Number of positions (curve points)**: The number of curve points refers to the main curve. Without startup cycle, the main curve can have 256 or 512 curve points. If a startup cycle curve was defined (startup cycle ≠ 0), the number of curve points of the main curve is always 256.
- **Startup cycle**: Entering "0" means there is no startup cycle. If you require a startup cycle, you have to enter the startup cycle distance in increments [inc] in relation to the master distance (H433 **StartupCycleScale**). The startup cycle must always be smaller than the master cycle.
- **MasterCycleScale** master cycle H432: Enter the number of increments [inc] in relation to the master distance that define a master cycle. The slave has run through exactly one slave cycle when the master has covered this number of increments from the start or end of the startup cycle.
- **ReSprintClose** direction stop H444: Entering "Off" means there is no direction block. You can enter "Lock right" or "Lock left" to prevent the slave from moving opposite to its defined movement direction when the master alters the counting direction.
- **NFilterTime**, H446 **MFilterTime** stiffness setpoint H443: You can use this setting to influence the control response of the electronic cam control. The setting range is 0.5 ... 2. The standard setting is 1. Increasing the stiffness causes the control speed to increase. The control loop starts to become unstable above a critical value. The control response is slower when the stiffness is reduced. The lag error increases. **Note**: Increase the stiffness in small increments (e.g. 0.05)!
- Master encoder: You have to enter the source of the master encoder in this field. Four choices are available:
  - **Encoder X14**; the encoder connected to X14 is used as the master encoder (H430 **MasterSource** = 0). A virtual encoder is added to encoder input X14 with **MasterTrimX14** H442. This way, positioning can be simulated, for instance during the startup phase when X14 does not yet receive master encoder pulses.
  - **SSI encoder** (H430 **MasterSource** = 509).
  - **Variable**; an IPOS$^{plus®}$ variable is used as the master encoder. At the same time, a selection window is activated for selecting the variable number (H430 **MasterSource** = variable number).
  - **Virtual encoder with ramp generator**, the IPOS$^{plus®}$ variable H376 is used as master encoder (H430 **MasterSource** = 376).
- **Show user parameters**: The program uses the entries to calculate all operating and initialization variables for the electronic cam application. These variables can be verified again before all data are downloaded to the inverter. "Show user parameters" must be selected for this purpose.

- Clicking "**Next >>**" opens the window where the curve is created.

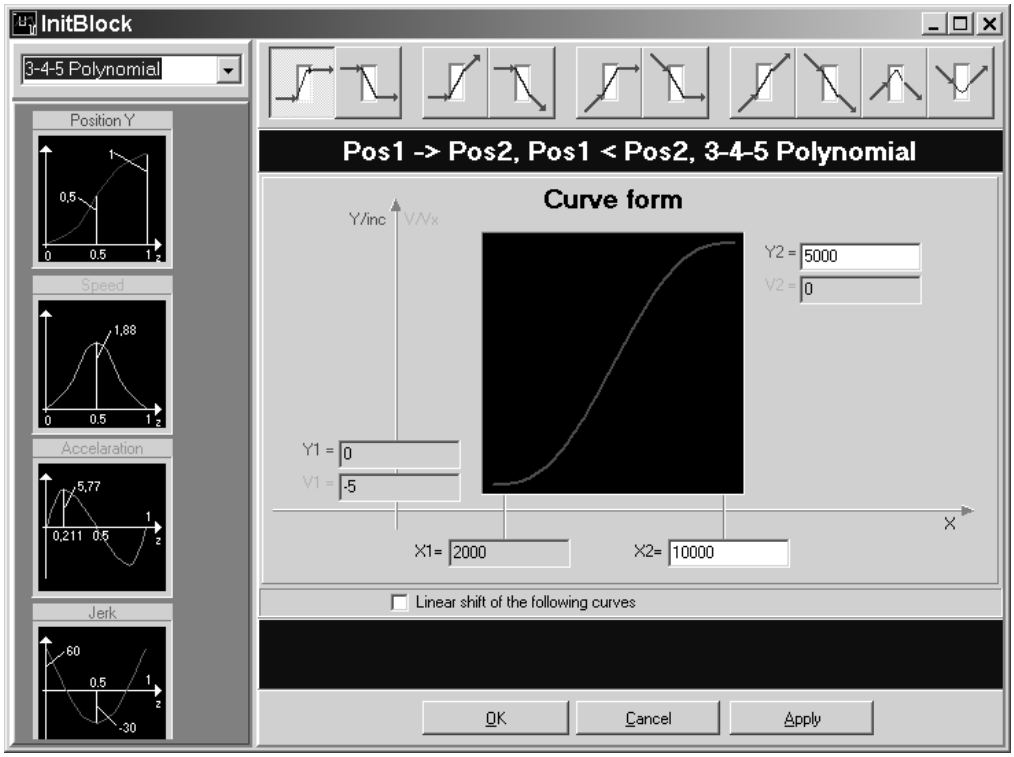


10733AEN

Figure 25: Creating the curve

You can create the curve from several blocks using **"Add right"** or **"Add left"**. You can assign one of the following motion profiles to each block:

- Linear
- Sine
- 3-4-5 polynomial
- Parabola
- Modified sine
- Accelerating trapezoid
- User defined, existing curve data are imported (see "Importing a curve)
- Spline



10734AEN

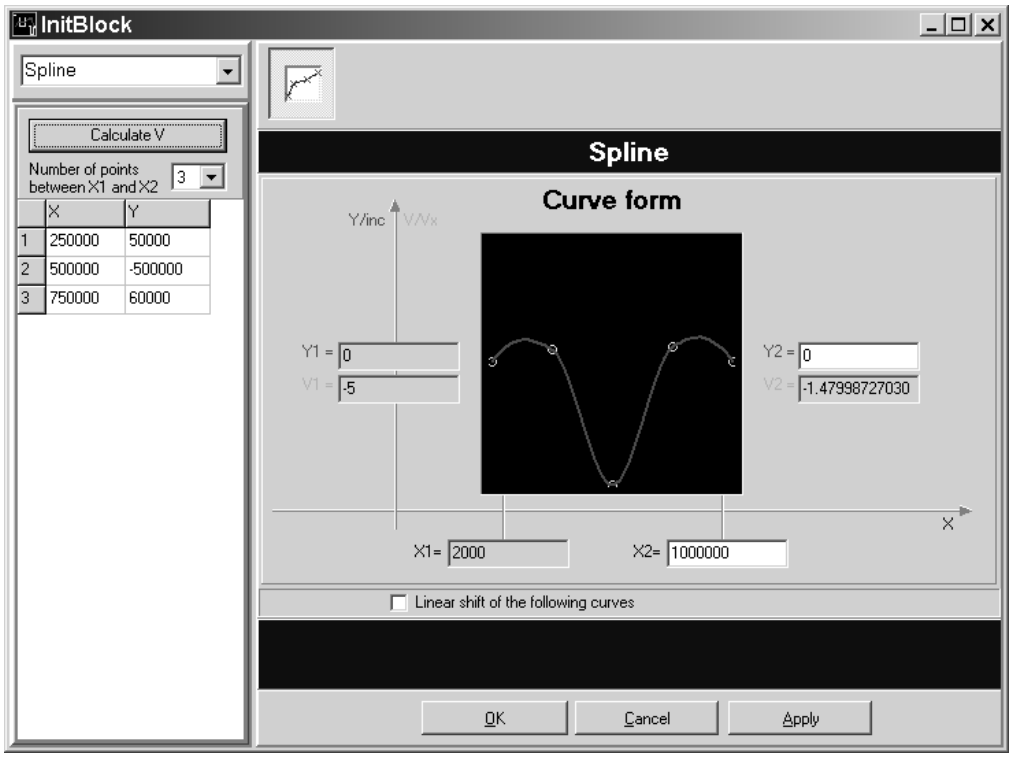
Figure 26: Creating segments of the curve

All curve blocks are connected to create the curve. Speed, acceleration and jerk can be represented using the switches for the 1st, 2nd and 3rd derivation.

The total number of master increments must correspond with the **MasterCycleScale** system variable H433 and, if necessary, in addition with **StartupCycleScale** variable H433. The startup cycle curve is entered in the same window.

Spline

Entering "Spline" allows for defining a curve with just a few control points. The intermediate values are calculated using spline interpolation.



10735AEN

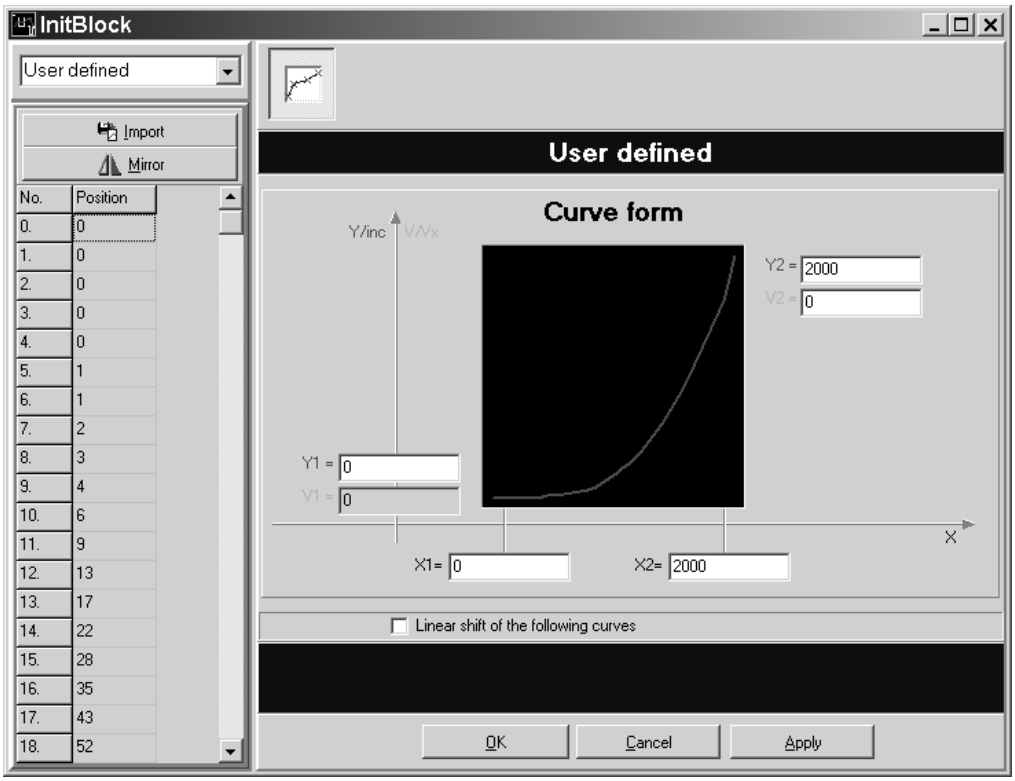
Figure 27: Spline

1. Determine the number of control points between X1 and X2. It is important that you define the first and last point in the right side of the window.
2. Enter X values and the corresponding Y values in the left-hand table.
3. Enter the first and last point in the right side of the window.
4. Click "Apply" to calculate and display the curve.
5. If necessary, you can adjust the speed of the first and last point by clicking "Calculate V". Two points will then be added each at the begin and end of the list.

Unreasonable speed values may easily cause the curve to exceed the value range.

Importing a curve

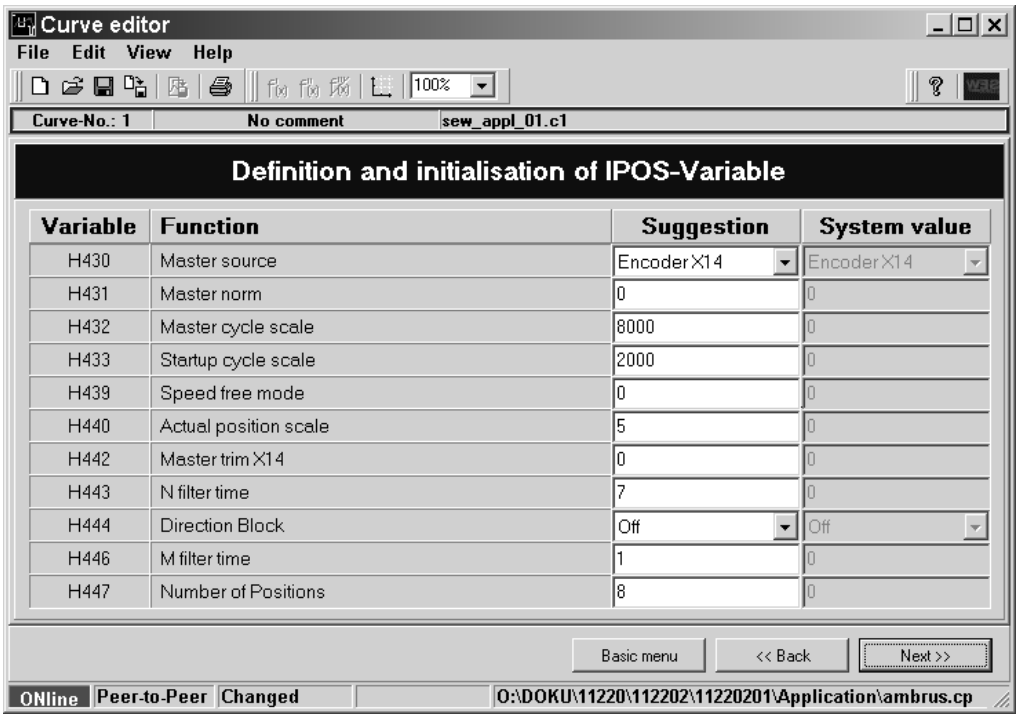
A slave curve that was not created using the curve editor can be imported into the curve editor. To do so, select "User defined" in the curve editor. Next, click the Import button to select the corresponding file (*.txt, *.prn). The file must contain 512 integer values in one column (curve points). The individual curve points are then displayed in ascending order. You also have to enter the values for the end points X2 and Y2 that are used for scaling the curve.



10736AEN

Figure 28: Importing a curve

When you have completed the curve, click "Next.>>" The window with the user parameters appears if you have selected it in the "General curve parameters" window.



10737AEN

Figure 29: User parameters

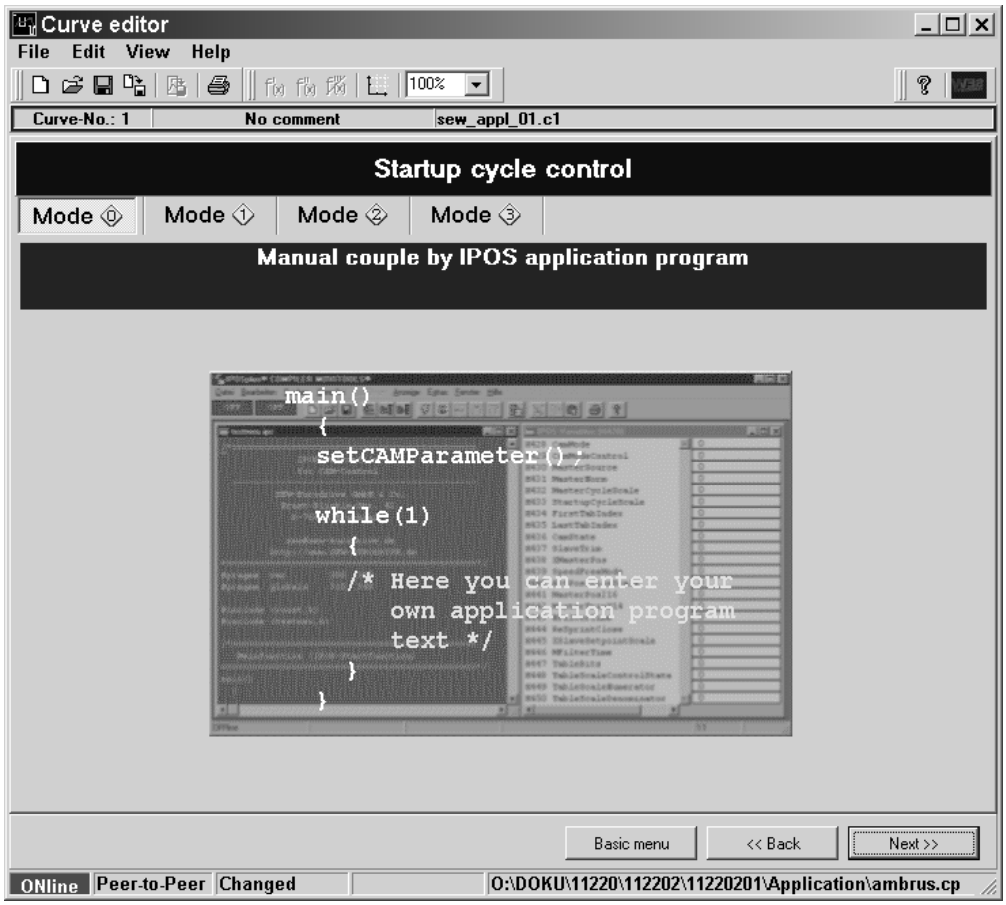
If necessary, you can make alterations in this window.

Clicking "Next>>" opens the Startup cycle control window if you have selected it in the "General curve parameters" window.

10.4 Startup cycle control

With startup cycle control, you can choose between four different modes:

Mode 0



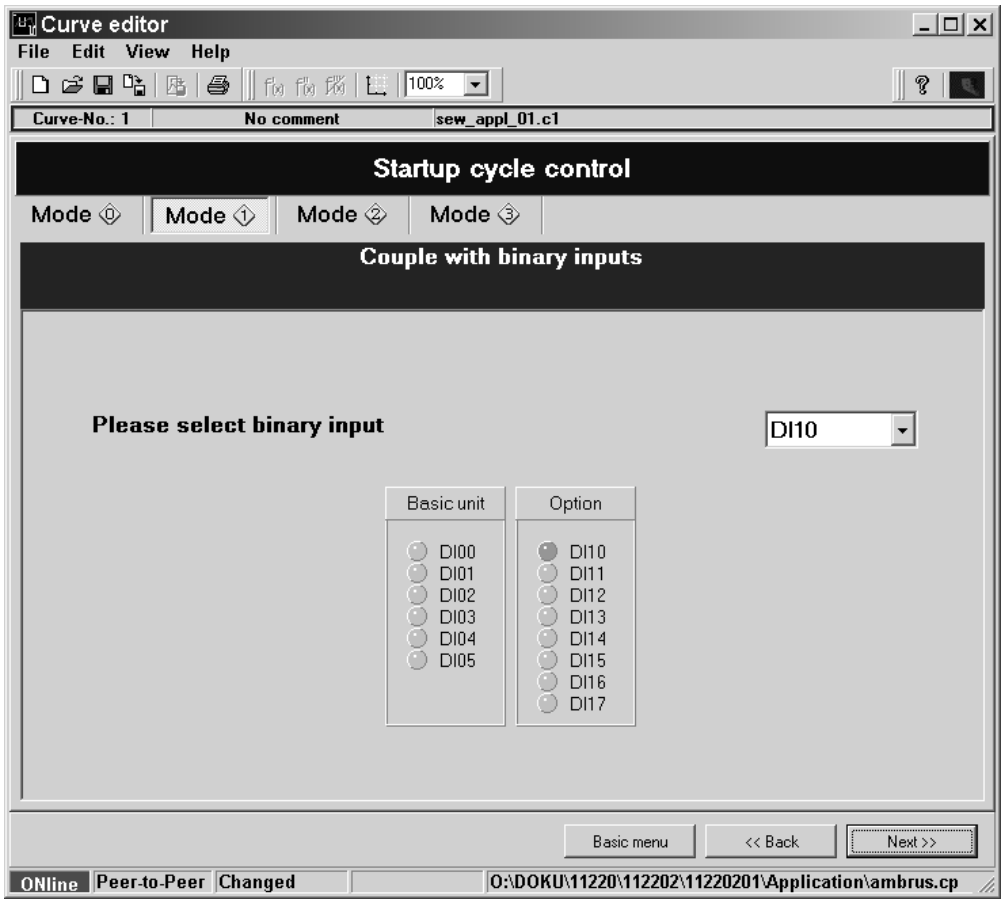
10738AEN

Figure 30: Startup cycle control mode 0

Mode 0 Manual engaging (H410 **StartupCycleMode** = 0)

No startup cycle event is being prepared via the startup interface. The user is prompted to trigger the startup cycle by writing the value 1 to the **CamState** variable H436 in the IPOS$^{plus®}$ program to be generated.

Mode 1



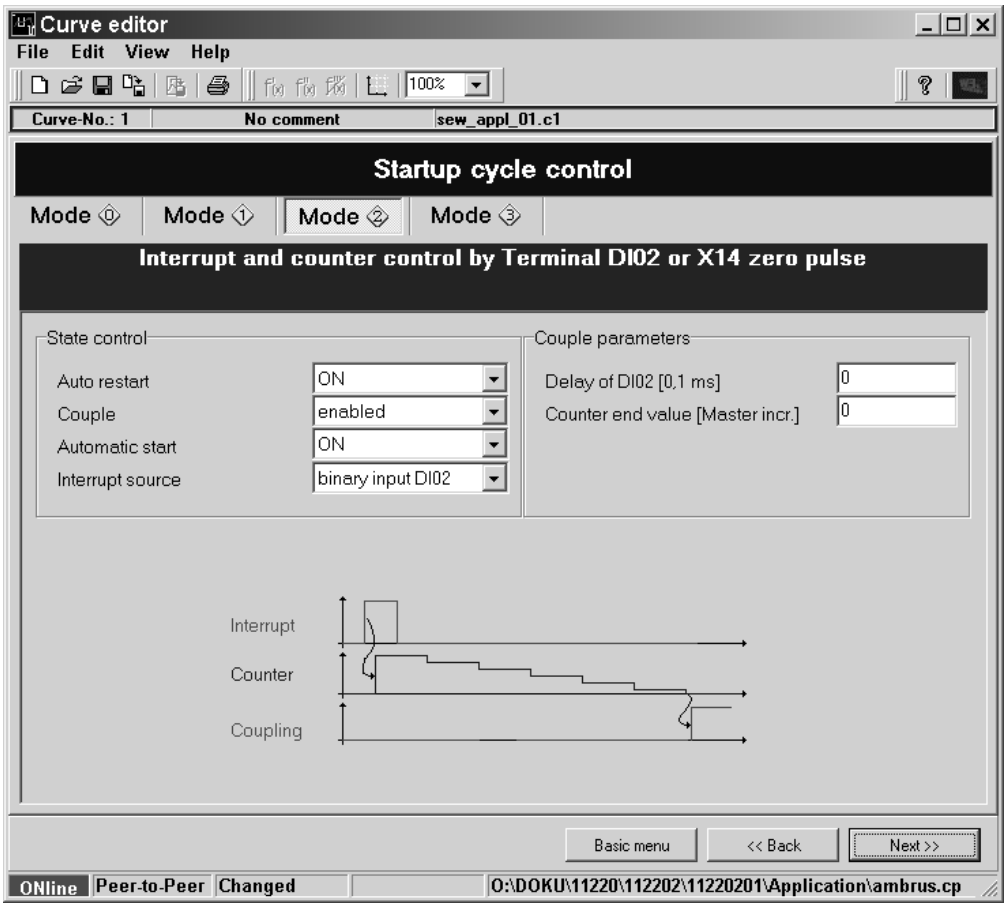
10739AEN

Figure 31: Startup cycle control mode 1

Mode 1 Engaging with binary input (H410 **StartupCycleMode** = 1)

You can select the binary input that triggers the startup cycle process (H413 **StartupCycleInputMask**).

Mode 2



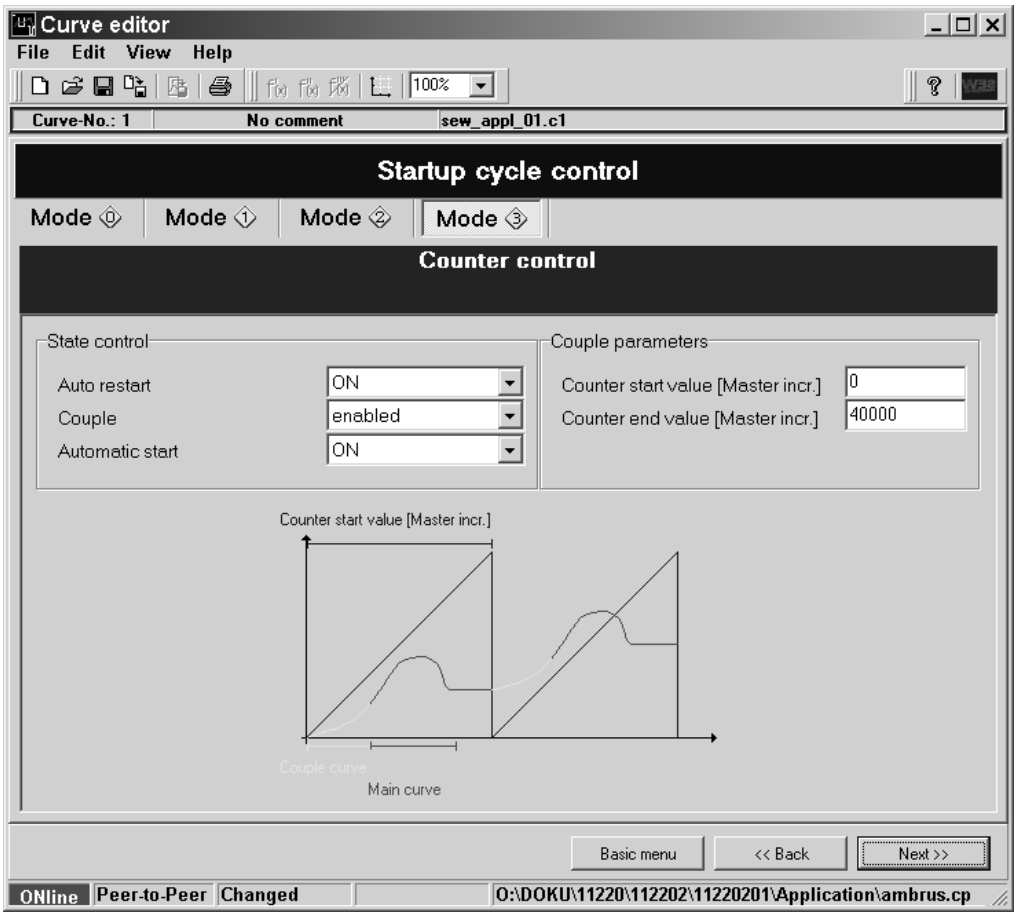
10740AEN

Figure 32: Startup cycle control mode 2

Mode 2 Engaging via interrupt (H410 **StartupCycleMode** = 2)

Auto restart	**OFF** -> Interrupt must be enabled manually after each cycle by assigning H412 = 1 in the application (H411 **StartupCycleModeControl.0** = 0). **ON** -> Interrupt is activated automatically after each cycle (H411 **StartupCycleModeControl.0** = 1).
Couple	**Enabled** -> (H411 **StartupCycleMode.1** = 0) Engaging enabled. **Disabled** -> (H411 **StartupCycleModeControl.1** = 1).
Automatic start	**ON** -> Enabled automatically. Only affects first start after initialization (H412 **StartupCycleState** = 1). **OFF** -> H412 must be enabled manually the first time in the application by assigning “1“.
Interrupt source	**Binary input DI02** -> A signal edge on binary input DIØ2 is used for engaging (H411 **StartupCycleModeControl.2** = 0). **X14 C track** -> X14 zero pulse is used for engaging (H411 **StartupCycleModeControl.2** = 1).
Delay of DI02	Delay in ms of the sensor. This value is compensated during the startup cycle process, i.e. a possible offset caused by the sensor delay is taken into account (H416 **StartupCycleDelayDI02**).
Counter end value [master inc.]	Offset that delays the startup cycle process with reference to the length (H415 **StartupCycleCounterMaxValue**).

Mode 3



10741AEN

Figure 33: Startup cycle control mode 3

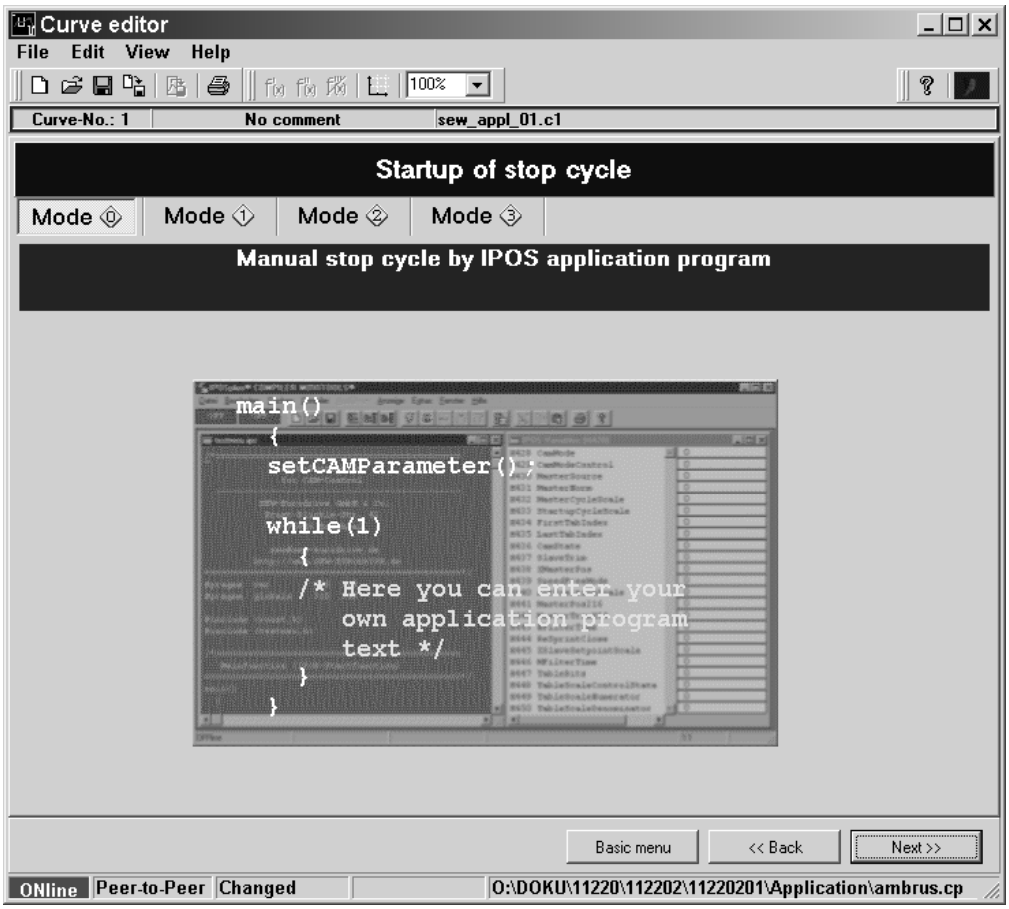
Mode 3 counter control (H410 **StartupCycleMode** = 3)

Auto restart	**OFF** -> Interrupt must be enabled manually in the application after each cycle by assigning H412 = 1 (H411 **StartupCycleModeControl**.0 = 0). **ON** -> Interrupt is activated automatically after each cycle (H411 **StartupCycleMode**Control.0 = 1).
Couple	**Enabled** -> (H411 **StartupCycleModeControl.1** = 0) Engaging enabled. **Disabled** -> (H411 **StartupCycleModeControl.1** = 1).
Automatic start	**ON** -> Automatically enabled. Only affects first start after initialization (H412 **StartupCycleState** = 1). **OFF** -> H412 must be enabled in the application the first time by assigning “1“.
Counter start value	Describes the value to which the startup cycle counter is initialized (H414 **StartupCycleCounter**).
Counter end value	Describes the distance when the next startup cycle process is started automatically (counter-controlled) (H415 **StartupCycleCounterMaxValue**).

Click **"Next>>"** to open the Startup of stop cycle window if selected in the "General curve parameters" window.

10.5 Stop cycle mode control

Mode 0



10742AEN

Figure 34: Startup of stop cycle mode 0

Mode 0 Manual disengaging

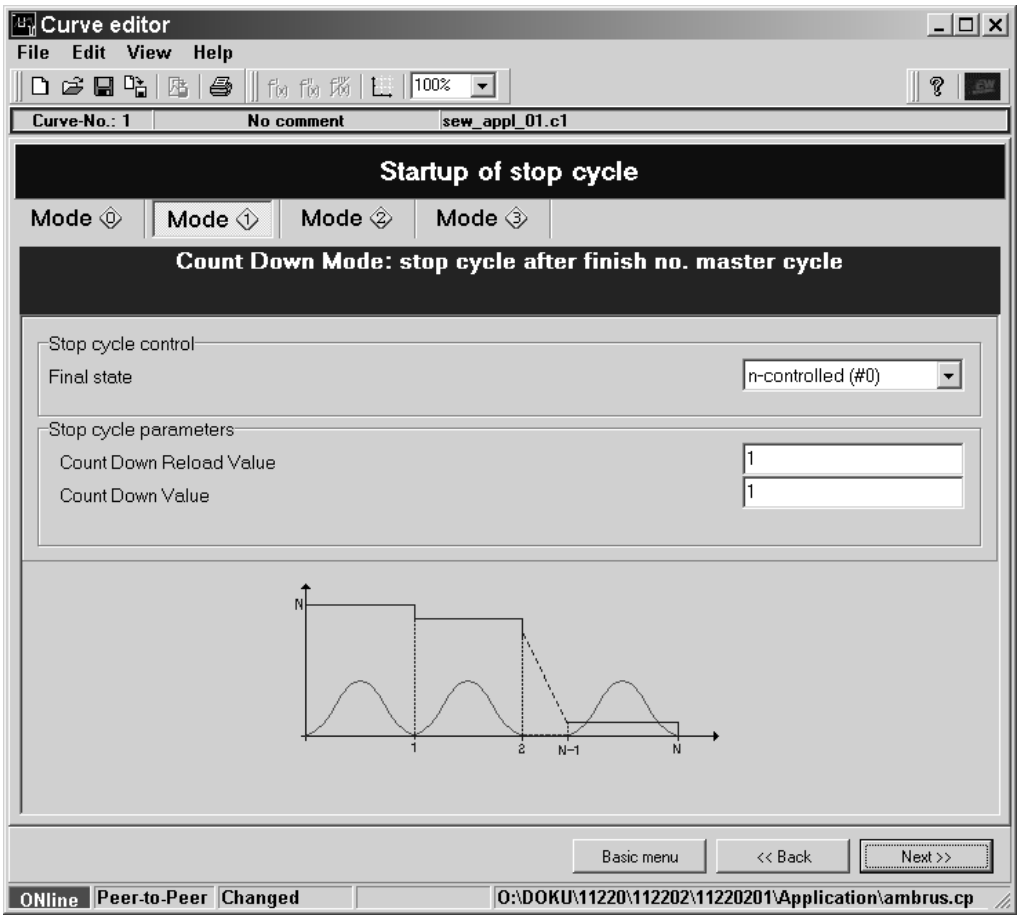
No stop cycle event is being prepared via the startup interface. The user is prompted to trigger the stop cycle process in the later IPOS$^{plus®}$ program by writing the value 0 to the **CamState** variable H436 (H400 **StopCycleMode** = 0).

You can, for example, change over to positioning mode (ramp type linear) "on the fly" in order to move the slave to a defined position.

Do not directly write 3 or 4 to CamState H436!

Stop cycle mode control allows for choosing from four different modes described in the following.

Mode 1



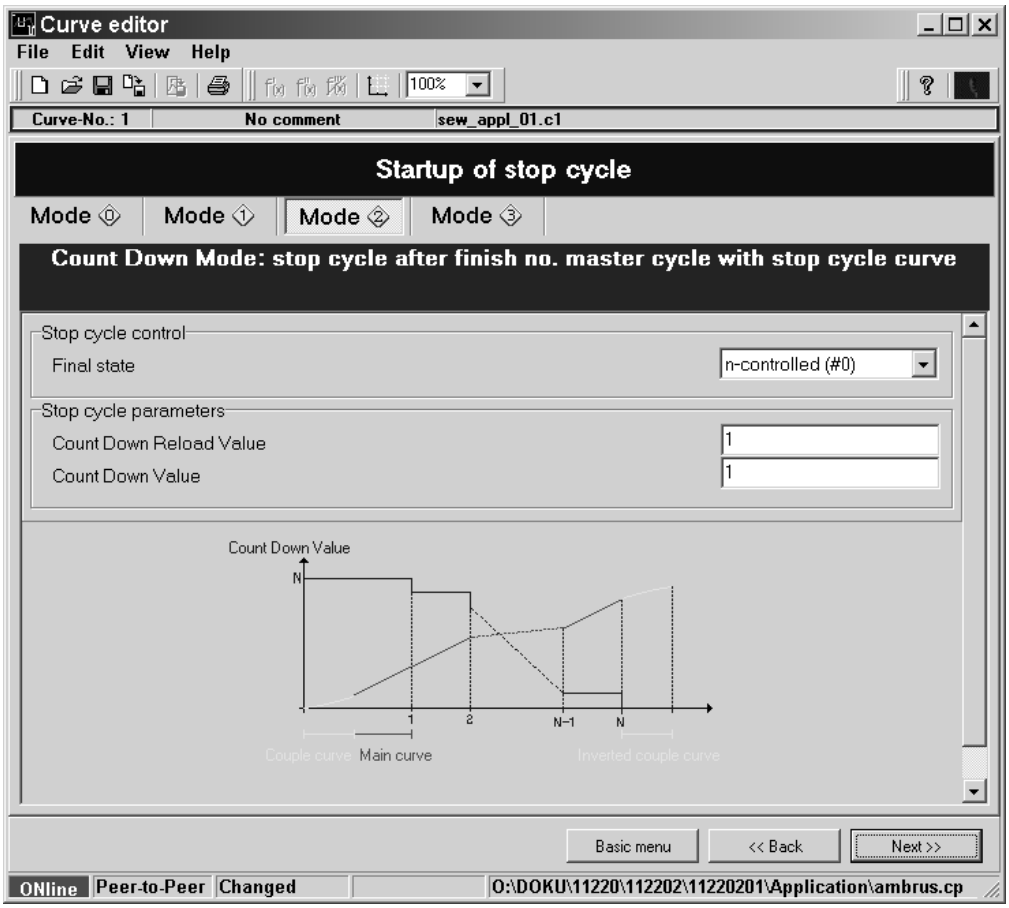
10743AEN

Figure 35: Startup of stop cycle mode 1

Mode 1 Stop cycle after finish no. master cycle (disengaging after n runs) (H400 **StopCycleMode** = 1)

Final state	**x controlled** -> (H401 **StopCycleModeControl.0** = 1). **n controlled** -> (H401 **StopCycleModeControl.0** = 0).
Countdown reload value	Reload value after the first cycle (H403 **StopCycleReload**).
Countdown value	Initialization value (H404 **StopCycleCounter**).

Mode 2



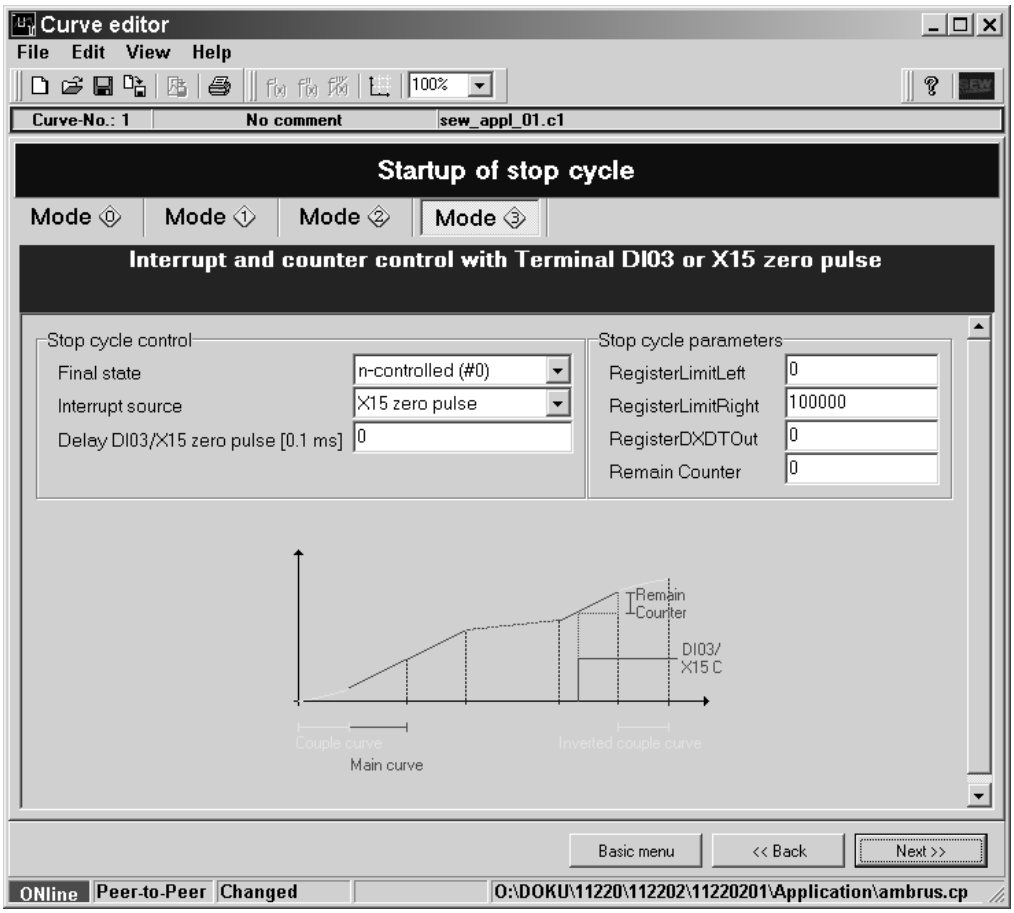
10744AEN

Figure 36: Startup of stop cycle mode 2

Mode 2 Stop cycle after finish no. master cycle (disengaging after n runs with stop cycle curve) (H400 **StopCycleMode** = 2).

Final state	**x controlled** -> (H401 **StopCycleModeControl.0** = 1). **n controlled** -> (H401 **StopCycleModeConntrol.0** = 0).
Countdown reload value	Reload value after the first cycle (H403 **StopCycleReload**).
Countdown value	Initialization value (H404 **StopCycleCounter**).

Mode 3



10745AEN

Figure 37: Startup of stop cycle mode 3

Mode 3 Interrupt and counter control with stop cycle curve (interrupt and position controlled disengaging with stop cycle curve) (H400 **StopCycleMode** = 3)

Final state	**x-controlled** -> (H401 **StopCycleModeControl.0** = 1). **n-controlled** -> (H411 **StopCycleModeControl.0** = 0).
Interrupt source	**Binary input DI02** -> A signal edge on binary input DIØ2 is used for disengaging (H401 **StopCycleModeControl.1** = 0). **X15 C track** -> X15 zero pulse is used for engaging (H401 **StopCycleModeControl.1** = 1).
Delay of DI03	Delay in ms of the sensor. This value is compensated during the stop cycle process. A possible offset caused by the sensor delay is taken into account (H406 **StopCycleDelayDI03**). Unit [0.1 ms].
RegisterLimitLeft	Definition of left-hand limit within which a valid interrupt must be detected (H383 **RegisterLimitLeft**).
RegisterLimitRight	Definition of right-hand limit within which a valid interrupt must be detected (H383 **RegisterLimitRight**).
RegisterDXDTOut	You have to specify the control range per ms (H390 **RegisterLimitDXDTOut**) in order to compensate for the delay DI03 (H406 **StopCycleDelayDI03**) and because of the relatively high speed during the stop cycle process compared to the start cycle process.
RemainCounter	Distance that the slave covers after the interrupt event until disengaging (H405 **StopCycleRemainWay**).

Click "Next >>" to open the Startup register loop control window, if selected in the "General curve parameters" window.

10.6 Register loop control

Register loop control distinguishes between three different modes described in this section.

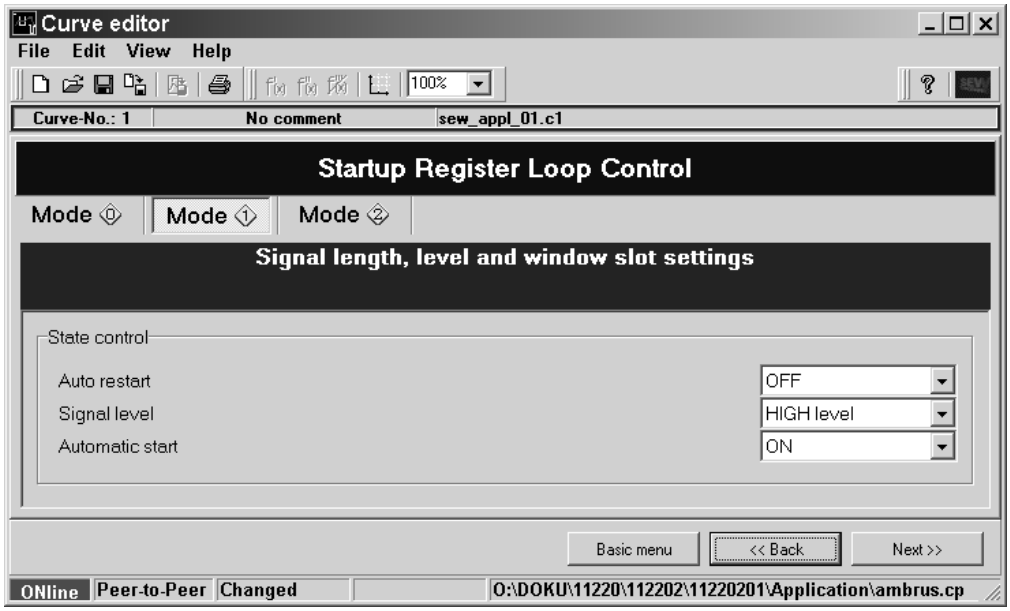
Mode 0

10746AEN

Figure 38: Startup register loop control mode 0

Mode 0 Register loop control off (H380 **RegisterMode**=0)

Mode 1



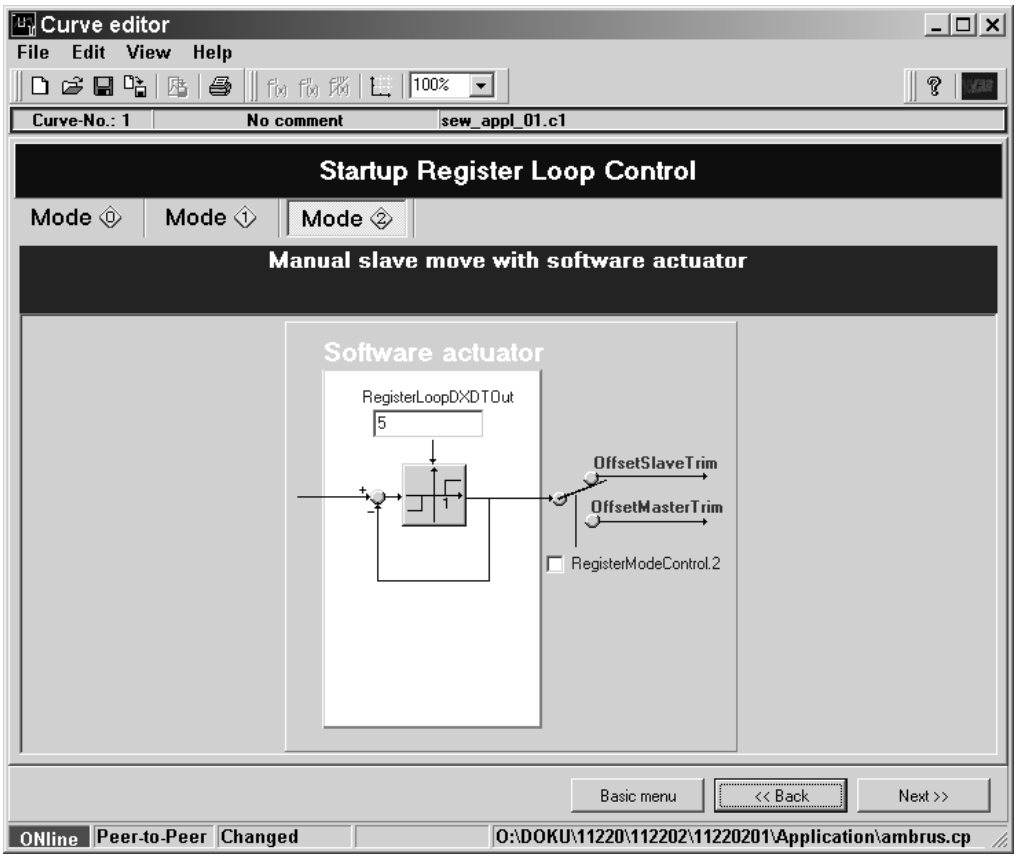
10747AEN

Figure 39: Startup register loop control mode1

Mode 1 Signal length, level and window slot settings (H380 **RegisterMode**=1)

Auto restart	**ON** -> causes restart to be enabled automatically after a valid register. **OFF** -> Value 1 must be assigned to H382 **RegisterState** (H381 **RegisterMode-Control.0**).
Signal level	**HIGH level** -> **RegisterModeControl.1** H381.1 = 1 rising edge is detected. **LOW level** -> **RegisterModeControl.1** H381.1 = 0 falling edge is detected.
Automatic start	**ON** -> Enabled automatically without manual assignment. This only affects the first start after the initialization (H382 **RegisterState** = 1). **OFF** -> H382 must be enabled manually in the application at the first time by assigning the value 1 to H382 **RegisterState**.

Mode 2



10748AEN

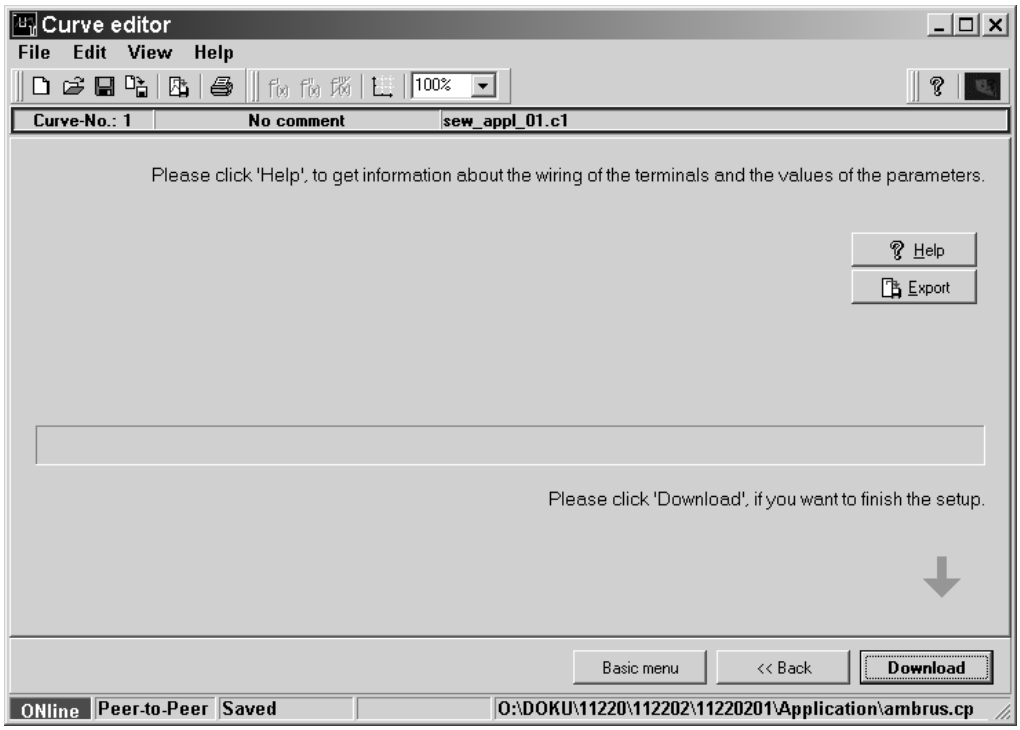
Figure 40: Startup register loop control mode 2

Mode 2 Manual slave correction with control element (H380 **RegisterMode** = 2)

Only the control element is enabled. Calculation of the setpoint/actual value difference variable is disabled. In this case you can perform corrections by writing the **RegisterLoopOut** variable H389.

RegisterModeControl.2	Control element either acts on the master or on the slave (H389 **RegisterModeControl.2**).
RegisterLoopDXDTOut	Maximum control range of the control element that is to be processed per ms.

Click" Next>>" to open the download window. Downloading involves loading all curve data into the inverter. If severel curves were selected, the next curve will be run through in the same process. If not, the IPOS$^{plus®}$ compiler opens that has already generated a program. This program contains different header files as well as various IPOS$^{plus®}$ files. Please also refer to the Project structure section for more details.

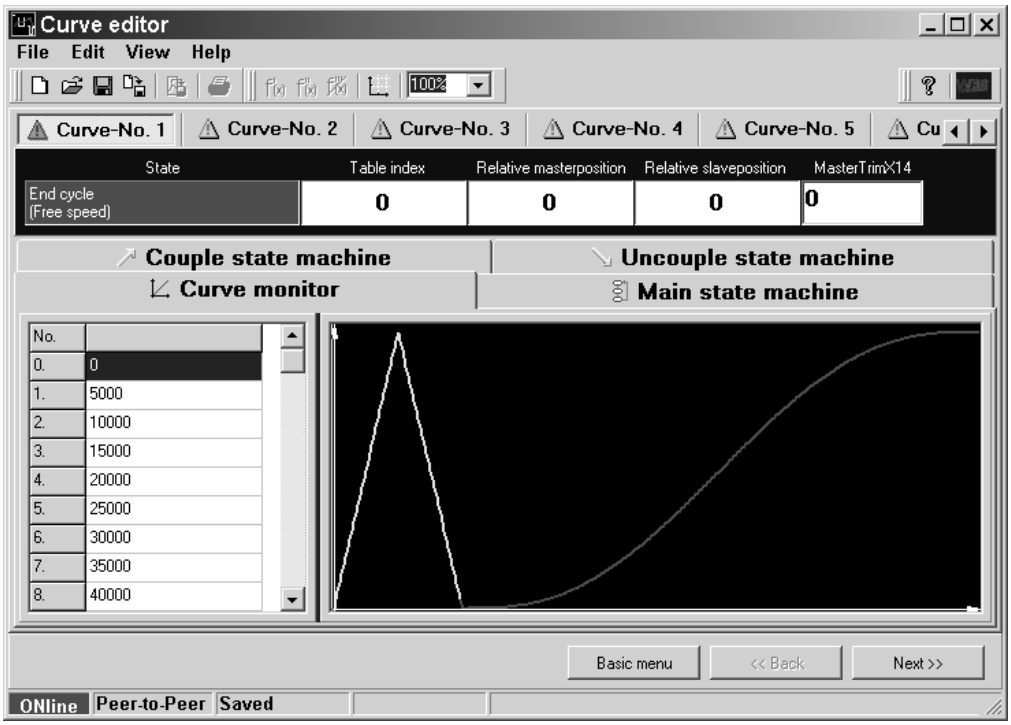


10749AEN

Figure 41: Download

10.7 Monitor

The electronic cam monitor displays operating states and the current cam profile.



10750AEN

Figure 42: Electronic cam monitor

11 Curve Variables

Virtual encoder with ramp generator

Variables	Name	Meaning	Coding / usage
H370	VEncoderMode	Virtual encoder operating mode	= 0: Virtual encoder with acceleration ramp = 1: Reserved = 2: Infinite counter = 3: Virtual encoder with acceleration and deceleration ramp
H371	VEncoderModeControl	State transition control in the event of a unit fault	Modes 0, 2 and 3 only Bit 0 == 0: No stop Bit 0 == 1: Virtual axis is stopped (H373 = 0).
H372	VEncoderState	No function	
H373	VEncoderNSetpoint	Speed setpoint	Unit master increments/ms
H374	VEncoderNActual	Actual speed value	Unit master increments/ms
H375	VEncoderXSetpoint	Setpoint position of virtual master value	Unit master increments
H376	VEncoderXActual	Actual position of virtual master value	Unit master increments
H377	VEncoderdNdT	Acceleration of virtual encoder	Unit master increments/ms^2 in mode 0. Unit 1/2^{16} master increments/ms^2 in mode 3.

Register state machine

Variables	Name	Meaning	Coding / usage
H380	RegisterMode	Activation of register loop control	= 0: Register loop control disabled = 1: Control element and register loop control active = 2: Only control element active, e.g. for manual trimming.
H381	RegisterModeControl	State transition control	Selecting a function within a mode Bit 0 == 0: Auto reset Bit 0 == 1: Manual restart Bit 1 == 0: Signal level high active (positive edge is evaluated). Bit 1 == 1: Signal level low active (negative edge is evaluated). Bit 2 == 0: Register loop controller has an effect on H437 SlaveTrim (to be used if slave is prone to slip). Bit 2 == 1: Register loop controller has an effect on H438 XMasterPos (to be used if master is prone to slip).
H382	RegisterState	Interrupt enable via software	Indicates the state of the register state machine. Register input is not enabled until RegisterState has value 1 or H381.0 == 0 auto restart enabled by IPOS$^{plus®}$ program.
H383	RegisterLimitLeft	Left-hand beginning of position window	Definition of left-hand limit within which the valid register must be detected. Unit: slave increments of the main curve. This variable is also used in stop cycle mode 3 (H400 StopCycleMode = 3) to define the left-hand limit within which a touch probe event is evaluated (stop cycle trigger).
H384	RegisterLimitRight	Right end of the reference window	Definition of right-hand limit within which a valid register must be detected. Unit: slave increments of the main curve. This variable is also used in stop cycle mode 3 (H400 StopCycleMode = 3) to define the right limit within which a touch probe event is evaluated (stop cycle trigger).

Variables	Name	Meaning	Coding / usage
H385	RegisterSetpoint	Definition for start of register	Setpoint for start of register. Unit: slave increments of the main curve.
H386	RegisterActValue	The determined start of register is stored	Actual value for start of register for a register which is detected as valid. Unit: slave increments of the main curve.
H387	RegisterLoopPGain	P-factor, evaluation with 0.01	> 0: If slave curve slope is positive <= 0: If slave curve slope is negative Value range: –30000 ... 0 ... 30000. == 0: Control element inactive. Start value ± 100 (which means no scaling is performed).
H388	RegisterLoopMaxOut	Limit for control element	Unit: slave increments of the main curve.
H389	RegisterLoopOut	Calculated control output for control element	Unit: slave increments of the main curve.
H390	RegisterLoopDXDTOut	Control range of control element per ms (rate of change)	Value must be selected for max. speed. This variable is additionally used in stop cycle mode 3 (H400 StopCycleMode = 3) to compensate the latency of the stop cycle sensor (H406 StopCycleDelayDI03).
H391	RegisterLengthMax		Maximum length of a “correct” register. Unit: slave increments of the main curve.
H392	RegisterLengthMin		Minimum length of a “correct” register. Unit: slave increments of the main curve.
H393	RegisterLengthActual		Measured register length (value = 0 if no register is detected). Unit: slave increments of the main curve.
H394	RegisterLoopCounter		Register detection counter. Value is incremented by 1 each time a register is determined correctly.
H395	RegisterLoopTimeConst		An I component can be activated in addition to the P component (P387). Value = Factor for the adjustment time of the register loop controller. Value range 0 ... 32000 with I component with 2^{16} normalization. With value = 0 -> Do not calculate “I component”.
H396	RegisterLoopIValue		I component of register loop controller with 2^{16} normalization Read only -> mere display value.
H397	RegisterLagPGain		== 0: disabled > 0: If slave curve slope is positive <= 0: If slave curve slope is negative Value range: –30000 ... 0 ... 30000.
H398	RegisterLoopSlavePos	Slave position	Unit: 4096 increments/revolution of the slave. Absolute slave position unscaled. Start cycle or stop cycle distance has a negative sign. Value can always be read (even without register control).

Stop cycle state machine

Variables	Name	Meaning	Coding / usage
H400	StopCycleMode	Stop cycle mode	= 0: No automatic disengaging = 1: Automatic disengaging according to the number of cycles defined in H404 (StopCycleCounter) = 2: Due to a touch probe event on DI03, the distance defined in variable H405 will be continued. Next, disengaging takes place as in mode 2.
H401	StopCycleModeControl	State transition control	Bit 0 == 0: Operating mode after disengaging, speed controlled Bit 0 == 1: Operating mode after disengaging, position controlled Bit 2 == 0: Interrupt enabled on DI03. Bit 2 == 1: Signal edge change of zero track on X15 as startup cycle source.

Variables	Name	Meaning	Coding / usage
H402	StopCycleState	State machine	Indicates the state of the stop cycle state machine
H403	StopCycleReload	Reload value	Reload value for counter H405 after the stop cycle process
H404	StopCycleCounter	Counter cycle	Signed value for cycle counter Main curves are run through, then disengaging takes place automatically. Important! Only active when H400 == 1.
H405	StopCycleRemainWay	Remaining distance when disengaging in mode 3	Remaining distance for mode 3, unit 4096 increments / revolutions of slave
H406	StopCycleDelayDI03	Delay on sensor DI03	Unit 0.1ms, bipolar; only active in mode 2

Startup cycle state machine

Variables	Name	Meaning	Coding / usage
H410	StartupCycleMode	Startup cycle mode	= 0: No engaging using input terminals = 1: Engaging due to input terminal signal, level sensitive, high-active, input terminals via bit mask in H413 = 2: Engaging due to edge change on terminal DI02 (interrupt) = 3: Counter control
H411	StartupCycleModeControl	State transition control, interrupt processing	**Enabled in modes 2 and 3 only** Bit 0 == 0: No auto restart; interrupt must be enabled in the application by assigning StartupCycleState (), i.e. StartupCycleState remains in state 0 (see startup cycle state machine). Bit 0 == 1: Auto restart, i.e. interrupt remains enabled and StartupCycleState automatically changes from state 0 to state 1 (see startup cycle state machine). Bit 1 == 0: Engaging with enabled input terminals (e.g. used for ring buffer management) Bit 1 == 1: Engaging with disabled input terminals **Mode 2 only** Bit 2 == 0: Interrupt enabled on DI02 Bit 2 == 1: Signal edge change of zero track on X14 as startup cycle source Bit 3 == 0: Ring buffer management not active Bit 3 == 1: Ring buffer management active
H412	StartupCycleState	State machine	Indicates the state of the startup cycle state machine Important! The StartupCycleCounter variable H414 does not increment until the value of StartupCycleState is set to 1. During the first run after initialization, 1 must be assigned to this value by IPOS program. All other runs are automatically started with AutoRestart (StartupCycleModeControl.0 == 1), i.e. state 0 is not reached anymore. When “Automatic start ON“, the startup function initializes the StartupCycleState = 1. This means state 0 is not reached.
H413	StartupCycleInputMask	Startup cycle terminal mask	Assignment identical to H483 (IPOS)
H414	StartupCycleCounter	Current distance offset when engaging	Unscaled master encoder pulses are counted; is zeroed upon startup cycle
H415	StartupCycleCounterMax Value		Mode 2 MasterOffset starting from the startup cycle process begins via DI02 / C track. Mode 3 length limit for automatic disengaging Must be greater than startup cycle + master cycle + stop cycle

Variables	Name	Meaning	Coding / usage
H416	StartupCycleDelayDI02	Delay of sensor DI02	Unit 0.1ms, bipolar; only active in mode 2
H417	StartupCycleBuffer	32-bit ring buffer Read only	Is “shifted left” once per cycle and assigned with the current product signal indicated by bit 0. A 0 at the defined input H413 results in a 0 in the buffer which then “moves“ to the higher value bits. Important! Only active when H411.3 == 1
H417	StartupCycleBuffer Length	Effective buffer length	Indicates which bit is transferred from the buffer H416 to the disable bit H411.1. The level changes with each transfer, i.e. a 0 in the buffer will become H411.1 = 1

Cam state machine

Variables	Name	Meaning	Coding / usage
H428	CamMode	Reserved	
H429	CamModeControl	Direction of rotation of master encoder; filter	Bit 0: Control of master encoder's direction of rotation CamModeControl.0 == 0 Do not invert master encoder's direction of rotation. CamModeControl.0 == 1 Invert master encoder's direction of rotation. Bit 1: Filter CamModeControl.1 == 0 Filter active CamModeControl.1 == 1 Filter inactive See Sec. Curve variables
H430 *	MasterSource	Master frequency source	MasterSource == 0: X14 + virtual axis; synchronous encoder input enabled; a virtual axis is added (can be activated via H442 MasterTrimX14), for example, to simulate the master at startup. Important! No ramp possible! MasterSource ≠ 0: Pointer to variable that represents the master encoder (e.g. H376 virtual encoder with ramp generator)
H431 *	MasterNorm	Master cycle scaling	Scales the master cycle internally to < 2^{16} –1 (acts on H441 MasterPos216). Specified as exponent n of 2^n. Is absolutely necessary and must be calculated manually (external) if you do not use the startup user interface! E.g. MasterCycleScale = 1000000 and as a result MasterNorm = 5: 1000000 / 2^{15} = 30.5 < 2^5.
H432 *	MasterCycleScale	Master cycle	Master increments multiplied by interpolation time H446
H433 *	StartupCycleScale	Startup cycle	Engaging distance multiplied by interpolation time H446
H434	FirstTabIndex	First curve point table pointer	The main curve starts from this table value (control point). This value is 0 if no startup cycle curve has been defined
H435	LastTabIndex	Last curve point table pointer	Total number of table values (curve points) of both startup cycle curve and main curve
H436	CamState	State of electronic cam	= 0: Reset electronic cam, disengaged at constant speed H439 SpeedFreeMode. = 1: Startup cycle state = 2: Main curve = 3: Stop cycle state = 4: Stop cycle state with inverted startup cycle curve Never directly assign CamState = 2 via IPOS! See Sec. Electronic cam state machine

Variables	Name	Meaning	Coding / usage
H437	SlaveTrim	Setpoint position offset (scaled with H440)	Runtime dependent slave setpoint position offset Is written by the firmware • after reset (CamState == 0) with H445 XSlaveSetpointScale • if the curve is run through more than once Can be written for trimming using IPOS, i.e. phasing of master / slave changes. Important! ActPosScale variable H440 must be taken into account for trimming purposes!
H438	XMasterPos	Master interpolator output	Value range 0 ... 2^{32}. Is written by the firmware and indicates the master position in the curve Can be used for correcting master / slave phasing. Important! Multiplication with H446 MFilterTime! Starts at 0 once the startup cycle curve has been run through once.
H439	SpeedFreeMode	Speed setpoint in freewheel state	Only active if H436CamState == 0 or stop cycle state H436CamState == 0 n-control is enabled (StopCycleModeControl.0) Unit 0.2 min^{-1}
H440 *	ActPositionScale	Slave cycle scaling	Specified as exponent n of 2^n; value range n = 0 ... 10. Serves for improving slave cycle resolution. Is absolutely necessary and must be calculated manually (external) if you do not use the startup interface! E.g. slave cycle (is not indicated on a variable) = 20000, as a result ActPositionScale = 3: 1FFFFhex / slave cycle = 6.55 < 2^3.
H441	MasterPos216	Normalized pointer to master interpolation table	MasterPos216 = MasterCycleScale / MasterNorm, value range 0 ... 2^{16}; internal system parameter Starts again at 0 after startup cycle curve has been run through.
H442	MasterTrimX14	Virtual axis without ramp generator	Speed of the virtual encoder Is added to the external encoder (X14); number of pulses 1 inc/ms; only active if H430 MasterSource == 0.
H443 *	NFilterTime	Feedforward interpolator	Value range 0 ... 30, can be used for filtering master encoder signal. Feedforward is disabled if value is 1; feedforward branch is separated if value is 0. Ist ein gutes Werkzeug um den Slave “ruhiger“ zu machen.
H444	ReSprintClose	Direction of rotation inhibit	= 0: Both directions of rotation are enabled = 1: Only CCW direction of rotation = 2: Only CW direction of rotation
H445	XSlaveSetpointScale	Absolute setpoint position of slave	Is written by the firmware and indicates the setpoint position of the slave. Important! Is scaled with H440 ActPositionScale and H437 SlaveTrim!
H446 *	MFilterTime	Interpolation time	Value range 0 ... 30. = 0: disabled, no master. = 1: without filter ≤=30: High scaling active, better resolution of master cycle, master cycle must be re-adjusted afterwards. Important! Multiplication with MFilterTime
H447 *	TabBits	Definition of number of curve points	= 8: 256 master curve points (mandatory entry if startup cycle has been defined) = 9: 512 master curve points (mandatory entry if without startup cycle).

Variables	Name	Meaning	Coding / usage
H448	TableScaleControlState	Control / state variable for scaling control point table stored in EEPROM	TableScaleControlState == 0: No scaling or scaling is canceled if already started (e.g. denominator = 0). TableScaleControlState == 1: Start curve scaling (control); entire data range is always scaled. TableScaleControlState == 2: Scaling in progress (state) TableScaleControlState == 3: Scaling completed successfully (state)
H449	TableScaleNominator	Scaling factor numerator	Signed 16 bit value range –32000 ... 0 ... 32000
H450	TableScaleDenominator	Scaling factor denominator	Unsigned 16 bit value range 0 ... 32000
H451	TableSelect	Selection of curve set	= 0: Curve set 0 active (index 20000 ... 20513 read and write possible). = 1 to 5: corresponding curve set active (no index access possible but via Vardata only; curve 1 can additionally be written via copy command H452 TableCopy).
H452	Table Copy	Copying a curve set	= 0: no copying or requested copying completed successfully = 1: copying requested (copy curve set 0 to curve set 1)

* Parameter is set by the startup function and should not be altered manually.

SHELL parameters of the electronic cam

Variables	Name	Comment
	Technology function	Select Shell / Startup / Technology function and set electronic cam
P700	Operating mode	CFC & IPOS or SERVO & IPOS
P916	Ramp type	Set to "Electronic cam"
P203	Filter accel. feedforward	This parameter filters the setpoints from the curve table. A speed-dependent lag error occurs as a result. This is not a problem in applications such as pick and place. In other applications, such as printing or cutting, this lag error is undesirable because it might impair the result. In such cases, this parameter must be set to the smallest possible value. This means, you have to decrease the value preset at startup until the current controller starts to vibrate. Then set the parameter to a higher value again. If the current controller starts vibrating too early, the stiffness must be reduced when adjusting the speed controller.
P206	Sample time n-control	You can use this parameter to set the cycle time of the speed controller. The default setting is 1 ms. Reducing the cycle time of the speed controller can improve the control quality, in particular in applications in which optimizing the stiffness does not improve the control response. Bear in mind that setting 0.5 only makes sense if you are using a high-resolution motor encoder (e.g. sin/cos encoder).
P601	Binary input DI02	Set to "No function" if the startup cycle is to be triggered by an interrupt.
P602	Binary input DI03	Set to "No function" if you want to use register control.

12 Project Structure

This project structure refers to MOVITOOLS® version 3.0.

The curve editor uses the compiler project structure. This means you always have to click the “Compile and load project“ button to ensure that all created *.ipc files will be compiled.

Two *.ipc files are created automatically when the compiler opens automatically from the curve editor:

- Project name_curves.ipc: Only header files are included in this file (includes)
  - #include <curve.h> // General header file for electronic cam (not created separately, is stored in MOVITOOLS®)
  - #include <curve name_of_curve1.h> // Header file for the second curve (contains function setCAMParameter2();)
  - #include <curve name_of_curve2.h> // Header file for the third curve (contains the function setCAMParameter3();)
  - ... etc. up to max. 6 curves

If only one curve set is loaded, this file only contains “ #include <curve.h>“. This is for compatibility reasons with earlier series.

- curve name_of_first_curve.ipc: This is the file that can contain the application program. This file also contains header files.
  - #include <const.h> // General header file for IPOS$^{plus®}$ (not created separately, is stored in MOVITOOLS®)
  - #include <io.h> // General header file for IPOS$^{plus®}$ (not created separately, is stored in MOVITOOLS®)
  - #include <curve name_of_curve0.h> // Header file for the first curve (contains the function setCAMParameter ();)

12.1 Header file: curve.h

```
/*****************************************************************
Name        : Data file for CAM control
Description : Data and startup header file for IPOS+ Compiler.
              For Startup after power on call "setCAMParameterx();"
Device      : MOVIDRIVE CAM control
Version     : 1.60
Author      : Automatically generated file from CAM-Editor
Company     : SEW-Eurodrive GmbH & Co.
*****************************************************************/
// ---------------------------------------------------------------
// Data from the integrated startup function:
// ---------------------------------------------------------------
#define  CamMode                    H428
#define  CamModeControl             H429
#define  MasterSource               H430
#define  MasterNorm                 H431
#define  MasterCycleScale           H432
#define  StartupCycleScale          H433
#define  SpeedFreeMode              H439
#define  ActPositionScale           H440
#define  MasterTrimX14              H442
#define  NFilterTime                H443
#define  ReSprintClose              H444
#define  MFilterTime                H446
#define  TableBits                  H447
// ---------------------------------------------------------------
```

```
// Data from block diagram:
// ------------------------------------------------------------------
#define  FirstTabIndex              H434
#define  LastTabIndex               H435
#define  CamState                   H436
#define  SlaveTrim                  H437
#define  XMasterPos                 H438
#define  MasterPos216               H441
#define  XSlaveSetpointScale        H445
#define  TableScaleControlState     H448
#define  TableScaleNumerator        H449
#define  TableScaleDenominator      H450
#define  TableSelect                H451
#define  TableCopy                  H452
// ------------------------------------------------------------------
// Variable of StopCycle
// ------------------------------------------------------------------
#define StopCycleMode               H400
#define StopCycleModeControl        H401
#define StopCycleState              H402
#define StopCycleReload             H403
#define StopCycleCounter            H404
#define StopCycleRemainWay          H405
#define StopCycleDelayDI03          H406
// ------------------------------------------------------------------
// Variable of StartupCycle
// ------------------------------------------------------------------
#define StartupCycleMode            H410
#define StartupCycleModeControl     H411
#define StartupCycleState           H412
#define StartupCycleInputMask       H413
#define StartupCycleCounter         H414
#define StartupCycleCounterMaxValue H415
#define StartupCycleDelayDI02       H416
#define StartupCycleBuffer          H417
#define StartupCycleBufferLength    H418
// ------------------------------------------------------------------
// Variable of RegisterControl H380..H399
// ------------------------------------------------------------------
#define RegisterMode                H380
#define RegisterModeControl         H381
#define RegisterState               H382
#define RegisterLimitLeft           H383
#define RegisterLimitRight          H384
#define RegisterSetpoint            H385
#define RegisterActValue            H386
#define RegisterLoopPGain           H387
#define RegisterLoopMaxOut          H388
#define RegisterLoopOut             H389
#define RegisterLoopDXDTOut         H390
#define RegisterLengthMax           H391
#define RegisterLengthMin           H392
#define RegisterLengthActual        H393
#define RegisterLoopCounter         H394
#define RegisterLoopTimeConst       H395
#define RegisterLoopIValue          H396
#define RegisterLagPGain            H397
#define RegisterLoopSlavePos        H398
// ------------------------------------------------------------------
// Variable virtual Encoder  H370..H379
// ------------------------------------------------------------------
#define VEncoderMode                H370
```

```
#define VEncoderModeControl         H371
#define VEncoderState               H372
#define VEncoderNSetpoint           H373
#define VEncoderNActual             H374
#define VEncoderXSetpoint           H375
#define VEncoderXActual             H376
#define VEncoderdNdT                H377
// ------------------------------------------------------------------
```

12.2 Header file: Curve name_of_curve0.h

```
/*******************************************************************
Name        : Data file for CAM control
Description : Data and startup header file for IPOS+ Compiler.
              For startup after power-on call "setCAMParameter();"
Device      : MOVIDRIVE CAM control
Version     : 1.60
Author      : Automatically generated file from CAM-Editor
Company     : SEW-Eurodrive GmbH & Co.
*******************************************************************/
// ------------------------------------------------------------------
//Startupdata from: 03.01.2003 - 11:08:45
setCAMParameter()
{
  for (H0=370; H0<=452; H0++)
    {
    *H0=0;
    }
  _Memorize(/*MEM_LDDATA*/ 6 );
  _Wait(100);
MasterSource                  = 0;
MasterNorm                    = 0;
MasterCycleScale              = 10000;
StartupCycleScale             = 0;
SpeedFreeMode                 = 0;
ActPositionScale              = 5;
MasterTrimX14                 = 0;
NFilterTime                   = 7;
ReSprintClose                 = 0;
MFilterTime                   = 1;
TableBits                     = 8;
StartupCycleMode              = 2;
_BitSet(StartupCycleModeControl, 0);
StartupCycleDelayDI02         = 0;
StartupCycleCounterMaxValue   = 0;
StartupCycleState             = 1;
StopCycleMode                 = 1;
_BitSet(StopCycleModeControl, 0);
StopCycleReload               = 1;
StopCycleCounter              = 1;
RegisterLagPGain              = 0;
}
```

13 Programmin Examples

13.1 Engaging in the curve

```
/*============================================
              IPOS source file
              for CAM control
               (MOVIDRIVE B)
  ------------------------------------------
          SEW-Eurodrive GmbH & Co.
           Ernst-Blickle-Str. 42
             D-76646 Bruchsal

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

/* The purpose of this example is to show how to
engage at any point in the curve other than the
beginning (e.g. after an emergency stop).
The curve editor must be run for this purpose*/

#pragma  var        300  309
#pragma globals    310  349

#include <curve.h>
#include <constb.h>
#include <iob.h>
#include <Kurve.h>
#define electronic cam 5
#define linear 0
long BinInputsNew, BinInputsOld, ramp type, target ;

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

  /*-------------------------------------
                 Startup
  -------------------------------------*/
  setCAMParameter();

  /*-------------------------------------
           Main program loop
  -------------------------------------*/
  while(1)
    {
      // For edge evaluation
      // _GetSys(BinInputsNew,GS_INPUTS) is also possible
      BinInputsNew = InputLevel;

      if((BinInputsNew & 0x4)&& !(BinInputsOld & 0x4)) //positive edges DI02
          {
          _BitSet( ControlWord, 1 );   // no enable
          Ramp type = linear;
          _SetSys( SS_RAMPTYPE, ramp type ); // Set ramp type to linear
          _BitSet( CamModeControl, 1 ); // disable internal filter
          CamState = 1;
          XMasterPos = ActPos_Ext / MFilterTime;
          SlaveTrim = 0;
          _BitClear( CamModeControl, 1 );  // enable internal filter
          _BitClear(ControlWord , 1 );     // enable
          _Wait (200);
          Target =  XSlaveSetpointScale>>ActPositionScale; //Set target position
          _GoAbs( GO_WAIT, target );  // Move to target position
          Ramp type = electronic cam;
          _SetSys( SS_RAMPTYPE, ramp type );  //Change-over to electronic cam
          H0++;
          }
       BinInputsOld = BinInputsNew; //for edge evaluation

    }
  }
```

13.2 Disengaging and transition to positioning mode

```
/*============================================
              IPOS source file
              for CAM control
               (MOVIDRIVE B)
  ------------------------------------------
          SEW-Eurodrive GmbH & Co.

         sew@sew-eurodrive.de
         http://www.SEW-EURODRIVE.de
============================================*/
/*The purpose of this programming example is to show how to
  position the master "on the fly" for example to the
  modulo home position. Modulo must be enabled in Shell
  for this purpose. The curve editor must have been
  run through.                  26.02.2004 Pfaadt */

#pragma  var        300  309
#pragma globals    310  349

#include <constb.h>
#include <curve.h>
#include <iob.h>
#include <hipp1.h>
#define electronic cam 5
#define linear 0
SSPOSSPEED Speed;
long positioning speed, ramp type ;
long Flag1, ModNumerator, ModDenominator;
/*============================================
   Main function (IPOS start function)
============================================*/
main()
  {

  /*--------------------------------------
                  Startup
  --------------------------------------*/
 setCAMParameter();

 //Ramp type electronic cam as default
 Ramp type = electronic cam;
 _SetSys( SS_RAMPTYPE, ramp type );

 ModNumerator = 10;  // Adjust modulo numerator with parameter 961
                  // if necessary, ggf. MOVILINK auf Index 8836 absetzen
 ModDenumerator = 1;    // Adjust modulo numerator with parameter 962
                  // ggf. MOVILINK auf Index 8837 absetzen

  /*--------------------------------------
            Main program loop
  --------------------------------------*/
  while(1)
    {
    // Disengaging and transition to positioning mode is started "on the fly" with DI02
    if(DI02)
      {
       //Read in current speed
      _GetSys(positioning speed ,GS_ACTSPEED );

      //Write current speed to positioning speed
      Speed.CW = positioning speed;
      Speed.CCW = Speed.CW;
      _SetSys( SS_POSSPEED,Speed);

      //Calculation of modulo home position and setting to target
      //The parameter H960 Modulo function must be set to "Short"
      // and the corresponding numerator / denominator factors must be entered
      Flag1 = ((0xFFFF-ModActPos)* ModNumerator) / (ModDenominator * 16);
      TargetPos = ActPos_Mot + Flag1;

      //Switch ramp type to linear
      //Reference to the curve is quit
      Ramp type = linear;
      _SetSys( SS_RAMPTYPE, ramp type );

      //Reset a curve
      CamState = 0;

      // Only if binary input is present for a longer time! Remedy: edge evaluation
      _Wait( 1000 );
      }

   //DI03 is used to set the ramp type to electronic cam again
   //Curve starts from the beginning because it is reset
   if(DI03)
     {
     Ramp type = electronic cam;
     _SetSys( SS_RAMPTYPE, ramp type);
     }
    }

 }
```

14 Index

How we're driving the world

With people who think fast and develop the future with you.

With a worldwide service network that is always close at hand.

With drives and controls that automatically improve your productivity.

With comprehensive knowledge in virtually every branch of industry today.

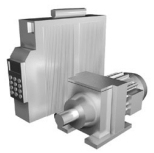
With uncompromising quality that reduces the cost and complexity of daily operations.

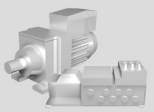
SEW-EURODRIVE
Driving the world

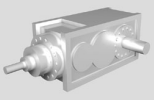
With a global presence that offers responsive and reliable solutions. Anywhere.

With innovative technology that solves tomorrow's problems today.

With online information and software updates, via the Internet, available around the clock.







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**