ENGINEERING TOMORROW

Programming Guide

VLT® Integrated Servo Drive ISD® 510 System

vlt-drives.danfoss.com

Contents

1 Introduction

1.1 Purpose of the Programming Guide

The purpose of this programming guide is to describe the programming of the VLT® Integrated Servo Drive ISD® 510 System.

This programming guide contains information about:

- Software installation
- Programming
- Operation
- Applications
- Troubleshooting

This programming guide is intended for use by qualified personnel. Read the document in full in order to use the servo system safely and professionally, and pay particular attention to the safety instructions and general warnings. This programming guide is an integral part of the ISD 510 servo system so keep it available with the servo system at all times.

Compliance with the information in this document is a prerequisite for:

- Trouble-free operation
- Recognition of product liability claims

Therefore, read this document before working with the servo system.

1.2 Additional Resources

Available manuals for the VLT® Integrated Servo Drive ISD® 510 System:

Document	Contents
VLT® Integrated Servo Drive ISD® 510 System Operating Instructions	Information about the installation, commissioning, and operation of the ISD 510 servo system.
VLT® Integrated Servo Drive ISD® 510 System Design Guide	Information about the set-up of the ISD 510 servo system and detailed technical data.
VLT® Integrated Servo Drive ISD® 510 System Programming Guide	Information about the programming of the ISD 510 servo system.

Table 1.1 Available Manuals for the ISD 510 Servo System

Technical literature for Danfoss drives is also available online at *drives.danfoss.com/knowledge-center/technical-documentation/*.

1.3 Copyright

VLT®, ISD®, and SAB® are Danfoss registered trademarks.

1.4 Software

The software for the ISD 510 servo system comprises:

- The firmware of the VLT® Integrated Servo Drive ISD® 510 that is already installed on the device.
- The firmware of the VLT® Servo Access Box that is already installed on the device.
- A package of PLC libraries for Automation Studio™ for operating the ISD 510 devices (see *chapter 6.4.1 Programming with Automation Studio ™* for further information).
- A PLC library for TwinCAT® 2 for operating the ISD 510 devices (see *chapter 6.3.1 Programming with TwinCAT®* for further information).
- ISD Toolbox: A Danfoss PC-based software tool for commissioning and debugging the devices.

1.4.1 Software Version

This programming guide can be used for the following software versions onwards:

- ISD 510 Servo Drive: Version 1.4.0
- Servo Access Box (SAB): Version 1.2.0
- ISD Toolbox: Version 2.0
- PLC libraries (Powerlink / EtherCAT): Version 1.0

The software version number can be read from object 0x4000 (see *chapter 7.22.4 Parameters 15-40, 15-41, and 15-43: Version log (0x4000)*).

1.4.2 Firmware Updates

Firmware updates may be available. When firmware updates are available, they can be downloaded from the *danfoss.com* website. Use the ISD Toolbox software to install the firmware in the servo drives.

1.5 Approvals and Certifications

The ISD 510 servo system fulfills the standards listed in *Table 1.2*.

IEC/EN 61800-3	Adjustable speed electrical power drive systems. Part 3: EMC requirements and specific test methods.
IEC/EN 61800-5-1	Adjustable speed electrical power drive systems. Part 5-1: Safety requirements – Electrical, thermal and energy.

IEC/EN 61800-5-2	Adjustable speed electrical power drive systems. Part 5-2: Safety requirements – Functional.
IEC/EN 61508	Functional safety of electrical/electronical/ programmable electronic safety-related systems.
EN ISO 13849-1	Safety of machinery – Safety-related parts of control systems. Part 1: General principles for design.
EN ISO 13849-2	Safety of machinery – Safety-related parts of control systems. Part 2: Validation.
IEC/EN 60204-1	Safety of machinery – Electrical equipment of machines. Part 1: General requirements.
IEC/EN 62061	Safety of machinery – Functional safety of safety-related electrical, electronic, and programmable electronic control systems.
IEC/EN 61326-3-1	Electrical equipment for measurement, control, and laboratory use – EMC requirements. Part 3-1: Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) – General industrial applications.
UL508C	UL Standard for Safety for Power Conversion Equipment.
2006/42/EC	Machinery Directive
CE	
2014/30/EU	EMC Directive
2014/35/EU	Low Voltage Directive
RoHS (2002/95/EC)	Restriction of hazardous substances.
EtherCAT®	Ethernet for Control Automation Technology. Ethernet-based fieldbus system.
Ethernet POWERLINK®	Ethernet-based fieldbus system:
PLCopen®	Technical specification. Function blocks for motion control (formerly Part 1 and Part 2) Version 2.0 March 17, 2011.

Table 1.2 Approvals and Certifications

1.6 Terminology

ISD	Integrated servo drive
ISD 510 Servo Drive	Decentral servo drive
VLT® Servo Access Box (SAB)	Unit that generates the DC-link voltage and passes the U_{AUX}, Real-Time Ethernet, and STO signals to the ISD 510 servo drives via a hybrid cable.
PLC	External device for controlling the ISD 510 servo system.
Loop cable	Hybrid cable for connecting drives in daisy-chain format.
Feed-in cable	Hybrid cable for connection from the SAB to the 1st servo drive.

Table 1.3 Terminology

An explanation of all terminology and abbreviations can be found in *chapter 10.1 Glossary*.

1.7 Safety

The following symbols are used in this guide:

WARNING

Indicates a potentially hazardous situation that could result in death or serious injury.

CAUTION

Indicates a potentially hazardous situation that could result in minor or moderate injury. It can also be used to alert against unsafe practices.

NOTICE

Indicates important information, including situations that can result in damage to equipment or property.

The following safety instructions and precautions relate to the ISD 510 servo system.
Read the safety instructions carefully before starting to work in any way with the ISD 510 servo system or its components.
Pay particular attention to the safety instructions in the relevant sections of this manual.

WARNING

HAZARDOUS SITUATION

If the servo drive, SAB, or the bus lines are incorrectly connected, there is a risk of death, serious injury, or damage to the unit.
Always comply with the instructions in this manual and national and local safety regulations.

⚠WARNING

GROUNDING HAZARD

The ground leakage current is >3.5 mA. Improper grounding of the ISD 510 servo system components may result in death or serious injury.

- **For reasons of operator safety, ground the components of the ISD 510 servo system correctly in accordance with national or local electrical regulations and the information in this manual.**

⚠WARNING

HIGH VOLTAGE

The ISD 510 servo system contains components that operate at high voltage when connected to the electrical supply network.
A hazardous voltage is present on the servo drives and the SAB whenever they are connected to the mains network.
There are no indicators on the servo drive or SAB that indicate the presence of mains supply.
Incorrect installation, commissioning, or maintenance can lead to death or serious injury.

- **Installation, commissioning, and maintenance may only be performed by qualified personnel.**

⚠WARNING

UNINTENDED START

The ISD 510 servo system contains servo drives and the SAB that are connected to the electrical supply network and can start running at any time. This may be caused by a fieldbus command, a reference signal, or clearing a fault condition. Servo drives and all connected devices must be in good operating condition. A deficient operating condition may lead to death, serious injury, damage to equipment, or other material damage when the unit is connected to the electrical supply network.

- **Take suitable measures to prevent unintended starts.**

⚠WARNING

UNINTENDED MOVEMENT

Unintended movement may occur when parameter changes are carried out immediately, which may result in death, serious injury, or damage to equipment.

- **When changing parameters, take suitable measures to ensure that unintended movement cannot pose any danger.**

⚠WARNING

DISCHARGE TIME

The servo drives and the SAB contain DC-link capacitors that remain charged for some time after the mains supply is switched off at the SAB. Failure to wait the specified time after power has been removed before performing service or repair work could result in death or serious injury.

- **To avoid electrical shock, fully disconnect the SAB from the mains and wait for at least the time listed in *Table 1.4* for the capacitors to fully discharge before carrying out any maintenance or repair work on the ISD 510 servo system or its components.**

Number	Minimum waiting time (minutes)
0–64 servo drives	10

Table 1.4 Discharge Time

NOTICE

Never connect or disconnect the hybrid cable to or from the servo drive when the ISD 510 servo system is connected to mains or auxiliary supply, or when voltage is still present. Doing so damages the electronic circuitry. Ensure that the mains supply is disconnected and the required discharge time for the DC-link capacitors has elapsed before disconnecting or connecting the hybrid cables or disconnecting cables from the SAB.

NOTICE

Full safety warnings and instructions are detailed in the *VLT® Integrated Servo Drive ISD 510 System Operating Instructions*.

2 Servo Drive Operation

2.1 Overview

The CiA CANopen standard *DS402 Drives and Motion Control Device Profile* is supported by both Ethernet POWERLINK® and EtherCAT®.

2.2 Firmware Update

The products are delivered with the most recent firmware version. See *chapter 1.4.2 Firmware Updates* for information on upgrading.

The servo drive firmware can be updated via the fieldbus. The download of new firmware is only allowed in the unpowered drive state *Switch on disabled*. If the servo drive is in another state, the transfer is refused. While the update is in progress, the servo drive signals the warning *Firmware update in progress*. After finishing, the servo drive signals the warning *Firmware update occurred*. Power cycle the servo drive to resume normal operation.

If the servo drive state machine is switched to another state than *Switch on disabled* after the firmware update has begun (that is, during file transfer or after flashing without a power-cycle), the servo drive switches to state *Fault*. This error indicates that a power-cycle is needed before the servo drive can resume operation. If, for example, a power failure occurs during upgrading, the servo drive remains in a state that allows the update process to resume. The currently installed version can be read from object 0x4000 (see *chapter 7.22.4 Parameters 15-40, 15-41, and 15-43: Version log (0x4000)*).

NOTICE

To change the supported fieldbus, update to the corresponding firmware. After changing the fieldbus, the original product code is no longer valid.

2.3 Basic Operation

2.3.1 State Machine

The servo drive uses the state machine described in the CiA DS402 standard. The state machine is operated either locally via the LCP or remotely via the network.

The state machine is operated by local signals and by the *Controlword* sent over the fieldbus. The state of the state machine is reported by the *Statusword* produced by the servo drive.

A single state represents a special internal or external behavior. The state of the state machine also determines which commands are accepted.

Illustration 2.1 shows the state machine of the servo drive with regard to control of the power electronics as a result of commands and internal servo drive faults.

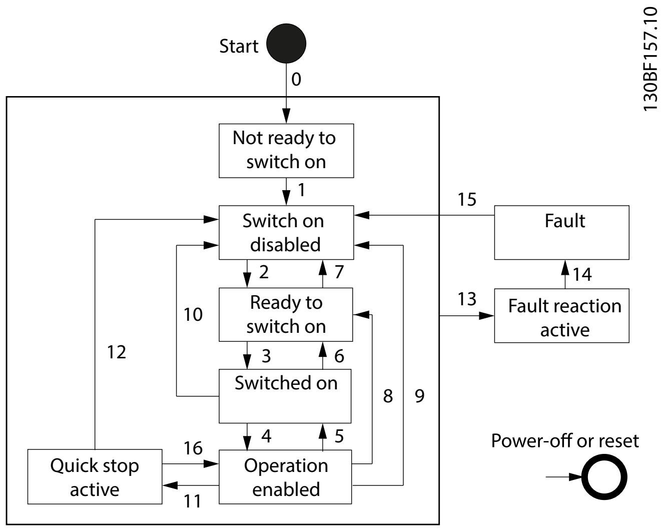


Illustration 2.1 DS402 State Machine

The states support the functions shown in *Table 2.1*. The *Start* state is a pseudo state indicating the start when the state machine is activated during the start-up sequence of the device drives application software.

Function	Not ready to switch on	Switch on disabled	Ready to switch on	Switched on	Operation enabled	Quick stop active	Fault reaction active	Fault
Brake applied, if present	Yes	Yes	Yes	Yes	No	No	No	Yes
Low-level power applied	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
High-level power applied	Yes/no	Yes/no	Yes/no	Yes	Yes	Yes	Yes	Yes/no
Drive function enabled	No	No	No	No	Yes	Yes	Yes	No
Configuration allowed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 2.1 DS402 States and Supported Functions

Quick stop active state is implemented, which is optional according to the standard. When entering this state, the behavior of the servo drive is according to the option code defined in object 0x605A (see *chapter 7.20.6 Parameter 50-46: Quick Stop Option Code (0x605A)*).

The transition from state *Quick stop active* to state *Operation enabled* (Transition 16 in *Illustration 2.1*) is not available, as recommended by the standard.

The servo drive supports the transitions and actions as given in *Table 2.2*. The events initiate the transition. The transition is terminated after the action has been performed.

High-level power applied means that UDC is applied at the input of the servo drive. *Yes/No* means that it is allowed but not necessary.

Configuration allowed means that the following configuration is allowed:

- Changes to the option code objects (see *chapter 7.20 Option Code Objects*).
- Changes to the mode of operation object (see *chapter 7.5.1 Parameter 52-00: Modes of Operation (0x6060)*).

Transition	Event	Action
0	Automatic transition after power-on or reset application.	Servo drive self-test and self-initialization are performed.
1	Automatic transition.	Communication is activated.
2	Shutdown command received from control device.	–
3	Switch on command received from control device.	High-level power is switched on, if possible.
4	Enable operation command received from control device.	The servo drive function is enabled and all internal setpoints are cleared. If the servo drive is rotating when the command to carry out transition 4 is received, the behavior is defined by option code *chapter 7.20.4 Parameter 50-44: Enable in Positioning Option Code (0x2052)*.
5	Disable operation command received from control device.	The configured disable operation reaction function is executed (see *chapter 7.20.9 Parameter 50-49: Disable Operation Option Code (0x605C)*).
6	Shutdown command received from control device.	The configured shutdown reaction function is executed (see *chapter 7.20.8 Parameter 50-48: Shutdown Option Code (0x605B)*).
7	Quick stop or disable voltage command received from control device.	–
8	Shutdown command received from control device.	The servo drive function is disabled and high-level power is switched off, if possible.
9	Disable voltage command received from control device.	The servo drive function is disabled and high-level power is switched off, if possible.
10	Disable voltage or quick stop command received from control device.	High-level power is switched off, if possible.
11	Quick stop command received from control device.	The quick stop function is started.
12	Automatic transition when: • Quick stop function is completed (see *chapter 7.20.6 Parameter 50-46: Quick Stop Option Code (0x605A)*). • Disable voltage command received from control device.	The configured quick stop reaction function is executed (see *chapter 7.20.6 Parameter 50-46: Quick Stop Option Code (0x605A)*).
13	Fault signal.	The configured fault reaction function is executed (see *chapter 7.20.1 Parameter 50-41: Fault Reaction Option Code (0x605E)*).
14	Automatic transition.	The servo drive function is disabled and high-level power is switched off, if possible.
15	Fault reset command received from control device.	If no fault exists on the servo drive, the fault condition is reset. After leaving state *Fault*, clear the fault reset bit in the *Controlword* via fieldbus or the LCP.
16	Not supported.	–

Table 2.2 Transition Events and Actions

If a state transition is requested, the related actions are processed completely before transitioning to the new state. For example, in state *Operation enabled*, when the disable operation command is received, the servo drive remains in state *Operation enabled* until the disable operation function (see *chapter 7.20.9 Parameter 50-49: Disable Operation Option Code (0x605C)*) is completed.

Drive function is disabled means that no energy is supplied to the motor. Target or setpoint values (for example, torque, velocity, position) are not processed. *Drive function is enabled* means that energy is supplied to the motor. Target or setpoint values are processed.

If a fault is detected in the servo drive, a transition to state *Fault reaction active* takes place. In this state, the state machine executes a special fault reaction (see *chapter 7.20.1 Parameter 50-41: Fault Reaction Option Code (0x605E)*). After the execution of this fault reaction, the servo drive automatically switches to state *Fault*. This state can only be left by using the fault reset command, but only if the fault is no longer active.

If a fatal error occurs, the servo drive is no longer able to control the motor, so the servo drive must be switched off immediately. If a fatal error has occurred, the servo is trip-locked and cannot be reset via fieldbus.

The behavior of drive disabling, quick stop, halt, and fault reaction functions are configurable via the objects defined in *chapter 7.20 Option Code Objects*.

If a brake is present, the high-level power is switched off after a delay time in order to apply the brake.

2.3.2 Factor Group

Use the factor group to set the user-defined units required in the application.

The user-defined units are:

- Position units
- Velocity units
- Acceleration units

These units are used for all objects that support user-defined units (for example, position actual value, profile velocity, and profile acceleration).

Changing the objects in the factor group has an immediate effect on all objects that support user-defined units. Their numerical values stay the same, but they are interpreted differently (according to the new scaling factors of the factor group). All numerical values are interpreted using the current settings of the factor group.

NOTICE

If the factor group is changed, then the default values are interpreted differently.

The formulae in this chapter show the calculation of the units. Objects, whose values are not dependent on the factor group, have fixed units specified with the objects.

The objects of the factor group can be found in *chapter 7.4 Factor Group Objects*.

Position units:

The position value is calculated as:

$$\text{Position value} = \frac{\text{position internal value} \times \text{feed constant}}{\text{position encoder resolution} \times \text{gear ratio}}$$

Position value means all objects containing values in user-defined position units.
Position internal value is given in encoder increments.

Velocity units:

The velocity value is calculated as:

$$\text{Velocity value} = \frac{\text{velocity internal value} \times \text{feed constant}}{\text{velocity encoder resolution} \times \text{gear ratio}} \times \text{velocity factor}$$

Velocity internal value is the position internal value(s), resulting in the following formula:

$$\text{Velocity value} = \frac{\text{position value}}{s} \times \text{velocity factor}$$

Velocity value means all objects containing values in user-defined velocity units.

Acceleration units:

The acceleration value is calculated as:

$$\text{Acceleration value} = \frac{\text{velocity value}}{s} \times \text{acceleration factor}$$

Acceleration value means all objects containing values in user-defined acceleration units. The acceleration unit is also used for deceleration.

2.3.3 Positions and Offsets

Inside the servo drive, there are several logical positions. *Illustration 2.2* shows the relationships between them.

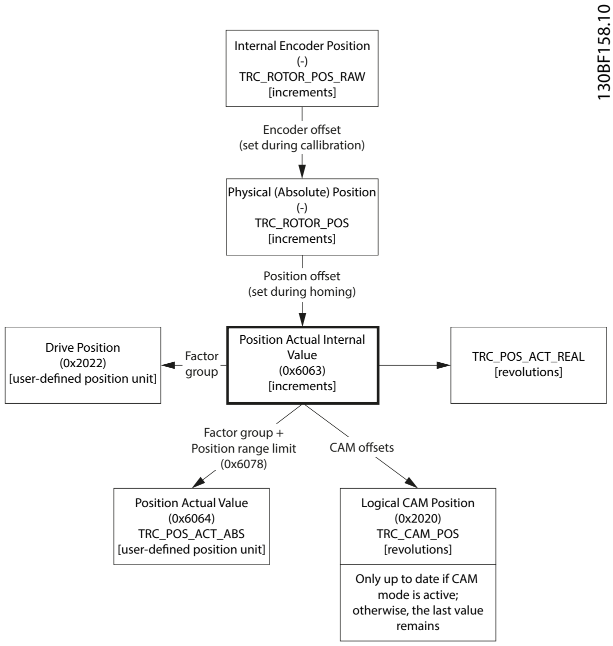


Illustration 2.2 Servo Drive Logical Positions

The object index is given in round brackets. The positions without index numbers are not available in the object dictionary but are used internally in the firmware of the servo drive. The units are given in square brackets.

The *Position offset* is the offset that is calculated during a homing procedure (see *chapter 2.4.4 Homing Mode*). For applications where the zero position only needs to be set once during the lifetime of the servo drive, this offset can be saved to non-volatile memory (see *chapter 7.7.8 Parameters 51-02, 52-04, and 52-49: Application Settings (0x2016)*).

2.3.4 Position Limits

2.3.4.1 Hardware Limit Switch

One method to limit the positions of the servo drive is to use limit switches (left/negative or right/positive), which are also referred to as hardware limit switches. The limit switches must be configured using object 0x200F (see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*). When the servo drive reaches the *Left (Right) Limit switch*, it ramps down to standstill using the value set in object 0x6085 (see *chapter 7.5.9 Parameter 50-13: Quick Stop Deceleration (0x6085)*). It is possible to command the servo drive out of the limit switch in the opposite direction. The states of the limit switches are indicated in object 0x2006 (see *chapter 7.22.12 Parameter 50-08: Motion and Input Status (0x2006)*).

The servo drive remains in state *Operation enabled*. If a motion command is issued that would direct the servo drive further in the wrong direction, the command is rejected by setting the command error bit in the *Statusword*. The monitoring of the limit switch is edge-triggered because the signal does not need to remain high for the duration of the servo drive ramp-down time.

The hardware limit switch is monitored in all modes of operation.

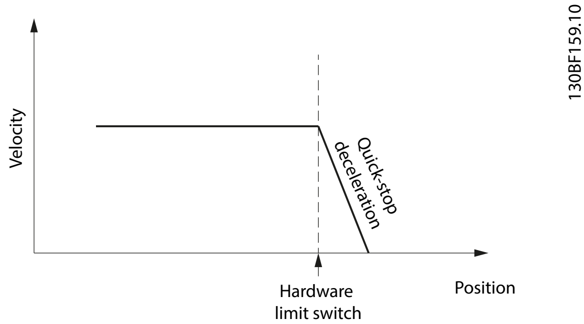


Illustration 2.3 Hardware Limit Switch

2.3.4.2 Software Position Limit

The valid positions of the servo drive can also be limited using software position limits (object 0x607D: Software position limit). This object indicates the configured maximum and minimum software position limits and is used to monitor the position limits in all available modes of operation.

Supervision of software position limits requires a defined home position (the *Is homed* bit in the *Statusword* must be set).

The behavior of the servo drive in a position-controlled mode of operation differs to other modes. In a position-controlled mode of operation, the drive does not pass over the software position limit. The target position is limited to the software position limit. In all other modes of operation, the servo drive immediately ramps down using the *Quick stop deceleration value* (see *chapter 7.5.9 Parameter 50-13: Quick Stop Deceleration (0x6085)*) when the software position limit is passed. This means that the servo drive always stops after the *Software position limit*.

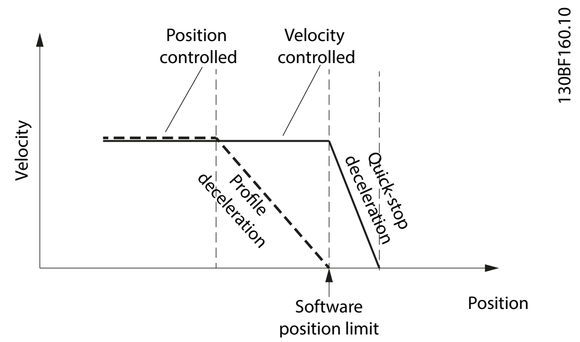


Illustration 2.4 Software Position Limit

Illustration 2.5 to *Illustration 2.9* show the behavior of the servo drive around the position limits.

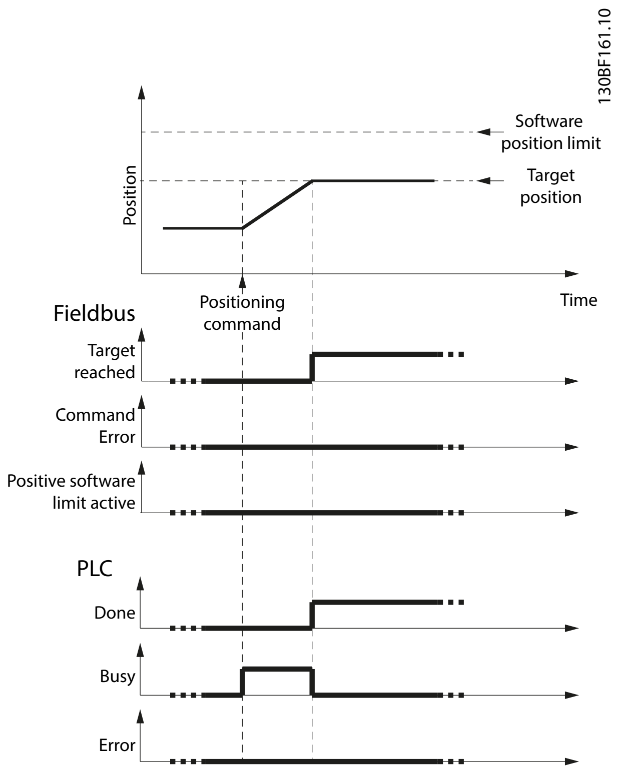


Illustration 2.5 Normal Positioning: Target Position is in the Valid Position Range

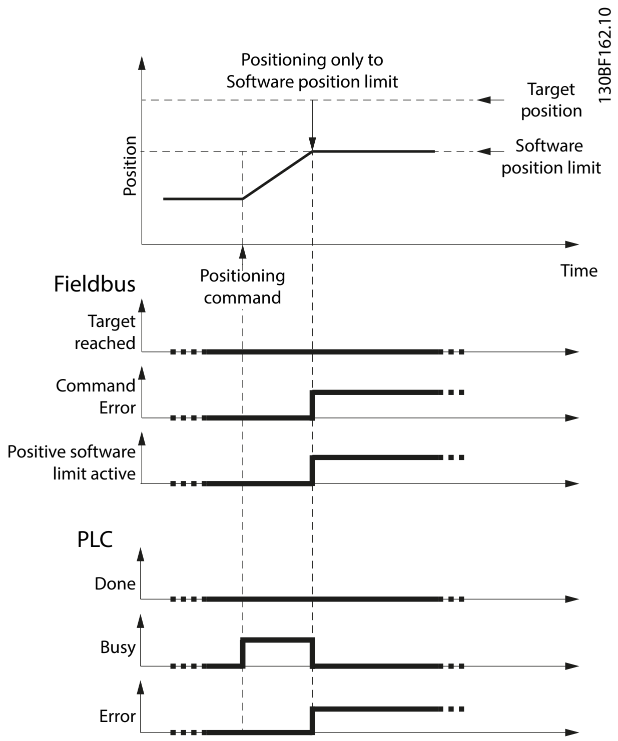


Illustration 2.6 Position Command: Target Position is Behind the Software Position Limit

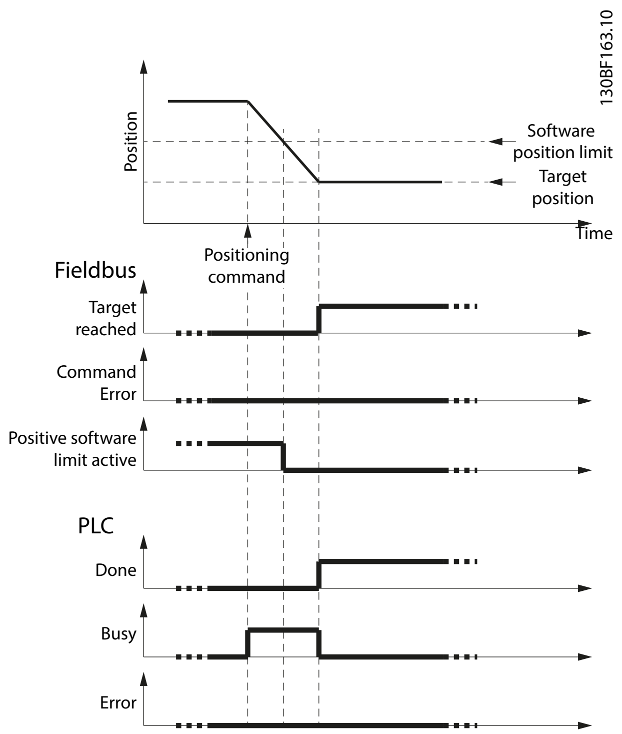


Illustration 2.7 Servo Drive is Outside the Valid Position Limit and the Target Position is in a Valid Area

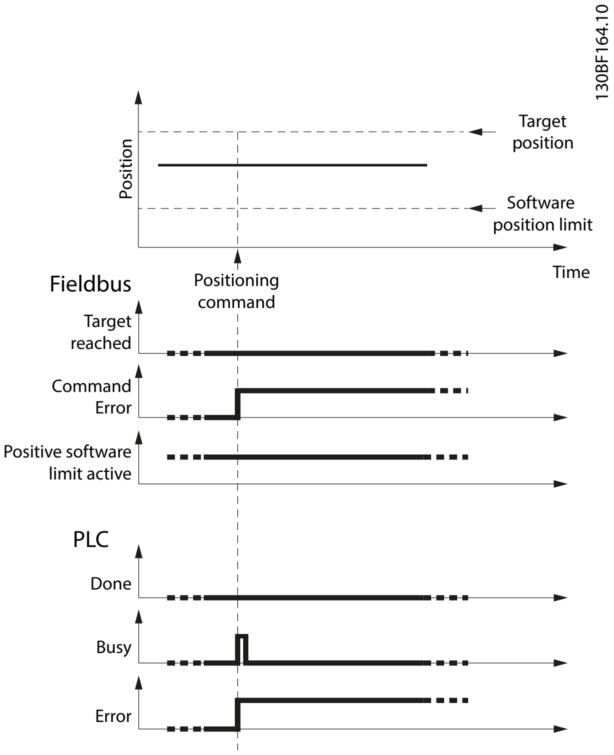


Illustration 2.8 Servo Drive is Outside the Valid Position Limit and the Target Position is in the Wrong Direction

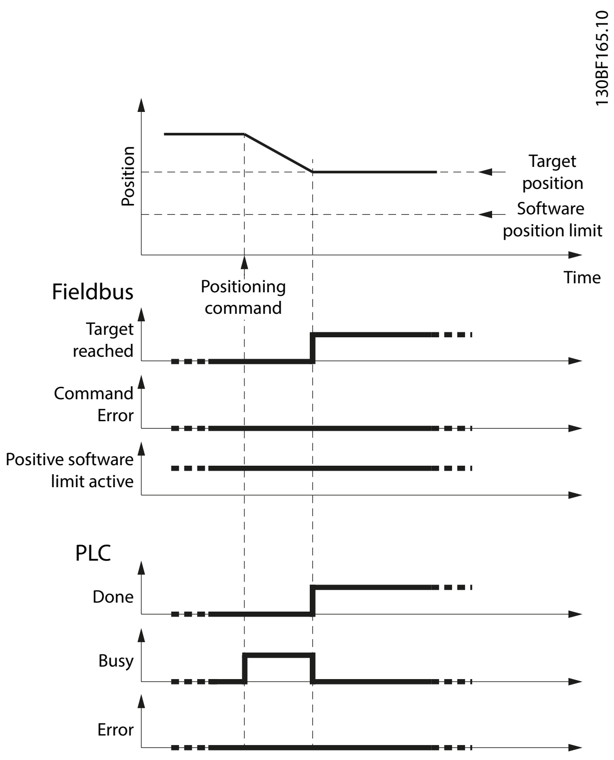


Illustration 2.9 Servo Drive is Outside the Valid Position Limit. The Target Position is Still Not in a Valid Area, but is Nearer to it than the Previous Position

2.3.5 Brake Handling

When the servo drive enters state *Operation enabled*, it automatically lifts the brake. The servo drive reports the new state after the brake is lifted.

When the servo drive leaves state *Operation enabled*, it automatically releases the brake so that the axis cannot sag down. The servo drive reports the new state after the brake is unreleased.

The brake state can be overwritten using the digital output object (see *chapter 7.21.4 Parameter 16-66: Digital Outputs (0x60FE)*). This is only allowed in unpowered state. The valid commands and the reactions are shown in *Illustration 2.10*.

⚠WARNING

UNINTENDED MOTION

Releasing the brake in an unpowered state may result in unintended motion leading to death, serious injury, damage to equipment, or other material damage.

- **Do not release the brake in an unpowered state.**

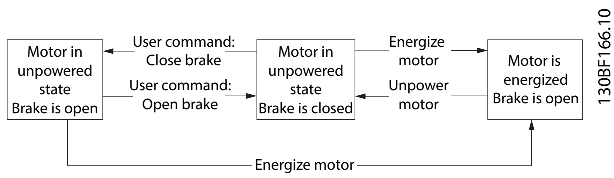


Illustration 2.10 Valid Brake Commands and Reactions

It is not possible to have an energized motor with a closed brake. For further information about the current state, see *chapter 7.22.8 Parameter 50-09: STO Voltage and Brake Status (0x2007)*.

2.3.6 Control Loops

Servo motor control takes place using 3 cascaded control loops (position controller, speed controller, and current controller) with trajectory generators for position and velocity. The control loops run synchronously with the fieldbus cycles. The cycle times shown in *Table 2.3* are possible with Ethernet POWERLINK® and EtherCAT®:

Fieldbus cycle [μs]	Position control cycle [μs]	Speed control cycle [μs]	Current control cycle [μs]
400	200	200	100
500	250	250	125
800	200	200	100

Fieldbus cycle [µs]	Position control cycle [µs]	Speed control cycle [µs]	Current control cycle [µs]
1000	250	250	125

Table 2.3 Ethernet POWERLINK® and EtherCAT® Cycle Times

The used cycle times can be read using object 0x201D (see *chapter 7.6.1 Parameter 51-07 to 51-09: Used Task Cycle Times (0x201D)*). The values are given in microseconds.

There are 2 control parameter sets in the servo drive, however only 1 of them can be active at any time. Use bit 15 (cs) in the *Controlword* to switch from 1 set to the other.

Linear blending occurs from the parameter of the currently active set to the new one. The blending time is defined in object 0x201B (see *chapter 7.6.2 Parameter 51-01: Control Parameter Blending Time (0x201B)*).

No blending takes place when writing to a value of the currently active control parameter set. The new value is used immediately, which could cause a jerk on the shaft.

Blending is used when updating a whole set of parameters at the same time (for example, when activating *CAM mode*, which uses its own sets of control parameters).

2

2.3.6.1 Position Controller

The controller uses PD control. The D constant is the derivative time constant. The controller provides 2 sets of control parameters that can be switched during operation (see *chapter 7.7.8 Parameters 51-02, 52-04, and 52-49: Application Settings (0x2016)* and *chapter 7.6.4.2 Parameters 51-26 and 51-27: Position Controller Parameters 2 (0x2015)*).

Both sets are available as read-write objects in the object dictionary. Use a manufacturer-specific bit in the *Controlword* to switch between the 2 sets of parameters.

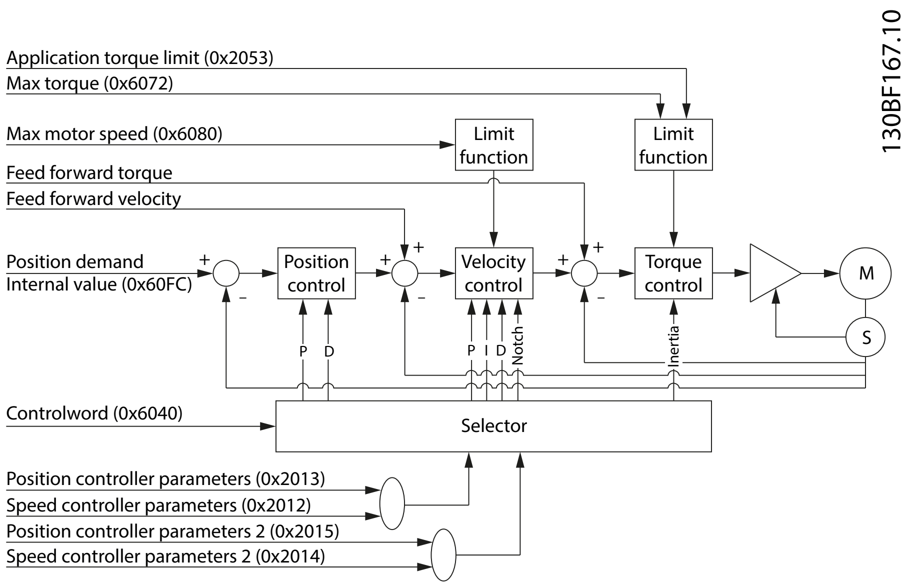


Illustration 2.11 Position Control Loop

2.3.6.2 Speed Controller

The controller uses PID control. The D constant is the derivative time constant. The speed controller has a Notch-Filter (IIR) that can be parameterized (center frequency/bandwidth) to suppress resonance. The controller provides 2 sets of control parameters (see *chapter 7.6.5.1 Parameters 51-10 to 51-15: Speed Controller Parameters (0x2012)* and *chapter 7.6.5.2 Parameters 51-20 to 51-25: Speed Controller Parameters 2 (0x2014)*) that can be switched spontaneously.

Both sets are available as read-write objects in the object dictionary. Use a manufacturer-specific bit in the *Controlword* to switch between the 2 sets of parameters.

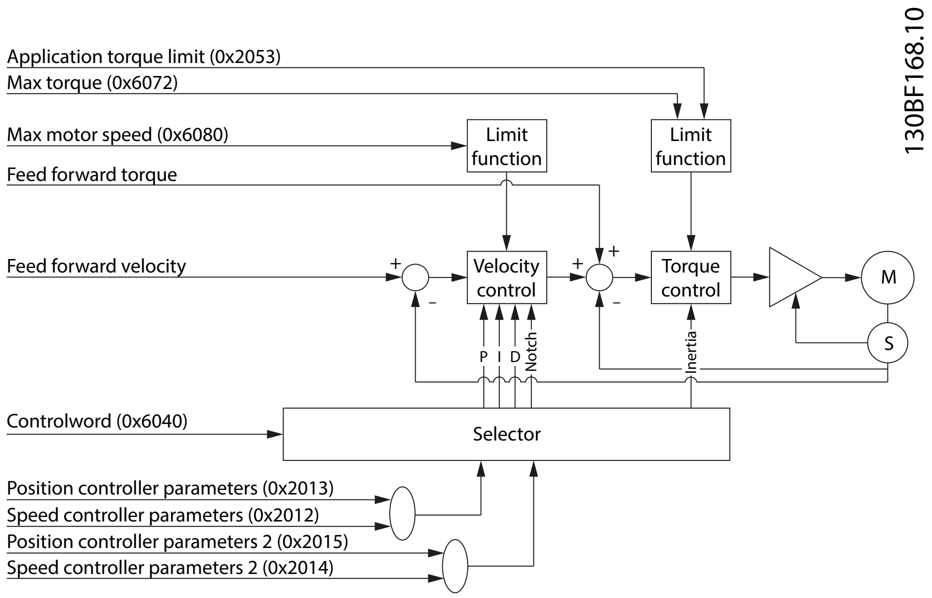


Illustration 2.12 Speed Control Loop

2.3.6.3 Current Controller

The current controller runs synchronous to the fieldbus cycle time. It cannot be parameterized.

2.4 Operating Modes

The servo drive implements several modes of operation. The behavior of the servo drive depends on the activated mode of operation. It is possible to switch between the modes while the servo drive is enabled. The supported modes of operation are according to CANopen® CiA DS402 and there are also ISD-specific modes of operation. All supported modes of operation are available for *EtherCAT®* and Ethernet *POWERLINK®*.

2.4.1 Profile Position Mode

In *Profile position mode*, the servo drive is operated under position control and executes absolute and relative movements. Parameters such as velocity, acceleration, and deceleration can be parameterized. The servo drive provides a buffer to queue a following move while another move is already executing.

This functionality can be commanded using the function blocks MC_MoveAbsolute_ISD51x (see *chapter 6.5.5.4 MC_MoveAbsolute_ISD51x*) and MC_MoveRelative_ISD51x (see *chapter 6.5.5.5 MC_MoveRelative_ISD51x*). This functionality can also be used via the LCP (see section *Position mode* in *chapter 4.3.5.1 Servo Drive*).

When switching to *Profile position mode* from *Profile velocity mode*, *CAM mode*, *Gear mode*, or *Profile torque mode*, the servo drive continues rotating with the current velocity. As soon as there is a new setpoint (handed over using the handshaking between *Controlword* and *Statusword*), the new setpoint is processed with the corresponding parameters.

When switching from a torque or velocity controlled mode to *Profile position mode*, the last target position is set to the position actual value. This is relevant when starting a relative movement from the last target position after switching to this mode, because no last target position from the previous mode is available. If the previous mode ended with a velocity unequal to 0, the last target position is the position actual value at the time of the mode switch.

If the trajectory is completed (target position is reached) and the end velocity (see *chapter 7.10.2 Parameter 52-16: End Velocity (0x6082)*) is unequal to 0, the servo drive continues rotating at the specified end velocity until a further trajectory is set.

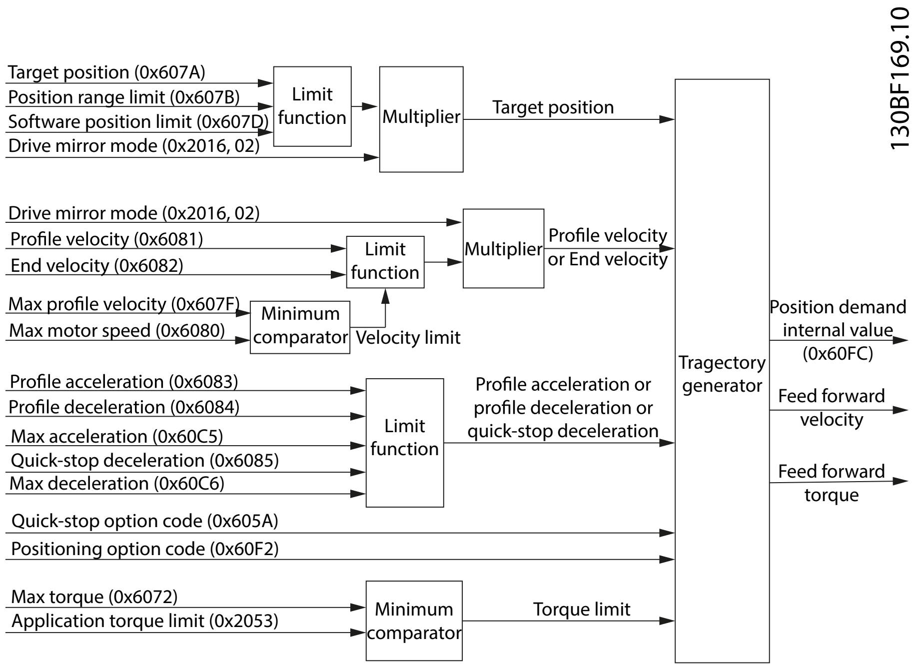


Illustration 2.13 Profile Position Mode Control Function

Target position activation

The activation of a setpoint is controlled by the timing of:

- The *new setpoint* bit and the *change set immediately* bit in the *Controlword*.
- The *setpoint acknowledge* bit in the *Statusword*.

If the *Change set immediately* bit of the *Controlword* is set to *1*, a potentially ongoing motion is interrupted and the new setpoint is used immediately. If the *Change set immediately* bit of the *Controlword* is set to *0*, the ongoing positioning command is finished first and the new setpoint is executed afterwards.

After a setpoint is applied to the servo drive, the control device signals that the setpoint is valid by a rising edge of the *new setpoint* bit in the *Controlword*. The servo drive sets the *setpoint acknowledge* bit in the *Statusword* to *1*. Afterwards, the servo drive with the *setpoint acknowledge* bit set to *0* signals its ability to accept new setpoints. An example is shown in *Illustration 2.14*.

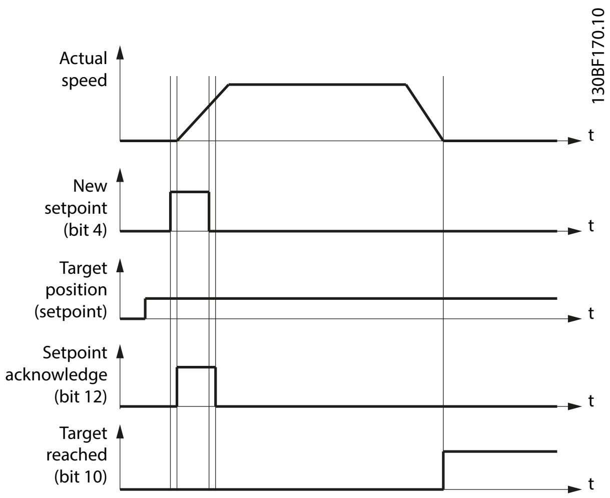


Illustration 2.14 Handshaking Procedure for Setpoint Activation

The servo drive supports 2 setpoints: a setpoint that is currently being processed, and a buffered setpoint. If a setpoint is still in progress (has not been reached) and a new setpoint is activated by the *new setpoint* bit in the *Controlword*, 2 methods of handling are supported. The new setpoint is activated immediately if the *Change set immediately* bit of the *Controlword* is set to *1*. If the *Change set immediately* bit of *Controlword* is set to *0*, the currently active setpoint is finished first and the new setpoint is started afterwards.

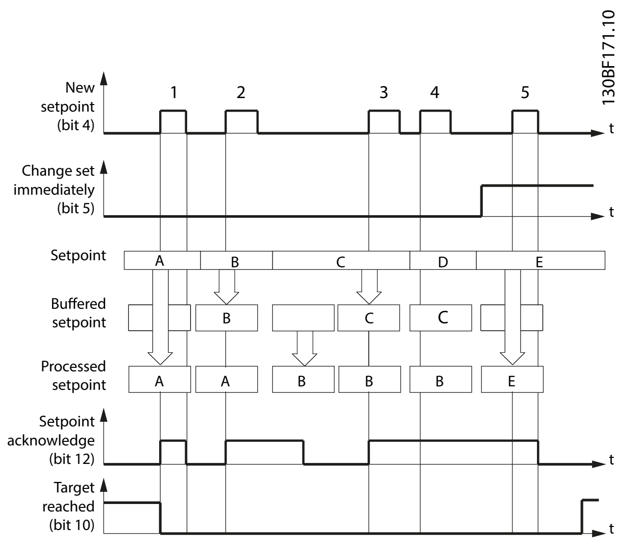


Illustration 2.15 Setpoint Handling for 2 Setpoints

New setpoints are buffered as long as a free setpoint buffer is available in the axis. If no setpoint is in progress, the new setpoint becomes active immediately (case 1 in *Illustration 2.15*). If a setpoint is in progress, the new setpoint is stored in the setpoint buffer (cases 2 and 3 in *Illustration 2.15*).

If all setpoint buffers are busy (*Setpoint acknowledge* bit is set to *1*), the reaction depends on the *Change set immediately* bit. If the *Change set immediately* bit is set to *0*, the new setpoint is rejected (case 4 in *Illustration 2.15*). If the *Change set immediately* bit is set to *1*, the new setpoint is processed immediately. The currently running setpoint profile is discarded (case 5 in *Illustration 2.15*).

The *Target reached* bit in the *Statusword* remains as *0* until all setpoints are processed.

The *Buffered setpoint* is not available as an object for readout.

When a setpoint is in progress and a new setpoint is set to start afterwards (*New setpoint* bit is set to *0*), the new setpoint is only processed after the previous setpoint has been reached. The handshaking procedure shown in *Illustration 2.16* is used for this scenario. The additional gray line in the graph *Actual speed* shows the actual speed if the *Change of setpoint* bit (bit 9 in the *Controlword*) is set to *1*.

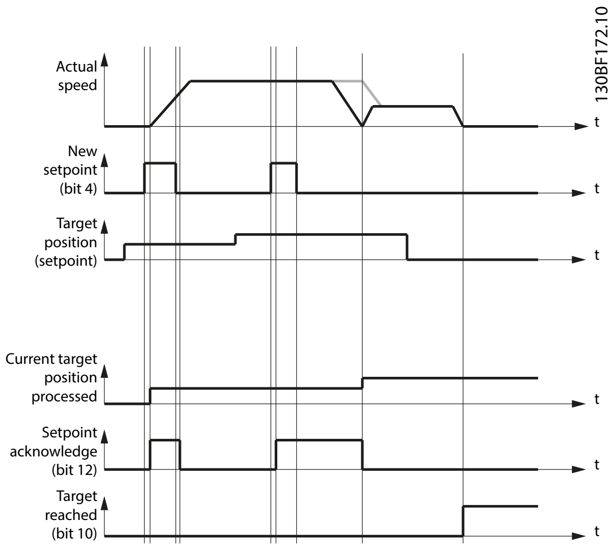


Illustration 2.16 Influence of Change of Setpoint Bit in Profile Position Mode

Position reached function

The position reached function offers the possibility to define a range around a *position demand value* to be regarded as valid. If the position of the servo drive is within this area for a specified time (the *position window time*), the related control bit *Target reached* (bit 10) in the *Statusword* is set to *1*.

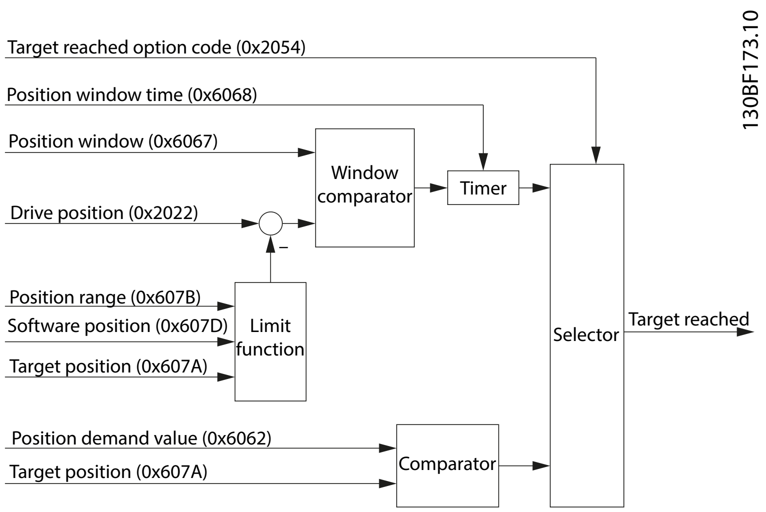


Illustration 2.17 Position Reached – Functional Overview

Illustration 2.18 shows the definition of the sub-function position reached. A window is defined for the accepted position range symmetrically around the *target position*. If a servo drive is situated in the accepted position range over the time *position window time*, the bit *Target reached* (bit 10) in the *Statusword* is set to *1*.

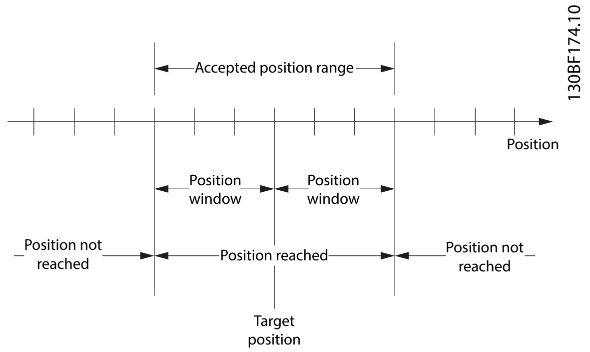


Illustration 2.18 Position Reached Window

2.4.2 Profile Velocity Mode

In *Profile velocity mode*, the servo drive is operated under velocity control and executes a movement with a defined velocity (see *chapter 7.11.1 Parameter 52-20: Target Velocity (0x60FF)*). Parameters such as acceleration (see *chapter 7.5.7 Parameter 50-11: Profile Acceleration (0x6083)*) and deceleration (see *chapter 7.5.8 Parameter 50-12: Profile Deceleration (0x6084)*) can be parameterized. Parameters that influence the *Profile velocity mode* can be found in *Illustration 2.19*.

This functionality can be commanded using function block *MC_MoveVelocity_ISD51X* (see *chapter 6.5.5.7 MC_MoveVelocity_ISD51x*). This functionality can also be used via the LCP (see the *Velocity mode* section in *chapter 4.3.5.1 Servo Drive*). In *Profile velocity mode*, the velocity control loop is used to reach the target velocity (see *chapter 7.11.1 Parameter 52-20: Target Velocity (0x60FF)*).

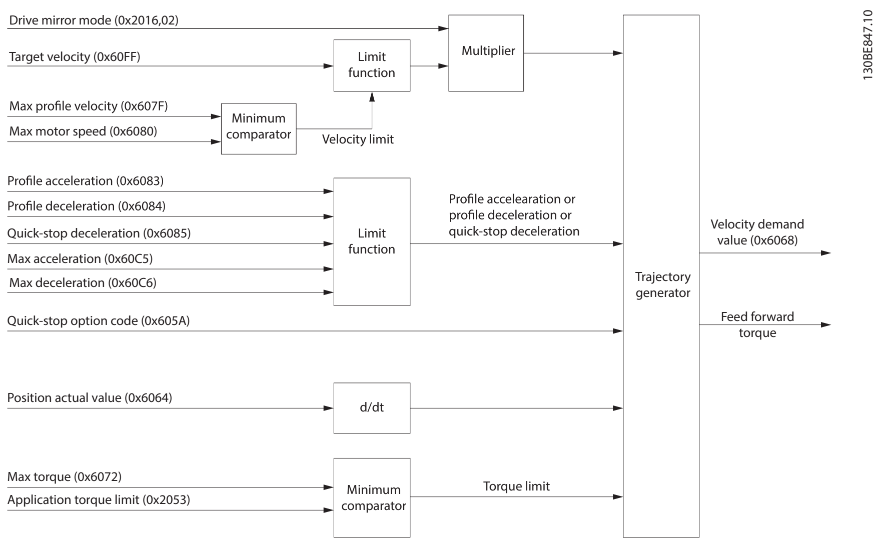


Illustration 2.19 Profile Velocity Mode Control Function

The usage of acceleration and deceleration for the calculation of the trajectory is shown in *Illustration 2.20*.

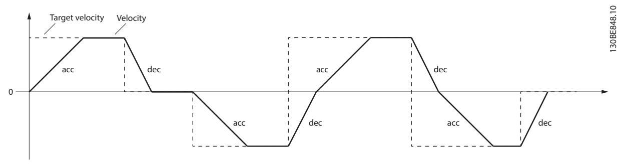


Illustration 2.20 Usage of Acceleration and Deceleration in Velocity Control

This principle on using the acceleration and deceleration value applies to all velocity controlled modes of operation. The ramp bends when reversing the velocity. If this behavior is undesired, set the value of the acceleration and deceleration to the same value.

Velocity reached function

The velocity reached function offers the possibility to define a velocity range around a velocity demand value to be regarded as valid. If the velocity of the servo drive is within this area for a specified time (see *chapter 7.11.4 Parameter: Velocity Window (0x606D)*), the velocity window time (see *chapter 7.11.5 Parameter: Velocity Window Time (0x606E)*), the related control bit *Target reached* (bit 10) in the *Statusword* is set to *1*.

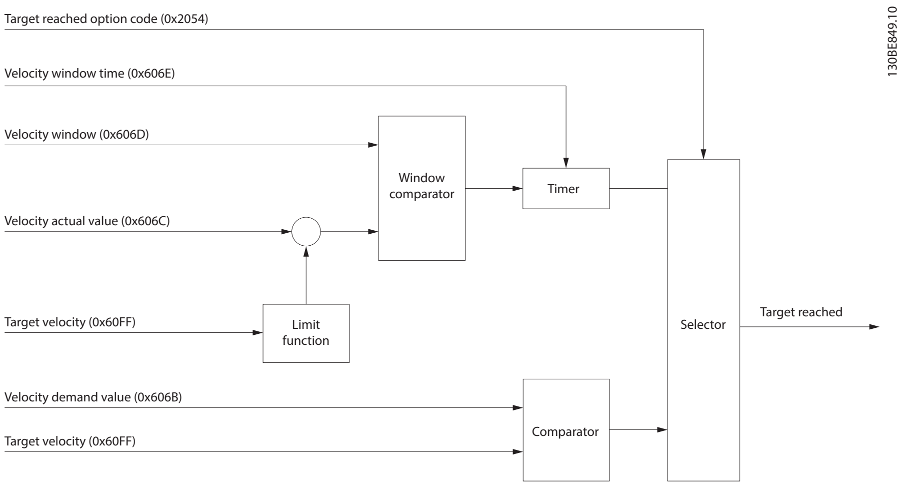


Illustration 2.21 Velocity Reached - Functional Overview

Illustration 2.22 shows the definitions of the sub-function *Velocity reached*. A window is defined for the accepted velocity range symmetrically around the velocity. If a servo drive is running within the accepted velocity range over the time velocity window time, the bit *Target reached* (bit 10) in the *Statusword* is set to *1*.

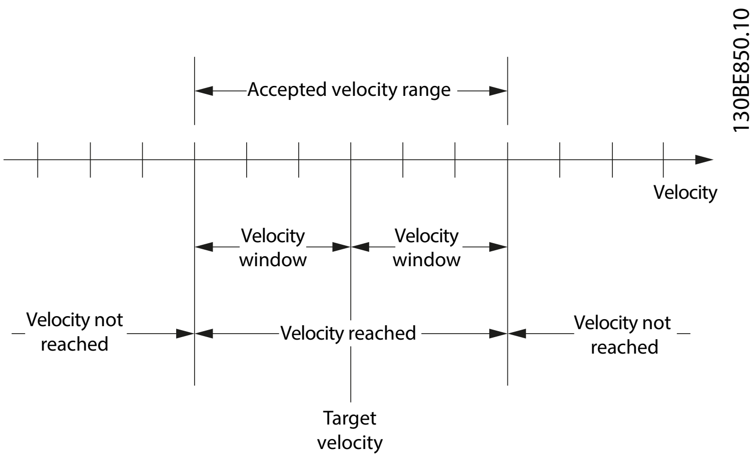


Illustration 2.22 Velocity Reached Window

2.4.3 Profile Torque Mode

In *Profile torque mode*, the servo drive is operated under torque control and executes a movement with constant torque. Linear ramps are used. Additional parameters, such as the torque ramp and maximum velocity can be parameterized. This functionality can be commanded using function block *MC_TorqueControl_ISD51X* (see *chapter 6.5.5.8 MC_Torque-Control_ISD51x*).

The *Profile torque mode* allows transmitting the target torque value (see *chapter 7.12.1 Parameter 52-30: Target Torque (0x6071)*), which is processed via the trajectory generator. The torque slope (see *chapter 7.12.7 Parameter 52-32: Torque Slope (0x6087)*) is required. The servo drive supports linear ramps for calculation of the trajectory generation. If the *Controlword* bit 8 (Halt) is switched from *0* to *1*, or from *1* to *0*, then the trajectory generator ramps its control effort output down to *0*, or up to the target torque. In both cases, the trajectory generator uses the torque slope for the ramp calculation.

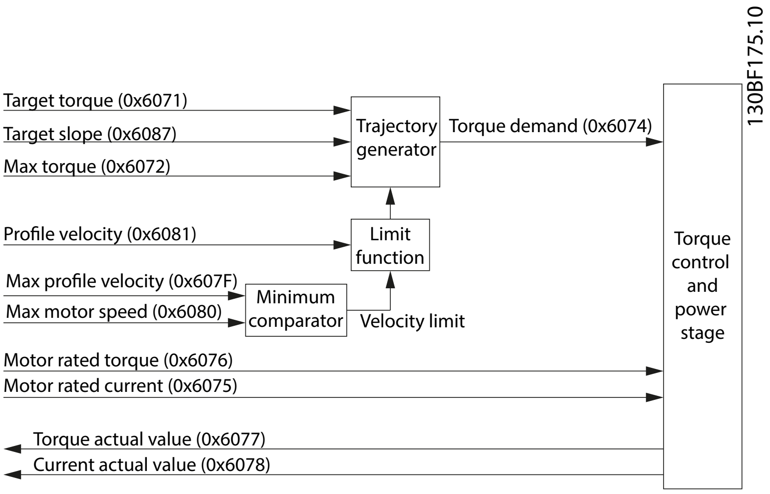


Illustration 2.23 Profile Torque Mode Control Function

Torque reached function

The *Torque reached* function offers the possibility to define a torque range around a torque demand value to be regarded as valid. If the torque of the servo drive is within this area (see *chapter 7.12.8 Parameter: Torque Window (0x2050)*) for a specified time, the torque window time (see *chapter 7.12.9 Parameter: Torque Window Time (0x2051)*) and the related control bit 10 *Target reached*, in the *Statusword* is set to *1*.

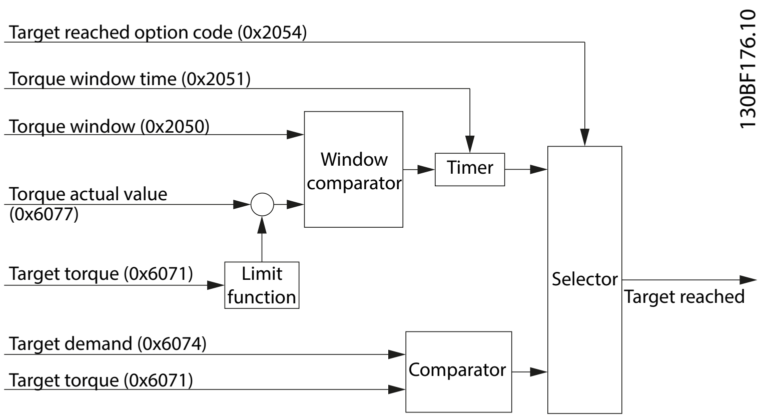


Illustration 2.24 Torque Reached - Functional Overview

Illustration 2.25 shows the definitions of the sub-function *Torque reached*. A window is defined for the accepted torque range symmetrically around the velocity. If a servo drive is running within the accepted torque range over the time torque window time, the bit target reached (bit 10) in the *Statusword* is set to 1.

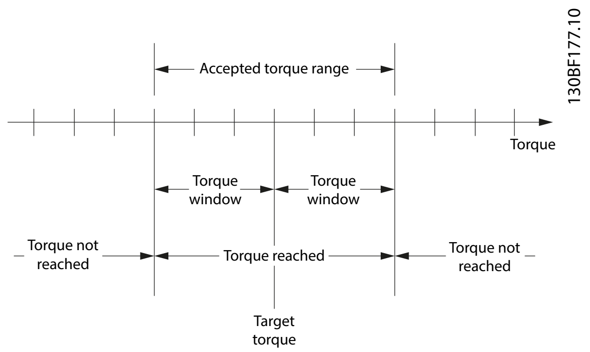


Illustration 2.25 Torque Reached Window

2.4.4 Homing Mode

In *Homing mode*, the application reference position of the servo drive can be set. Several homing methods, described in this chapter, are available.

This functionality can be commanded using MC_Home_ISD51x (see *chapter 6.5.5.1 MC_Home_ISD51x*).

The *home position* is the position where an event was triggered. The type of event depends on the homing method (for example, detection of an edge of a switch). Based on this *home position* and the *home offset* (see *chapter 7.13.1 Parameter 52-40: Home Offset (0x607C)*), the new *zero position* is calculated (see *Illustration 2.26*).

Illustration 2.26 Home Offset Definition

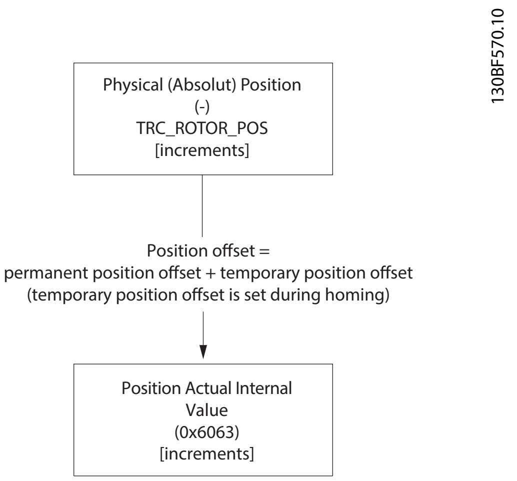


Illustration 2.27 Position Offset Definition

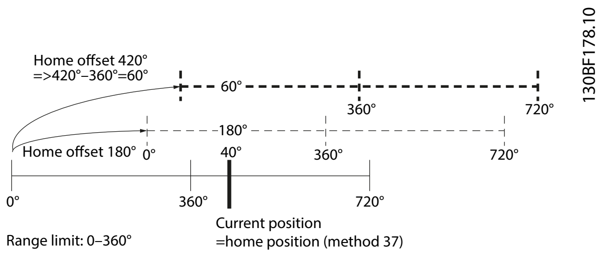


Illustration 2.28 Behavior of Homing with Software Range Limit Applied

In *Illustration 2.28*, the lowermost solid line shows the current physical position of the servo drive. The software range limit is applied so that the servo drive shows position actual values between 0° and 360°. The bold vertical line shows the current/ reference position, where the servo drive shows 40°. The fine dashed line in the middle shows the situation when activating homing method 37 (*Homing on current position*) with a value for the home offset (0x607C) of 180°. The position actual value (0x6064) shows 180°. The multi-turn revolutions are discarded. The bold dashed line at the top shows the situation when activating homing method 37 (*Homing on current position*) with a value for the home offset (0x607C) of 420°. The position actual value (0x6064) shows 60°. The multiples of the software range limit from the home offset are discarded.

The reference position found during homing is lost after a reset. However, it is possible to save this reference position permanently (see sub-index 3 in *chapter 7.7.8 Parameters 51-02, 52-04, and 52-49: Application Settings (0x2016)* for details). The homing bit is not set after a power-cycle, however the position is preserved.

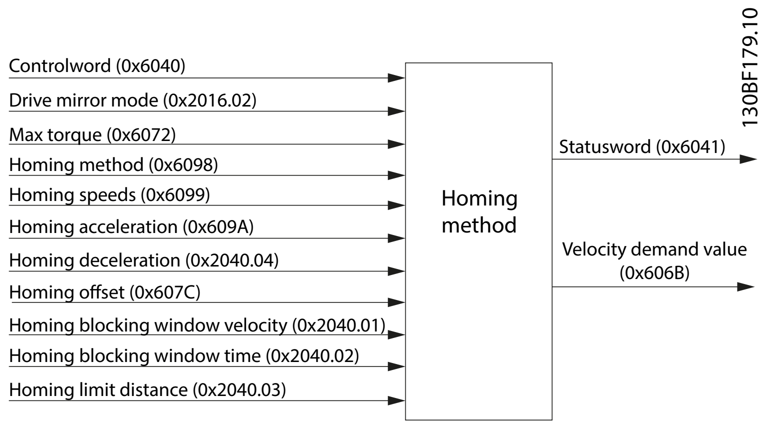


Illustration 2.29 Homing Mode Function

The homing methods that require a physical input (home switch or limit switches) are available depending on the configuration of the analog inputs (see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*).

The methods are described in detail in the corresponding sub-chapters.

Value	Definition
–3	Homing on actual position.
–2	Homing on positive block.
–1	Homing on negative block.
+17	Homing on negative limit switch.
+18	Homing on positive limit switch.
+19	Homing on positive home switch.
+21	Homing on negative home switch.
+37	Homing on current position.

Table 2.4 Supported Homing Methods

The successful completion of a homing procedure is indicated by bit 8 of the *Statusword* (home bit). This bit remains set until the servo drive is power-cycled (U_{AUX}), reset, or a new homing procedure is started.

Switching to *Homing mode* while the servo drive is in state *Operation enabled* is only allowed in standstill. In all other states, the mode of operation can always be changed.

Exiting *Homing mode* (and switching to any other mode of operation) is allowed without restrictions. If homing procedure is being carried out at that time, it is automatically aborted. In this case, the home bit in the *Statusword* is not set.

2.4.4.1 Homing on Actual Position

In method –3 *Homing on actual position*, the temporary part of the position offset is set to *0* (see *Illustration 2.26*). This method does not require the servo drive to be in state *Operation enabled*, as there is no movement. If the servo drive is in state *Operation enabled* during activation, it must be in standstill.

2.4.4.2 Homing on Positive/Negative Block

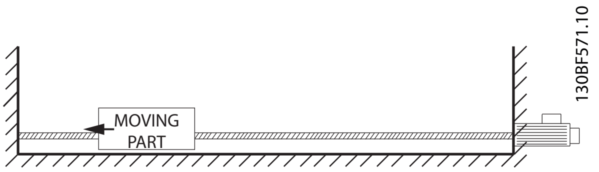


Illustration 2.30 Example of Homing Method on Block

Method –1 *Homing on negative block* and method –2 *Homing on positive block* perform a homing against a physical object that mechanically blocks the movement. A limit switch or home switch is not required.

NOTICE

An inadequate torque limit during the homing process may result in damage to mechanics.

The servo drive is considered as blocked if the actual speed falls below the *Homing blocking window velocity* for the specified *Homing blocking window time* (see *chapter 7.13.6 Parameter 52-45 to 52-48: Additional Homing objects (0x2040)*) and the torque limit is reached (see *chapter 7.5.12 Parameter: Maximum Torque (0x6072)* and *chapter 7.5.13 Parameters 52-15, 52-23, and 52-36: Application Torque Limit (0x2053)*).

When the motor is blocked, the actual position is the home position. The motor then ramps down to 0 velocity using the homing deceleration value and the successful homing procedure is reported.

The differences between the 2 methods are:

- Homing on negative block (–1): Motor moves with negative speed.
- Homing on positive block (–2): Motor moves with positive speed.

2.4.4.3 Homing on Positive/Negative Limit Switch

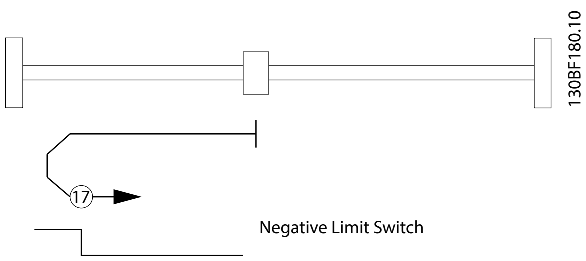


Illustration 2.31 Homing Method 17: Homing on Negative Limit Switch

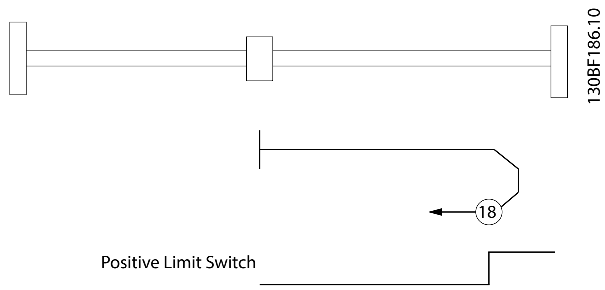


Illustration 2.32 Homing Method 18: Homing on Positive Limit Switch

Homing methods 17: Homing on negative limit switch or 18: Homing on positive limit switch can be used if a limit switch is available (and configured using object 0x200F, see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*), so that the limit switch signals the home reference point.

The differences between the 2 methods are:

- 17: Homing on negative limit switch: Motor moves with negative speed to reach the negative limit switch.
- 18: Homing on positive limit switch: Motor moves with positive speed to reach the positive limit switch.

When starting the homing procedure, the servo drive starts moving with the defined velocity value set in object 0x6099 sub-index 01: Speed during search for switch (see *chapter 7.13.3 Parameters 52-42 and 52-43: Homing Speeds (0x6099)*). The direction depends on the selected method (positive or negative). As soon as a rising edge is detected on the limit switch, the motor reverses direction and ramps to the velocity set in object 0x6099 sub-index 2: Speed during search for zero (see

chapter 7.13.3 Parameters 52-42 and 52-43: Homing Speeds (0x6099)) until the switch is no longer active (falling edge). The home position of the servo drive is at this edge. The motor ramps down to 0 velocity and the successful homing procedure is reported. If the homing procedure is started and the limit switch is already set, the servo drive immediately signals a homing error.

2.4.4.4 Homing on Positive/Negative Home Switch

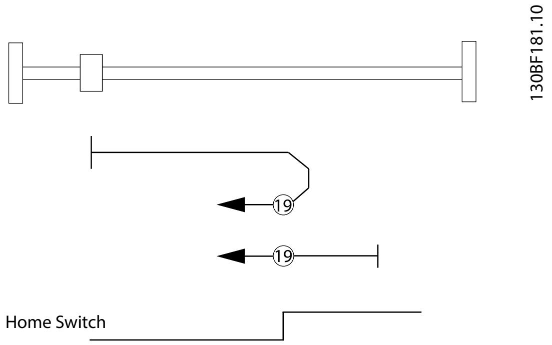


Illustration 2.33 Homing Method 19: Homing on Positive Home Switch

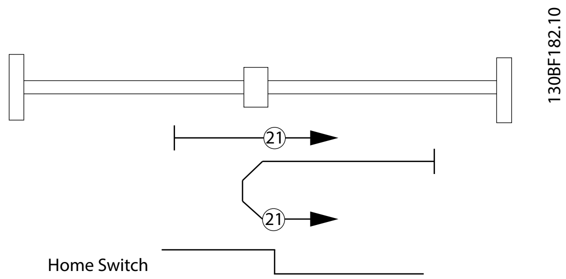


Illustration 2.34 Homing Method 21: Homing on Negative Home Switch

Homing method 19 (positive) or 21 (negative) can be used if a home switch is available and can be configured using object 0x200F (see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*) so that the home switch signals the home reference point.

The initial movement depends on the logical state of the home switch at activation. In all cases, the servo drive turns with the velocity set in object 0x6099, sub-index 01: Speed during search for switch (see *chapter 7.13.3 Parameters 52-42 and 52-43: Homing Speeds (0x6099)*) until it encounters a signal change of the home switch. The servo drive reverses direction and ramps to the velocity set in object 0x6099, sub-index 02: Speed during search for zero until the home switch changes states again. The home position of the servo drive is at this edge. The motor ramps down to 0 velocity and the successful homing procedure is reported.

The differences between the 2 methods are:

- Positive home switch (19): During activation of the homing procedure, a low state of the home switch leads to the servo drive moving in a positive direction. A high state of the home switch leads to the servo drive moving in a negative direction.
- Negative home switch (21): During activation of the homing procedure, a low state of the home switch leads to the servo drive moving in a negative direction. A high state of the home switch leads to the servo drive moving in a positive direction.

2.4.4.5 Homing on Current Position

In this method (37), the current position of the servo drive is used as the home position. This method does not require the servo drive to be in state *Operation enabled* because no movement occurs. If the servo drive is in state *Operation enabled* during activation, it must be in standstill.

At the home position, the *Position offset* is calculated so that the value of the *Position actual value* (see *chapter 7.7.5 Parameter 50-03: Position Actual Value (0x6064)*) equals the *Home offset* (see *chapter 7.13.1 Parameter 52-40: Home Offset (0x607C)*):

Position actual value (0x6064) = Home offset (0x607C)

If the value of the *Home offset* is higher than the *Position range limit*, only the modulo part is used.

2.4.4.6 Error Behavior in Homing Mode

For the methods where *Homing limit distance* (see sub-index 03: in *chapter 7.13.6 Parameter 52-45 to 52-48: Additional Homing objects (0x2040)*) is used for supervising, the following error behavior applies:

If the limit is exceeded, the homing procedure is aborted. The servo drive signals a *Homing error*. If the servo drive is in motion at this point of time, it ramps down with the quick stop deceleration (see *chapter 7.5.9 Parameter 50-13: Quick Stop Deceleration (0x6085)*) to standstill but stays in state *Operation enabled*. Additionally, a warning is issued (Warning bit in *Statusword* and setting of warning code).

The following situations can also lead to a warning:

- Entering *Homing mode* when not in standstill.
- Starting a homing procedure while not in standstill. A warning and a homing error are reported. If the axis reaches standstill, the homing method is started.
- If the homing procedure reaches the homing distance, the homing is aborted and a warning is reported.

2.4.5 CAM Mode

In *CAM mode*, the servo drive executes a synchronized movement based on a master axis (guide value). The synchronization takes place by means of a CAM profile that contains slave positions corresponding to master positions. CAMs are designed with either the CAM Editor of the ISD Toolbox *chapter 5.7.7 CAM Editor (Servo Drive only)* or by using special structures in the PLC library *chapter 6.5.7 Drive – CAM Creation*. The guide value can be provided by an external encoder, virtual axis, or the position of another axis.

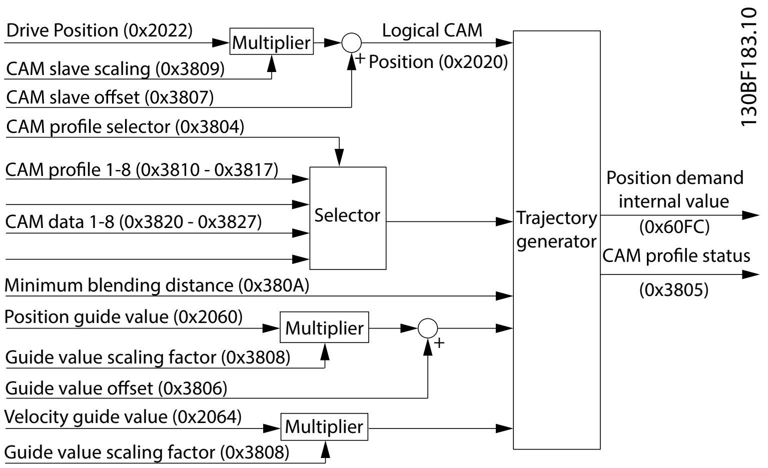


Illustration 2.35 Inputs for CAM Mode

When switching to *CAM mode* when the servo drive is not in standstill, it continues rotating with its current velocity. As soon as a new CAM profile is activated, the new CAM profile is processed with the corresponding behavior. The servo drive can hold a maximum of 8 CAM profiles (see *chapter 7.14.4 Parameters: CAM Profile 1–8 (0x3810–0x3817)*). A CAM profile consists of the CAM itself and its CAM configuration. CAM profiles are automatically stored inside the servo drive.

There are 2 types of CAMs:

- Basic CAM

  A basic CAM is a list of data points that describe the relationship between the slave position and the master position. Each data point consists of:
  - Master position
  - Slave position
  - Slave velocity
  - Slave acceleration
- Advanced CAM

  An advanced CAM is represented by nodes, segments, actions, and exit conditions. There are different segment types that each have a special functionality to provide intelligent application functionality within the servo drive.

The CAM is represented in an XML file, which contains the following information:

- Master scaling (optional)
- Slave scaling (optional)
- Control loop parameter (optional; both sets)
- Following error settings (optional)
- Basic cam or advanced cam definition

The CAM configuration consists of the following information:

- Cyclic/non-cyclic
- Master absolute/relative
- Slave absolute/ relative (only applicable for basic CAM)

For general information about the format of an XML file, see *chapter 10.2 General XML Conventions*.
When parsing a CAM profile, the servo drive checks the format and the plausibility. The result is available in the sub-indexes 3 and 4 of the same objects (see *chapter 7.14.4 Parameters: CAM Profile 1–8 (0x3810–0x3817)*). These sub-indexes contain detailed information about the parsing state, error result, and more detailed information for debugging.

The factor group (see *chapter 2.3.2 Factor Group*) is not used in *CAM mode*. The velocity must be given as a unitless factor between rotor angle and angle of guide value. The acceleration must be given as velocity per degree of guide value. A CAM profile is running based on the guide value. This value always runs from *0* to *1*. To adjust this to the real application environment, it is possible to specify a factor (master scaling) to reduce or increase the value that is considered as a full cycle.

There are 2 possible CAM buffer layouts available:

- 8 CAM profiles
- 2 CAM profiles with more data

The CAM buffer layout can be selected in object 0x380F (see *chapter 7.14.1 Parameter: CAM Profile Memory Layout (0x380F)*). Carry out a power-cycle to activate the selection.
All nodes are non-signaling nodes. The axis does not automatically signal if it passes a node. However, for example for debugging purpose, it is possible to enable this signaling for selected nodes.

NOTICE

The factor group (feed constant, gear ratio, and so on) has no effect in CAM mode.

Terminology

Name	Description
CAM profile	Consists of 1 CAM and 1 CAM configuration. A valid CAM profile is automatically stored in the servo drive (maximum 8 CAM profiles).
CAM	XML file (basic CAM or advanced CAM) containing the data points or the nodes and segments.
CAM configuration	Contains the following information: - Cyclic/non-cyclic - Master absolute/relative - Slave absolute/relative
Basic CAM	List of data points that describe the relationship between the slave position and the master position.
Advanced CAM	Describes the relationship between the slave and the master based on nodes, segments, actions, and exit conditions.
CAM profile activation request (Handshaking)	Handshaking procedure with *Controlword* and *Statusword* to activate a valid CAM profile. This does not necessarily mean that the profile starts immediately. This depends on the CAM configuration, time of CAM activation request, and so on.
Change CAM immediate/ delayed	In the *Controlword* there are 2 options for changing the CAM: - Immediate: Abort the currently running CAM and immediately change to the new CAM. - Delayed: The currently running CAM profile finishes first before the CAM is activated (see *Illustration 2.37*).
CAM profile activation	All setpoints of the CAM profile are calculated and the servo drive is able to run the CAM profile. This mainly contains the calculations for blending.
Full CAM	A basic CAM that is defined with the 1st data point at guide value *0* and last data point at guide value *1*. The CAM is defined over the whole guide value cycle. Not applicable for advanced CAM.

Name	Description
Partial CAM	A CAM that is defined with the 1st data point not at guide value *0* or last data point not at guide value *1* (or both). The CAM is only defined on part of the guide value cycle. Parts of the guide value are "undefined". Not applicable for advanced CAM.
P5	Polynomial of 5th degree.
End of profile	Output signaling the end of the CAM profile. For cyclic processing of the CAM, it is displayed every time the end of the CAM profile is reached. This signal is only high for 1 fieldbus cycle. For basic CAMs, the end of profile is signaled at the last data point. For advanced CAMs, the end of profile is signaled at each end node.
InSync	Output *InSync* is high as long as the slave follows the commanded CAM profile.
Blending	Blending occurs whenever the servo drive automatically calculates a P5 when switching between CAMs, or it is used to fill up the undefined parts in cyclic processing of CAMs.

Table 2.5 Terminology

2.4.5.1 Activating a CAM profile

Perform the following steps to activate a CAM profile:

1. Write the CAM data to 1 of the objects 0x3820–0x3827: CAM data 1–8 (see *chapter 7.14.5 Parameters: CAM Data 1–8 (0x3820–3827)*).
2. Write the CAM configuration and activate the CAM parsing to the corresponding object 0x3810–0x3817: CAM profile, sub-index 01 (see *chapter 7.14.4 Parameters: CAM Profile 1–8 (0x3810–0x3817)*).
3. Check the CAM parsing state in objects 0x3810–0x3817: CAM profile, sub-index 02 and 03 (see *chapter 7.14.4 Parameters: CAM Profile 1–8 (0x3810–0x3817)*).
4. Write the number of the CAM and the delay code that should be used into object 0x3804: CAM profile selector (see *chapter 7.14.7 Parameter: CAM Profile Selector (0x3804)*.
5. Switch to *CAM mode* (this can also be done earlier).
6. Perform handshaking to send the CAM activation request.

To transfer a CAM profile, use function block *MC_CamTable-Select_ISD51x* (see *chapter 6.5.6.1 MC_CamTableSelect_ISD51x*).

CAM profile activation request (Handshaking)

The activation of a CAM profile is controlled by the timing of the *New CAM* bit in the *Controlword*, and the *CAM ack* bit in the *Statusword*. After a CAM profile is transferred and successfully parsed, the control device signals that the CAM profile will be activated (CAM profile activation request) by a rising edge of the *New CAM* bit in the *Controlword*. The axis internally calculates all necessary parameters and afterwards sets the *CAM ack* bit in the *Statusword* to *1*. With the *CAM ack* bit set to *0*, the axis signals its ability to accept new CAM profiles. An example is shown in *Illustration 2.36*. After activation of the CAM profile, the CAM is not necessarily executed immediately. This depends on the CAM configuration and the change immediate bit in the *Controlword*.

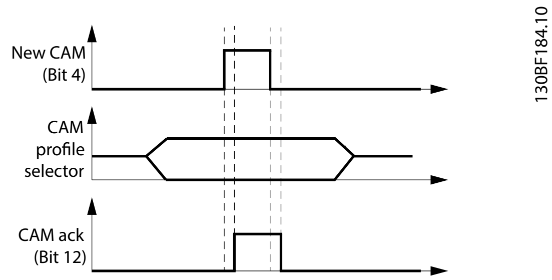


Illustration 2.36 Handshaking Procedure for CAM Profile Activation

The CAM profile can also be activated using function block *MC_CamIn_ISD51x* (see *chapter 6.5.6.2 MC_CamIn_ISD51x*).

The axis supports a set of 2 CAM profiles numbers: a CAM profile that is currently being processed, and a buffered profile.
If a CAM profile is still in progress and a new CAM profile is validated by the new CAM (bit 4) in the *Controlword*, 2 methods of handling are supported:

- The new CAM profile is activated immediately (*Change CAM immediately* bit of the *Controlword* is set to *1*).
- The currently active CAM profile is finished first and afterwards the new CAM profile is started (*Change CAM immediately* bit of the *Controlword* is set to *0*).

When a new CAM profile is activated, all specific parameters are activated at the start of the new profile (this is the beginning of the blending). This can lead to jumps in position and velocity, for example when using different master scaling values.

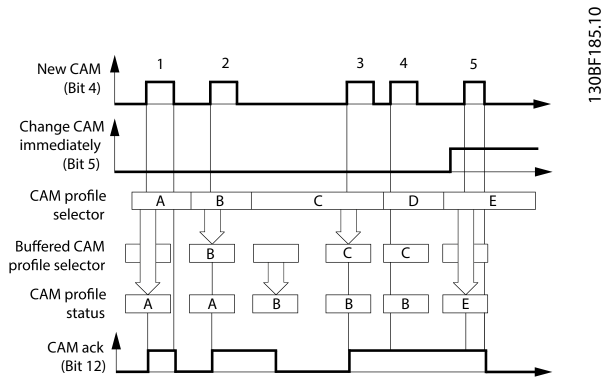


Illustration 2.37 CAM Profile Handling for 2 CAM Profiles

New CAM profile numbers are buffered in the buffered CAM profile selector as long as there is a free CAM profile selector buffer available in the axis. If no CAM is in progress, the new CAM profile becomes active immediately (case 1 in *Illustration 2.37*).
If a CAM profile is in progress, the new CAM profile number is stored in the CAM profile buffer (cases 2 and 3 in *Illustration 2.37*). If all profile number buffers are busy (*CAM ack* bit is 1), the reaction depends on the *Change CAM immediately* bit. If the *Change CAM immediately* bit is set to *0*, the new CAM profile is rejected (case 4) with a command error indication (*Statusword*). If the Change *CAM immediately bit* is set to *1*, the new CAM profile number is processed immediately. The currently running CAM profile is discarded (case 5 in *Illustration 2.37*).
The *Buffered CAM profile selector* is not available as an object for readout. There are cases where it is necessary to do a compensation movement when switching between CAMs. This movement is called blending and it is calculated automatically by the servo drive. The blending takes place using a polynomial of 5th degree.

2.4.5.2 CAM Configuration: Master Absolute/Relative

If the master and slave positions are configured to be absolute positions, it is necessary to have a synchronization movement that aligns the position at the point of activation with the set-position of the profile. This is called blending. For blending, a polynomial of 5th degree is used. It is automatically calculated by the servo drive.

The blending can be influenced using bit *Use blend distance*. When set to *0*, the blending is done to the 1st data point of a basic CAM, or the start node of an advanced CAM. This distance can be very short, which leads to high velocity or acceleration.

When a concrete blend distance is used, set the *Use blend distance* bit. Then, the value given in the minimum blending object 0x380A (see *chapter 7.14.11 Parameter: Minimum Blending Distance (0x380A)*) is used to calculate a synchronization movement within the axis. This distance should be regarded as a minimum value, as there are situations where the servo drive automatically enlarges this distance (for example, if the end of the distance does not lead to a point of a defined CAM, see *Illustration 2.50*).
When using non-cyclic CAM profiles, the influence of *Master relative* versus *Master absolute* is only an offset in guide value direction. This is dependent on the point of activation of the CAM profile.

2.4.5.3 CAM Header Information

All parameters defined in this header information have corresponding parameters in the object dictionary. These objects are updated at the point of activation of the CAM. If an element is not included in the header (which is allowed for optional elements), the parameter in the object dictionary remains unchanged. When leaving a CAM, the values in the object dictionary persist; so they are not switched back to their old values before the CAM activation. The header information is the same for both CAM types.

```
<?xml version="1.0"?>
<CamProfile version="0.0.0.1">
    <masterScaling numerator="1" denominator="2" />
    <slaveScaling  numerator="1" denominator="2" />

    <controlParam1 speedP="0.1" speedI="0.1" speedD="0.0"
                  inertia="0.0004" positionP="6" positionD="0" />
    <controlParam2 speedP="0.2" speedI="0.05" speedD="0.0"
                 inertia="0.0008" positionP="12" positionD="0" />
    <followingError windowRev="0.01" time="5" />

…
</CamProfile>
```

Illustration 2.38 CAM Header Information

Each file can only contain 1 *CamProfile* element.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
Version	O	x.x.x.x	Gives the version of the CAM profile definition.

Table 2.6 Attribute for Element CamProfile

The *CamProfile* element contains an optional element *masterScaling* which defines the length of a guide value cycle. This parameter is used as scaling factor. If this element is missing, the values from the object dictionary are used (see *chapter 7.8.4 Parameter: Guide Value Scaling Factor (0x3808)*).

2

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
numerator	M	Same as for object 0x3808, sub-index 1.	See object 0x3808, sub-index 1.
denominator	M	Same as for object 0x3808, sub-index 2.	See object 0x3808, sub-index 2.

Table 2.7 Attributes for Element masterScaling

The *CamProfile* element contains an optional element *slaveScaling* which defines the scaling factor for the axis. If this element is missing, the values from the object dictionary are used (see *chapter 7.14.10 Parameter: CAM Slave Scaling (0x3809)*). The same value range applies as described for the objects in the object dictionary.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
numerator	M	Same as for object 0x3809, sub-index 1.	See object 0x3809, sub-index 1.
denominator	M	Same as for object 0x3809, sub-index 2.	See object 0x3809, sub-index 2.

Table 2.8 Attributes for Element slaveScaling

Another optional element is the *controlParam1*, and/or *controlParam2* element, where the control loop parameters are defined. Those 2 elements allow the automatic *overwriting* of the 2 sets of control parameters (in the object dictionary) on activation of the CAM. Both objects are optional and can be present independently of each other.

controlParam1 refers to the 1st set of control parameters (see *chapter 7.6.5.1 Parameters 51-10 to 51-15: Speed Controller Parameters (0x2012)* and *chapter 7.6.4.1 Parameters 51-16 and 51-17: Position Controller Parameters (0x2013)*), whereas *controlParam2* refers to the 2nd set of control parameters (see *chapter 7.6.5.2 Parameters 51-20 to 51-25: Speed Controller Parameters 2 (0x2014)* and *chapter 7.6.4.2 Parameters 51-26 and 51-27: Position Controller Parameters 2 (0x2015)*).

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
speedP, speedI, speedD, inertia	M	Float, same as for object 0x2012.	See object 0x2012.
positionP, positionD	M	Float, same as for object 0x2013.	See object 0x2013.

Table 2.9 Attributes for Element controlParam1

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
speedP, speedI, speedD, inertia	M	Float, same as for object 0x2014.	See object 0x2014.
positionP, positionD	M	Float, same as for object 0x2015.	See object 0x2015.

Table 2.10 Attributes for Element controlParam2

The optional element *followingError* defines the following error settings for the CAM. The mandatory attribute *windowRev* refers to object 0x6065 (see *chapter 7.22.1.1 Parameter: Following Error Window (0x6065)*), but the value must be given in revolutions. The mandatory attribute time gives the time in milliseconds (see *chapter 7.22.1.2 Parameter: Following Error Time Out (0x6066)*). The following error behavior applies here.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
windowRev	M	Float	See object 0x6065 for description, however this value must be given in revolutions. The servo drive automatically recalculates the value to the value required for object 0x6065.
time	M	Same as for object 0x6066.	See object 0x6066.

Table 2.11 Attributes for Element followingError

The rest of the *CamProfile* element depends on the profile type (basic or advanced) and is described in the following chapters.

2.4.5.4 Basic CAM

Data points

The basic CAM consists of data points, which all have the same structure. There can be a maximum of 256 or 1024 data points inside 1 basic CAM (see *chapter 7.14.1 Parameter: CAM Profile Memory Layout (0x380F)*).

```
<basicCam>
      <dataPoint  masterPos="0.0278"  slavePos="0.08333"  vel="0"    acc="0" />
      <dataPoint  masterPos="0.0833"  slavePos="0.25"     vel="1"    acc="0" />
      <dataPoint  masterPos="0.1111"  slavePos="0.1667"   vel="0.5"  acc="0" />
      <dataPoint  masterPos="0.1667"  slavePos="0.0417"   vel="0"    acc="0" />
      …
</basicCam>
```

Illustration 2.39 Basic CAM Data Points

Attribute	Mandatory/optional (+default value)	Value range/allowed values	Description
masterPos	M	Float: [0;1]	Master position for this data point. Given in revolutions of guide value. The *masterPos* inside a CAM profile is always defined from *0* to *1*.
slavePos	M	Float	Axis position for this data point. Given in revolutions of rotor position. *SlavePos* describes the position on the motor side.
vel	O; default = 0	Float	Velocity of the axis in this data point. The velocity must be given as a factor between the velocity of the axis in relation to the velocity of the guide value (1 revolution of the axis per 1 round of guide value). Jumps in velocity are not possible.
acc	O; default = 0	Float	Acceleration of the axis in this data point. The acceleration must be given as a factor between the acceleration of the axis in relation to the velocity of the guide value (1 revolution of axis per square of round of guide value). Jumps in acceleration are not possible.

Table 2.12 Attributes for a Data Point

All data points are non-signaling data points. The axis does not automatically signal if it passes a data point. However, it is possible to enable this signaling for selected data points, for example for debugging purposes.

CAM configuration: Slave absolute/relative

When using the slave absolute option for a basic CAM, the values of the *slavePos* attribute in the data point are used. When using the slave relative option, the start of the CAM is transferred to the current position of the slave.

CAM configuration: cyclic/non-cyclic

The different configurations are explained using illustrations. There are 3 basic CAMs defined to show all situations:

- *Illustration 2.40* shows a full CAM (1st data point at *masterPos* 0, last data point at *masterPos* 1), which has a velocity unequal to *0* in the 1st data point.
- *Illustration 2.41* and *Illustration 2.42* show partial CAMs (1st data point not at *masterPos* 0, last data point not at *masterPos* 1).

In the following chapters, several situations are defined. The mentioned CAMs are used throughout the description of the basic CAM to cover all situations.

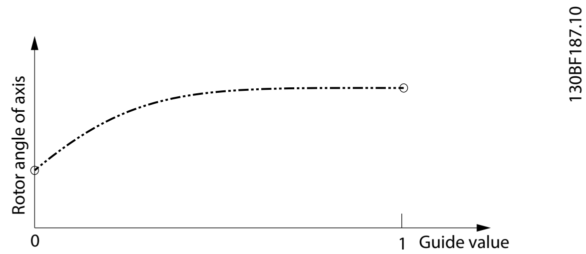


Illustration 2.40 CAM 1 - Full CAM with Velocity of Last Node/ point = 0

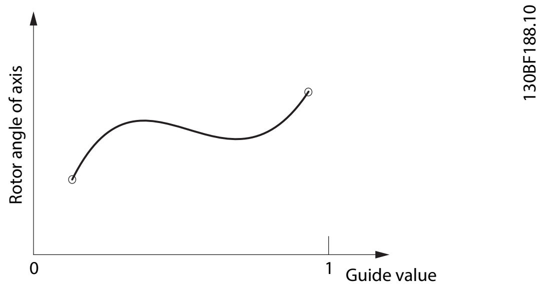


Illustration 2.41 CAM 2 - Partial CAM

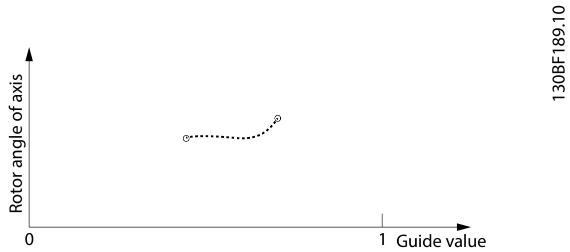


Illustration 2.42 CAM 3 - Partial CAM

Non-cyclic CAM execution

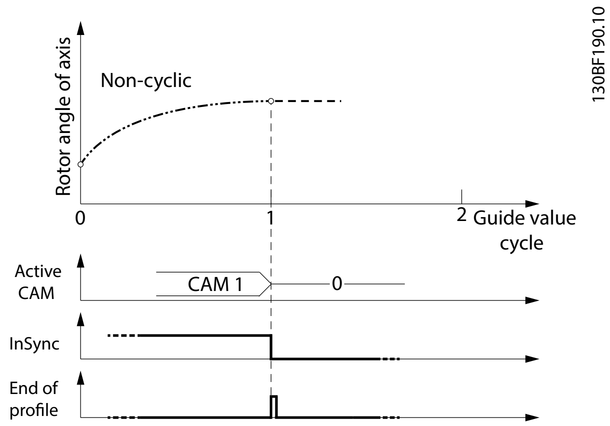


Illustration 2.43 CAM 1: Full CAM - Non-cyclic: Velocity at the End is 0

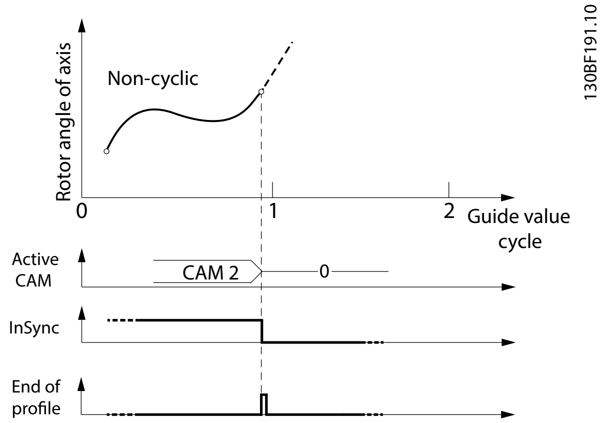


Illustration 2.44 CAM 2: Partial CAM - Non-cyclic: Velocity at the End is Unequal to 0

If the last data point of a CAM profile has a velocity other than *0*, and ends in this data point (for example, because of non-cyclic configuration), the axis keeps on turning at the velocity of this last data point (see *Illustration 2.44*). However, the velocity is still related to the guide value. The acceleration of this last data point is automatically set to *0*.

Cyclic CAM execution

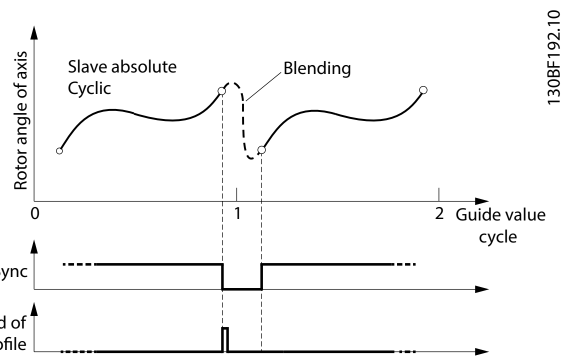


Illustration 2.45 CAM 2: Partial CAM - Cyclic: Blending Segment is an Automatically Calculated P5

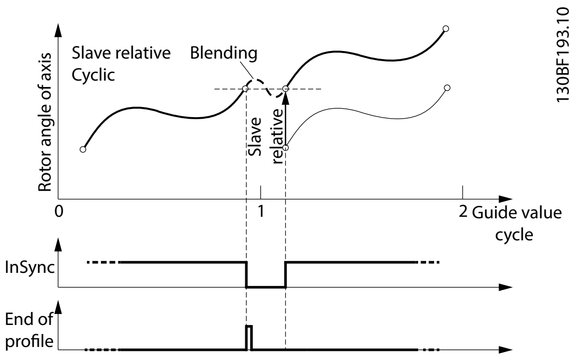


Illustration 2.46 CAM 2: Partial CAM - Cyclic, Slave relative: Blending Segment is an Automatically Calculated P5.
The CAM is adjusted in a way that the 1st rotor angle value matches the rotor angle of the last node.

A special case in *Illustration 2.46* is when the velocity of the last point and the velocity of the 1st point are both *0*. Then the P5 is actually a P0.

For smooth movements with cyclic use of fully defined CAM profiles, the 1st and the last data point of the profiles must match each other. "Matching" means having the same velocity value and, depending on the configuration, also the slave position value (see *Illustration 2.47* and *Illustration 2.48*).

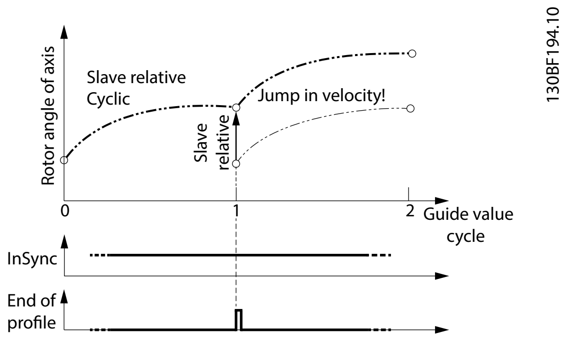


Illustration 2.47 Cyclic, Full CAM Profile Defined Over the Whole Guide Value Cycle.
The velocity of the 1st and the last node are not equal. During execution, a jump may occur.

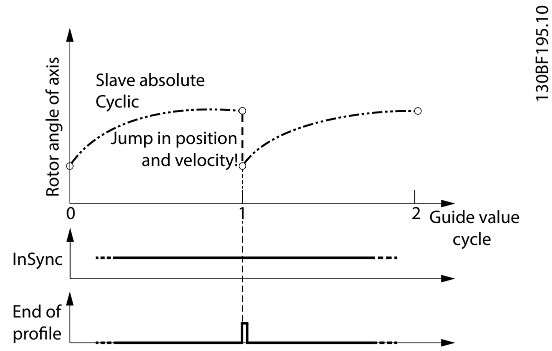


Illustration 2.48 Cyclic, Full CAM Profile.
The velocity and the position of 1st and last nodes do not match.

Switching between CAM profiles
Depending on the CAM configuration options *Master absolute/relative* and *Slave absolute/relative*, there are several methods to transition from 1 running CAM profile to the next. All the possibilities are described in the illustrations in this section. The examples all show the starting point based on the time of the CAM activation request, or when *CAM ack* (bit 12) is set by the axis (see *chapter 2.4.5.1 Activating a CAM profile*).

All illustrations in the following sub-chapters show the transition from currently running CAM 2 (see *Illustration 2.41*) to a newly activated CAM 1 (see *Illustration 2.40*) or CAM 3 (see *Illustration 2.42*). The CAM itself is always the same, but the illustrations show the behavior with different configurations and settings.

The following conventions are used for transitions between profiles:

- The blending distance has no influence on the position of the CAM (for example, regarding the automatically calculated relative offset).
- If a CAM profile is aborted (Change CAM imm = 1), the current slave position is considered as end slave position.
- When activating a non-cyclic CAM profile with *Use blend distance* = 0, the processing takes place in the same master cycle (as the CAM activation request) or in the next one (depending on the end point of the currently running CAM profile and the start point of the new CAM profile). In both cases, the profile is processed as 1 complete cycle (starting with the next upcoming start node).
- When activating a non-cyclic CAM profile with *Use blend distance* = 1, the processing of it (at least the start point) takes place in the same master cycle (as the CAM activation request), otherwise a CAM error is issued.

Absolute master position, absolute slave position

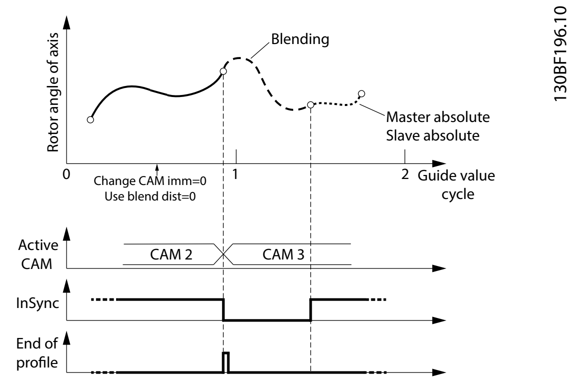


Illustration 2.49 Change CAM immediately = 0.
Do not use blending distance

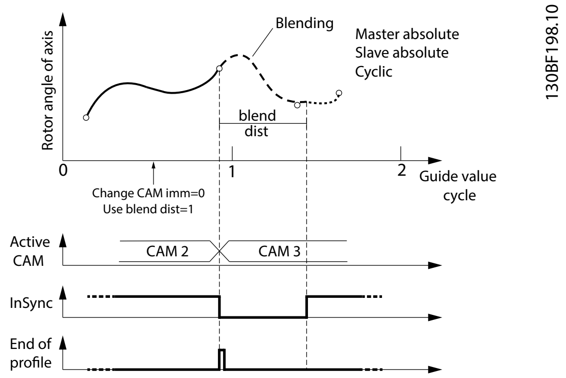


Illustration 2.51 Change CAM immediately = 0.
Use blending distance; Blending distance is long enough to cover the gap to the next CAM.

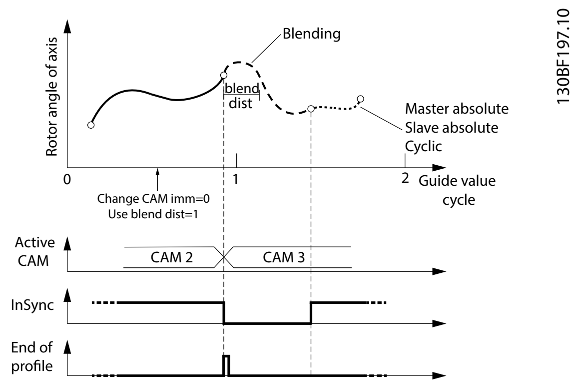


Illustration 2.50 Change CAM immediately = 0.
Use blending distance; Blending distance is not long enough to reach the next CAM.

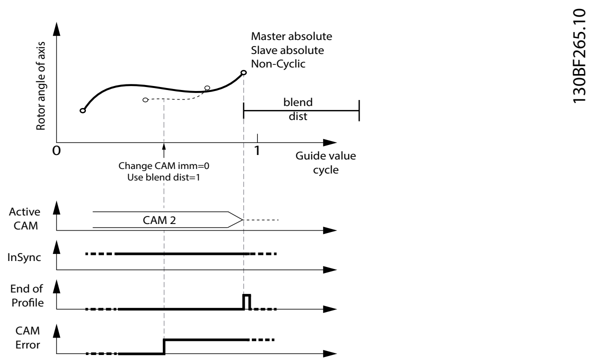


Illustration 2.52 The end of the blend distance is not on the new CAM in the same guide value cycle.
This situation leads to a rejection of the transition. The servo drive acts as if the command has never been issued.

In *Illustration 2.50*, the blending distance is not long enough to reach the 1st data point of the next CAM. The axis therefore automatically increases the blending up to the 1st point of the next CAM.

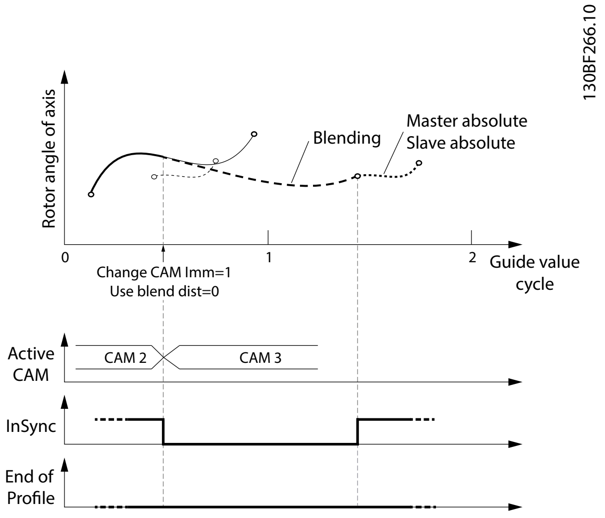


Illustration 2.53 Change CAM immediately = 1.
No blending distance used. The blending is then done automatically to the beginning of the new CAM. It does not necessarily mean that this is in the next cycle (for example, see *Illustration 2.57*).

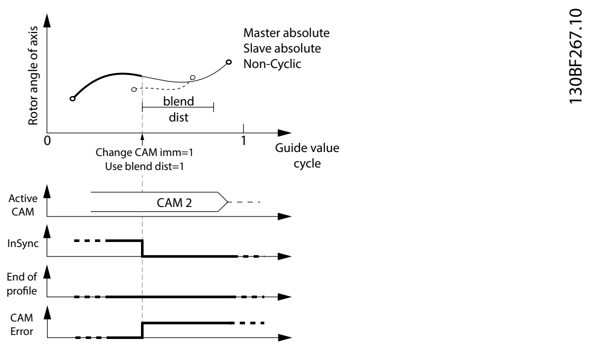


Illustration 2.54 The end of the blend distance is not on the new CAM in the same guide value cycle.
This situation leads to a reject of the transition. An error is issued because the 1st CAM was aborted with Change CAM imm = 1.

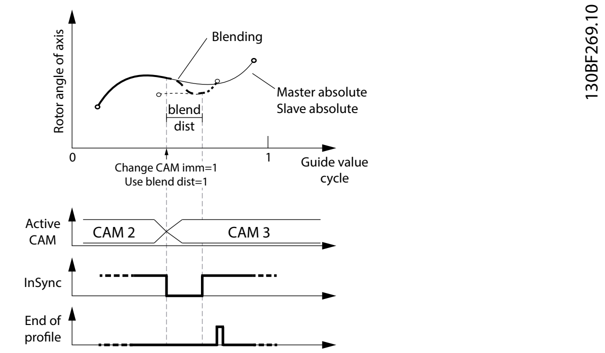


Illustration 2.55 Change CAM immediately = 1.
Use blending distance; Blending is possible to the new CAM profile in the same cycle.

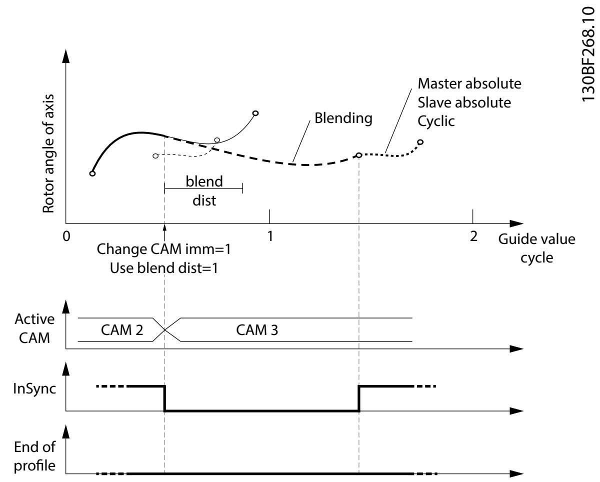


Illustration 2.56 Change CAM immediately = 1.
Use blending distance; Blending distance ends after the new CAM profile ends; Blending is then extended to the starting point of the next cycle.

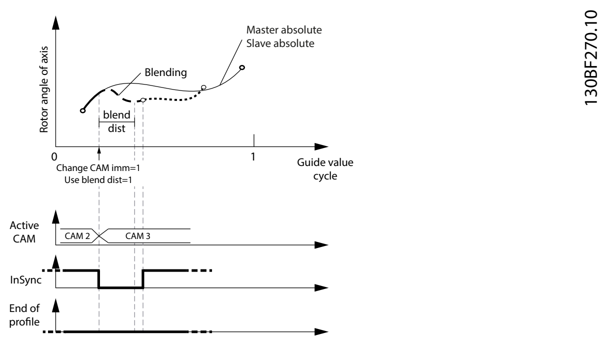


Illustration 2.57 Use blending distance; Blending distance is not long enough to reach the next CAM.

Absolute master position, relative slave position
Change CAM immediately = 0:
The currently running CAM is processed until the end. The 1st slave position of the new CAM is adjusted so that it matches the slave position of the end point of the old CAM profile.

Change CAM immediately = 1:
The currently running CAM is aborted immediately. The slave position of the new CAM at the current guide value is adjusted so that it matches the current slave position. If the new CAM has not defined at this current guide value, the slave value of the 1st point of the new CAM is adjusted so that it matches the current slave position.

The behavior when switching to non-cyclic CAM profiles is the same as detailed in section *Absolute master position, absolute slave position* in *chapter 2.4.5.4 Basic CAM*. Options *slave absolute* and *slave relative* have no influence in these cases and are therefore not mentioned again in this section.

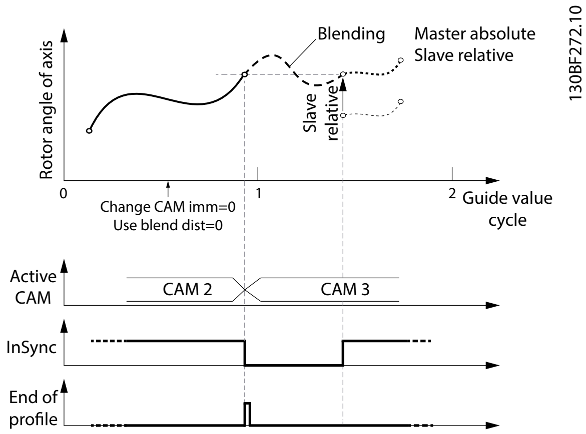


Illustration 2.58 Transition with absolute master and relative slave positioning;
No blending distance used.

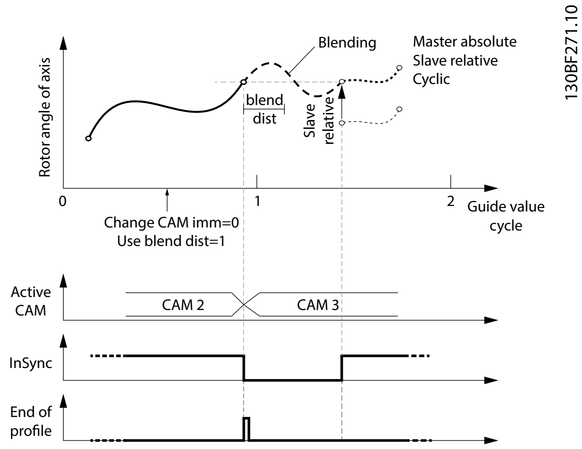


Illustration 2.59 Change CAM immediately = 0.
Use blending distance; Blending distance is not long enough to reach the next CAM.

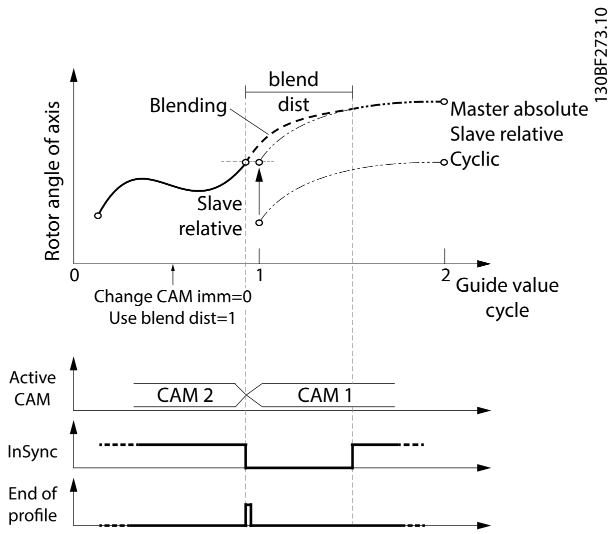


Illustration 2.60 Change CAM immediately = 0.
Use blending distance; Blending distance is long enough to cover the gap to the next CAM.

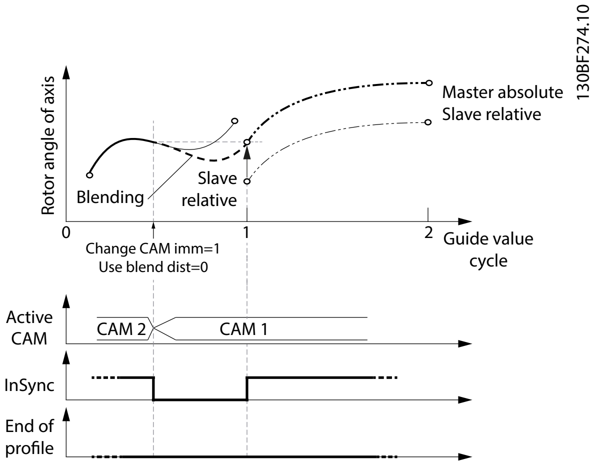


Illustration 2.61 Change CAM immediately = 1.
No blending distance used. The blending is then done automatically to the beginning of the new CAM.
It does not necessarily mean that this is in the next cycle (for example, see *Illustration 2.62*).

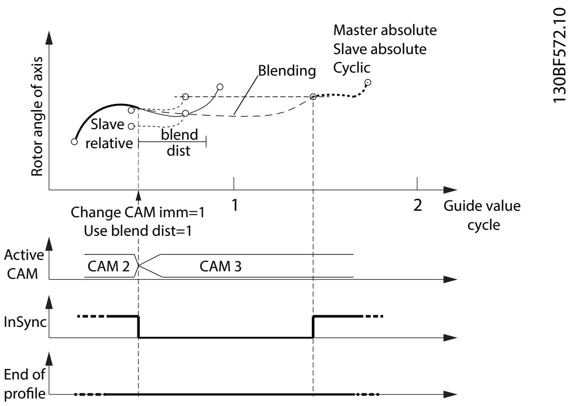


Illustration 2.62 Change CAM immediately = 1.
Use blending distance; Blending distance ends after the new CAM profile ends;
Blending is then extended to the starting point of the next cycle.

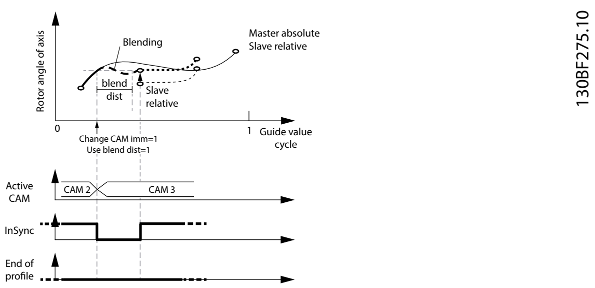


Illustration 2.63 Change CAM immediately = 1.
Use blending distance; Blending distance is not long enough to reach the next CAM.

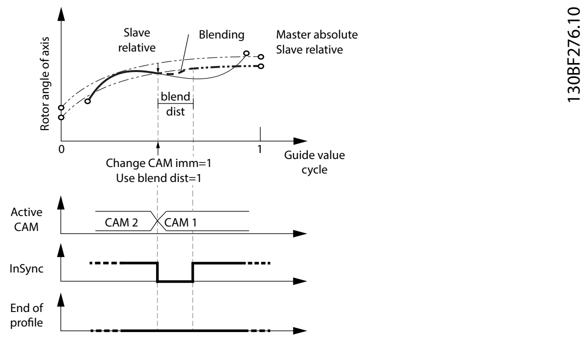


Illustration 2.64 Change CAM immediately = 1.
Use blending distance; Blending is possible to the new profile of the next CAM.

Relative master position, absolute slave position

In this case, the option *Use blend distance* is ignored. The minimum blending distance (see *chapter 7.14.11 Parameter: Minimum Blending Distance (0x380A)*) is used to calculate a polynomial of 5th degree for the synchronization movement to align the current rotor angle of the axis to the slave position of the 1st data point in the CAM profile.

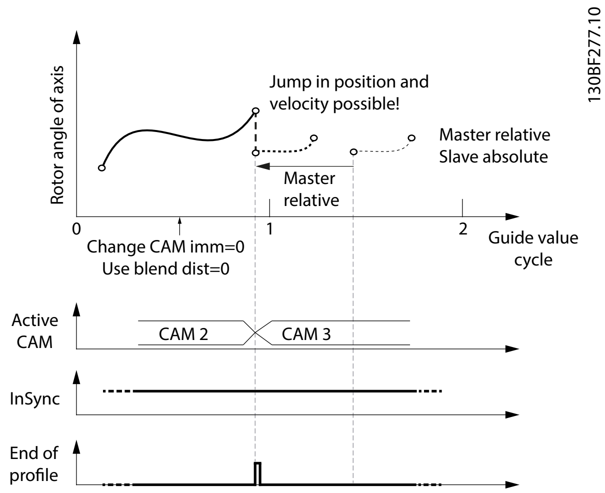


Illustration 2.65 Change CAM immediately = 0.
Do not use blending distance. If the CAMs do not match in slave position and velocity, a jump may occur.
This would probably lead to a following error.

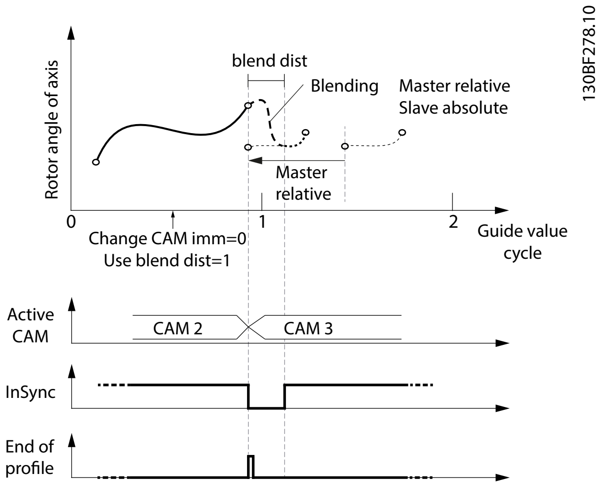


Illustration 2.66 Change CAM immediately = 0.
Use blending distance; Blending distance is inside the new CAM.

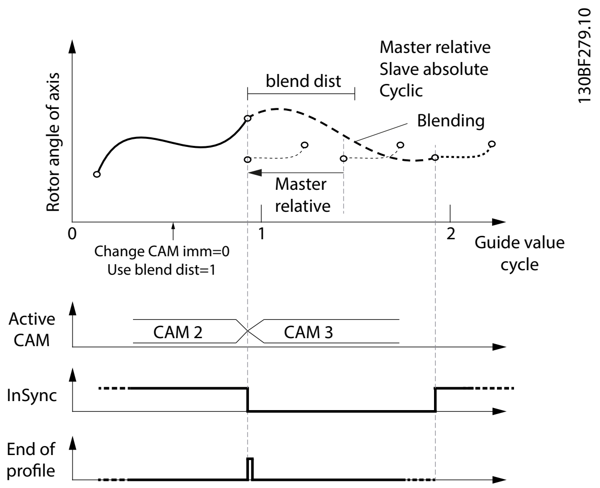


Illustration 2.67 Change CAM immediately = 0.
Use blending distance; Blending distance ends after the end point of the new CAM.

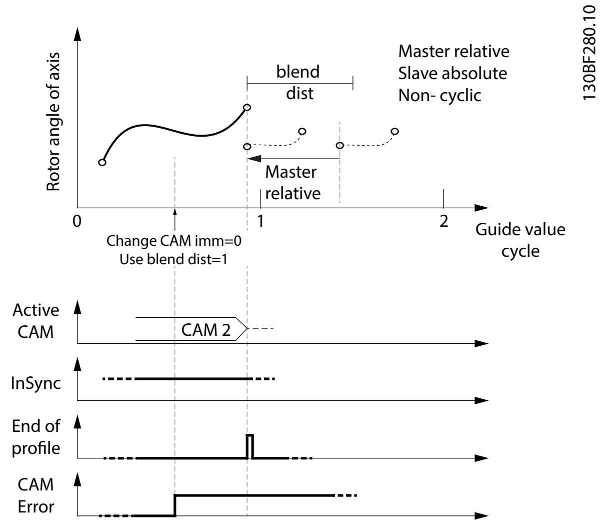


Illustration 2.68 The end of the blend distance is not on the new CAM.
This situation leads to rejection of the transition. An error is issued
because the 1st CAM was aborted with Change CAM immediately = 1.

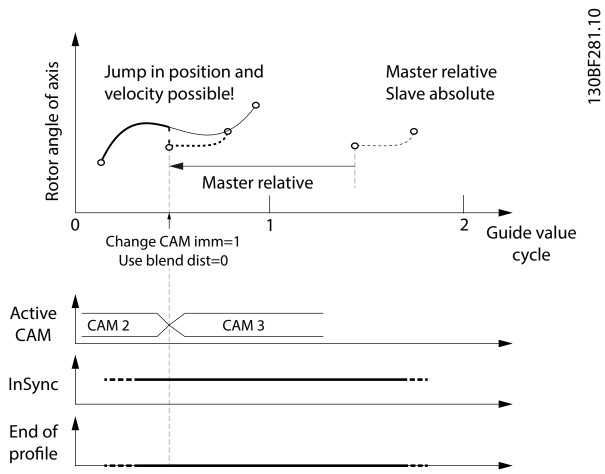


Illustration 2.69 Change CAM immediately = 1.
No blending distance used. If the CAMs do not match in slave position and velocity, a jump may occur.
This may lead to a following error.

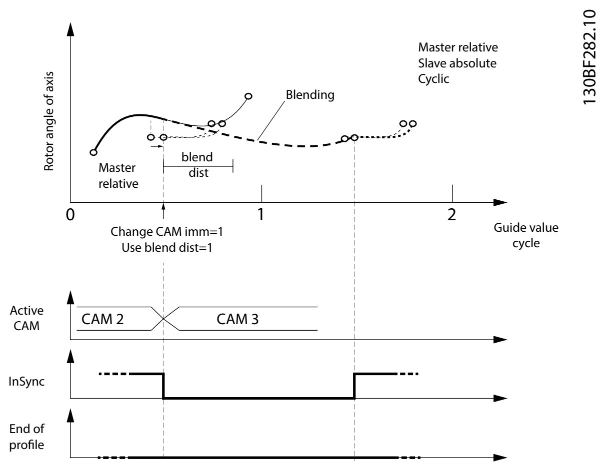


Illustration 2.70 Change CAM immediately = 1.
Use blending distance; Blending distance ends after the end point of the new CAM.

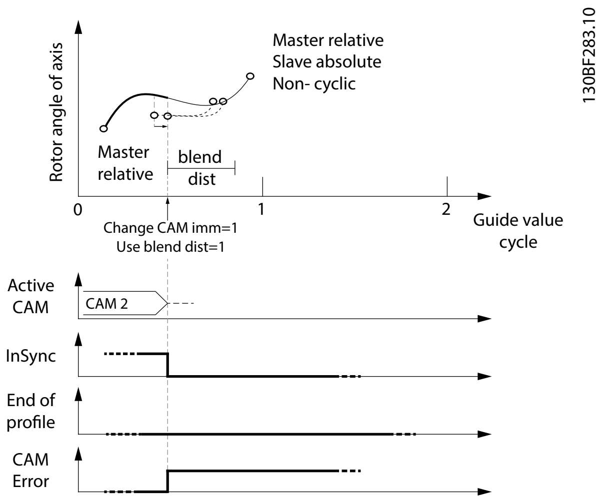


Illustration 2.71 The end of the blend distance is not on the new CAM.
The transition is rejected and the servo drive acts as if the command was never issued.

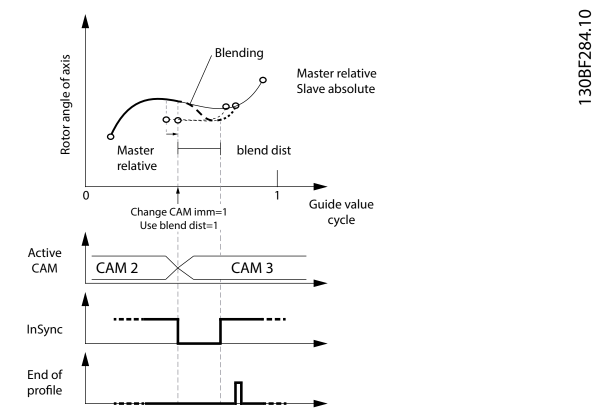


Illustration 2.72 Change CAM immediately = 1.
Use blending distance; Blending distance is shorter than the new CAM profile.

Relative master position, relative slave position

The processing starts as soon as the CAM profile is activated. The 1st point of the CAM profile is moved to the current position and guide value. The behavior when switching to non-cyclic CAM profiles is the same as shown in section *Relative master position, absolute slave position* in *chapter 2.4.5.4 Basic CAM*. Option *slave absolute* or *slave relative* has no influence in these cases and is therefore not mentioned again in this chapter.

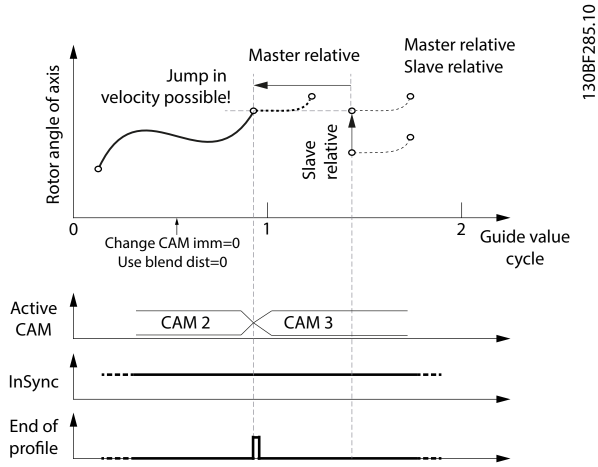


Illustration 2.73 CAM profile with relative master and relative slave positioning.
A jump in velocity may occur if the CAMs do not match.

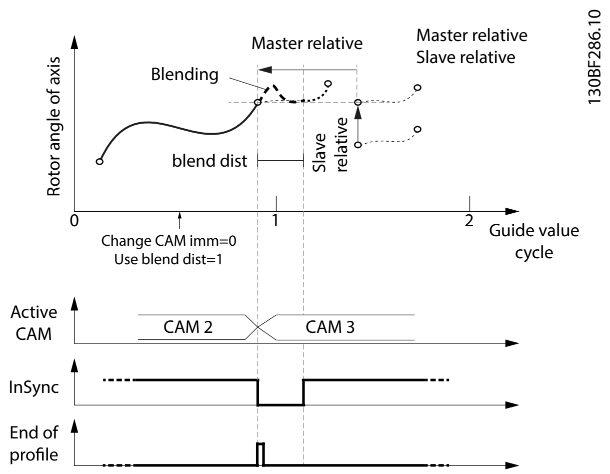


Illustration 2.74 Change CAM immediately = 0. Use blending distance.

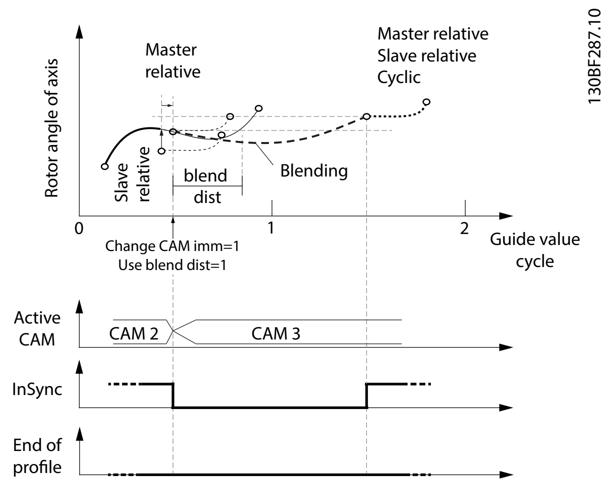


Illustration 2.75 Change CAM immediately = 1. Use blending distance; Distance is longer than the new CAM.

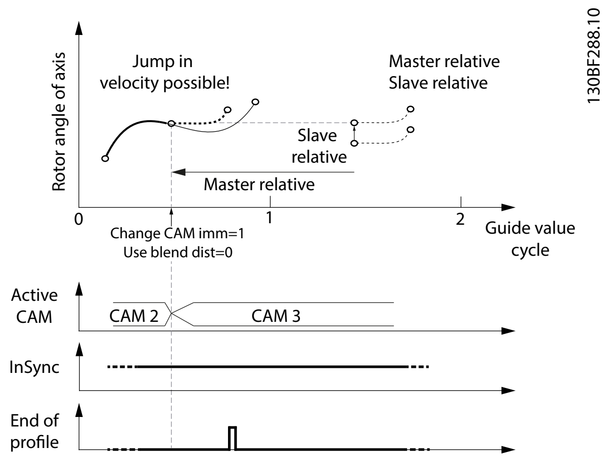


Illustration 2.76 Change CAM immediately = 1. Do not use blending distance. Jumps in velocity may occur.

2.4.5.5 Advanced CAM

The advanced CAM is represented by nodes and segments. The available node types are described in *Table 2.13*. The available segment types are described in *Table 2.14*.

The CAM configuration options *Slave absolute* and *Slave relative* are not available for advanced CAM. In advanced CAM, the behavior of an absolute or relative movement is built in to the different segment types.
A differentiation between full and partial CAM is not applicable for advanced CAM. An advanced CAM can contain paths that form a circle, and alternative paths (see *Illustration 2.78*), which end at nodes without further following segment.

An advanced CAM profile can consist of several nodes, segments, actions, and exit conditions. The size of a CAM profile highly depends on the number of elements and, for example, on the segment types (some require more and others require fewer parameters).

Name	Description
GuideNode	Connects GuideSegments
EventNode	Connects EventSegments

Table 2.13 Available Node Types

All nodes are non-signaling nodes. The axis does not automatically signal if it passes a node. However, for example for debugging purposes, it is possible to enable this signaling for selected nodes.

GuidePoly	GuideSegment Polynomial of 5^{th} order based on guide value.
MoveDistance-Segment	GuideSegment Uses run-time calculated polynomial of 5^{th} order; the angle is sent over fieldbus at run-time.

FlyingStop-Segment	GuideSegment Constant speed, followed by braking ramp, angle of constant movement is sent over fieldbus at run-time.
ReturnSegment	GuideSegment Turns shaft to a symmetric angle (absolute position) to eliminate rounding errors.
EventSegment-Container	GuideSegment All time-related movements must be encapsulated by this segment type.
TimePoly	EventSegment Polynomial of 5th order based on time.
VelocitySegment	EventSegment Constant velocity, independent of the guide value.
SyncSegment	EventSegment Constant velocity, depending on the guide value.
TorqueSegment	EventSegment Constant torque, independent of the guide value.
PwmOffSegment	EventSegment Turns off the PWM.
FrictionSegment	EventSegment Determines the friction of the system.

Table 2.14 Available Segment Types

This CAM profile type can only be used with forward turning guide values.

The advanced CAM profile consists of a list of nodes (containing *GuideNodes*), a list of segments (containing *GuideSegments*), an optional list of actions, and an optional list of exit conditions.

```
<advancedCam>
        <nodes>
        ... list of GuideNodes
        </nodes>

        <segments>
        ... list of GuideSegments
        </segments>

        <actions>
        ... list of actions
        </actions>

        <exits>
        ... list of exit conditions
        </exits>

</advancedCam>
```

Illustration 2.77 Advanced CAM Profile

Nodes

Nodes are defined by their position on the guide value. The slave position is defined, where necessary, inside the segments. The starting node of a CAM is the node with *nodeID* 0. In a CAM, there must be exactly 1 starting node (1 node with ID *0*). However, this starting node does not need to be the 1st node of the CAM (see *Illustration 2.78*). The starting node must be a guide node.

End nodes define the end of a non-cyclic CAM, or the end when switching non-immediate to another CAM. Only guide nodes can be end nodes. A guide node that has no following segment is automatically defined as an end node.

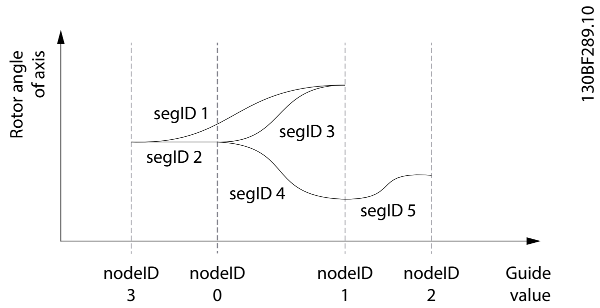


Illustration 2.78 NodeID 2 has no following segment: It automatically becomes an end node.

An advanced CAM must have at least 1 end node, however it is possible to have >1 end node within a CAM. If no end node is explicitly defined, and there is no node without a following segment (that implicitly would be an end node), the start node becomes an end node.

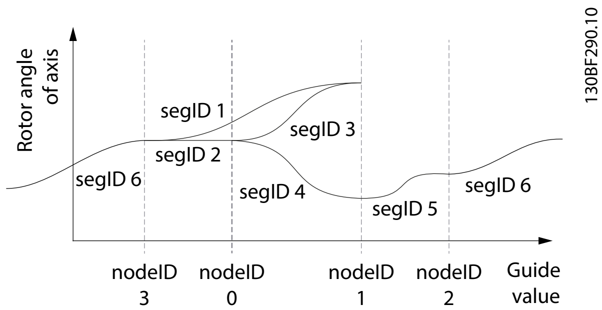


Illustration 2.79 No end node explicitly defined and no node without following segment defined in the CAM. NodeID 0 (start node) automatically becomes the end node.

A non-cyclic CAM ends at the 1st end point that is processed. This can take several cycles of guide value or continue infinitely if there is no end node within the currently processed path. For example, in *Illustration 2.79*, when the path is set in a way that *segID* 1 is used instead of *segID* 2, the start node is not in the active path.
A cyclic CAM just passes an end node like every normal node; the *End of Profile* bit (see *chapter 7.14.8 Parameter: CAM Profile Status (0x3805)*) is set. This bit is set for every end node that is passed within a cyclic CAM. So, the end of profile bit can also be set several times within 1 cycle. It is also possible that it is not set at all if there is no end node within the processed path.

2

A cyclic CAM that passes an end node without following segment blends to the start node of the CAM profile. For example, in *Illustration 2.78*, the *nodeID* 2 has no following segment and when executing this CAM as cyclic CAM, the axis blends from *node ID* 2 to *node ID* 0.
Non-immediate switching to another CAM takes place when the currently running CAM passes the next occurring end node (cyclic and non-cyclic). In *Illustration 2.78*, this would be when passing *node ID* 2 and in *Illustration 2.79*, this would be, when passing *node ID* 0. The switching only takes place when *node ID* 0 is in the processed path. Otherwise, a command is needed for segment ID 2 to be the following segment of *node ID* 3.

GuideNode

GuideNodes are similar to data points within a machine cycle. However, in contrast to data points of a basic CAM, a *GuideNode* is only defined by its guide value position (master position). The slave position, velocity, and acceleration are not defined inside a *GuideNode*. This information is given in the connected segments. The velocity (and acceleration) of 2 segments that are connected to the same node must match. Otherwise a jump in velocity (and/or acceleration) occurs. Each node has a unique ID for referencing to it. A *GuideNode* combines *GuideSegments* so therefore, represents starting and ending points of segments.

```
<guideNode  nodeID="0"  masterPos="0"
      signal="FALSE"    endNode="FALSE"
      action="0" />
```

Illustration 2.80 XML Representation of a GuideNode

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
nodeID	M	Integral number; 0–65535	Integral number to uniquely identify this node. The *nodeID* must be unique across all *GuideNodes* and *EventNodes*. The same *nodeID* cannot be used twice. The node with *nodeID* 0 is the starting node.
masterPos	M	Float; 0.0–1.0	Master position for this *GuideNode*. Given in revolutions of guide value.
signal	O; default = FALSE	FALSE or TRUE	Defines if this node is signaled by the axis. This attribute is optional. If it is not present, the default behavior is not to signal this node.
endNode	O; default = FALSE	FALSE or TRUE	Defines if this node is an end node of the CAM. This attribute is optional. If it is not present, the node is no end node.
action	O; default = no action	0, 1, or more existing action IDs	Defines if 1 or multiple actions are attached to this node. This attribute is optional. If it is not present, no action is assigned to this node. To define multiple actions for this node, all action IDs must be listed inside the attribute, separated by a white space. If a non-existing action ID is used, an error is issued during parsing.

Table 2.15 Attributes for GuideNode

EventNode

Like *GuideNodes*, *EventNodes* are data points within a time-related movement. They combine *EventSegments* in an *EventSegmentContainer*. Each *EventSegmentContainer* has exactly 1 first *EventNode*, which has no preceding *EventSegment*, and at least 1 ending *EventNode*, which has no succeeding *EventSegment*.

```
<eventNode  nodeID="1"
      signal="FALSE"  action="0" />
```

Illustration 2.81 XML Representation of an EventNode

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
nodeID	M	Integral number; 1–65535	Integral number to uniquely identify this node. The *nodeID* must be unique across all *GuideNodes* and *EventNodes*. The same *nodeID* cannot be used twice.
signal	O; default = FALSE	FALSE or TRUE	Defines if this node is signaled by the axis. This attribute is optional. If it is not present, the default behavior is not to signal this node.
action	Same as in *Table 2.15*.		

Table 2.16 Attributes for EventNode

Segments

There are 2 types of segment:

- *GuideSegments*: All segment types that are defined based on the guide value.
- *EventSegments*: All segment types that are defined based on time.

There are 2 types of XML representation for some of the segments:

- Start/Endpoint representation
- Coefficient representation

The availability of each type is stated in the corresponding section.

All segments always have exactly 1 preceding and 1 succeeding node. Multiple segments can have the same node as the preceding node. This is used to design alternative paths. The selection between those paths takes place during run-time (see *chapter 2.4.5.6 Commands During Operation*).
In *Illustration 2.82*, an example is given where the segment with ID 3 is an alternative to segment 4. Both have the same preceding and succeeding nodes. It is also possible to overleap a node, as shown in segment 1: it is an alternative path to segments 2 and 3 or segments 2 and 4. Multiple segments can also have the same node as the succeeding node. The alternative paths are then combined again and the further movement is common.
In *Illustration 2.82*, an example is given where segments with ID 1, 3, and 4 all have the same succeeding node. Regardless of which segment the servo drive is coming from, segment 5 is processed afterwards.

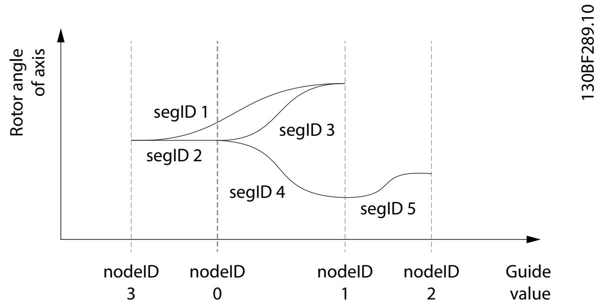


Illustration 2.82 Example of Alternative Segments

GuideSegments

GuideSegments are all segment types that are defined based on the guide value. *GuideSegments* can only have *GuideNodes* as preceding and succeeding nodes.
There are some attributes that are common to all *GuideSegments* (see *Table 2.17*).

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	M	Integral number; 0–50000	Integral number to uniquely identify this segment. The *segID* must be unique across all *GuideNodes* and *EventSegments*. The same *segID* cannot be used twice.
precNode	M	An existing *nodeID* of a *GuideNode*	ID of the *GuideNode* at the beginning of this segment. If a non-existing node ID is used, an error is issued during parsing.
succNode	M	An existing *nodeID* of a *GuideNode*	ID of the *GuideNode* at the end of this segment. If a non-existing node ID is used, an error is issued during parsing.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
default	O; default = FALSE	TRUE/FALSE	Defines if this segment is the default segment for the referenced preceding node. This attribute is not necessary if only 1 segment has this *precNode* as preceding node. If >1 segment has this *precNode* as preceding node, and none of them claims to be the default one, the segment with the lowest segment ID is used. If >1 segment claims to be the default segment of a specified *precNode*, a parsing error is issued.
startAction	O; default = no action	0, 1, or more existing action IDs	Defines if 1 or multiple actions are attached to the beginning of this segment. This attribute is optional. If it is not present, no action is assigned to the beginning of this segment. To define multiple actions, all *actionIDs* must be listed inside the attribute, separated by a white space. If a non-existing action ID is used, an error is issued during parsing.
endAction	O; default = no action	0, 1, or more existing action IDs	Defines if 1 or multiple actions are attached to the end of this segment. This attribute is optional. If it is not present, no action is assigned to the end of this segment. To define multiple actions, all *actionIDs* must be listed inside the attribute, separated by a white space. If a non-existing action ID is used, an error is issued during parsing.

Table 2.17 Common Attributes for all GuideSegments

GuidePoly:

The *GuidePoly* defines a movement that relates the rotor angle of the axis with the guide value. Position, velocity, and acceleration at the preceding and the succeeding node can be selected without restrictions. It is therefore possible to realize many movements already with a single *GuidePoly*.

Complex movements can be combined by a number of *GuidePolys*. When combining *GuidePolys*, the end velocity of the segment and the start velocity of the next segment must match, otherwise a jump in velocity occurs. It is possible to define absolute and relative movements.

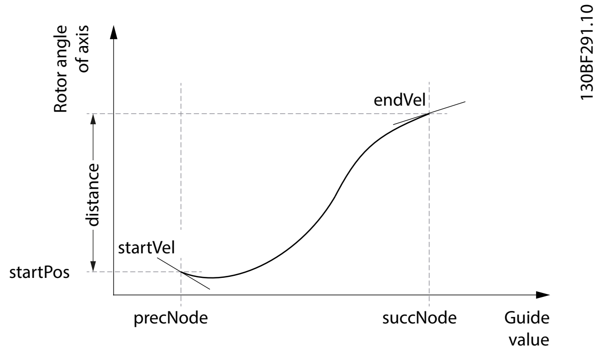


Illustration 2.83 GuidePoly

```
<guidePoly  segID="0"   precNode="0"
      succNode="1"   default="FALSE"   type="absolute"
      startPos="0"   distance="0"      startVel="0"
      endVel="0"     startAcc="0"      endAcc="0"
      startAction="0"   endAction="0" />
```

Illustration 2.84 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.17*.		
precNode	Same as in *Table 2.17*.		
succNode	Same as in *Table 2.17*.		
type	M	Absolute/ relative	Defines if the segment is executed at an absolute slave position or if the segment is executed relative to the previous position.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
startPos	M for *type*= *absolute*; O for *type*= *relative*	Float	Axis position at the beginning of this segment. Given in revolutions of rotor position. *startPos* describes the position on the motor side. If it is a relative segment, the *startPos* attribute only modifies the *Logical CAM position*. If *startPos* is not present (in a relative segment), the *Logical CAM position* from the previous segment is used as *startPos*.
distance	M	Float	Rotor angle of the axis during this segment. Given in revolutions of rotor position. Use negative values for backward movements.
startVel	O; default = 0	Float	Velocity of the axis at the beginning of this segment. The velocity must be given as a factor between the velocity of the axis in relation to the velocity of the guide value (1 revolution of the axis per 1 round of guide value). To ensure smooth movements, the velocities of all segments that are connected in the same node should be the same. If not parameterized correctly, a jump in velocity may occur.
endVel	O; default = 0	Float	Same as *startVel* but at the end of the segment.
startAcc	O; default = 0	Float	Acceleration of the axis at the beginning of this segment. The acceleration must be given as a factor between the acceleration of the axis in relation to the velocity of the guide value (1 revolution of axis per square of round of guide value). Jumps in acceleration may occur when 2 succeeding segments have different *startAcc* and *endAcc* values.
endAcc	O; default = 0	Float	Same as *startAcc* but at the end of the segment.
startAction	Same as in *Table 2.17*.		
endAction	Same as in *Table 2.17*.		

Table 2.18 Attributes for GuidePoly in Start/Endpoint Representation

```
<guidePoly  segID="0"   precNode="0"
      succNode="1"   default="FALSE"   type="absolute"
      a0="0"         a1="0"            a2="0"
      a3="0"         a4="0"            a5="0"
      startAction="0"   endAction="0" />
```

Illustration 2.85 Coefficient Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.17*.		
precNode	Same as in *Table 2.17*.		
succNode	Same as in *Table 2.17*.		
type	Same as in *Table 2.17*.		
a0	*type = absolute*: M else O	Float	Polynomial coefficients for the movement described by a5x5 + a4x4 + $a3x^3$ + $a2x^2$ + a1x + a0 a0 is the same as *startPos* in the Start/ Endpoint represen- tation.
a1-a5	M	Float	
startAction	Same as in *Table 2.17*.		
endAction	Same as in *Table 2.17*.		

Table 2.19 Attributes for GuidePoly in Coefficient Representation

2

MoveDistanceSegment:

MoveDistanceSegments are used for movements with no predefined rotation angle. The desired rotor angle is given to the axis during run-time. It must be given before the beginning of the segment. It must be given in every machine cycle (see *Table 2.53* in *chapter 2.4.5.6 Commands During Operation*).

This segment is mostly used together with an external camera for object alignment. The start and end velocity, and the start and end acceleration can be parameterized. The parameter that is sent during run-time must be given in revolutions of rotor angle. The rotor angle must be sent at least 5 ms before the segment begins.

If no parameter is sent for this segment, the axis reports an error (see *chapter 2.4.5.7 Notifications from the Servo Drive*) and assumes a distance of 0. An error message is sent when passing the *precNode* of this segment. A new parameter message, meant for the next cycle, can be sent to the servo drive when the *succNode* of this segment has been passed.

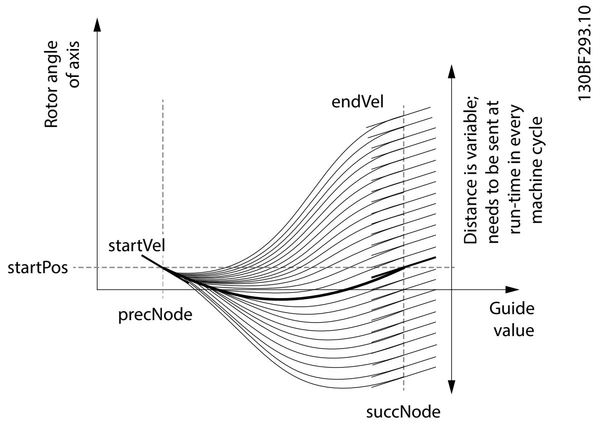


Illustration 2.86 MoveDistanceSegment

```
<moveDistSeg segID="0"  precNode="0"
      succNode="1"   default="FALSE"
      startPos="0"   startVel="0"   endVel="0"
      startAcc="0"   endAcc="0"
      startAction="0"   endAction="0" />
```

Illustration 2.87 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.17*.		
precNode	Same as in *Table 2.17*.		
succNode	Same as in *Table 2.17*.		
startPos	O	Float	See *Table 2.18* for *type= relative.*
startVel	Same as in *Table 2.18*.		
endVel	Same as in *Table 2.18*.		
startAcc	Same as in *Table 2.18*.		
endAcc	Same as in *Table 2.18*.		
startAction	Same as in *Table 2.17*.		
endAction	Same as in *Table 2.17*.		

Table 2.20 Attributes for MoveDistanceSegment in Start/Endpoint Representation

Coefficient representation: This representation is not available.

FlyingStopSegment:

The *FlyingStopSegment* is used to stop the servo drive out of a synchronous movement at a position, which can be determined at run-time. This angle is usually determined by a camera system. The motion consists of 2 parts, a constant rotation, which length is defined by the sent parameter, and a deceleration polynomial for stopping the servo drive (a polynomial of 3rd degree is used). The angle must be passed before the segment has started. The parameter can be in a range from 0° to *maxConstantDist*. The value is given as an absolute value. The direction is determined by the direction of the velocity.

The rotor angle must be sent during run-time but before the beginning of this segment. When the constant part has been processed for the parameter, which was given, the stopping part of the segment starts. This braking polynomial is always the same, independent of the remaining distance to the end of the segment.

The parameter that is sent during run-time must be given in revolutions of rotor angle. The rotor angle must be sent at least 5 ms before the segment begins. If no parameter is sent for this segment, the axis reports an error (see section *Notifications from the servo drive* in this sub-chapter) and assumes a distance of *maxConstDist*. The error message is sent when passing the *precNode* of this segment. A new parameter message, meant for the next cycle can be sent to the servo drive when the *succNode* of this segment was passed.

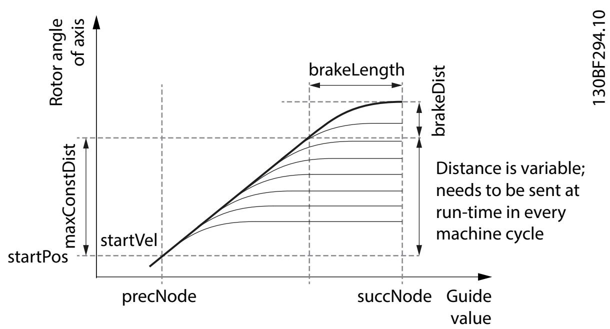


Illustration 2.88 FlyingStopSegment

```
<flyingStop segID="0"   precNode="0"
      succNode="1"   default="FALSE"
      startPos="0"   startVel="1"   maxConstDist="1"
      brakeDist="0.1"   brakeLength="0.1"
      startAction="0"   endAction="0" />
```

Illustration 2.89 Start/Endpoint Representation

```
<flyingStop segID="0"   precNode="0"
      succNode="1"   default="FALSE"
      a0="0"    maxConstDist="1"    brakeLength="0.1"
      a1="1"    a2="0"    a3="0"
      startAction="0"   endAction="0" />
```

Illustration 2.90 Coefficient Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.17*.		
precNode	Same as in *Table 2.17*.		
succNode	Same as in *Table 2.17*.		
startPos	O	Float	See *Table 2.18* for *type= relative*.
startVel	Same as in *Table 2.18*.		
maxConstDist	M	Float >0	Defines the maximum rotor angle that the axis turns if no parameter is sent during run-time. Given in revolutions of rotor position. Only positive values are allowed. The value is considered as absolute value in the direction of the start velocity.
brakeDist	M	Float >0	Rotor angle of the axis during the deceleration phase of this segment. Given in revolutions of rotor position. Only positive values are allowed. The value is considered as absolute value in the direction of the start velocity. There must be enough space to be able to brake also in worst case situations.
brakeLength	M	Float >0	Guide value for the length of the deceleration phase of this segment. Given in revolutions of guide value. The segment must be long enough to run the *maxConstDist* and have enough guide value left for at least the *brakeLength*. If there is space left, the servo drive remains in standstill until the succeeding *GuideNode* is reached.
startAction	Same as in *Table 2.17*.		
endAction	Same as in *Table 2.17*.		

Table 2.21 Attributes for FlyingStopSegment in Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.17*.		
precNode	Same as in *Table 2.17*.		
succNode	Same as in *Table 2.17*.		
a0	type = absolute: M else O	Float	Polynomial coefficients for the movement described by a5x5 + a4x4 + $a3x^3$ + $a2x^2$ + a1x + a0 a0 is the same as startPos in the Start/ Endpoint represen-tation.
maxConstDist	Same as in *Table 2.21*.		
brakeLength	Same as in *Table 2.21*.		
a1-a3	M	Float	Polynomial coefficients for the movement described by $a3x^3$ + $a2x^2$ + a1x It is not necessary to give a0 as this information comes from the position value of the beginning of the decelerating part. a1 is also the velocity for the constant part of the segment. This value must be unequal to 0. The coefficients must be given so that the braking part ends in a standstill.
startAction	Same as in *Table 2.17*.		
endAction	Same as in *Table 2.17*.		

Table 2.22 Attributes for FlyingStopSegment in Coefficient Representation

ReturnSegment:

The *ReturnSegment* is used to return from any position to a defined absolute position. In this way, all offsets of the logical rotor angle are discarded and a fixed relation

2

between logical and absolute rotor angle is established or re-established. This is useful, if the axis has to be moved to an absolute position after a loss of the reference position by using a variable movement (for example, *MoveDistance-Segment*).
Usually the *ReturnSegment* is used at the beginning of the CAM to start from a defined absolute position. The *Return-Segment* is used in conjunction with devices which have multiple, equidistant, and equivalent starting positions, for example, a square device.

The axis automatically selects the shortest way and calculates a polynomial of 5th degree to reach the next valid position. A backward movement of the servo drive is possible. Valid positions are calculated by the formula:

$$\frac{partition}{revolutions} + offsetRev$$

The *ReturnSegment* should be the first segment in a CAM profile. It provides a means to return to the next equivalent starting position and eliminate all rounding errors. This segment must always start in standstill and it also stops in standstill.

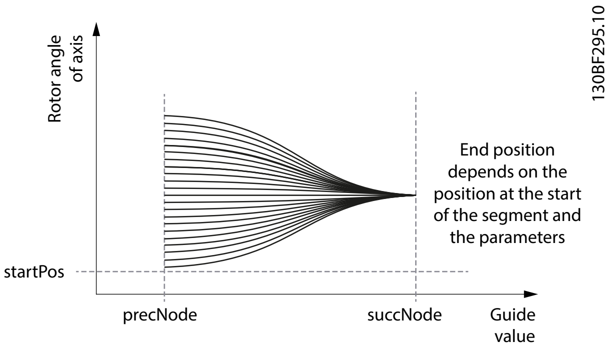


Illustration 2.91 Return Segment

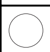
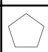
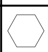
Partition	0	1	2	3	4	5	6	...
Possible shapes								
valid phys. rotor angles	each	0°	0° 180°	0° 120° 240°	0° 90° 180° 270°	0° 72° 144° 216° 288°	0° 60° 120 180° 240° 300°	
worst case angle to turn	0°	±180°	±90°	±60°	±45°	±36°	±30°	

Table 2.23 Partition Example of ReturnSegment for Single-turn Axis (revolutions = 1; offsetRev = 0)

```
<returnSeg  segID="0"    precNode="0"
      succNode="1"       default="FALSE"
      startPos="0"       partition="0"
      revolutions="1"    offsetRev="1"
      startAction="0"    endAction="0" />
```

Illustration 2.92 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.17*.		
precNode	Same as in *Table 2.17*.		
succNode	Same as in *Table 2.17*.		
startPos	O	Float	See *Table 2.18* for *type=relative*.
partition	O; default = 0	Integer: (0;16)	Can be used for shaped plates when several equal, valid starting positions are allowed. The worst case movement is influenced by this parameter.
revolutions	O; default = 1	Integer >0	Number of revolutions that are used when calculating valid positions, for example, if there is a gear.
offsetRev	O; default = 0	Float	Desired end rotor position relative to the nearest physical position. The reference-position is determined by the absolute position at the beginning of this segment and the partition/revolutions. Given in revolutions of the axis.
startAction	Same as in *Table 2.17*.		
endAction	Same as in *Table 2.17*.		

Table 2.24 Attributes for ReturnSegment in Start/Endpoint Representation

Coefficient representation: This representation is not available.

EventSegmentContainer:

The *EventSegmentContainer* embeds a time-related movement (composed by *EventNodes* and *EventSegments*) into the guide value-related process. It provides a certain guide value position for the beginning of the time-related movement and so that the further guide value-related movement can be resumed at the end of the *EventSegmentContainer*. For that, the *EventSegmentContainer* must be long enough, so that even at the highest speed of the guide value, the time-related movement of the *EventSegmentContainer* can still be processed completely.

Otherwise, the time-related movement is aborted at the end of the *EventSegmentContainer*, leading to possible jumps in velocity and position. The time-related movement must start and end in standstill (the 1st *EventSegment* must start in standstill and the last *EventSegment* must end in standstill). The guide value-related movement that is before the *EventSegmentContainer* must end in standstill. The guide value-related movement after the *EventSegmentContainer* must start in standstill. If 1 of the conditions is not fulfilled, a jump in velocity occurs.

```
<eventSegmentContainer segID="0"
      precNode="0"   succNode="1"   default="FALSE"
      startingEventNode="0"         startAction="0"
      endAction="0">

      <EventNodes>
      … list of event nodes
      </EventNodes>

      <EventSegments>
      … list of event segments
      </EventSegments>

</EventSegmentContainer>
```

Illustration 2.93 Event Segment Container

There is no special list of actions or exit conditions inside the *EventSegmentContainer* element. All actions defined in the CAM profile can be used for time-related nodes and segments as well.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.17*.		
precNode	Same as in *Table 2.17*. This needs to be a *GuideNode*.		
succNode	Same as in *Table 2.17*. This needs to be a *GuideNode*.		
startAction	Same as in *Table 2.17*.		
endAction	Same as in *Table 2.17*.		
startingEventNode	M	An existing *nodeID* of an *EventNode*.	ID of the starting *EventNode* at the beginning of this segment. If a non-existing node ID is used, an error is issued during parsing.

Table 2.25 Attributes for EventSegmentContainer

Additionally, an *EventSegmentContainer* has subelements to describe its embedded time-related movement. There is a (mandatory) list of *EventNodes* and a (mandatory) list of *EventSegments*.
The beginning of the first time-related segment and the end of the last time-related segment must have velocity *0*.

EventSegments

EventSegments are all segment types that are defined based on time. *EventSegments* must be embedded into an *EventSegmentContainer*.

EventSegments may only have *EventNodes* as preceding and succeeding nodes. There are some attributes that are common to all *EventSegments*. Those attributes can be found in *Table 2.26*.

GuideSegments always run from one guide value position to the next. *EventSegments* are more flexible. It is possible to define additional *supervising* parameters that serve as exit conditions. If such an exit condition appears, the axis proceeds with the next segment.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	M	Integral number; 0–50000	Integral number to uniquely identify this segment. The *segID* must be unique across all Guide- and *EventSegments*. The same *segID* cannot be used twice.
precNode	M	An existing *nodeID* of an *EventNode*.	ID of the *EventNode* at the beginning of this segment. If a non-existing node ID is used, an error is issued during parsing.
succNode	M	An existing *nodeID* of an *EventNode*.	ID of the *EventNode* at the end of this segment. If a non-existing node ID is used, an error is issued during parsing.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
default	O; default = FALSE	TRUE/FALSE	Defines if this segment is the default segment for the referenced preceding node. This attribute is not necessary if only 1 segment has this *precNode* as the preceding node. If >1 segment has this *precNode* as preceding node and none of them claims to be the default one, the segment with the lowest segment ID is used. If >1 segment claims to be the default segment of a specified *precNode*, a parsing error is issued.
duration	M	Integer >3 or 0: disable	Time given in ms counted from the beginning of this segment (duration). This is the maximum time if the segment has not been exited otherwise.
exitCond	O; default = only duration	0, 1, or more existing exit condition IDs	Defines if 1 or multiple exit conditions are attached to this segment. This attribute is optional. If it is not present, there is no exit condition assigned to the segment. The duration attribute is then the only exit condition. To define multiple exit conditions, all *exitIDs* must be listed inside the attribute, separated by a white space. If there are multiple exit conditions, the segment is aborted as soon as 1 of them applies (logical OR). If a non-existing exit condition ID is used, an error is issued during parsing.
startAction	O; default = no action	0, 1 or more existing action IDs	Defines if 1 or multiple actions are attached to the beginning of this segment. This attribute is optional. If it is not present, no action is assigned to the beginning of this segment. To define multiple actions, all *actionIDs* must be listed inside the attribute, separated by a white space. If a non-existing action ID is used, an error is issued during parsing.
endAction	O; default = no action	0, 1 or more existing action IDs	Defines 1 or multiple actions attached to the end of this segment. This attribute is optional. If it is not present, no action is assigned to the end of this segment. To define multiple actions, all *actionIDs* must be listed inside the attribute, separated by a white space. If a non-existing action ID is used, an error is issued during parsing.

Table 2.26 Common Attributes for all EventSegments

TimePoly:

The *TimePoly* is the time-related correspondent to the *GuidePoly*. It defines a time-related movement. In general, advanced CAM profiles are related to a guide value; therefore, the time-related movements must be embedded into an *EventSegmentContainer*.

Start and ending position, velocity, and acceleration at the start and the end of the segment can be selected without restrictions. Complex movements can be combined by a number of *TimePolys*.

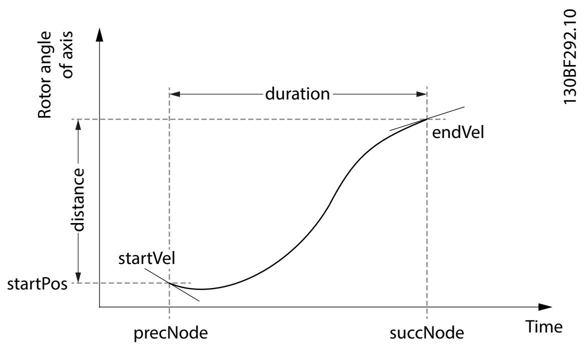


Illustration 2.94 TimePoly

```
<timePoly  segID="0"  precNode="0"
      succNode="1"  default="FALSE"
      startPos="0"  duration="500"
      type="absolute"  distance="0"
      startVel="0"    endVel="0"
      startAcc="0"    endAcc="0" exitCond="0"
      startAction="0"  endAction="0" />
```

Illustration 2.95 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.26*.		
precNode	Same as in *Table 2.26*.		
succNode	Same as in *Table 2.26*.		
default	Same as in *Table 2.26*.		
startPos	Same as in *Table 2.18*.		
duration	Same as in *Table 2.26*.		
type	Same as in *Table 2.18*.		
distance	Same as in *Table 2.18*.		
startVel	O; default = 0	Float	Velocity of the axis at the beginning of this segment. The velocity must be given in rps. To ensure smooth movements, the velocities of all segments that are connected in the same node should be the same. If this is not parameterized correctly, a jump in velocity may occur.
endVel	O; default = 0	Float	Same as *startVel* but at the end of the segment.
startAcc	O; default = 0	Float	Acceleration of the axis at the beginning of this segment. The acceleration must be given in rps per second. It is possible to parameterize jumps in acceleration when 2 succeeding segments have different *startAcc* and *endAcc* values.
endAcc	O; default = 0	Same as *startAcc*	Same as *startAcc* but at the end of the segment.
exitCond	Same as in *Table 2.26*.		
startAction	Same as in *Table 2.26*.		
endAction	Same as in *Table 2.26*.		

Table 2.27 Attributes for TimePoly in Start/Endpoint Representation

```
<timePoly  segID="0"    precNode="0"
      succNode="1"      default="FALSE"
      duration="500"    type="absolute"
      a0="0"   a1="0"   a2="0"   a3="0"
      a4="0"   a5="0"   exitCond="0"
      startAction="0"   endAction="0" />
```

Illustration 2.96 Coefficient Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.26*.		
precNode	Same as in *Table 2.26*.		
succNode	Same as in *Table 2.26*.		
default	Same as in *Table 2.26*.		
duration	Same as in *Table 2.26*.		
type	Same as in *Table 2.18*.		
a0	*type = absolute*: M else O	Float	Polynomial coefficients for the movement described by $a_5x^5 + a_4x^4 + a_3x^3 + a_2x^3 + a_1x + a_0$. a_0 is the same as *startPos* in the Start/ Endpoint representation.
a1-a5	M	Float	
exitCond	Same as in *Table 2.26*.		
startAction	Same as in *Table 2.26*.		

2

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
endAction	Same as in *Table 2.26*.		

Table 2.28 Attributes for TimePoly in Coefficient Representation

VelocitySegment:

The *VelocitySegment* is used for a movement with constant velocity, independent from the velocity of the guide value. It is similar to a P1 *TimePoly* of type relative, but velocity controlled instead of position controlled.

```
<velocitySeg segID="0"      precNode="0"
       succNode="1          default="FALSE"
       duration="500"       startPos="0"
       velocity="100"       acceleration="500"
       deceleration="500"   torqueLimit="0"
       exitCond="0"
       startAction="0"      endAction="0" />
```

Illustration 2.97 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.26*.		
precNode	Same as in *Table 2.26*.		
succNode	Same as in *Table 2.26*.		
default	Same as in *Table 2.26*.		
duration	Same as in *Table 2.26*.		
startPos	Same as in *Table 2.18*.		
velocity	M	Float	Velocity of the axis during this segment. The velocity must be given in rps. To ensure smooth movements, the velocities of all segments that are connected in the same node should be the same. If this is not parameterized correctly, a jump in velocity may occur.
acceleration	M	Float >0	Acceleration of the axis when increasing the velocity. The acceleration must be given in rps per second. It is possible to parameterize jumps in acceleration when 2 succeeding segments have different *startAcc* and *endAcc* values.
deceleration	O; default = value of *acceleration*	Float >0	Deceleration of the axis when decreasing the velocity. The deceleration must be given in rps per second. It is possible to parameterize jumps in acceleration when 2 succeeding segments have different *startAcc* and *endAcc* values.
torqueLimit	O; default = maximum	Positive integer (0; 32767)	Configures the maximum torque used during this segment. The value is given per mNm.
exitCond	Same as in *Table 2.26*.		
startAction	Same as in *Table 2.26*.		
endAction	Same as in *Table 2.26*.		

Table 2.29 Attributes for VelocitySegment in Start/Endpoint Representation

Coefficient representation: This representation is not available.

SyncSegment:

The *SyncSegment* is used for a synchronized, velocity controlled movement in relation to the velocity of the guide value. It is similar to a *VelocitySegment*, but with a coupling factor for the velocity (*velocityRatio*).

```
<syncSeg    segID="0"    precNode="0"
      succNode="1"       default="FALSE"
      duration="500"     startPos="0"
      velocityRatio="100"  acceleration="0"
      deceleration="0"     torqueLimit="0"
      exitCond="0"
      startAction="0"    endAction="0" />
```

Illustration 2.98 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.26*.		
precNode	Same as in *Table 2.26*.		
succNode	Same as in *Table 2.26*.		
default	Same as in *Table 2.26*.		
duration	Same as in *Table 2.26*.		
startPos	Same as in *Table 2.18*.		
velocity Ratio	M	Float	Velocity of the axis during this segment. The velocity must be given as a factor between the velocity of the axis in relation to the velocity of the guide value (1 revolution of the axis per 1 round of guide value). To ensure smooth movements, the velocities of all segments that are connected in the same node should be the same. If this is not parameterized correctly, a jump in velocity may occur.
acceleration	M	Float	Acceleration of the axis when increasing the velocity. The acceleration must be given in rps per second. It is possible to parameterize jumps in acceleration when 2 succeeding segments have different *startAcc* and *endAcc* values.
deceleration	O; default = value of *acceleration*	Float	Deceleration of the axis when decreasing the velocity. The deceleration must be given in rps per second. It is possible to parameterize jumps in acceleration when 2 succeeding segments have different *startAcc* and *endAcc* values.
torqueLimit	Same as in *Table 2.29*.		
exitCond	Same as in *Table 2.26*.		
startAction	Same as in *Table 2.26*.		
endAction	Same as in *Table 2.26*.		

Table 2.30 Attributes for SyncSegment in Start/Endpoint Representation

Coefficient representation: This representation is not available.

TorqueSegment:

The *TorqueSegment* is used for a torque controlled movement, independent of the guide value.

```
<torqueSeg  segID="0"    precNode="0"
      succNode="1"       default="FALSE"
      startPos="0"       duration="500"
      torque="100"       torqueRamp="1"
      velocityLimit="500"  exitCond="0"
      startAction="0"    endAction="0" />
```

Illustration 2.99 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.26*.		
precNode	Same as in *Table 2.26*.		
succNode	Same as in *Table 2.26*.		
default	Same as in *Table 2.26*.		
duration	Same as in *Table 2.26*.		
startPos	Same as in *Table 2.18*.		
torque	M	Integer (–32768; 32767)	Configures the target torque. The value is given in mNm.
torqueRamp	O; default = maximum	Integer (1; 2147483648)	Configures the rate of change of torque. The value is given in mNm per second.

2

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
velocity Limit	O; default = maximum	Float >0	Configures the maximum velocity that can be used during this segment (absolute value). The velocity must be given in rps. When limit is reached, no more torque is generated until velocity is below limit again.
exitCond	Same as in *Table 2.26*.		
startAction	Same as in *Table 2.26*.		
endAction	Same as in *Table 2.26*.		

Table 2.31 Attributes for TorqueSegment in Start/Endpoint Representation

Coefficient representation: This representation is not available.

PwmOffSegment:

The *PwmOffSegment* is used to turn off the PWM. Enabling the PWM again afterwards takes some time.

```
<pwmOffSeg  segID="0"   precNode="0"
      succNode="1"      default="FALSE"
      startPos="0"      duration="500"
      exitCond="0"
      startAction="0"   endAction="0" />
```

Illustration 2.100 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.26*.		
precNode	Same as in *Table 2.26*.		
succNode	Same as in *Table 2.26*.		
default	Same as in *Table 2.26*.		
startPos	Same as in *Table 2.18*.		
duration	Same as in *Table 2.26*.		
exitCond	Same as in *Table 2.26*.		
startAction	Same as in *Table 2.26*.		
endAction	Same as in *Table 2.26*.		

Table 2.32 Attributes for PwMOffSegment in Start/Endpoint Representation

Coefficient representation: This representation is not available.

FrictionSegment:

The *FrictionSegment* is used to first measure the friction of the servo drive system at 2 different velocities. This friction can either be used for long-term monitoring or the servo drive can use it for an automatic compensation. The measurement occurs alternating (over the guide value cycles) with *velocityLow* and with *velocityHigh*.

This segment ends either with the defined *velocityLow* or *velocityHigh*.

```
<frictionSeg segID="0"  precNode="0"
      succNode="1"      default="FALSE"
      startPos="0"      duration="500"
      velocityLow="0"   velocityHigh="500"
      doCompensation="TRUE"   acceleration="500"
      deceleration="500"      exitCond="0"
      startAction="0"   endAction="0" />
```

Illustration 2.101 Start/Endpoint Representation

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
segID	Same as in *Table 2.26*.		
precNode	Same as in *Table 2.26*.		
succNode	Same as in *Table 2.26*.		
default	Same as in *Table 2.26*.		
startPos	Same as in *Table 2.18*.		
duration	Same as in *Table 2.26*.		
velocityLow	M	Float	Velocity of the axis during the first part of the measurement. The velocity must be given in rps.
velocityHigh	O; no default exists	Float	Velocity of the axis during this segment. The velocity must be given in rps. To ensure smooth movements, the velocities of all segments that are connected in the same node should be the same. If this is not parameterized correctly, a jump in velocity will occur.
doCompen-sation	O; default = FALSE	TRUE/FALSE	If TRUE, the measured friction is compensated automatically by the servo drive. If FALSE, the value can be used for diagnostics.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
acceleration	M	Float >0	Acceleration of the axis when increasing the velocity. The acceleration must be given in rps per second. It is possible to parameterize jumps in the acceleration when 2 succeeding segments have different *startAcc* and *endAcc* values.
deceleration	O; default = value of *acceleration*	Float >0	Deceleration of the axis when decreasing the velocity. The deceleration must be given in rps per second. It is possible to parameterized jumps in the deceleration when 2 succeeding segments have different *startAcc* and *endAcc* values.
timeout	M	Uint32	Timeout in ms for reaching *guideValue* offset and start of measuring.
guideValue	M	Float 0–0.9999	*guideValue* Offset for starting the measuring.
exitCond	Same as in *Table 2.26*.		
startAction	Same as in *Table 2.26*.		
endAction	Same as in *Table 2.26*.		

Table 2.33 Attributes for FrictionSegment in Start/Endpoint Representation

Coefficient representation: This representation is not available.

Switching between CAM profiles

Depending on the CAM configuration option master abs/rel and especially on the advanced CAM itself, there are several ways to go from 1 running CAM profile to the next. All the possibilities are described in the graphics in this section.

The following examples all show the starting point based on the time of the CAM activation request, respectively when *CAM ack* (bit 12) is set by the axis (see *chapter 2.4.5.5 Advanced CAM*).

In the following sub-chapters, it is assumed, that the servo drive is already running on the first shown CAM. The behavior that is interesting here is the transition to the second (advanced) CAM based on the point of activation request and the configuration of the second (advanced) CAM.

All illustrations in the following sub-chapters show the transition from a currently running CAM 2 (see *Illustration 2.41*) to a newly activated CAM.

The following conventions are basically used for transitions between profiles:

- If a CAM profile is aborted (Change CAM imm = 1), the current slave position is considered as end slave position.
- Master absolute uses the *GuideNode* positions as specified in the CAM.
- Master relative moves the starting node of the CAM (*nodeID* = 0) to the end point of the previous CAM. This can be the end position or the point where it has been aborted (using Change CAM imm = 1), see *Illustration 2.102*).
- When activating a non-cyclic CAM profile with *Use blend distance* = 0, the processing of it takes place in the same master cycle (as the CAM activation request) or in the next one (depending on the end point of the currently running CAM profile and the starting node of the new CAM profile). In both cases, the CAM is processed as 1 complete cycle (starting with the next upcoming starting node).
- When activating a non-cyclic CAM profile with *Use blend distance* = 1, the processing of (at least the starting node) it takes place in the same master cycle (as the CAM activation request) or a CAM error is issued. This means that the starting node must be in the same master cycle.
- When option *Use blend distance* = 0, it leads to a blending to the starting node of the CAM (*nodeID* = 0). However, this is not necessarily the next node (seen from the current guide value position).

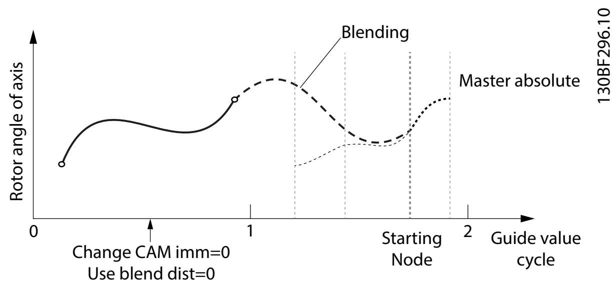


Illustration 2.102 Blending is Done to the Starting Node of the CAM; Master Absolute

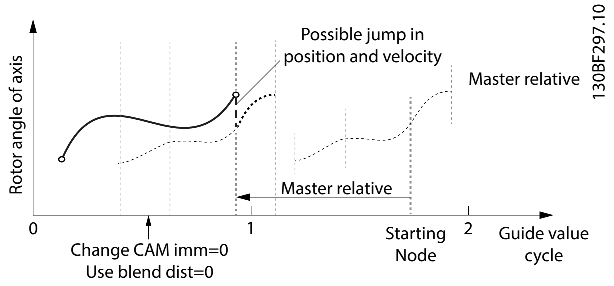


Illustration 2.103 Blending is Done to the Starting Node of the CAM; Master Relative.
Jump depends on the following segment of the starting node of the CAM.

- When option *Use blend distance* = 1, there are 2 possible cases:
  - Minimum blending distance ends before the CAM definition starts: The blending distance is extended to the next node (seen from the current guide value position, based on the default CAM).

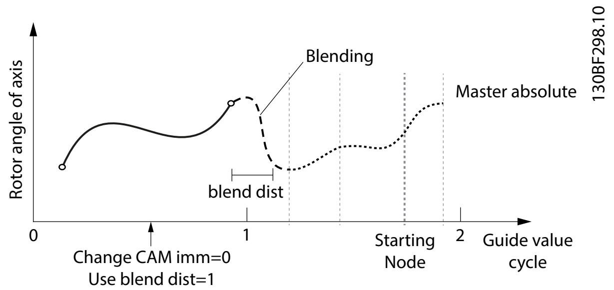


Illustration 2.104 Blending is Extended to the Next GuideNode (not necessarily the starting node)

  - Minimum blending distance ends within a segment: The behavior depends on the segment type where the blending would end (see the following sub-chapters).

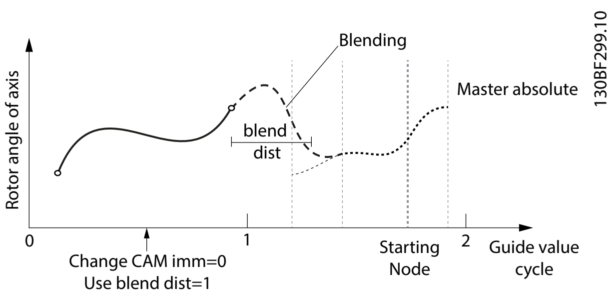


Illustration 2.105 Minimum Blending Distance Ends within a Segment

The blending behavior depends on the segment type where it ends (see *Table 2.34*). In *Illustration 2.105*, the blending is extended to the next *GuideNode* (not necessarily the starting node).

Segment type	Start position	End position
GuidePoly of type absolute	determined	determined
GuidePoly of type relative	undetermined	undetermined
MoveDistanceSegment	undetermined	undetermined
FlyingStopSegment	undetermined	undetermined
ReturnSegment	undetermined	determined
EventSegmentContainer	undetermined	undetermined
TimePoly of type absolute	determined	determined
TimePoly of type relative	undetermined	undetermined
All other *EventSegments*	undetermined	undetermined

Table 2.34 Segment types and their Classifications of Start and End Position

Blending ends inside segment with determined end position

If the segment is a segment with a determined end position (see *Table 2.34*), the blending distance is extended to the end of the segment and the blending is done to this absolute (determined) position.

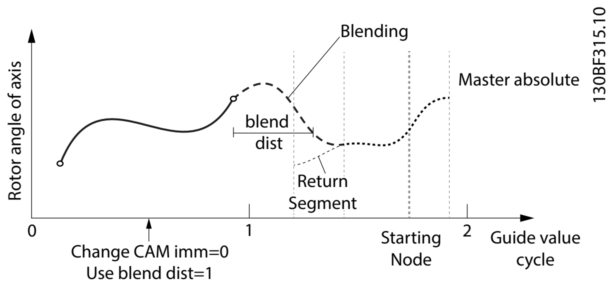


Illustration 2.106 Blending Ends inside a Segment with Determined End Position (Here: ReturnSegment)

A special case is the *GuidePoly* of type absolute. Here, the whole segment (not only the end position) is determined. So for *GuidePolys* the blending distance is not extended.

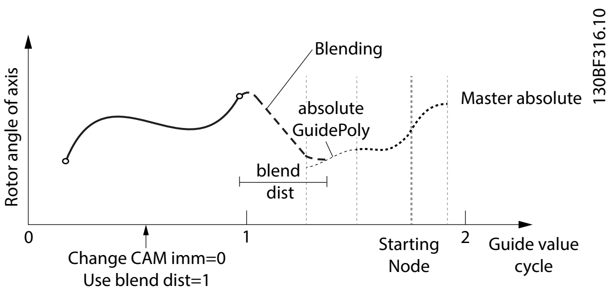


Illustration 2.107 Blending Ends inside a GuidePoly of Type Absolute.
The blending is done to that exact position.

Blending ends inside segment with undetermined end position

If the segment has an undetermined end position (see *Table 2.34*), the blending distance is extended to the end of the segment. The blending is done to the preceding node of the segment.

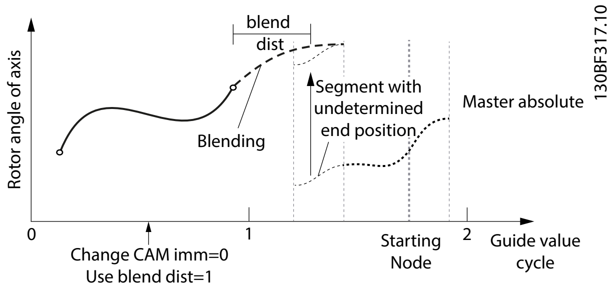


Illustration 2.108 Blending Ends inside a Segment with Undetermined End Position.
The blending is extended to the next node. The blending behavior then depends on the node.

A special case is the *GuidePoly* of type relative. Here, the servo drive calculates a P4 to adjust the velocity and the acceleration to match at the point where the blending distance ends. The position is not relevant here.

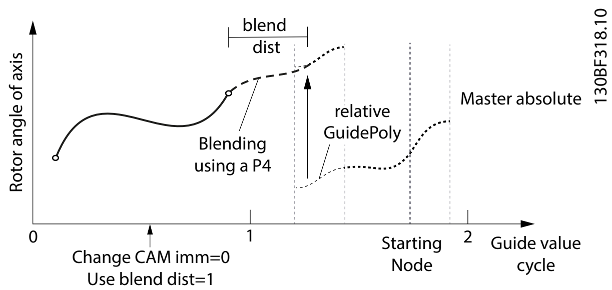


Illustration 2.109 Blending Ends inside a GuidePoly of Type Relative.
The blending distance is as specified.

The slave position is not relevant here, but is determined automatically by the P4 that is calculated by the servo drive to adjust the velocity and acceleration.

Blending ends at a node

The behavior is all the same, independent if this node is the starting node of a CAM, or some other node. It is also the same, if the blending distance has been extended to the node or not. When blending to a node, the following segment of this node is relevant.

For the *EventSegmentContainer*, the first *EventSegment* is relevant. If the following segment is a segment with a determined start position (see *Table 2.34*), a P5 is used to blend to this position.

If the following segment is a segment with an undetermined start position (see *Table 2.34*), the servo drive calculates a P4 to do the blending in order to adjust the velocity and acceleration of this start condition. The slave position itself is not relevant.

A CAM error is issued if there is no following segment to a node (as it is for example: the last node of a non-cyclic CAM).

Actions

A list of actions can be attached to several events. These events can be:

- A node.
- The beginning of a segment.
- The end of a segment.

The order of executing actions when processing segment A, node B, and segment C is the following:

- Start actions of segment A.
- End actions of segment A.
- Actions of node B.
- Start actions of segment C.
- End actions of segment C.

An action is described with a surrounding element to define an *actionID* which is used for referencing inside the CAM profile. This *actionID* must be unique across all defined actions. Inside this action element, there can be 1 or more sub-elements.

Available actions are listed in the following sub-chapters.

```
<action     actionID="0">

        ... specific action(s) with corresponding attributes

</action>
```

Illustration 2.110 Actions

Action: Change set of control loop parameters

To define an action that changes a set of control parameters, the following element must be inserted inside the action. The definition and value ranges are equal to the general definition of a control parameter set within a CAM profile.

To change the control parameters for the 1st set use:

```
<controlParam1    speedP="0.1"   speedI="0.1"
                  speedD="0.0"   inertia="0.0004"
                  positionP="6"  positionD="0" />
```

Illustration 2.111 Control Parameters for Set 1

To change the control parameters for the 2nd set use:

```
<controlParam2    speedP="0.1"  speedI="0.1"
                  speedD="0.0"  inertia="0.0004"
                  positionP="6" positionD="0" />
```

Illustration 2.112 Control Parameters for Set 2

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
speedP, speedI, speedD, inertia	M	Float, same as for object 0x2012.	See object 0x2012 (*chapter 7.6.5.1 Parameters 51-10 to 51-15: Speed Controller Parameters (0x2012)*).
positionP, positionD	M	Float, same as for object 0x2013.	See object 0x2013 (*chapter 7.6.4.1 Parameters 51-16 and 51-17: Position Controller Parameters (0x2013)*).

Table 2.35 Attributes for controlParam1

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
speedP, speedI, speedD, inertia	M	Float, same as for object 0x2014.	See object 0x2014 (*chapter 7.6.5.2 Parameters 51-20 to 51-25: Speed Controller Parameters 2 (0x2014)*).
positionP, positionD	M	Float, same as for object 0x2015.	See object 0x2015 (*chapter 7.6.4.2 Parameters 51-26 and 51-27: Position Controller Parameters 2 (0x2015)*).

Table 2.36 Attributes for controlParam2

Select set of control loop parameters

To define an action that changes the used set of control parameters, the following element must be inserted inside the action.

<selControlParam set="1"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
set	M	1/2	Switches/selects control parameter set 1 or 2.

Table 2.37 Attributes for selControlParam

Action: Set/Reset Digital Output

To define an action that changes the digital output, the following element must be inserted inside the action.

<setDigOut value="on"/>

The attribute *value* is mandatory and the allowed values are *on*, *off*, and *toggle*, where *toggle* inverts the current state of the digital output. The polarity of the digital output can be configured using object 0x200F (see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*).

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
value	M	on/off/toggle	Switches the digital output on, off, or changes the current state.

Table 2.38 Attributes for setDigOut

Action: Rounding Compensation

This action is used to compensate the rounding errors that necessarily appear during calculations. The behavior is similar to the *ReturnSegment* behavior, but there should not be an explicit movement. This means that the servo drive must be near to the correct position (so only small rounding errors can be compensated), otherwise the servo drive jumps to the corrected position.

<compensateRounding partition="1" revolutions="1" offsetRev="0.25"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
partition	M	Integer: (0;16)	Can be used for shaped plates when several equal, valid starting positions are allowed. The worst case movement is influenced by this parameter.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
revolutions	O; default = 1	Integer >0	Number of revolutions that are used when calculating valid positions, for example, if there is a gear.
offsetRev	O; default = 0	Float	Desired end rotor position relative to the nearest physical position. The reference-position is determined by the absolute position at the beginning of this segment and the *partition/ revolutions*. Given in revolutions of the axis.

Table 2.39 Attributes for compensateRounding

Action: Log Value

This action is used to log values at specific points in the CAM for later readout. All parameters that are available in the object dictionary can be logged.

There are 16 memory cells available for logging. The information is not automatically read out. This must be done by the application. Memory cells are in object 0x3870 (see *chapter 7.14.16 Parameter: Logged Values (0x3870)*).

<logValue index="0x2020" sub-index="0x01" memory="1"/>

The data must be interpreted according to the data type of the value.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
index	M	Existing parameter index	Index of the parameter to be logged.
Sub-index	O; default = 0	Existing sub-index	Sub-index of the parameter to be logged.
memory	M	Integer: [1;16]	Memory cell the parameter should be logged to.

Table 2.40 Attributes for logValue

Action: Digital Input Counter

These actions control the counters of the digital input.

<resetCounter input="1"/>

Element *resetCounter* resets the counter value to *0*.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the 1st or the 2nd digital input counter is affected.

Table 2.41 Attributes for resetCounter

<startCounter input="1" edge="rising"/>

Element *startCounter* starts the counting of the specified digital input events of the specified digital input.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the 1st or the 2nd digital input counter is affected.
edge	M	Rising/falling/ both	Indicates which input events are counted.

Table 2.42 Attributes for startCounter

<stopCounter input="1"/>

Element *stopCounter* stops the counting of any digital input events of the specified digital input.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the 1st or the 2nd digital input counter is affected.

Table 2.43 Attributes for stopCounter

The counter values can be read from the object 0x3860 (see *chapter 7.14.17 Parameter: Digital Input Counters (0x3860)*). The values are read/write for manually modifying the counters.

Action: Set Follow Segment

Instructs the servo drive to change the used succeeding segment of a node. It is only possible to select a segment ID that has this node ID defined as preceding node. This change is preserved over the guide value cycles, so no automatic switching back takes place.

<setFollowSegment nodeID="1" segID="2"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
nodeID	M	An existing node ID.	The mode ID to get another following segment. When using a non-existing *nodeID*, a notification from the axis is sent.
segID	M	An existing segment ID.	The segment ID that will be processed after the specified node. When using a non-existing *segID*, a notification from the axis is sent.

Table 2.44 Attributes for setFollowSegment

Exit conditions

The following exit conditions are used to monitor several variables. The axis proceeds with the next segment as soon as the condition is met. Exit conditions can only be defined for *EventSegments*.

An exit condition is described with a surrounding element to define an *exitID* which is used for referencing inside the CAM profile. This *exitID* must be unique across all defined exit conditions. Inside this exit element, there can be 1 or more *subelements*. Available exit conditions are listed in the following sub-chapters.

<exit exitID="0">
… specific exit condition(s) with corresponding attributes
<exit>

Exit: Rectangle Mark Detection

This exit condition is used to start the search for a rectangle mark, using the sensor interface. This exit condition is used for alignment, depending on a sensor signal. When using this exit condition, the axis waits for a rectangle input on the sensor interface with a length between the specified minimum and maximum.

When using an analog sensor, a threshold for the height of the impulse must be defined. Positive and negative impulses can be processed. This equates to light and dark marks with optical sensors. The axis proceeds with the next segment as soon as the impulse is found or the maximum duration of the segment is reached.

If the mark has been found and the following segment is a braking segment (*TimePoly* which leads to a *standstill*), the servo drive always stops at the same distance to the mark.

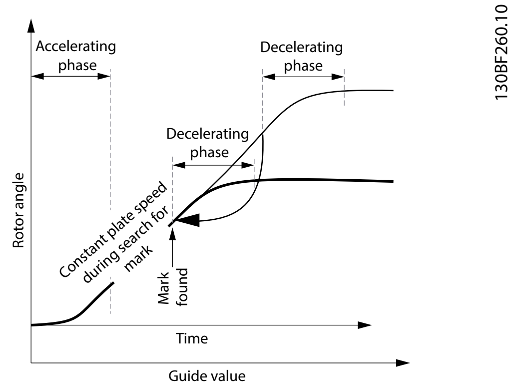


Illustration 2.113 Behavior when Mark was Found

The time at which the mark is found depends on the position of the mark. The black line shows an example for the case that the mark is found right at the point in time that is marked with the black arrow.
The duration of the segment determines the latest point in time when the search is aborted. If the mark is found before this duration is over, the axis proceeds with the following segment immediately after the mark is found.

Proceeding to the next segment always takes place in relation to the middle of the impulse. To make this possible, the point in time for proceeding depends on the parameterized maximal length of the mark.

<rectMark input="1" mode="analogue" threshold="50" minLength="300" maxLength="400"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the 1st or the 2nd digital input is affected.
mode	M	Analogue/ digital	Specifies the signal source. Allowed values are analog or digital.
threshold	M	Float: [-100; 100]%	Threshold for the sensor signal in %. Negative values are used for inverse mark polarity.
minLength	M	Integer: 0 to maxLength	Specifies the maximum length that is recognized as a mark; Given in number of samples.
maxLength	M	Integer: minLength to 65535	Specifies the maximum length that is recognized as a mark; Given in number of samples.

Table 2.45 Attributes for rectMark Search

Exit: Pattern detection

Just like the search for a rectangle mark, also the search for a pattern is used for alignment, depending on a sensor signal. In contrast to the rectangle mark, here it is possible to search for any mark. Therefore, it is necessary to download the reference signal to the axis together with the CAM profile. A pattern search can only be done using an analog sensor.

The behavior of the search for pattern exit condition is more or less equivalent to the search for a rectangle mark (see *Illustration 2.113*). As soon a pattern is recognized, the axis proceeds with the next segment. In addition to the reference pattern, the axis only needs a threshold for the expected correlation. It is usually placed in the middle between the highest disturbing signal and the expected desired signal.

The time at which the pattern is found, depends on the position of the pattern. The black line shows an example for the case that the pattern is found right at the point in time that is marked with a black arrow. The position of the succeeding node determines the latest point in time when the search is aborted. If the pattern is found before reaching the succeeding node, the axis proceeds with the following segment right after the pattern is found, that means before the succeeding node is reached.

Proceeding to the next segment always takes place in relation to the end of the pattern. When changing the position or the length of the reference pattern, the position where the axis stops is also changed.

<pattern input="1" threshold="50" subsample="1" checkLength="1"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the first or the second digital input is affected.
threshold	M	Float: [-100; 100]%	Threshold for the minimum correlation in %. Negative values are used for inverse mark polarity.
subsample	M	0–4	Subsampling factor for sensor input. 0: 16 kHz or 20 kHz 1: 8 kHz or 10 kHz 2: 4 kHz or 5 kHz 3: 2 kHz or 2.5 kHz 4: 1 kHz or 1.25 kHz
checkLength	M	Integer: [1; 1000]	Number of consecutive descending correlation samples after correlation maximum.

Table 2.46 Attributes for Pattern Search Action

Each CAM profile has 1 pattern file associated. This means that, if there is >1 pattern search action inside 1 CAM, they would use the same pattern file.

The pattern information is transmitted to objects 0x3830 to 0x3837 (see *chapter 7.14.6 Parameters: CAM Pattern 1–8 (0x3830–3837)*).

Check Digital Input Event

This exit condition checks for the state of the digital input. As soon as the specified state is reached, the axis proceeds with the next segment.

<checkDigInput input="1" value="off"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the 1st or the 2nd digital input is affected.
value	M	On/off/toggle	Switches the digital output on, off, or changes the current state.

Table 2.47 Attributes for Pattern Search Action

Counter Exceeds Limit

This exit condition checks for the digital input counters that are controlled via actions (see section *Digital Input Counter* in *chapter 2.4.5.5 Advanced CAM*). As soon as the threshold value is reached or exceeded, the axis proceeds with the next segment.

<checkCounter input="1" threshold="500"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the 1st or the 2nd digital input counter is affected.
threshold	M	Positive integer	Defines the threshold of the counter (greater or equal).

Table 2.48 Attributes for checkCounter

Check Analog Input Event

This exit condition checks for the state of the analog input. As soon as the specified state is reached, the axis proceeds with the next segment.

<checkAnaInput input="1" threshold="0.5" condition="above">

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
input	M	1/2	Selects if the 1st or the 2nd analog input is affected.
threshold	M	0–1	Threshold to be exceeded or underrun. Scaled from 0 to 1.
condition	M	Above/below	Selects if the segment should be left if the threshold has been exceeded or underrun.

Table 2.49 Attributes for checkAnaInput

Exit: Velocity Below/Above Limit

This exit condition checks if the velocity is below or above the specified absolute. As soon as the value is above or below the threshold, the axis proceeds with the next segment.

<checkVelocity threshold="500" condition="above"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
threshold	M	Float	Velocity threshold to be exceeded or underrun. The velocity must be given in rps.
condition	M	Above/below	Selects if the segment should be left if the threshold has been exceeded or underrun.

Table 2.50 Attributes for checkVelocity

Torque Below/Above Limit

This exit condition checks if the torque is above or below the specified absolute value. As soon as the value is above or below the threshold, the axis proceeds with the next segment.

<checkTorque threshold="500" condition="above"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
threshold	M	Float	Torque threshold to be exceeded or underrun. The torque must be given in mNm.
condition	M	Above/below	Selects if the segment should be left if the threshold has been exceeded or underrun.

Table 2.51 Attributes for checkTorque

Distance Above Limit

This exit condition checks if the distance that has been processed during the current segment is above the specified absolute value. As soon as the value is above the threshold, the axis proceeds with the next segment.

<checkDistance threshold="500"/>

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
threshold	M	Float	Distance threshold to be exceeded. Given in revolutions of rotor position.

Table 2.52 Attributes for checkDistance

2.4.5.6 Commands During Operation

The commands listed in this chapter are provided by the servo drive to control the functionality during the operation of a CAM. Some commands are only available if an advanced CAM is used. The CAM control data information is represented in 4 16 bit objects (see *chapter 7.14.3 Parameter: CAM Control (0x3800)* for object description). One of them is the control code, whereas the rest contain additional parameters (see *Table 2.53*). The detailed descriptions are given in the following sub-chapters.

Bit 16 (MSB) of the control code is a toggle bit. As synchronous fieldbuses are supported, it is not possible to distinguish between a new and a resent command. Therefore, the edge of the toggle bit is used for this purpose.

<table>
<tr><th>Control code</th><th>Meaning</th><th>Availa-bility</th><th>Control parameter 1</th><th>Control parameter 2</th><th>Control parameter 3</th></tr>
<tr><td>0x0000</td><td>Reserved</td><td>–</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td rowspan="2">0x0001</td><td rowspan="2">Rotation stop</td><td rowspan="2">Basic & Advanced</td><td rowspan="2">Rotation stop option code (see Table 2.54)</td><td colspan="2">Deceleration [float; rps per second]</td></tr>
<tr><td>Low byte</td><td>High byte</td></tr>
<tr><td rowspan="2">0x0002</td><td rowspan="2">Segment parameter during run-time</td><td rowspan="2">Advanced only</td><td rowspan="2">SegmentID</td><td colspan="2">Parameter [float; in revolutions]</td></tr>
<tr><td>Low byte</td><td>High byte</td></tr>
<tr><td>0x0003</td><td>Set follow segment</td><td>Advanced only</td><td>nodeID</td><td>SegmentID</td><td>Reserved</td></tr>
<tr><td>0x0004</td><td>Node signaling status (leads to a status information with status code 0x0005)</td><td>Basic & Advanced</td><td>nodeID/No. of data point</td><td>1: Enable
0: Disable</td><td>Reserved</td></tr>
<tr><td>0x0005</td><td>Go to setpoint (while guide value velocity is 0)</td><td>Basic & Advanced</td><td>Direction option code (see Table 2.55)</td><td>Time in ms</td><td>Reserved</td></tr>
</table>

Table 2.53 CAM Control Data Information

When using the PLC, the libraries provide function blocks to send the commands. The function blocks are described in *chapter 6.5.6 Drive – CAM Operation*.

Rotation stop

This command issues a stop of the servo drive for 1 CAM cycle. The stopping takes place according to the *Table 2.54*.

Value	Definition
0	Coasting and stay in *Operation enabled.*
+1	Slow down on specified ramp and stay in *Operation enabled.*
+2	*Slow down* on current limit and stay in *Operation enabled.*

Table 2.54 Rotation Stop Option

The CAM processing is resumed at the starting node of the CAM. Ensure that the resuming can take place without jumps.
For advanced CAMs, this can be done by:

- Starting the CAM with a relative movement.
- Starting the CAM with a *ReturnSegment* (suggested solution).

For basic CAMs, use the slave relative option. A jump occurs if the CAM starts with an absolute movement and the servo drive is at a different position. No blending occurs.

Segment parameter during run-time

This command sends a parameter to a specific segment. The parameter influences the behavior of the segment during run-time. Segment types that expect an angle parameter at every guide value cycle (master cycle) are *Move Distance segments* and *Flying Stop segments* (see *chapter 5.7.7.7 Editing Advanced CAM Profiles*).
The specific segment is addressed using its ID. The parameter is a floating point value given in rotor shaft revolutions.

Set follow segment

This command instructs the servo drive to change the used succeeding segment of a node. It is only possible to select a segment ID that already has this node ID defined as preceding node. This change is preserved over the guide value cycles. Therefore, no automatic switching back takes place.

Node signaling status

This command enables/disables the signaling of a node/ data point. That means that the servo drive can send a notification when passing a node/data point. If there are too many notifications coming from the axis, others could be delayed. Basic CAMs do not signal the passing of a data point per default.

Go to setpoint

This command issues a movement to the setpoint of the CAM while the *Guide Value velocity* is *0*. This is used, for example, when starting up a CAM with slave absolute and the servo drive is at another position. The required movement is then calculated by the servo drive automatically, based on the direction option code (see *Table 2.55*) over the specified time. This movement takes place using a polynomial of 5th degree. The *Guide Value velocity* must stay at *0* until this movement is finished.

Value	Definition
0	Normal movement similar to linear axis.
+1	Movement only in negative direction.
+2	Movement only in positive direction.
+3	Movement the shortest way. Assuming a rotary axis. Cam slave scaling is considered (regarding a possible gear).
+4	Movement in last direction.

Table 2.55 Direction Option Code

2.4.5.7 Notifications from the Servo Drive

The servo drive sends out information about the currently ongoing CAM execution or as a reaction on a command. Bit 16 (MSB) of the status code is a toggle bit. As synchronous fieldbuses are supported, it is not possible to distinguish between a new and a resent notification. Therefore, the edge of the toggle bit must be monitored. The CAM status information is represented in 4 16-bit objects (see *chapter 7.14.2 Parameter: CAM Status (0x3801)*). One of them is the status code, whereas the rest contains additional information (see *Table 2.56*).

Status code	Meaning	Status parameter 1	Status parameter 2	Status parameter 3
0x0000	Reserved	Reserved	Reserved	Reserved
0x0002	Result of dynamic alignment	*SegmentID* of alignment segment (pattern or mark)	1: Success 0: Failure	Reserved
0x0004	Following error (also signaled in the *Statusword*)	*SegmentID* of the segment in which the following error occurred	Reserved	Reserved
0x0005	Node/data point passed	*nodeID*/data point that was passed	*SegmentID* of current segment/not available for basic CAM	Reserved
0x0006	Bad parameter sent to a segment or segment does not exist	Sent *SegmentID*	Sent parameter [float]	
			Low byte	High byte
0x0007	Bad parameter sent: Error when setting following segment of a node (node or segment not valid)	Sent *SegmentID*	Sent *nodeID*	Reserved
0x0009	Correction angle indication	ID of *MoveDistanceSegment*	Logical rotor angle [float; given in revolutions]	
			Low byte	High byte
0x000A	Flying stop angle indication	ID of *FlyingStopSegment*	Logical rotor angle [float; given in revolutions]	
			Low byte	High byte
0x000B	Forced Time-exit; *EventSegmentContainer* too short	ID of *EventSegmentContainer*	Reserved	Reserved

Table 2.56 CAM Control Data Information

The PLC library provides the information in function block *chapter 6.5.6.9 DD_ReadCamInfo_ISD51x*.

2.4.6 Gear Mode

In *Gear mode*, the servo drive executes a synchronized movement based on a master axis by using a gear ratio between the master and the slave position. The guide value can be provided by an external encoder, virtual axis, or the position of another axis. This functionality can be commanded using function block MC_GearIn_ISD51x (see *chapter 6.5.5.9 MC_GearIn_ISD51x*) and MC_GearInPos_ISD51x (see *chapter 6.5.5.10 MC_GearInPos_ISD51x*).

The slave axis calculates its position out of the master position value (see *chapter 7.8.1 Parameter: Position Guide Value (0x2060)*). The slave axis sets its target position corresponding to the configured gear ratio (see *chapter 7.15.1 Parameter: Gear Ratio (0x3900)*). The principle of the *Gear mode* is shown in *Illustration 2.114*.

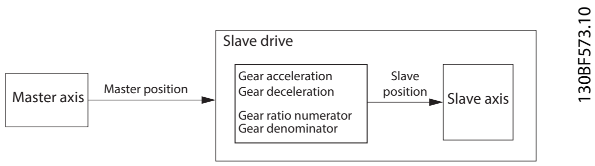


Illustration 2.114 Gear Mode Description

The slave velocity is calculated as:

$$Slave\ velocity = \frac{gear\ ratio\ numerator}{gear\ ratio\ denominrator} \ x\ \ master\ velocity$$

To start a geared movement, the acceleration (see *chapter 7.5.7 Parameter 50-11: Profile Acceleration (0x6083)*) and deceleration (see *chapter 7.5.8 Parameter 50-12: Profile Deceleration (0x6084)*) can also be configured. These parameters are also used to link up the gear. The slave ramps up or down to the ratio of the master velocity according to the given acceleration or deceleration value and locks in when this velocity is reached.

There are 2 synchronization methods:

- The relative synchronization between the master and the slave is important (*Gear In* functionality). For the *Gear In* functionality, the synchronization phase is velocity controlled, so any lost distance during synchronization is not caught up.
- The absolute relation between master and slave is important (*Gear In Pos* functionality), as shown in *Illustration 2.115*.

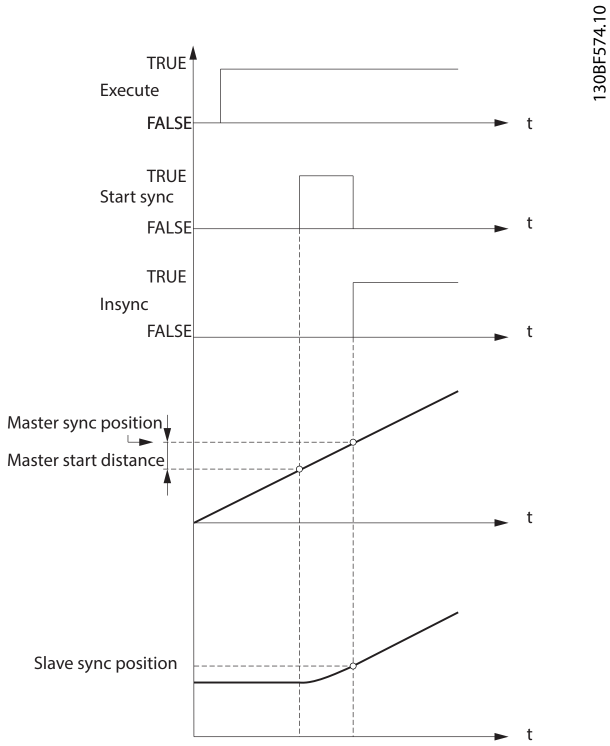


Illustration 2.115 Timing Diagram for Gear In Position Procedure

A polynomial of maximum 5th degree is used for the synchronization phase.

The mode is activated by writing –7 to object 0x6060.

For the *Controlword* (see *chapter 7.2.1 Parameter 16-00 Controlword (0x6040)*) and the *Statusword* (see *chapter 7.3.1 Parameter 16-03 Statusword (0x6041)*), the bits that usually hold the operating mode-specific bits are defined here.

Depending on the value of the *Guide value option code* object (see *chapter 7.8.3 Parameter: Guide Value Option Code (0x2061)*), the guide value (backward or forward movement) must be handled. The parameters specific to this mode are listed in *chapter 7.15 Gear Mode Objects*.

2.4.7 ISD Inertia Measurement Mode

This mode measures the inertia of an axis. It is used to measure the inertia of the servo drive and the external load, and can be used to optimize the control loop settings. The friction effects are eliminated automatically.

This functionality can be commanded using function block DD_GetInertia_ISD51x (see *chapter 6.5.5.11 DD_GetInertia_ISD51x*). It can also be used via LCP parameter *52-6* Inertia Measurement*. The measured inertia is written to object 0x2009 (see *chapter 7.16.1 Parameter 52-60: Measured Inertia (0x2009)*).

The measured value is not automatically used by the control loop.

WARNING

DANGER OF MOVING PARTS

The servo drive moves during the measurement.

- **Limit the maximum velocity to be used during measurement using object 0x200A, sub-index 01 (see *chapter 7.16.2 Parameters 52-61 and 52-62: Inertia Measurement Parameters (0x200A)*).**
- **Limit the maximum torque to be used during measurement using object 0x200A, sub-index 02 (see *chapter 7.16.2 Parameters 52-61 and 52-62: Inertia Measurement Parameters (0x200A)*).**

To start the measurement, the servo drive must be switched to *ISD Inertia Measurement mode*. Switching is always possible when the servo drive is disabled. If the servo drive is in state *Operation enabled*, it must be in *Standstill* (defined in *chapter 10.1 Glossary*). Start the measurement by using bit 4 in the *Controlword* (see *chapter 7.2.1 Parameter 16-00 Controlword (0x6040)*). The end of the measurement is reported in the *Statusword* (see *chapter 7.3.1 Parameter 16-03 Statusword (0x6041)*). After the measurement, the value can be read from object 0x2009 (see *chapter 7.16.1 Parameter 52-60: Measured Inertia (0x2009)*).

If an error occurred during the measurement, the servo drive signals the error in the *Statusword* and the value of object 0x2009 (see *chapter 7.16.1 Parameter 52-60: Measured Inertia (0x2009)*) is used for the error reason.

2.4.8 Cyclic Synchronous Position Mode

In *Cyclic synchronous position mode*, the trajectory generator of the position is located in the control device, not in the servo drive. The overall structure for this mode is shown in *Illustration 2.116*. The servo drive provides actual values for position, velocity, and torque to the control device. In cyclic synchronous manner, it provides a target position to the servo drive, which performs position control, velocity control, and torque control.

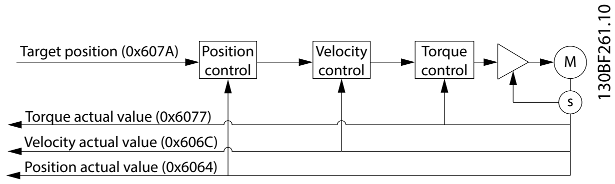


Illustration 2.116 Cyclic Synchronous Position Mode Overview

Illustration 2.116 shows the inputs and outputs of the servo drive control function. The input value (from the control function point of view) is the target position.

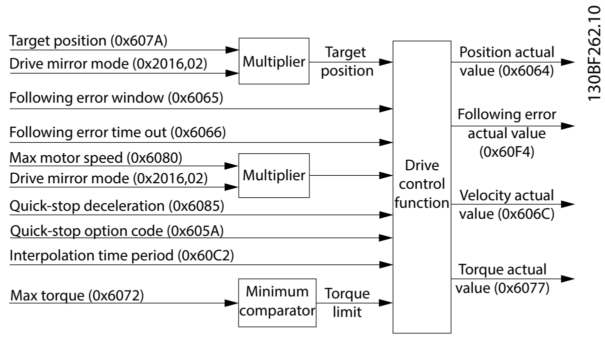


Illustration 2.117 Cyclic Synchronous Position Control Function

The servo drive monitors the following error. Other features specified in this mode are limitation of motor speed and a quick stop function for emergency reasons. The torque is limited as well. The interpolation time period defines the time period between 2 updates of the target position and is used for intercycle interpolation.
The target position is interpreted as absolute value. The position actual value is used as output to the control device. Further outputs are the velocity actual value, torque actual value, and the following error actual value. All values are given in user-defined units.
A target position value outside the allowed range of the *following error window* around a *position demand value* for longer than the *following error time-out* results in setting bit 13 (*Following error*) in the *Statusword* to *1*. Object 0x2055: Following error option code is not supported in this mode of operation.

2.4.9 Cyclic Synchronous Velocity Mode

In *Cyclic synchronous velocity mode*, the trajectory generator of the velocity is located in the control device, not in the servo drive. The overall structure for this mode is shown in *Illustration 2.118*. The servo drive provides actual values for position, velocity, and torque to the control device. In cyclic synchronous manner, it provides a target velocity to the servo drive, which performs velocity control and torque control.

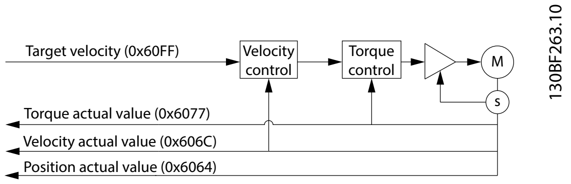


Illustration 2.118 Cyclic Synchronous Velocity Mode Overview

Illustration 2.119 shows the inputs and outputs of the servo drive control function. The input value (from the control function point of view) is the target velocity.

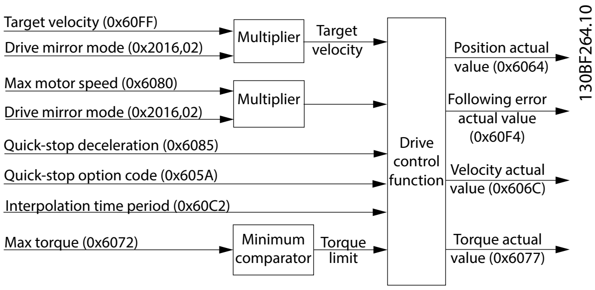


Illustration 2.119 Cyclic Synchronous Velocity Control Function

The servo drive supports limitation of motor speed and a quick stop function for emergency reasons. The torque is limited as well. The interpolation time period defines the time period between 2 updates of the target velocity and/or additive velocity and is used for intercycle interpolation. The position actual value is used as mandatory output to the control device. The PLC calculates the actual velocity from the changes to the actual position changes. All values are given in user-defined units.

2.5 Motion Functions

Function	Description
Digital CAM switch	This functionality controls whether the digital output is enabled or disabled, depending on the axis position. It performs a function comparable to switches on a motor shaft. Forward and backward movements of the axis position are allowed. On and off compensation and hysteresis can be parameterized.
ISD touch probe	This functionality stores the position actual value at a rising or falling edge of the configured digital input.
Guide value	The guide value is used in all synchronous modes of operation (*CAM mode* and *Gear mode*). It is used as the master position within the synchronous modes.

Table 2.57 Motion Functions

2.5.1 Digital CAM Switch

This functionality controls whether the digital output is enabled or disabled, depending on the axis position. It performs a function comparable to switches on a motor shaft. Forward and backward movements of the axis position are allowed. On and off compensation and hysteresis can be parameterized.

The digital CAM switches are stored and handed over to the servo drive using the contents of an XML file. The content is stored automatically in the servo drive. There is only 1 configuration for the digital CAM switches and a maximum of 100 switches are supported.

The calculation of the digital CAM switches is based on the *Position actual value* (see *chapter 7.7.5 Parameter 50-03: Position Actual Value (0x6064)*) in all modes of operation except *CAM mode*. In *CAM mode*, the calculation is based on the *Logical CAM position* (see *chapter 7.14.12 Parameter: Logical CAM Position (0x2020)*). The cyclic usage of switches is based on the range of the *Position actual value* and/or the *Logical CAM position*.

Information about the state of the digital CAM switching functionality is given in object 0x2005 (see *chapter 7.22.13 Parameter 50-07: Overlaying Motion Status (0x2005)*).

A compensation time with which the switching on (see *chapter 7.17.1 Parameter: On Compensation (0x3840)*) or the switching off (see *chapter 7.17.2 Parameter: Off Compensation (0x3841)*) can be advanced or delayed in time.

A hysteresis can be defined by using object 0x3842 (see *chapter 7.17.3 Parameter: Hysteresis (0x3842)*) to avoid jittering around the switching point.

To use the digital CAM switch, transfer the file content to object 0x3844 (see *chapter 7.17.5 Parameter: Digital CAM Switches Data (0x3844)*). Afterwards, parse the profile using object 0x3843 (see *chapter 7.17.4 Parameters: Digital CAM Switch Parsing Control (0x3843)*). When the status signals that the data is valid, the functionality can be enabled by using the *Controlword* (see *chapter 7.2.1 Parameter 16-00 Controlword (0x6040)*).

NOTICE

The digital output must be configured to be used for the digital CAM switching functionality by using object 0x2FFF (see *chapter 7.21.5 Parameter 52-05: Digital Output Configuration (0x2FFF)*). Otherwise, the activation of the digital CAM switch has no effect.

This functionality can be commanded via PLC by using function blocks *DD_PrepareDigCamSwitch_ISD51x* and *DD_DigitalCamSwitch_ISD51x*.

```
<?xml version="1.0"?>
<DigitalCamSwitch version="0.0.0.1">
    <OnCompensation> -125 </OnCompensation>
    <OffCompensation> 250 </OffCompensation>
    <Switches Unit="User"     Hysteresis="150">
    <Switch CamSwitchMode="Position"
     FirstOnPosition="2000" LastOnPosition="3000"
     AxisDirection="Positive" />
    <Switch CamSwitchMode="Position"
     FirstOnPosition="2500" LastOnPosition="3000"
     AxisDirection="Negative" />
    <Switch CamSwitchMode="Position"
     FirstOnPosition="4000" LastOnPosition="1000"
     AxisDirection="Both" />
    <Switch CamSwitchMode="Time"
     FirstOnPosition="3000" AxisDirection="Both"
     Duration="1350" />
       ...
    </Switches>
</DigitalCamSwitch>
```

Illustration 2.120 XML Representation of Digital CAM Switches

Each file can only contain 1 *DigitalCamSwitch* element.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
version	O	x.x.x.x	Gives the version of the digital CAM switches definition.

Table 2.58 Attributes for DigitalCamSwitch Element

The *DigitalCamSwitch* element contains the following optional elements:

- *OnCompensation* (see *chapter 7.17.1 Parameter: On Compensation (0x3840)*).
- *OffCompensation* (see *chapter 7.17.2 Parameter: Off Compensation (0x3841)*).

If those elements are present, the parameters are written to the object dictionary at the point of enabling of the digital CAM.

When disabling the digital CAM switching functionality, the values persist (that is, they do not switch back to values, before the enabling). It is also possible to change those values during the operation of the digital CAM by writing different values to the objects. This does not change the content of object 0x3844 (see *chapter 7.17.5 Parameter: Digital CAM Switches Data (0x3844)*).
Switches is a mandatory element.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
Unit	M	User/ revolutions	Defines the unit in which all the position values in this file are given. Allowed values are: • User: The position values are given in user-defined position units. All numerical values accept integers only. • Revolutions: The position values are given in shaft revolutions (matching the units that are used in *CAM mode*). The numerical values accept floats and integers.
Hysteresis	M	If unit = User: integer ≥0 If unit = Revolutions: float ≥0	See *chapter 7.17.3 Parameter: Hysteresis (0x3842)* for the description. This parameter is written to the object dictionary at the point of enabling of the digital CAM. When disabling the digital CAM switching functionality, the value persists (that is, it does not switch back to value before the enabling). It is also possible to change this value during the operation of the digital CAM by writing a different value to the object. It is not possible to change the parameter in the file on-the-fly. The unit of this value depends on the *Unit* attribute of the *Switches* element.

Table 2.59 Attributes for Switches Element

The *Switches* element itself contains 1–100 switch elements.

Attribute	Mandatory/ optional (+default value)	Value range/ allowed values	Description
CamSwitchMode	M	Position/ time	Position and time-based switches are possible and a mixture of both can be used. Possible values are: • Position: Position based. Attribute *LastOnPosition* is mandatory, attribute *Duration* is not used. • Time: Time based. Attribute *LastOnPosition* is not used, attribute *Duration* is mandatory.
FirstOnPosition	M	If unit = User: integer ≥0 If unit = Revolutions: float ≥0	Lower boundary where the switch is ON. The unit of this value depends on the *Unit* attribute of the *Switches* element.
LastOnPosition	M if CamSwitchMode = Position; not used otherwise	If unit = User: integer >0 If unit = Revolutions: float >0	Upper boundary where the switch is ON. The unit of this value depends on the *Unit* attribute of the *Switches* element.
AxisDirection	M	Both/ positive/ negative	Defines in which directions the switches are used: • Both: Switch is active in both directions. • Positive: Switch is only active, when the servo drive moves in positive direction. • Negative: Switch is only active, when the servo drive moves in negative direction.
Duration	M if CamSwitchMode = Time; not used otherwise	integer >0	Duration that the output is ON. Given in milliseconds.

Table 2.60 Attributes for Switches Element

```
<?xml version="1.0"?>
<DigitalCamSwitch version="0.0.0.1">
    <OnCompensation> -125 </OnCompensation>
    <OffCompensation> 250 </OffCompensation>
    <Switches Unit="User"     Hysteresis="150">
    <Switch CamSwitchMode="Position"      FirstOnPosition="2000"
     LastOnPosition="3000"    AxisDirection="Positive" />
    <Switch CamSwitchMode="Position"      FirstOnPosition="2500"
     LastOnPosition="3000"    AxisDirection="Negative" />
    </Switch CamSwitchMode="Position"     FirstOnPosition="4000"
     LastOnPosition="1000"    AxisDirection="Both" />
    </Switches>
</DigitalCamSwitch>
```

Illustration 2.121 Example 1

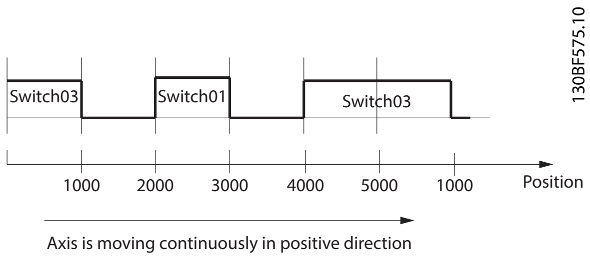


Illustration 2.122 Example 1 - Behavior of Output in Positive Direction

Illustration 2.122 shows the behavior of the output when the axis is moving continuously in a positive direction. The axis is a modulo axis with a modulo length of 5000. It does not include on/off compensation or hysteresis.

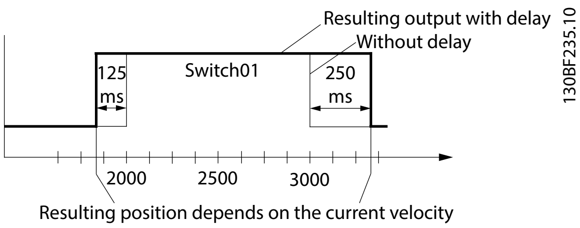


Illustration 2.123 Example 1 - Use of On/Off Compensation

Illustration 2.123 shows the additional use of on compensation of 125 ms and off compensation of 250 ms. The axis is moving continuously in a positive direction.

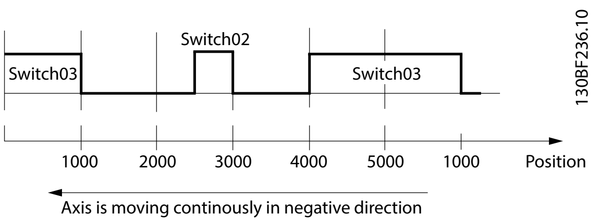


Illustration 2.124 Example 1 - Behavior of Output in Negative Direction

Illustration 2.124 shows the behavior of the output when the axis is moving continuously in a negative direction. The axis is a modulo axis with a modulo length of 5000. It does not include on/off compensation or hysteresis.

Example 2:

```
<?xml version="1.0"?>
<DigitalCamSwitch version="0.0.0.1">
    <OnCompensation> 0 </OnCompensation>
    <OffCompensation> 0 </OffCompensation>
 <Switches Unit="User"  Hysteresis="150">
 <Switch CamSwitchMode="Time" FirstOnPosition="3000"
  AxisDirection="Both"        Duration="1350" />
 </Switches>
</DigitalCamSwitch>
```

Illustration 2.125 Example 2

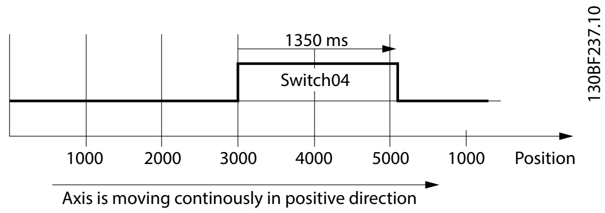


Illustration 2.126 Example 2- Behavior of Output in Positive Direction

Illustration 2.126 shows the behavior of the output when the axis is moving continuously in a positive direction. The axis is a modulo axis with a modulo length of 5000. It does not include on/off compensation or hysteresis.

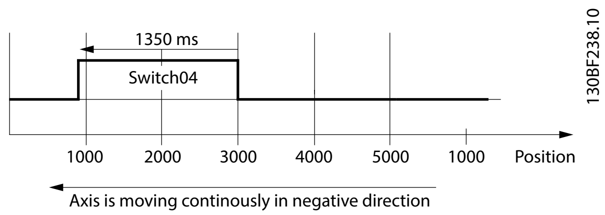


Illustration 2.127 Example 2- Behavior of Output in Negative Direction

Illustration 2.127 shows the behavior of the output when the axis is moving continuously in a negative direction. The axis is a modulo axis with a modulo length of 5000. It does not include on/off compensation or hysteresis.

2.5.2 ISD Touch Probe

This functionality stores the position actual value and a time stamp at the rising or falling edge of the configured digital input. This functionality is available in all modes of operation for each input individually.

The function is configured using object 0x60B8 (see *chapter 7.18.1 Parameter: Touch Probe Function (0x60B8)*), where the different options regarding the trigger event can be selected.

The status of the touch probe can be obtained using object 0x60B9 (see *chapter 7.18.2 Parameter: Touch Probe Status (0x60B9)*). The position results are given in objects 0x60BA–0x60BD (see *chapter 7.18.3 Parameter 51-51: Touch Probe 1 Positive Edge (0x60BA)* to *chapter 7.18.6 Parameter 51-64: Touch Probe 2 Negative Edge (0x60BD)*). The corresponding time stamps can be read using the objects 0x60D1–0x60D4 (see *chapter 7.18.10 Parameter 51-53: Touch Probe Time Stamp 1 Positive Value (0x60D1)* to *chapter 7.18.13 Parameter 51-66: Touch Probe Time Stamp 2 Negative Value (0x60D4)*).

2.5.2.1 Touch Probe Window

For touch probe events it is possible to define a window. If this functionality is activated (for touch probe 1: object 0x60B8 bit 6 = 1, and for touch probe 2: object 0x60B8 bit 14 = 1), touch probe events are only accepted within this window. The window is configured using objects 0x3853: First position (see *chapter 7.18.8 Parameter: First Position (0x3853)*) and 0x3854: Last position (see *chapter 7.18.9 Parameter: Last Position (0x3854)*).

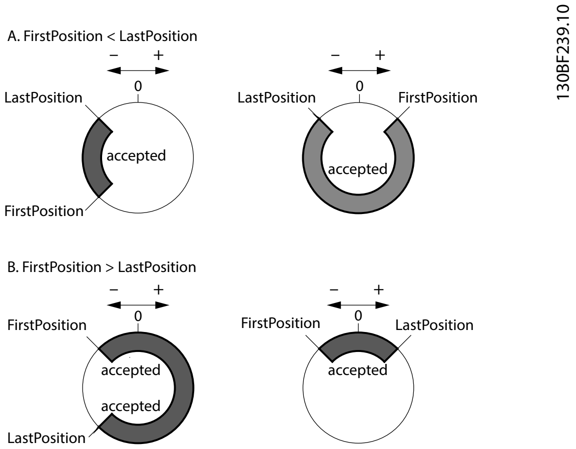


Illustration 2.128 Examples of Windows where Trigger Events are Accepted (For Modulo Axes)

2.5.2.2 Touch Probe Edge Counter for Continuous Mode

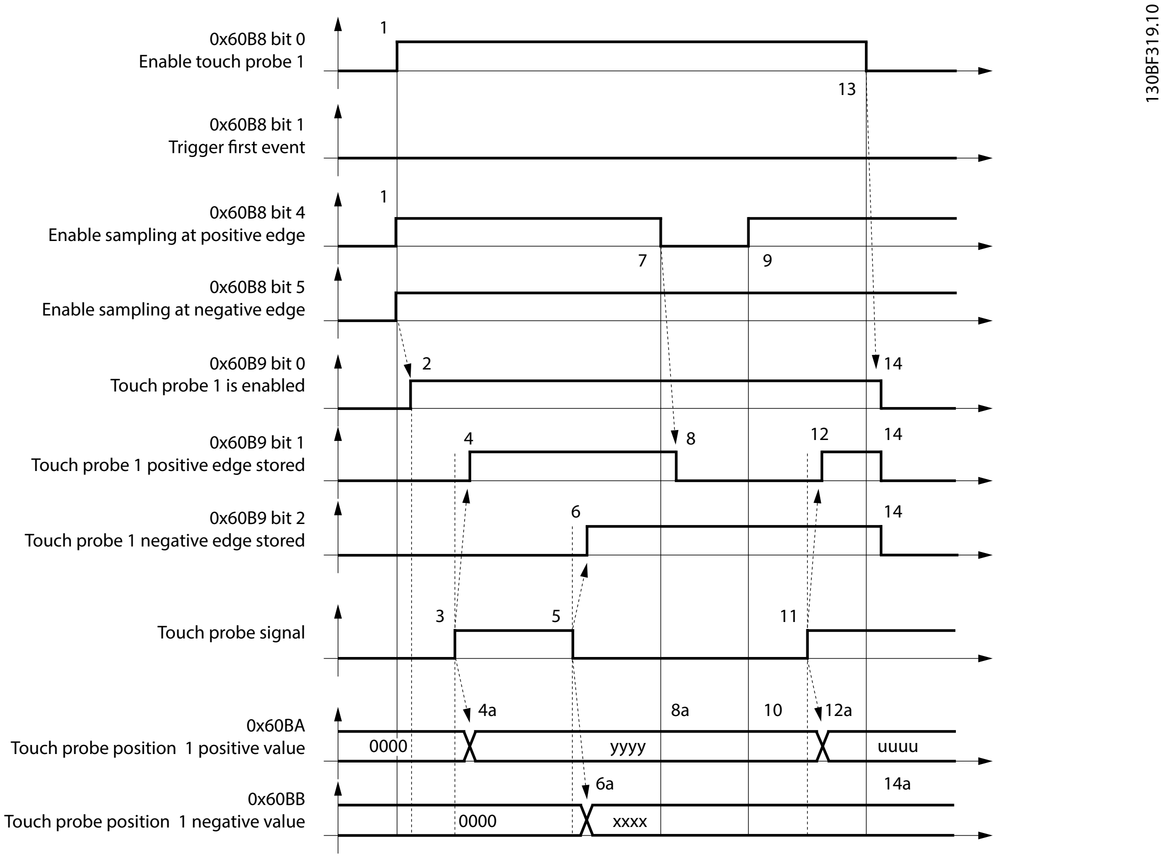
Touch probe edge counter for continuous mode

For continuous touch probe mode (0x60B8 bit 1 = 1, or 0x60B8 bit 9 = 1), a counter per touch probe channel is incremented on each touch probe event. Therefore, the control device may check how many touch probe events occur between the control cycles. A counter object is defined per touch probe and per edge. See objects:

- 0x60D5 (*chapter 7.18.14 Parameter 51-52: Touch Probe 1 Positive Edge Counter (0x60D5)*)
- 0x60D6 (*chapter 7.18.15 Parameter 51-55: Touch Probe 1 Negative Edge Counter (0x60D6)*)
- 0x60D7 (*chapter 7.18.16 Parameter 51-62: Touch Probe 2 Positive Edge Counter (0x60D7)*)
- 0x60D8 (*chapter 7.18.17 Parameter 51-65: Touch Probe 2 Negative Edge Counter (0x60D8)*)

2.5.2.3 Timing Example

Illustration 2.129 shows a timing diagram for an example touch probe configuration and the corresponding behavior.



Number	Touch probe behavior	
1	0x60B8, bit 0 = 1_b	Enable touch probe 1.
	0x60B8, bit 1, 4, 5	Configure and enable touch probe 1 positive and negative edge.
2	→ 0x60B9 bit 0 = 1_b	Status *Touch probe 1 enabled* is set.
3	External touch probe signal has positive edge	
4	→ 0x60B9 bit 1 = 1_b	Status *Touch probe 1 positive edge stored* is set.
4a	→ 0x60BA	*Touch probe position 1 positive value* is stored.
5	External touch probe has negative edge	
6	→ 0x60B9 bit 2 = 1_b	Status *Touch probe 1 negative edge stored* is set
6a	→ 0x60BB	Touch probe position 1 negative value is stored.
7	0x60B8, bit 4 = 0_b	Sample positive edge is disabled.
8	→ 0x60B9 bit 0 = 0_b	Status *Touch probe 1 positive edge stored* is reset.
8a	→ 0x60BA	*Touch probe position 1 positive value* is not changed
9	0x60B8, bit 4 = 1_b	Sample positive edge is enabled.
10	→ 0x60BA	*Touch probe position 1 positive value* is not changed.
11	External touch probe signal has positive edge.	
12	→ 0x60B9 bit 1=1_b	Status *Touch probe 1 positive edge stored* is set.
12a	→ 0x60BA	*Touch probe position 1 positive value* is stored.
13	0x60B8, bit 0 = 0_b	Touch probe 1 is disabled.
14	→ 0x60B9 bit 0, 1, 2 = 0_b	Status bits are reset.
14a	→ 0x60BA, 0x60BB	*Touch probe position 1 positive/negative values* are not changed.

Illustration 2.129 Timing Diagram for Touch Probe Example

2.5.3 Guide Value

The guide value is used in all synchronous modes of operation (*CAM mode* and *Gear mode*). It is used as the master position within the synchronous modes. The guide value consists of a position value (see *chapter 7.8.1 Parameter: Position Guide Value (0x2060)*) and an optional velocity value (see *chapter 7.8.2 Parameter: Velocity Guide Value (0x2064)*).

The servo drive also supports a scaling of the guide value. The scaling factor (see *chapter 7.8.4 Parameter: Guide Value Scaling Factor (0x3808)*) consists of a numerator and a denominator.

The *Position guide value* is multiplied by the quotient of numerator and denominator.

Guide value scaled = Guide value x Scaling factor + Guide value offset

(Internally used = 0x2060 x 0x3808.01/02 + 0x3806)

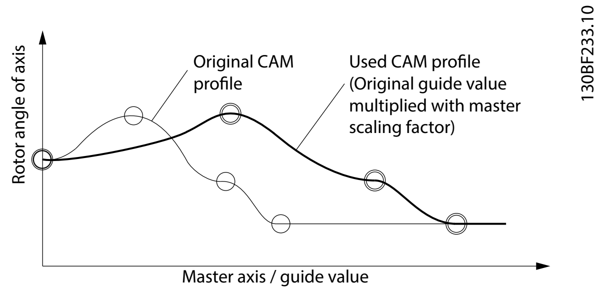


Illustration 2.130 Example of Guide Value Scaling in CAM Mode

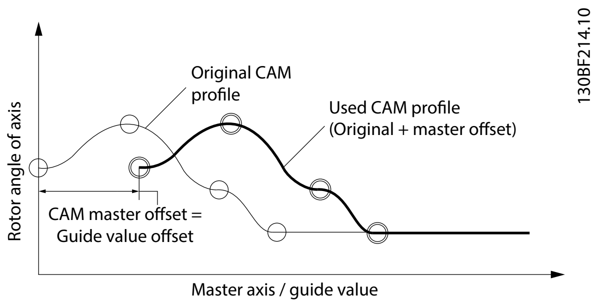


Illustration 2.131 Example of Position Guide Value Offset in CAM Mode

When receiving a guide value, the servo drive can optionally check the value against reversing and jumps in the position using object 0x2061 (see *chapter 7.8.3 Parameter: Guide Value Option Code (0x2061)*). Using this object, the servo drive can also be instructed to calculate the guide value velocity.

The guide value objects can be found in*chapter 7.8 Guide Value Objects*.

2.5.3.1 Guide Value Reference

The servo drive is also able to provide a guide value that can be used, for example, by the PLC. This generated guide value is called *Guide value reference*. It consists of the position and the velocity. The servo drive can provide these values based on different sources.

Possible guide value reference sources are:

- External encoder
- Simulation
- Actual target position
    - Actual position
    - Set position

Select 1 of these using object 0x2063 (see *chapter 7.9.3 Parameter: Guide Value Reference Option Code (0x2063)*). The objects to influence the guide value reference can be found in *chapter 7.9 Guide Value Reference Objects*.

2.5.3.2 Guide Value Reference Simulation

A guide value reference simulation functionality is provided (activate/deactivate guide value reference simulation) by the servo drive. Velocity, acceleration, and speed limits can be parameterized.

The guide value reference simulation can also be used as a virtual axis in a real application. The guide value reference simulation can be useful in commissioning scenarios when it is not possible/appropriate to use the entire machine. The guide value reference simulation can then be used to simulate that the main axis is moving.

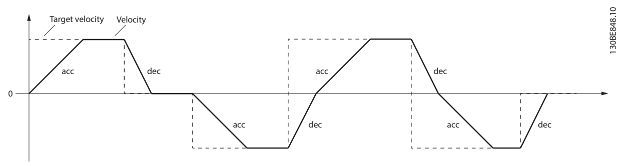


Illustration 2.132 Usage of Acceleration and Deceleration in GuideValue Reference Simulation

The device puts the calculated position and the velocity into the reference objects (see *chapter 7.9.1 Parameter: Position Guide Value Reference (0x2062)* and *chapter 7.9.2 Parameter: Velocity Guide Value Reference (0x2065)*) every cycle while increasing or decreasing the simulation speed with the desired ramp acceleration until the demanded speed has been reached.

The guide value reference simulation can be parameterized with the objects given in *chapter 7.9.6 Guide Value Reference Simulation*.

2.6 Peripherals

2.6.1 Inputs

The servo drive supports 2 inputs. The functionality of the inputs can be configured for various purposes. Each input can be defined as being:

- Analog input (for example, usable in *CAM mode* as analog sensor for alignment).
- Digital input (for example, usable in *CAM mode* as trigger).
- Left/right limit switch (for example, usable in *Homing mode*).
- Homing switch (for example, usable in *Homing mode*).
- Touch probe input.

Also the logical polarity of the input can be configured using object 0x200F (see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*). Use objects 0x60FD, 0x200D, or 0x2006 to read the values of the inputs (see *chapter 7.21.1 Parameter 16-60: Digital Inputs (0x60FD)*, *chapter 7.21.2 Parameters 16-62 and 16-64: Analog Inputs (0x200D)*, and *chapter 7.22.12 Parameter 50-08: Motion and Input Status (0x2006)*).

2.6.2 Output

The servo drive supports 1 output. This output can be influenced by various functionalities of the servo drive. Use object 0x2FFF (see *chapter 7.21.5 Parameter 52-05: Digital Output Configuration (0x2FFF)*) to configure the functionality that controls the output. Use object 0x2006 to read the value of the output (see *chapter 7.22.12 Parameter 50-08: Motion and Input Status (0x2006)*).

2.6.3 External Encoder

The advanced servo drive supports an interface to connect an external encoder. The parameters for configuration are described in *chapter 7.21.6 External Encoder Objects*. The external encoder can be used as source for the guide value (see *chapter 7.9.3 Parameter: Guide Value Reference Option Code (0x2063)*). Encoders of type BiSS-B and SSI are supported.

2.7 Monitoring

2.7.1 Errors and Warnings

If an error occurs the servo drive signals it. Depending on the reason of the error or warning, the servo drive changes its state. Some events provide warning messages before disabling the servo drive through an error. An application has the possibility then to react on the warning to avoid a shutdown of the machine, so that the downtime can be reduced. The required warning messages are listed in *chapter 9.2.2 Error Codes*. When an error occurs, its code is recorded in non-volatile memory, along with the actual guide value, IGBT temperature, winding temperature, and operating time. There are maximum 128 entries available and older entries are overwritten. Use the ISD Toolbox to read the error history (see *chapter 5.7.5 Get Error History (Servo Drive and SAB)*).

The last error code and the last warning code are given in objects 0x603F (see *chapter 7.22.9 Parameter 15-30: Error Code (0x603F)*) and/or 0x5FFE (see *chapter 7.22.10 Parameter 16-92: Warning Code (0x5FFE)*).

2.7.2 Trace

The servo drive has a built-in real-time signal tracer component which can record up to 8 internal signals into internal memory for later upload over the fieldbus. The trace process is controlled via parameters over the fieldbus, using a PLC library function block (see *chapter 6.5.4.24 DD_Trace_ISD51x*), or using the *Scope* subtool of the ISD Toolbox (see *chapter 5.7.3 Scope (Single and Multi-device for Servo Drive and SAB)*) for graphical representation. The available trace signals are listed in *chapter 9.2.3 Trace Signals*).

There are 3 different task levels for sampling:

- Real-time task: 100 µs or 125 µs
- Fast task: 200 µs or 250 µs
- Slow task: 400 µs or 500 µs

The sampling is always done synchronous to the fieldbus cycle.

The servo drive supports recording of up to 64000 samples in total (sum of all recorded signals). The complete number of configured samples must be recorded before it is possible to read the data. All samples are represented as floating point values. Use subsampling to get longer traces, so only every nth sample is recorded by the servo drive.

Tracing can be started instantly or a trigger can be configured. The trigger can be every trace signal, together with a trigger level, the slope, and the length of pre-trigger history. The trigger signal itself can be recorded but it is not required. The servo drive provides a status readout for the trace process.

Workflow to parameterize a trace

1. Write the IDs of the signals to trace into object 0x5001, sub-indexes 1 to N. Use the channels in ascending order without gaps.
2. Use the signal tracer control object (0x5000) to configure the trace settings:
    - Number of used channels
    - Task level
    - Trigger slop and mode
    - Number of samples to record
    - Subsampling
    - Amount of pre-trigger history
    - ID of the Trigger signal
    - Trigger level
3. Start the trace by writing to object 0x5000, sub-index 2.
4. Poll object 0x5000, sub-index 1 for the appearance of the data ready flag.
5. Upload the trace data from object 0x5002.
6. Separate the samples into the different channels.

2.7.3 Following Error Detection

A following error is signaled in all position controlled modes of operation (see *chapter 7.5.1 Parameter 52-00: Modes of Operation (0x6060)*). A position actual value (see *chapter 7.7.5 Parameter 50-03: Position Actual Value (0x6064)*) outside the allowed range of the following error window (see *chapter 7.22.1.1 Parameter: Following Error Window (0x6065)*) around a position demand value (see *chapter 7.7.1 Parameter: Position Demand Value (0x6062)*) for longer than the following error timeout (see *chapter 7.22.1.2 Parameter: Following Error Time Out (0x6066)*) results in setting bit 13: *Following error* in the *Statusword* to *1*. This window for the accepted following error tolerance is defined symmetrically around the reference position (see *Illustration 2.134*).

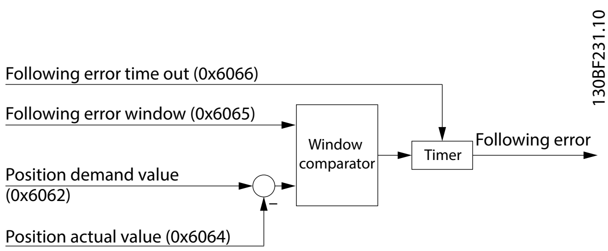


Illustration 2.133 Following Error - Functional Description

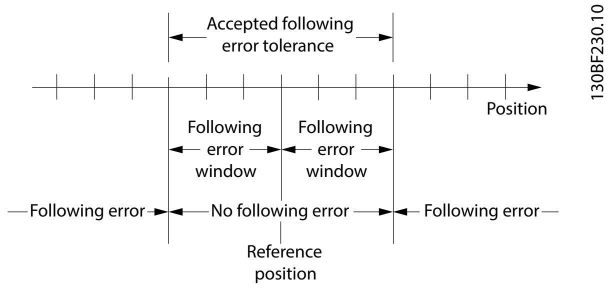


Illustration 2.134 Following Error Window

The behavior of the servo drive when a following error occurs, can be influenced by using the *Following error option code* (see *chapter 7.20.3 Parameter 50-43: Following Error Option Code (0x2055)*).

2.7.4 Standstill Detection

The standstill reached function offers the possibility to define a velocity range around velocity 0 to be regarded as standstill. If the velocity of a servo drive is within this area for a specified time (velocity window time), the servo drive is regarded to be in standstill.

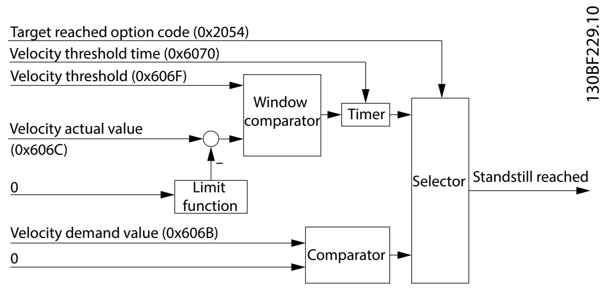


Illustration 2.135 Standstill Reached - Functional Description

Illustration 2.136 shows the definitions for the sub-function *Standstill reached* (see *chapter 7.22.2 Standstill Detection Objects*). A window is defined for the accepted velocity range symmetrically around 0 velocity. If a servo drive is running within the accepted standstill range over the *Velocity threshold time* (see *chapter 7.22.2.2 Parameter: Velocity Threshold Time (0x6070)*), the servo drive is regarded to be in standstill.

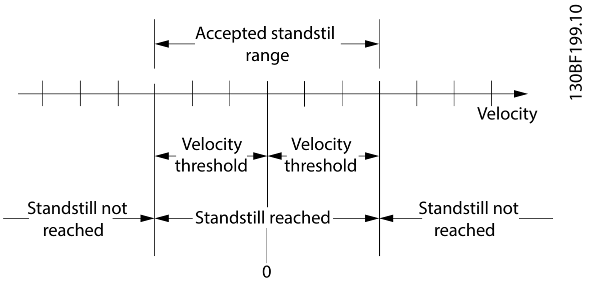


Illustration 2.136 Standstill Reached Window

2.7.5 Constant Velocity Detection

The constant velocity detection function offers the possibility to define a symmetrical range of accepted velocity changes (see *chapter 7.22.3.1 Parameter 51-70: Constant Velocity Window (0x2030)*), relative to the last velocity. If the velocity of a servo drive is within this area for a specified time, the constant velocity window time (see *chapter 7.22.3.2 Parameter 51-71: Constant Velocity Window Time (0x2031)*), the servo drive is regarded to be running at constant velocity.

The working principle is the same as for standstill detection (see *chapter 2.7.4 Standstill Detection*).

2.7.6 STO and Brake Status

The voltage is checked against a defined threshold (by the hardware) and the state is made available to the application for utilization with the DS402 state machine. The safety functionality itself is not part of the software.

The STO (Safe Torque Off) voltage state influences the DS402 state machine. The servo drive cannot be operated if the STO voltage is not active.

If the servo drive receives the command to enter DS402 state *Operation enabled*, it checks if the STO voltage is present or not. If it is not present, the servo drive enters state *Fault* and signals the occurrence of an error as described in *chapter 2.7.1 Errors and Warnings*.

If the servo drive is already in DS402 state *Operation enabled*, it continuously monitors whether the STO voltage is present or not. If it is not present, the servo drive enters state *Fault* and signals the occurrence of an error as described in *chapter 2.7.1 Errors and Warnings*.

The error code used for these 2 situations is the same and is detailed in *chapter 9.2.1 Troubleshooting*. STO information is available in objects 0x6041 (see *chapter 7.3.1 Parameter 16-03 Statusword (0x6041)*) and 0x2007 (see *chapter 7.22.8 Parameter 50-09: STO Voltage and Brake Status (0x2007)*).

3 Servo Access Box (SAB) Operation

3.1 Overview

The Servo Access Box (SAB) is the central component in the ISD 510 servo system, together with ISD 510 servo drives. It is the connection point for several servo drives. Up to 64 servo drives can be connected to 1 SAB (maximum of 2 lines, each with 32 servo drives), depending on the total power load. *Illustration 3.1* shows the typical system set-up:

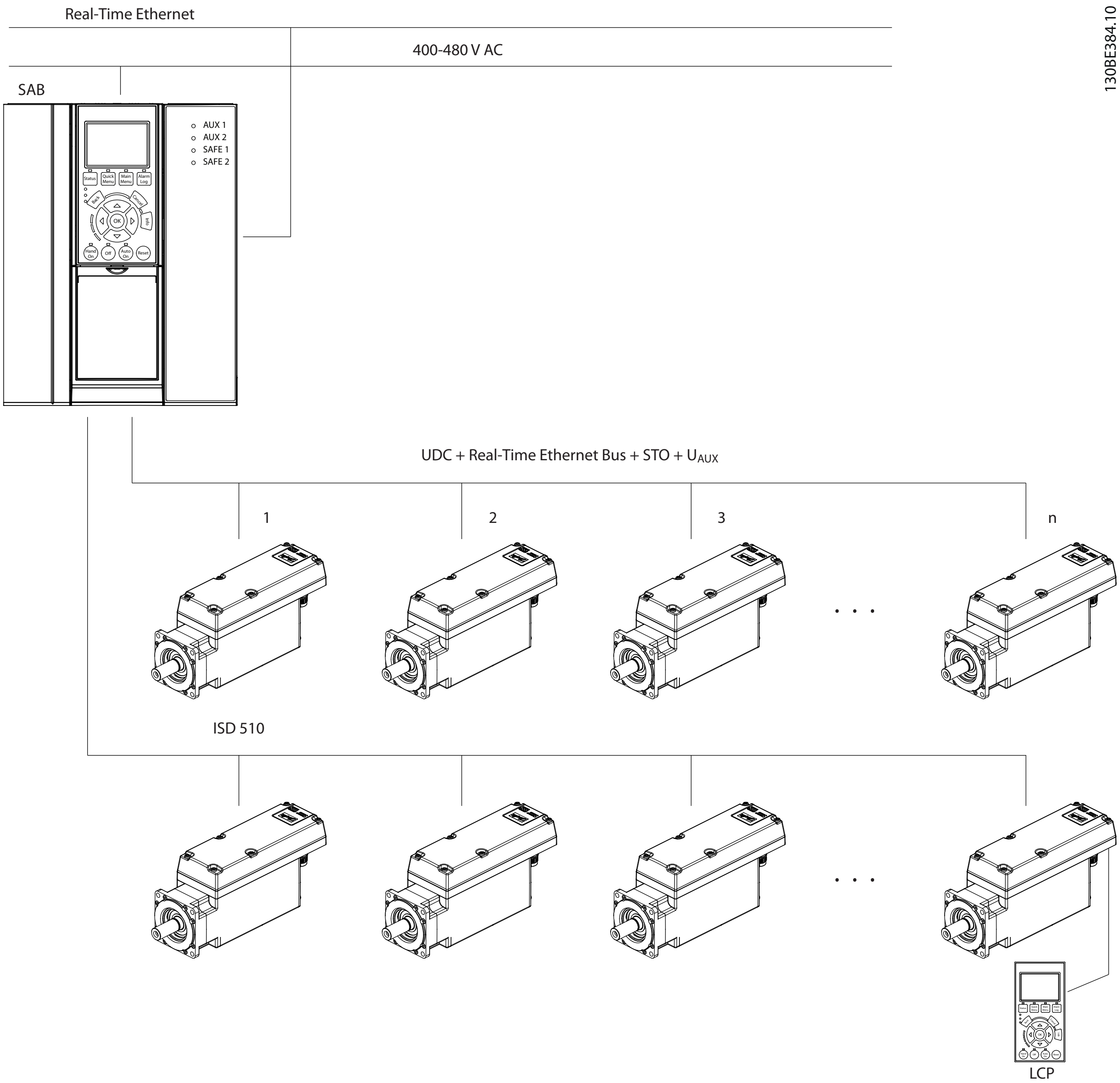


Illustration 3.1 ISD 510 Servo System

The functionalities of the SAB include:

- Distributes and switches the U_{AUX} voltage (24–48 V) that powers the control logic of the servo drives.
- Rectifies the 400 V AC 3-phase mains input and switches the resulting DC bus voltage.
- Distributes the fieldbus to the hybrid cable outputs and routes it to a 2nd RJ45-connector for simple cabinet cabling.
- Connects the STO (Safe Torque Off) voltages to the hybrid cables.
- Provides a master guide value for the whole system.
- When required, assigns the Ethernet POWERLINK® IDs to the connected servo drives.
- Controls the 2 relays.
- Dissipates the recuperation energy using an external brake resistor.

3.2 Control

To change the state of the SAB, write to the *Controlword* (see *chapter 8.1 Object 0x4040: Controlword*). This can be done in 2 ways:

- Using the PLC via the fieldbus.
- Via a connected LCP in local control mode.

The actual state can be read back from the *Statusword* (see *chapter 8.2 Object 0x4041: Statusword*). After power-up, the U_{AUX} output is activated by default and communication with the connected servo drives is possible.

UDC cannot be activated unless U_{AUX} is enabled. Deactivating U_{AUX} always also deactivates UDC.
Illustration 3.2 shows the possible transitions.

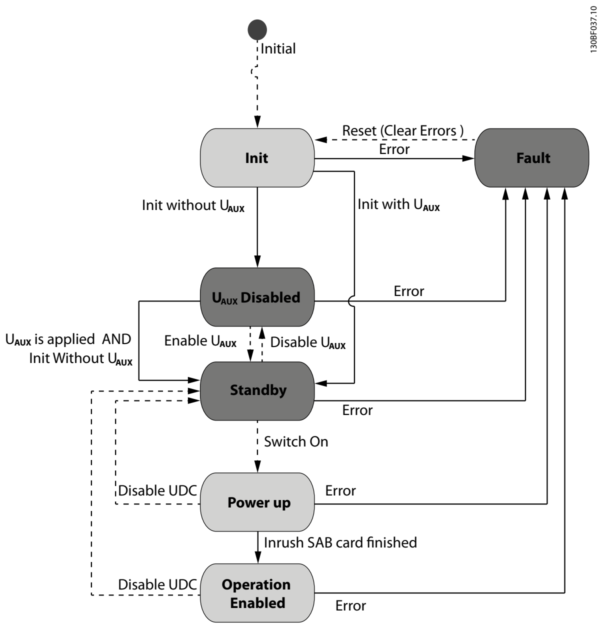


Illustration 3.2 State Diagram with Possible Transitions

Legend:

Transitions with dashed lines: Commands (Reset, Errors, U_{AUX} control, UDC control).
Transitions with solid lines: Automatic transitions with specified conditions.
States shaded dark gray: The control can be changed between *remote* and *local* via the LCP.

The defined states and the possible transitions, along with the executed actions, are defined in *Table 3.1*. The actual encoding is shown in *chapter 8.2 Object 0x4041: Statusword*.

Current state	Transition command	Required condition	Executed action	Following state
Init	(Auto)	Initialization finished	–	Standby
	Error	–	–	Fault
U_{AUX} disabled	U_{AUX} enable	U_{AUX} applied	Enable U_{AUX}	Standby
	Error	–	–	Fault
Standby	U_{AUX} disable	–	Disable U_{AUX}	Standby
	UDC enable	U_{AUX} 1 & 2 ready	–	Power-up
	Error	–	–	Fault
Power-up	(Auto)	Inrush finished	–	Operation enabled
	U_{AUX} disable	–	Disable U_{AUX} Disable UDC	U_{AUX} disabled
	UDC disable	–	Disable UDC	Standby
	Error	–	Disable UDC	Fault
Operation enabled	U_{AUX} disable	–	Disable U_{AUX} Disable UDC	U_{AUX} disabled
	UDC disable	–	Disable UDC	Standby
	Error	–	Disable UDC	Fault

Current state	Transition command	Required condition	Executed action	Following state
Fault	Reset errors	No further error conditions	–	Init

Table 3.1 States, Transitions, and Actions

NOTICE

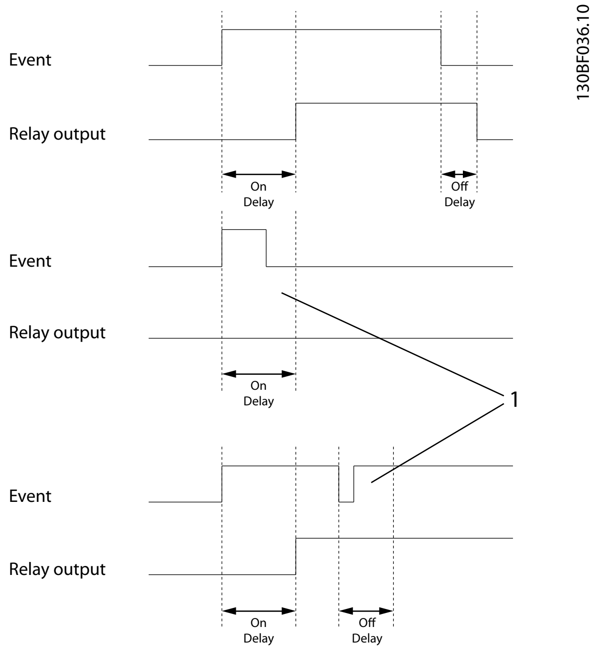
If the U_{AUX} input is not supplied on start-up, the state machine only transitions to state *U_{AUX} disabled*, even if the *Enable U_{AUX}* control bit is set to *0* (active). If, at a later stage, the U_{AUX} input is provided, then the state machine advances to state *Standby*.

3.2.1 Relay Outputs

The SAB has 2 relays:

- Relay 1: 0x200D (see *chapter 8.8 Object 0x200D: Relay 1 Control*).
- Relay 2: 0x200E (see *chapter 8.9 Object 0x200E: Relay 2 Control*).

Select the functionality of the relay via the corresponding parameter (see *Table 3.2*). There is 1 parameter for each relay for setting the *On delay time* and 1 for the *Off delay time*. Both delay times are independently adjustable from 0 s (default) to 10 minutes and the adjustment is possible in 1-second steps. The default setting for the relays is *No operation*. The function of the on and off delay time is described in *Illustration 3.3*.



1	If the condition changes before the on or off delay times expire, the relay is not affected.

Illustration 3.3 Relay Control Functions

Number	Description
0 (default)	No operation
1	SAB ready
2	SAB ready/remote control
3	SAB ready/local control
4	Enable/no warning
5	Alarm
6	Warning
7	Out of current range, U_{AUX}
8	Below current low, U_{AUX}
9	Above current high, U_{AUX}
10	Above current high, UDC
11	Encoder fault
12	Encoder simulation fault
13	Thermal warning
14	Thermal fault
15	Bus OK
16	Brake resistor ready, no warning
17	Brake resistor warning
18	Brake resistor ready, no fault
19	Brake resistor fault
20	Brake energy too high/fault (IGBT)
21	Controlword
22	Remote control
23	Local control
24	Standby/no alarm
25	Standby/no warning

Number	Description
26	Outputs UDC active

Table 3.2 Relay Control Functions

3.3 Monitoring

The SAB monitors the voltages and currents in the input and output lines, for example the auxiliary line and the DC-link, so that a phase loss or other errors can be detected. These values provide detailed information on the current load. The information is available via the fieldbus, the LCP, and the ISD Toolbox.

The SAB monitors the signals detailed in *Table 3.3* and provides the measured values via the fieldbus and the LCP.

Signal name	Description
DC-link voltage	DC-link voltage
Warning code	Warning code
Error code	Error code
Temperature control card	Temperature control card
Temperature power card	Temperature power card
Temperature SAB card	Temperature SAB card
External encoder position	External encoder position
External encoder speed	External encoder speed
UDC 1 current	Current flow on DC-link line 1
UDC 2 current	Current flow on DC-link line 2
AUX line 1 current	Current on AUX Line 1
AUX line 2 current	Current on AUX Line 2
AUX line voltage	AUX line voltage
Brake chopper gate	Brake chopper gate
Brake chopper feedback	Brake chopper feedback
UDC Over-Inrush	UDC current over-inrush
UDC Bypass-Inrush	UDC Current bypass-inrush
UDC back current	Current (link voltage) back from the servo drives
Inrush relay power card	Inrush relay power card
Inrush relay SAB card	Inrush relay SAB card
DC leakage current	DC leakage current
Brake resistor power monitoring	Brake resistor power monitoring
DC link total current	DC link total current
DC link total current raw	DC link total current (unfiltered)
UDC 1 flow (filtered)	UDC 1 current readout
UDC 2 flow (filtered)	UDC 2 current readout
Controlword	Controlword

Table 3.3 Signals Monitored by the SAB

Use the ISD Toolbox *Scope* subtool to perform a trace on these signals. If the overload situation is critical, the SAB protects itself and the servo drives by shutting down to prevent any damage. In such cases, warnings may not be visible if the shutdown occurs quickly.

3.3.1 AUX Output

The SAB protects the servo drives connected to the AUX lines against overcurrent, overvoltage, and undervoltage. If an overload occurs, the outputs are disabled and an alarm is issued. If the UDC outputs are enabled, they are disabled first.

For overvoltage and undervoltage conditions, an associated warning with a different threshold is triggered before the error. Set a user current limit in steps of 0.1 amperes for each of the auxiliary lines in object 0x2003, sub-indexes 4 and 5 (see *chapter 8.5 Object 0x2003: U_{AUX} Related Values*). If this limit is reached, the SAB disables the auxiliary lines. If the current reaches 90% of the limit set, a warning is issued. The SAB has additional hardware detection/protection in case a hard short circuit occurs on the auxiliary lines.

3.3.2 DC Output

The SAB protects itself and the servo drives connected to the UDC lines against overcurrent, overvoltage, and undervoltage. If an overload occurs, the outputs are disabled and an alarm is issued.

For overvoltage and undervoltage conditions, an associated warning with a different threshold is triggered before the error. The SAB provides short-term overload capabilities; it is possible to run at 160% load for 60 s. However, afterwards, the SAB must run at a reduced output load to compensate the overload. The SAB has an internal monitoring logic for the overload condition and its duration.

3.3.3 Brake Control and Monitoring

Connect a brake resistor to the SAB to limit the UDC voltage when the connected servo drives are in recuperation mode and acting as a generator. If configured, the SAB limits the UDC voltage by connecting the resistor via an internal IGBT switch. To monitor the functionality of the brake, the brake circuitry, and the brake power dissipation, make the following settings:

- Enter the correct resistance in object 0x2031 (see *chapter 8.11 Object 0x2031: Brake Resistor*).
- Enter the correct power limit in object 0x2032 (see *chapter 8.12 Object 0x2032: Brake Resistor Power Limit*).

If any configured limit is overstepped, the SAB issues a warning or alarm. If configured to report an alarm, the SAB transitions to fault state. Reinitialize the SAB using the error recover command in the SAB state machine.

NOTICE

If a short circuit of the IGBT switch occurs, the brake resistor is powered continuously and the external mains input must be cut by external means.

3.3.4 Input Voltages

The SAB monitors the phase balance of the input voltages. If the imbalance becomes too high, a phase loss warning is issued. If the situation remains, an error is issued and the SAB enters the *Fault* state.

3.3.5 Temperatures

The SAB can operate within a temperature range of 5–50 °C. If the temperature rises above the upper limit, the lifetime of the electronics decreases at a forced pace and there is a risk of malfunction. Therefore, the SAB has 3 temperature sensors that are placed on the power card, the control card, and the SAB card. The measured temperatures are visual in the LCP and can be read via the following fieldbus objects:

- 0x2000, sub-index 1 = Power card
- 0x2000, sub-index 2 = Control card
- 0x2000, sub-index 3 = SAB card

If 1 of the temperatures becomes too high, the SAB issues a warning. If the temperature continues to rise and passes a 2nd limit, the SAB protects itself by shutting down and issuing an error.

3.3.6 Cooling Fans

The SAB has 2 cooling fans to control the internal temperature; 1 is on the power card and the other 1 on the SAB card. The SAB controls the speed of the cooling fans to maintain a sufficiently low temperature. The speed of the power card fan can be read back via object 0x2009 (see *chapter 8.7 Object 0x2009: Fan Speed Power Card*).

3.4 External Encoder and Guide Value

It is possible to connect an external BiSS or SSI Encoder to provide a system global guide value. Alternatively, it is possible to generate a synthetic guide value. Operation is identical to the servo drive. See objects 0x2062 (*chapter 7.9 Guide Value Reference Objects*), 0x2063 (*chapter 8.18 Object 0x2063: Guide Value Reference Option Code*), and 0x3000 (*chapter 7.21.6.1 Parameters 51-30 and 51-34 to 51-40: External Encoder Configuration (0x3000)*).

3.5 Signal Tracing

For information on signal tracing, see *chapter 2.7.2 Trace*.

3.6 Multiple Device ID Assignment

For information on multiple device ID assignment, see *chapter 6.1.2.2 Multiple Device ID Assignment*.

3.7 Software Version

For information about the software, see *chapter 7.22.4 Parameters 15-40, 15-41, and 15-43: Version log (0x4000)*.

3.8 Firmware Update

The SAB firmware can be updated remotely via the fieldbus. The procedure for the SAB is identical to that of the servo drive (see *chapter 2.2 Firmware Update*).

4 Local Control Panel (LCP) Operation

4.1 Overview

The LCP is the graphical user interface on the SAB for diagnostic and operating purposes. It is included as standard with the SAB but can also be connected to the advanced servo drives using an optional cable (M8 to LCP D-SUB extension cable).

The LCP display provides the operator with a quick view of the state of the servo drive or SAB, depending on which device it is connected to. The display shows parameters and alarms/errors and can be used for commissioning and troubleshooting. It can also be used to perform simple functions, for example activating and deactivating the output lines on the SAB.

The LCP can be mounted on the front of the control cabinet and then connected to the SAB via a SUB-D cable (available as an accessory).

4.2 Local Control Panel (LCP) Layout

The local control panel is divided into 4 functional groups (see *Illustration 4.1*).

A. Display area.

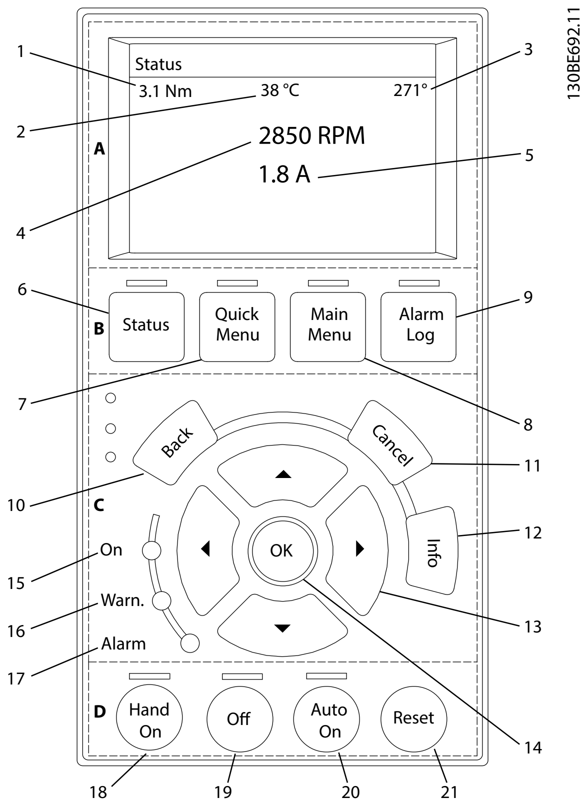
B. Display menu keys.

C. Navigation keys and indicator lights (LEDs).

D. Operation keys and reset.

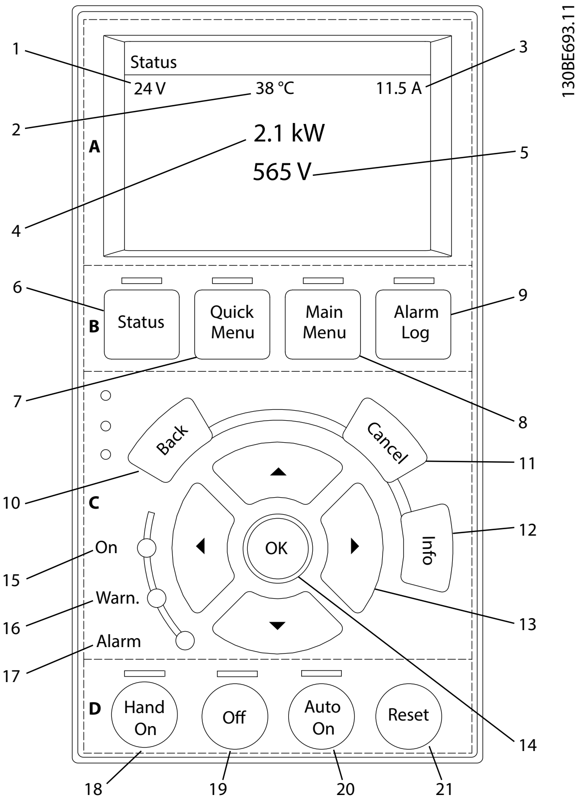
A. Display area
The values in the display area differ depending on whether the LCP is connected to an ISD 510 servo drive or the SAB, as shown in *Illustration 4.1* and *Illustration 4.2*.

The display area is activated when the servo drive or SAB it is connected to receives power from the mains supply, a DC bus terminal, or U_{AUX}.



Callout number	Description
1	Actual torque
2	Temperature module
3	Position
4	Speed
5	Current

Illustration 4.1 Display Area when Connected to an ISD 510 Servo Drive



Callout number	Description
1	AUX line voltage
2	Temperature power card
3	Actual UDC (current)
4	ISD power consumption
5	Actual UDC (voltage)

Illustration 4.2 Display Area when Connected to the SAB

B. Display menu keys

Menu keys are used to access menus for parameter set-up, toggling through status display modes during normal operation, and viewing fault log data.

Callout number	Key	Function
6	Status	Shows operational information.
7	Quick Menu	Allows access to parameters.
8	Main Menu	Allows access to parameters.
9	Alarm Log	Shows the last 10 alarms.

Table 4.1 Display Menu Keys

C. Navigation keys and indicator lights (LEDs)

Navigation keys are used to move the display cursor and provide operation control in local operation. There are also 3 status LEDs in this area.

Callout number	Key	Function
10	Back	Reverts to the previous step or list in the menu structure.
11	Cancel	Cancels the last change or command (unless the display mode is changed).
12	Info	Gives a definition of the current function.
13	Navigation keys	The 4 keys enable navigation between menu items.
14	OK	Accesses parameter groups or enables a selection.

Table 4.2 Navigation Keys

Callout number	LED	Color	Function
15	On	Green	The *On* LED activates when the servo drive or SAB it is connected to receives power from the mains, auxiliary supply, or a DC bus terminal.
16	Warn.	Yellow	When a warning is issued, the yellow *Warn.* LED activates and text appears in the display area identifying the problem.
17	Alarm	Red	A fault condition causes the red *Alarm* LED to flash and an alarm text is shown.

Table 4.3 Indicator Lights (LEDs)

D. Operation keys and reset

The operation keys are at the bottom of the LCP.

Callout number	Key	Function
18	Hand On	Enables the connected servo drive or SAB to be controlled via the LCP. See *chapter 4.3.5 Hand On Mode* for further information. Switching between *Hand On* mode and *Auto On* mode is only possible in certain states.
19	Off	Puts the SAB into state *Standby* and the servo drive to state *Switch on Disabled*. This only works in *Hand On* mode. *Off* mode enables transition from *Hand On* mode to *Auto On* mode.

Callout number	Key	Function
20	Auto On	Puts the system in remote operational mode. In *Auto On* mode, the device is controlled by fieldbus (PLC). Switching between *Auto On* mode and *Hand On* mode is only possible when the servo drive is in state *Switch on disabled* and/or the SAB is in state *Standby*.
21	Reset	Resets the servo drive or SAB after a fault has been cleared. The reset is only possible when in *Hand On* mode.

Table 4.4 Operation Keys and Reset

4.3 Graphical User Interface

4.3.1 Supported Languages

The user interface language is English, regardless of whether the LCP is connected to the servo drive or the SAB.

4.3.2 LCP Display

The display is backlit and has a total of 6 alphanumeric lines. The display lines show the direction of rotation (arrow), the selected set-up, and the programming set-up. The display is divided into 3 sections (see *Illustration 4.3*).

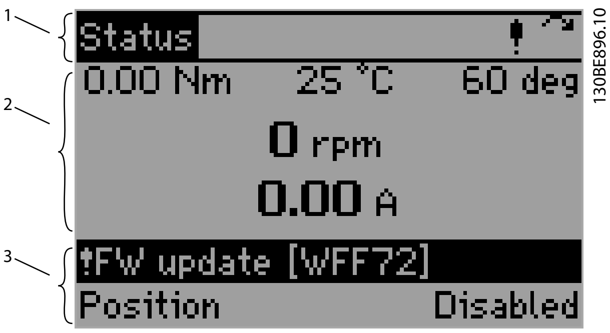


Illustration 4.3 LCP Display Overview

1	**Top section** Shows up to 2 measurements in normal operating status.
2	**Middle section** The top line shows up to 5 measurements with related unit, regardless of status (except in the case of alarm/warning).
3	**Bottom section** Shows the state of the device when the *Status* view is active: • If an alarm or warning is active, its number and short description are shown. • For the servo drives, the mode of operation is shown on the left and the servo drive state on the right. • For SAB, the SAB state is shown on the right.

The state names and the mode of operation names have been shortened for the LCP display, as defined by *Table 4.5* and *Table 4.6*.

Full name	Short name
Switch on disabled	Disabled
Ready to switch on	Ready
Switched on	Switched on
Operation enabled	Enabled
Fault	Fault
Quick stop active	Quick stop

Table 4.5 ISD 510 Servo Drive State Names

Full name	Short name
Profile position mode	Position
Profile velocity mode	Velocity
Homing mode	Homing
Inertia measurement mode	Inertia
Torque mode	Torque
Gear mode	Gear
CAM mode	CAM

Table 4.6 Mode of Operation Names

NOTICE

To adjust the display contrast, press the [Status] and the [▲] or [▼] key.

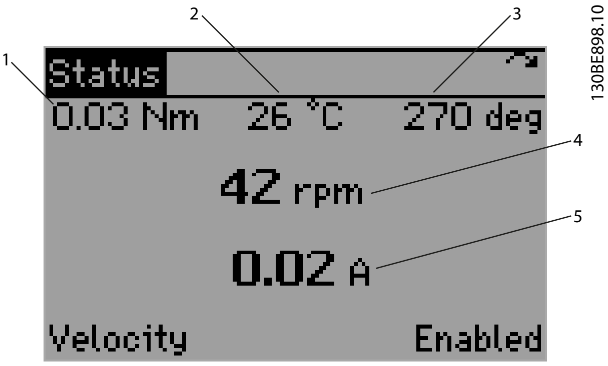
4.3.3 Status Menu (Auto On Mode)

The *Auto On* mode is the default mode after power up. It is also activated using the [Auto On] key (see *chapter 4.4.11 Auto On Key*). In *Auto On* mode, it is only possible to read parameters – the parameter values cannot be changed.
The *Status* menu in *Auto On* mode shows readout parameters.
Press the [Status] key while the *Status* menu is shown to toggle the readout mode between *Single-line readout* and *Double-line readout*.

The [▲] and [▼] keys can be used to toggle between the values shown in *Readout 2* on *Double-line readout* or *Single-line readout*.

Double-line readout

This readout state is a default mode after start-up or initialization. Use the [Info] key to obtain information about the measurement links to the shown operating variables (1.1, 1.2, 1.3, 2, and 3). See the operating variables shown in *Illustration 4.4*.

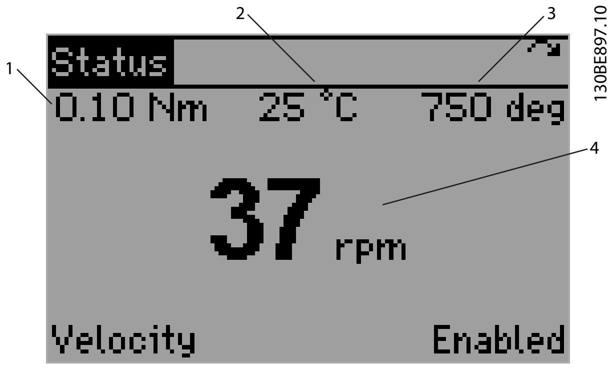


1	Readout 1.1
2	Readout 1.2
3	Readout 1.3
4	Readout 2
5	Readout 3

Illustration 4.4 Double-line Readout

Single-line readout

Use the [Info] key to obtain information about the measurement links to the shown operating variables (1.1, 1.2, 1.3, and 2). See the operating variables shown in *Illustration 4.5*. The dynamic data (readout parameters) on the *Status* screen is updated 3 times per second.



1	Readout 1.1
2	Readout 1.2
3	Readout 1.3
4	Readout 2

Illustration 4.5 Single-line Readout

4.3.3.1 Default Readouts for ISD 510 Servo Drive

The following parameters are the default readout configuration:

Operating variable	Name	Parameter number
1.1	Torque	[16-16]
1.2	Temperature module	[16-34]
1.3	Drive position	[16-20]
2	Speed	[16-17]
3	Current actual value	[16-14]

Table 4.7 Default Readouts for ISD 510 Servo Drive

The readout configuration can be changed and is retained after a power cycle.

4.3.3.2 Default Readouts for SAB

The following parameters are the default readout configuration:

Operating variable	Name	Parameter number
1.1	AUX line voltage	[50-61]
1.2	Temperature power card	[16-31]
1.3	Actual UDC (current)	[50-73]
2	ISD power consumption	[16-10]
3	Actual UDC (voltage)	[16-30]

Table 4.8 Default Readouts for SAB

The readout configuration can be changed and is retained after a power cycle.

4.3.3.3 Alarms and Warnings

Alarms and warnings are indicated on the LCP by the alarm overlay of the *Status* menu. Whenever an alarm or warning appears on the device, the *Status* menu is shown on the screen and the bottom section shows the alarm or warning indication using color-inverted text.

The alarm or warning screen consists of:

- Alarm or warning symbol:
    - Alarm: If an alarm is active on the device, a bell symbol is shown in the upper right corner (see *Illustration 4.6*). Also, the *Alarm* LED flashes red.
    - Warning: If a warning is active on the device, an exclamation mark symbol is shown in the upper right corner (see *Illustration 4.3*). Also, the *Warning* LED flashes yellow.
- Short alarm or warning text.
- Alarm or warning number: the 4-digit hexadecimal identifier of the alarm or warning, preceded by the capital letter *A* for alarm (see *Illustration 4.6*) or the capital letter *W* for warning (see *Illustration 4.3*).

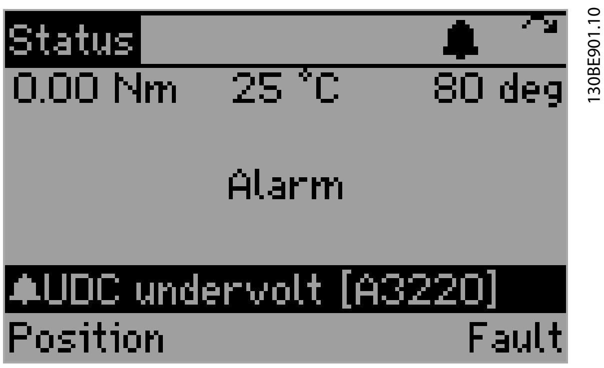


Illustration 4.6 Alarm Display

NOTICE

If multiple alarms occur, the alarm that occurred last is shown. See *chapter 4.3.6 Alarm Log* for further information on the history of alarms.

4.3.4 Main Menu

The LCP *Main Menu* is the interface for browsing through all available device parameters. The LCP parameters are organized in groups (level 1) and subgroups (level 2). At the *Main Menu* root, the LCP screen shows all groups (level 1), as depicted in *Illustration 4.7*. Select a group and press the [OK] key to show its subgroup (level 2). Select the subgroup and press the [OK] key to show all parameters belonging to the subgroup.

When the LCP *Main Menu* is shown, the 1st line is the menu head line that shows readout 1.1 and 1.2 (see *Illustration 4.5* and *Illustration 4.4*), and the actual motor direction. It also shows an alarm or warning symbol if an alarm or warning is present.

The motor direction is illustrated by an arrow in the upper-right corner:

- A left arrow indicates that the motor is turning in a negative direction.
- A right arrow indicates that the motor is turning in a positive direction.

For more information on the servo drive directions, see *parameter 52-04 Drive Mirror Mode* in *chapter 7.7.8 Parameters 51-02, 52-04, and 52-49: Application Settings (0x2016)*.

The 2nd line is the menu line and shows the color-inverted menu name on the left – *Main Menu* for the root menu (see *Illustration 4.7*), or the group or subgroup name. The rest of the screen is made up of the item selector (a control that shows the group numbers and names), and a scroll bar. Use the [▲] and [▼] keys on the LCP to navigate to the desired group, subgroup, or parameter.

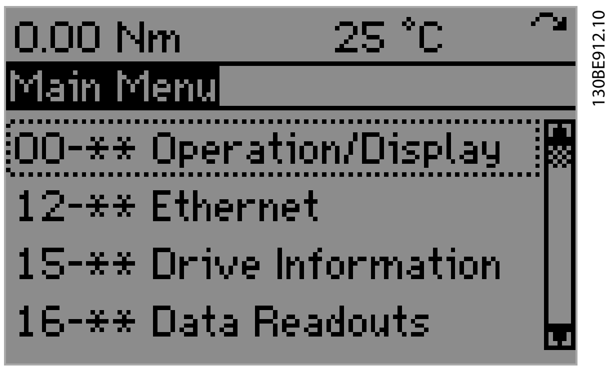


Illustration 4.7 Main Menu Level 1

Press the [OK] key when a parameter group or subgroup is selected to enter the group or subgroup.
When navigating to *parameter group 00-** Operation/Display*, press the [OK] key to access the subgroups belonging to this group (see *Illustration 4.7* and *Illustration 4.8*).

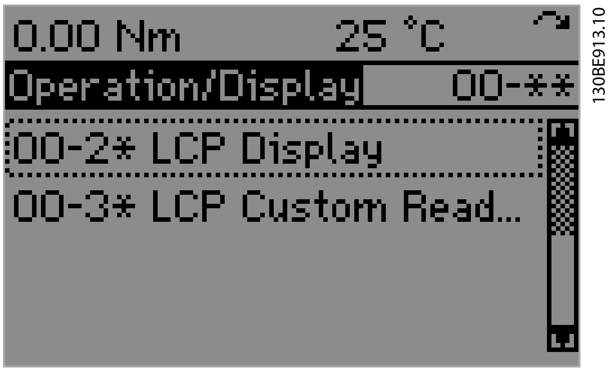


Illustration 4.8 Main Menu Level 2

Press the [OK] key while in the screen shown in *Illustration 4.8* to show all parameters in the subgroup, as depicted in *Illustration 4.9*. In this case, the 1st line is the menu head line, the 2nd line is the menu line, and the rest of the screen is the parameter selector that shows the parameter. The name of the selected parameter is shown in the first row of the parameter selector (*00-34 Unit for velocity readout* in *Illustration 4.9*). The value of the parameter is shown in the lower row (*[0] rpm* in *Illustration 4.9*).

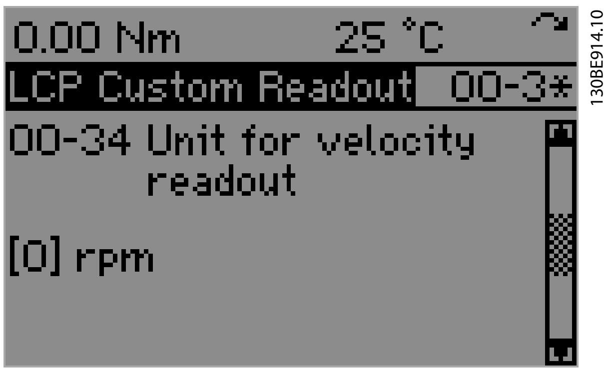


Illustration 4.9 Main Menu Level 3 (Enter Parameter)

Use the [▲] and [▼] keys on the LCP to navigate within the current group or subgroup.

4.3.4.1 Displaying and Editing Values

Parameter values can be read in all modes, however parameters can only be edited in *Hand On* mode.

Values can be edited using the [◄], [►], [▲] and [▼] keys digit by digit. Press the [◄] or [►] key to shift the selected digit then press the [▲] or [▼] key to increment or decrement the value.

The selected digit is color-inverted to indicate the position of the cursor.

Illustration 4.10 shows editing the value of *parameter 52-12 Profile velocity*. The set value is *600* and the 2nd digit from the right is highlighted. Press the [▲] key to increase the set value to *610*, or the [▼] key to change the set value to *590*.

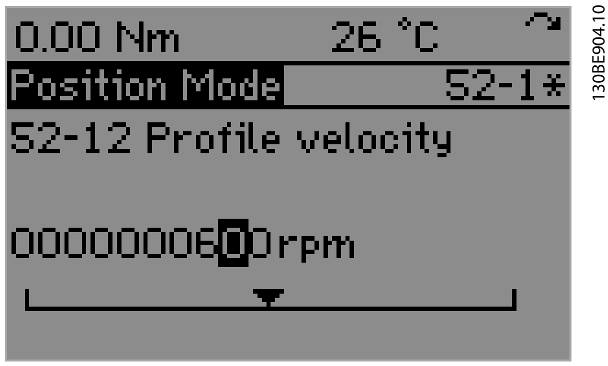


Illustration 4.10 Editing a Single Digit

Press the [OK] key to apply the value and return to parameter number mode.
Press the [Back] key to return to parameter number mode and discard any changes.

When edit mode is entered, the current edited digit starts from the last digit on the right. While editing parameter values, it is only possible to increase or decrease the value within its valid range.

Continuous value parameter

If the parameter is a continuous value, only the significant digits (without prefix zeros) of the value are shown, as shown in *Illustration 4.11*.

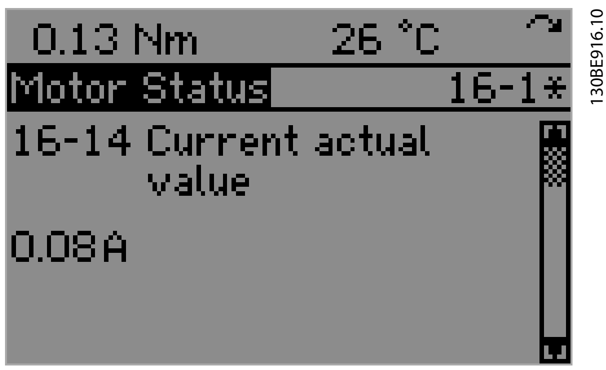
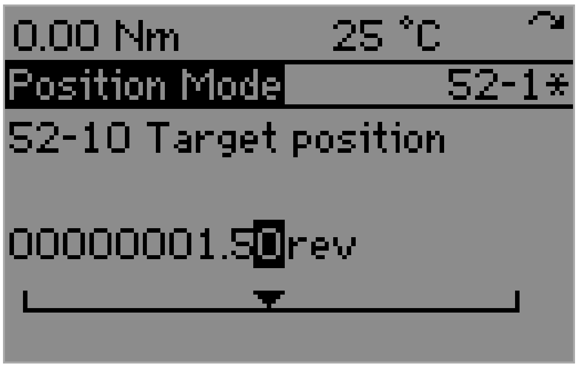


Illustration 4.11 Continuous Value Parameter Display

In *Edit* mode, all the possible digits of a parameter value are shown so the prefix zeros in front of the significant digit are required. The number of digits required depends on the minimum and maximum limits and the precision of the specific parameter. The color-inverted digit indicates the position of the cursor. A gauge control shows the current value in relation to the minimum and maximum values, as shown in *Illustration 4.12*.

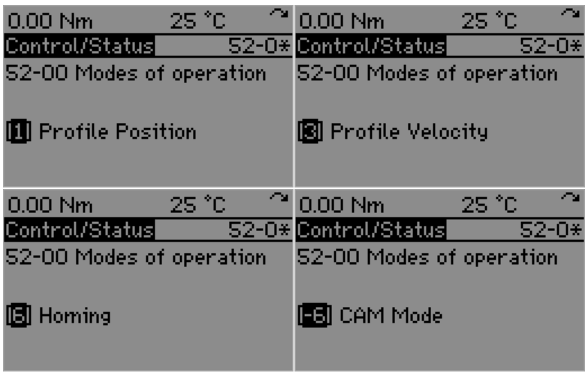


130BE917.10

Illustration 4.12 Edit Continuous Value Parameter

Enumerated parameters

Enumerated parameters show a meaningful text that is associated with discrete values. An enumerated parameter is represented by an integer variable that is limited to some allowed values. Each allowed value has a textual representation assigned to it. *Illustration 4.13* depicts the enumerated *parameter 52-00 Modes of operation* with 4 of its allowed values.



130BE899.10

Illustration 4.13 Enumerated Parameter 52-00

4.3.4.2 ISD 510 Drive Menu

Table 4.9 details the LCP menu structure for the ISD 510 servo drives. Most menu items are available for all fieldbuses. Where necessary, the fieldbus-specific availability of the menu items is indicated in *italics*.

00-**	**Operation/Display**	
	00-2*	LCP Display
	00-3*	LCP Custom Readout
12-**	**Ethernet**	
	12-0*	IP Settings
	12-5*	EtherCAT *(EtherCAT only)*
	12-6*	Ethernet POWERLINK *(Ethernet POWERLINK only)*
15-**	**Drive Information**	
	15-0*	Operating Data
	15-3*	Fault Log
	15-4*	Drive Identification
16-**	**Data Readouts**	
	16-0*	General Status
	16-1*	Motor Status
	16-3*	Drive Status
	16-6*	Inputs & Outputs
	16-9*	Diagnosis Readouts
50-**	**ISD General**	
	50-0*	General Readouts
	50-1*	General Control
	50-2*	Physical Limits
	50-3*	Position Limits
	50-4*	Option Codes
51-**	**ISD Config (ISD Configuration)**	
	51-0*	Control Loop
	51-1*	Ctrl Params 1 (Control Parameters 1)
	51-2*	Ctrl Params 2 (Control Parameters 2)
	51-3*	External Encoder
	51-5*	Touch Probe 1
	51-6*	Touch Probe 2
	51-7*	Const. Velocity Det. (Constant Velocity Detection)
52-**	**ISD Operation**	
	52-0*	Control/Status
	52-1*	Position Mode
	52-2*	Velocity Mode
	52-3*	Torque Mode
	52-4*	Homing Mode
	52-5*	Homing Methods
	52-6*	Inertia Measurement
	52-7*	Jog Mode
55-**	**Factor Group**	
	55-0*	Pos. Enc. Resolution (Position Encoder Resolution)
	55-1*	Gear Ratio
	55-2*	Feed Constant
	55-3*	Velocity Factor
	55-4*	Acceleration Factor
	55-5*	Jerk Factor

Table 4.9 LCP Menu Structure for ISD 510 Servo Drive

4.3.4.3 SAB Menu

Table 4.10 details the LCP menu structure for the SAB. Most menu items are available for all fieldbuses. Where necessary, the fieldbus-specific availability of the menu items is indicated in *italics*.

00-**	**Operation/Display**	
	00-2*	LCP Display
02-**	**Brakes**	
	02-1*	Brake Energy Func. (Brake Energy Functions)
05-**	**Digital In/Out**	
	05-4*	Relays
12-**	**Ethernet**	
	12-0*	IP Settings
	12-5*	EtherCAT (*EtherCAT* only)
	12-6*	Ethernet POWERLINK (Ethernet *POWERLINK* only)
15-**	**SAB Information**	
	15-0*	Operating Data
	15-4*	SAB Identification
16-**	**Data Readouts**	
	16-0*	General Status
	16-9*	Diagnosis Readouts
50-**	**SAB General**	
	50-0*	General Readouts
	50-1*	General Control
	50-5*	Task Cycle Times
	50-6*	Auxiliary Voltage
	50-7*	Output Line Status
	50-8*	Guide Value Ref. (Guide Value Reference)
51-**	**SAB Config (SAB Configuration)**	
	51-3*	External Encoder
	51-8*	Guide Val. Ref. Sim. (Guide Value Reference Simulation)
	51-9*	Fan Control
54-**	**ID Assignment** *(Ethernet POWERLINK only)*	
	54-0*	Automatic
	54-1*	Manual

Table 4.10 LCP Menu Structure for SAB

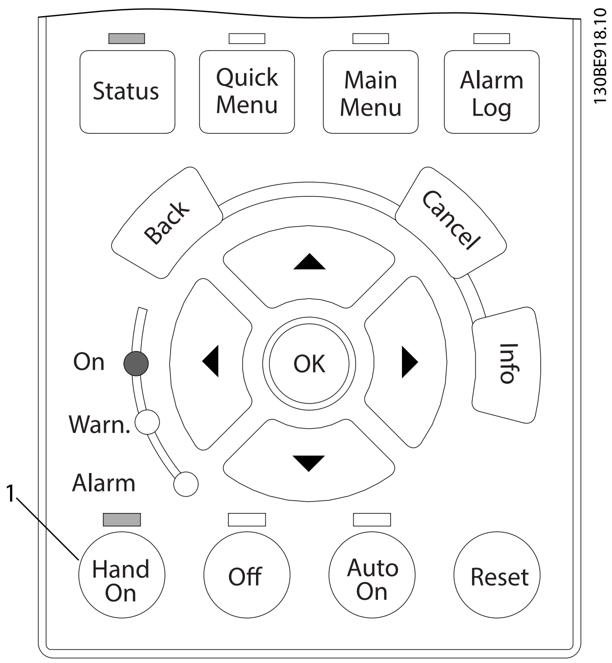
4.3.5 Hand On Mode

Hand On mode is a functionality that transfers the control of the servo drive or SAB from the master (for example, PLC) to the LCP. When a device is in *Hand On* mode, it cannot be controlled by a master via fieldbus. Any received PDO values are ignored and any SDO write requests are rejected with error code 0x8000021 (data cannot be transferred or stored to the application because of local control).

Illustration 4.14 shows an LCP connected to an ISD 510 servo drive with activated *Hand On* mode.

Hand On mode is signaled by the device to the fieldbus master (for example, PLC) by setting the remote bit (bit 9) in the *Statusword* to *0*. When *Hand On* mode is released, the bit is set back to *1*. After a power cycle, *Hand On* mode is always deactivated (see *chapter 7.3.1 Parameter 16-03 Statusword (0x6041)* and *chapter 8.2 Object 0x4041: Statusword*).

For the ISD 510 servo drive, the *Hand On* mode can only be activated or deactivated when the servo drive is not in state *Operation enabled* or *Quick stop active* – it must be in an unpowered state. For the SAB, the *Hand On* mode can only be activated or deactivated in all states except *Operation enabled*.



1	Hand On mode active

Illustration 4.14 Hand On Mode Active on an ISD 510 Servo Drive

4.3.5.1 Servo Drive

In *Hand On* mode, the servo drive can be controlled using 1 of the following modes:

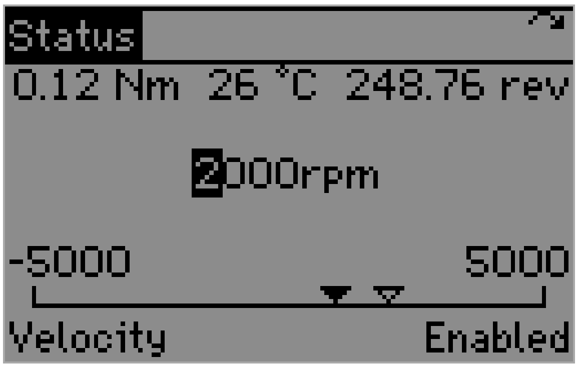
- Velocity mode
- Jog mode
- Position mode
- Inertia measurement mode

To operate the servo drive in *Velocity* mode, *Jog* mode, or *Position* mode, the servo drive must be in state *Operation enabled*. Enable the servo drive by setting *parameter 52-03 Hand On state* from *[0] disabled* to *[1] enabled*. Use either *parameter 52-03 Hand On state*, or the [Off] key to enter the state *Disabled*. When enabling the servo drive via *parameter 52-03 Hand On state*, set *52-20 Target Velocity* and *52-30 Target Torque* to *0*, otherwise the change is rejected for safety reasons.

Velocity mode

The *Hand On* mode for velocity allows moving the shaft by setting its target velocity. The functionality is provided by the *Status* screen when the LCP is in *Hand On* mode and the servo drive is in *Velocity* mode, as shown in *Illustration 4.15*.

Like the general *Status* screen, this customized screen contains readouts 1.1, 1.2, and 1.3, the mode of operation, and the state of the servo drive. Furthermore, it contains a numerical input field and a gauge control for setting and visualizing the target velocity. Edit the numerical value above the gauge to set the target velocity. The unit for the target velocity is specified by *parameter 00-34 Unit for velocity readout*.



130BE907.10

Illustration 4.15 Hand On Mode – Velocity

The minimum and maximum values of the gauge and numerical input are dynamically updated from *parameter 50-16 Max profile velocity*. Set the target velocity value and press the [OK] key to apply the velocity according to the velocity configuration. Switch to the device to state *Operation enabled* before performing velocity movement. The velocity behavior depends on the values of the following parameters. Set them accordingly before using the *Velocity* functionality.

- *Parameter 50-16 Max profile velocity*
- Ramp configuration
    - *Parameter 52-21 Profile acceleration* (default: 1000 RPM/s)
    - *Parameter 52-22 Profile deceleration* (default: 1000 RPM/s)
- *Parameter 52-23 Application torque limit*

Jog mode

The *Jog* mode turns the motor at a pre-defined velocity by using the [◂] and [▸] keys on the LCP. The functionality is based on the *Profile velocity* mode of operation. *Illustration 4.16* depicts the LCP *Status* screen in *Hand On* mode shown when *Jog* mode is active. The text "*Jog mode*" is shown in the center of the screen to indicate that the *Jog* functionality is active. Below this text, a control gauge with limits of *–1* and *1* is shown:

- –1 indicates jog in negative direction.
- 0 indicates standstill.
- 1 indicates jog in positive direction.

For more information on the servo drive directions, refer to *parameter 52-04 Drive Mirror Mode*.

130BE911.10

Illustration 4.16 Jog Mode

The *Jog* behavior depends on the values of the following parameters. Set them accordingly before using the *Jog* functionality.

- *Parameter 52-71 Jog speed* (default: 100 RPM)
- *Parameter 50-16 Max profile velocity*
- Ramp configuration
    - *Parameter 52-21 Profile acceleration* (default: 1000 RPM/s)
    - *Parameter 52-22 Profile deceleration* (default: 1000 RPM/s)
- *Parameter 52-23 Application torque limit*

Jog in positive direction is performed when the [▸] key is pressed – and stopped when it is released.
Jog in negative direction is performed when the [◂] key is pressed – and stopped when it is released.

Position mode

The *Hand On* mode functionality for positioning moves the shaft to an absolute or relative position, depending on the value of LCP *parameter 52-11 Positioning type* (default value: relative). The functionality is provided by the *Status* screen when the LCP is in *Hand On* mode and the servo drive is in *Position* mode, as shown in *Illustration 4.17*.

In addition to the information of the general *Status* screen, this customized screen contains a numerical input field and a gauge control for setting and visualizing the target position. The target position is set by editing the numerical value above the gauge. The unit for the target position is specified by *parameter 00-33 Unit for position readout*.

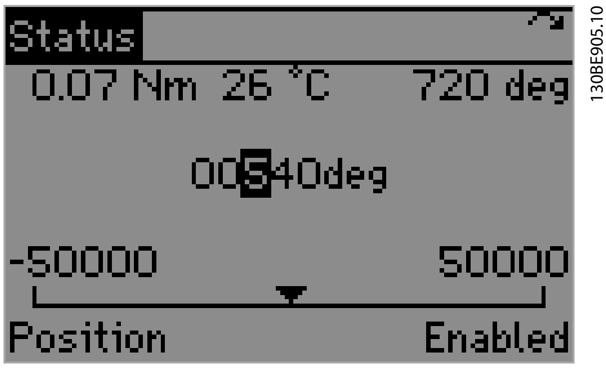


Illustration 4.17 Hand On Mode – Positioning

Set the target position and press the [OK] key. Now the shaft moves according to the positioning configuration. Before performing positioning, switch the device to state *Operation enabled*. The positioning takes place immediately. The positioning behavior depends on the values of the following parameters. Set them accordingly before using the *Positioning* functionality.

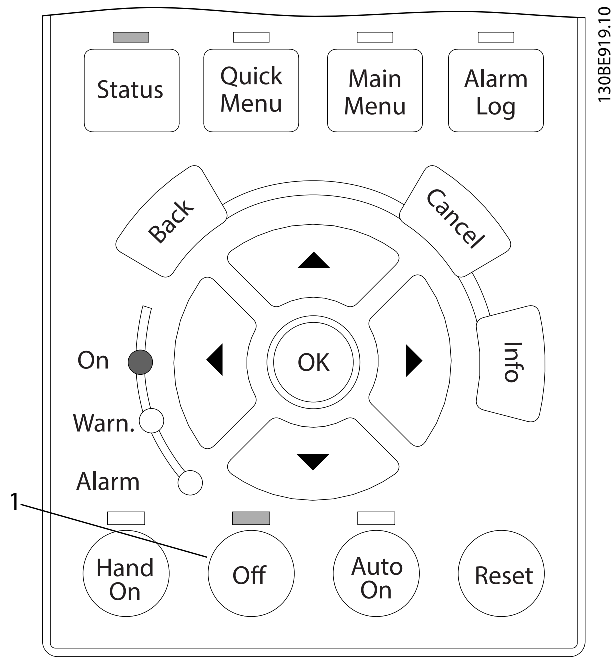
- *Parameter 52-12 Profile velocity* (default: 100 RPM)
- Ramp configuration
  - *Parameter 52-13 Profile acceleration* (default: 1000 RPM/s)
  - *Parameter 52-14 Profile deceleration* (default: 1000 RPM/s)
- *Parameter 52-11 Positioning type* (default: relative)
- *Parameter 52-15 Application torque limit*
- Limits
  - *Parameter 50-30 Min position range limit* (default: 0 deg)
  - *Parameter 50-31 Max position range limit* (default: 0 deg)

Off mode

Press the [Off] key when the servo drive is in *Hand On* mode to change the servo drive state to *Switch on Disabled*. If the motor is not in standstill, it stops according to the selected behavior set in *parameter 50-48 Shutdown option code*.

When entering *Off* mode from *Hand On* mode, the selected *Hand On* functionality (position, velocity, jog, or intertia) is retained. Therefore, the same status screen as in *Hand On* mode is shown after switching to *Off* mode.

Illustration 4.18 depicts the LCP after switching from *Hand On* mode with velocity to *Off* mode. The LED above the [Off] key indicates that *Off* mode is active – there is no other indication on the LCP display.



1	Hand On mode off

Illustration 4.18 Hand On Mode - Off

Inertia measurement

When *Hand On* mode is active, the LCP can be used to perform inertia measurement. The functionality is contained in *parameter group 52-6* Inertia Measurement*. The group contains the following parameters:

- *Parameter 52-60 Measured inertia* to read the measurement result. Positive values indicate that the measurement was carried out successfully and the inertia is shown in kg x m^2. Negative values indicate a measurement error.
- *Parameter 52-61 Inertia measurement velocity* and *parameter 52-62 Inertia measurement torque* for configuration of the inertia measurement procedure.
- *Parameter 52-63 Start inertia measurement* to perform the measurement operation. Set this parameter to *1* to trigger the operation.
- *Parameter 52-64 Inertia measurement result* to show the measurement result code and description if an error occurs during the inertia measurement.

NOTICE

The measured inertia is not automatically transferred to the control loop parameter set.

4

NOTICE

When the inertia measurement is carried out, ensure that the axis/mechanics connected to the drive can move freely.

4.3.5.2 SAB

Hand On screen

The *Hand On* screen for the SAB shows the *Controlword* parameter of the SAB as an editable hexadecimal value (see *chapter 8.1 Object 0x4040: Controlword*). Use this parameter to set the state of the SAB. In contrast to the *Hand On* screen on the ISD 510 servo drive, the *Hand On* screen on the SAB does not have different display modes.

Guide value reference simulation

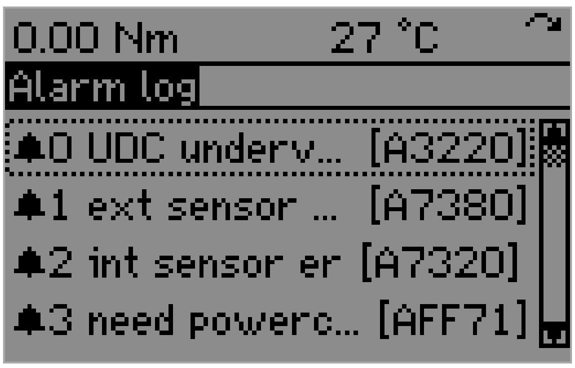
The simulation for the guide value reference (see *chapter 8.17 Object 0x2062: Position Guide Value Reference*) can be activated and deactivated via *parameter group 51-8* Guide value reference simulation* when in *Hand On* mode. The simulated position is readable from *parameter 50-81 Position guide value reference*.

Off mode

If the [Off] key is pressed when the SAB is in *Hand On* mode, the SAB state is changed to *Standby*.

4.3.6 Alarm Log

The *Alarm Log* menu shows the last 10 alarms from the device error history. The list is in descending order with the most recent alarm shown at the top. The error code and error text are shown for each entry.
An example of the *Alarm Log* is shown in *Illustration 4.19*.

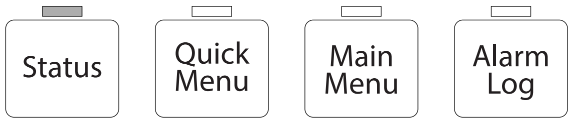


130BE902.10

Illustration 4.19 Alarm Log

4.4 Keys

The 4 main keys on the LCP (*Status, Quick Menu, Main Menu,* and *Alarm Log*) are used for navigation.



130BE920.10

Illustration 4.20 Main LCP Keys

4.4.1 Status Key

When the [Status] key is pressed, the *Status* menu is shown, depending on whether *Hand On* mode is active (see *chapter 4.3.3 Status Menu (Auto On Mode)* and *chapter 4.3.5 Hand On Mode*). If the key is pressed when the *Status* screen for *Auto On* mode is already shown, the *Status* display mode toggles between *Double Line Readout* and *Single Line Readout*. The [Status] key has no function when the *Status* screen for *Hand On* mode is shown.

4.4.2 Quick Menu Key

When the [Quick Menu] key is pressed, the *Main Menu* screen is shown (see *chapter 4.3.4 Main Menu*).

4.4.3 Main Menu Key

When the [Main Menu] key is pressed, the *Main Menu* screen is shown (see *chapter 4.3.4 Main Menu*). When switching the menu screen from *Status* or *Alarm Log* to *Main Menu*, the last viewed parameter is highlighted.

When entering the *Main Menu* for the 1st time after a power cycle, the root menu is shown with *parameter group 00-*** highlighted.

When the [Main Menu] key is pressed while the *Main Menu* screen is active and browsing through the parameter groups, the root menu is shown with *parameter group 00-*** highlighted.

When the [Main Menu] key is held down for 3 s, the parameter shortcut is shown (see *Illustration 4.21*), and it is possible to navigate to any parameter on the device by entering its number.

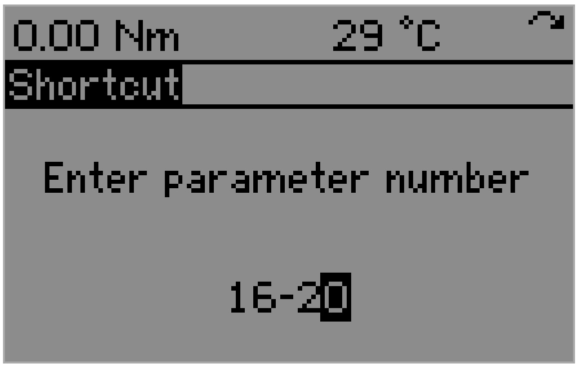


Illustration 4.21 Parameter Shortcut

4.4.4 Alarm Log Key

When the [Alarm Log] key is pressed, the *Alarm Log* screen is shown (see *chapter 4.3.6 Alarm Log*). If the *Alarm Log* screen is already active when the [Alarm Log] key is pressed, the latest alarm is highlighted (1st in the list).

4.4.5 Back Key

The [Back] key has no functionality when the *Status* screen is shown – both in *Hand On* mode and in *Auto On* mode. The [Back] key leads to the previous layer in the navigation structure when the *Main Menu* screen is shown, and while browsing through the device parameters.

If the [Back] key is pressed while editing the value of a parameter, the new value of the parameter is discarded and the edit mode is exited.

When the *Alarm Log* screen is shown, the [Back] key activates the previously shown screen (*Status* screen or *Main Menu* screen).

When the *Info* screen is shown for a parameter or alarm (see *chapter 4.4.7 Info Key*), the [Back] key closes the *Info* screen and shows the previous screen.

4.4.6 Cancel Key

If the [Cancel] key is pressed while changing the value of a parameter via the *Main Menu*, the previous value of the parameter is restored. This cancel functionality can be performed because the display has not been changed (for example, browsing through other parameters, or switching to another screen).

When the *Info* screen is shown for a parameter or alarm (see *chapter 4.4.7 Info Key*), the [Cancel] key closes the *Info* screen and shows the previous screen.

The [Cancel] key has no functionality when:

- The *Status* screen is shown – both in *Hand On* mode and in *Auto On* mode.
- The *Main Menu* screen is shown.
- Browsing through the device parameters.
- The *Alarm Log* screen is shown.

The cancel/undo functionality is not possible after navigating away from the changed parameter.

4.4.7 Info Key

The [Info] key activates the *Info* screen, which shows context-sensitive information. When the *Status* screen is shown – both in *Hand On* mode and in *Auto On* mode, the [Info] key shows the *Status info* screen. This shows the names and locations of the selected readout parameters (see *Illustration 4.22*).

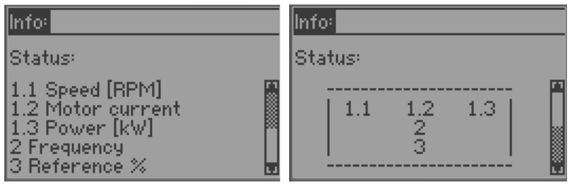


Illustration 4.22 Status Info Screens AutoOn_1 and AutoOn_2

When the *Main Menu* screen is active and currently showing a parameter, press the [Info] key to show the parameter info screen. It shows the name, value, and help text for the selected parameter (see *Illustration 4.23*).

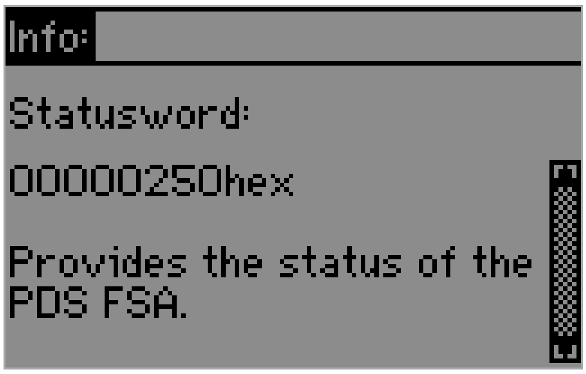


Illustration 4.23 Info Screen for a Parameter

When the *Main Menu* screen is active and currently showing a parameter group, press the [Info] key to show the *Parameter Group info* screen. It shows the name of the selected group.

When the *Alarm Log* screen is active and an alarm is highlighted, press the [Info] key to show the alarm info screen. It shows the name and help text of the selected alarm. This functionality is demonstrated in *Illustration 4.24*.

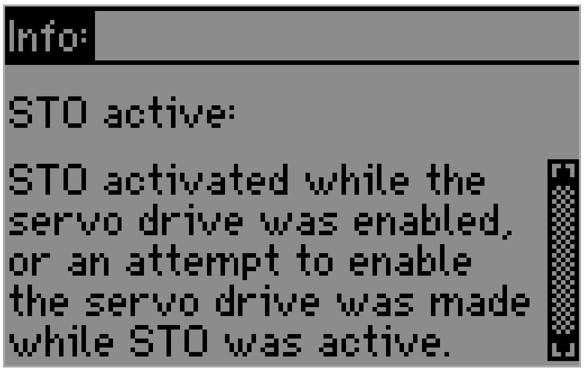


Illustration 4.24 Info Screen for an Alarm

Press either the [Back], [Info], or [Cancel] key to close the *Info* screen and return to the previous screen.

4.4.8 OK Key

Use the [OK] key to select a parameter marked by the cursor and to enable the change of a parameter. When editing a parameter value, press the [OK] key to store the present value as the new value for that parameter and return to the *Main Menu* screen.

When the *Main Menu* screen is showing a parameter subgroup, press the [OK] key to enter the parameter subgroup.

When the *Alarm Log* screen is shown, the [OK] key has the same functionality as the [Info] key.

The [OK] key has no functionality when the *Status* screen is shown – both in *Hand On* mode and in *Auto On* mode.

4.4.9 Hand On Key

Use the [Hand On] key to enter *Hand On* mode and show the *Hand On* status screen. This change is not possible if the device is in state *Operation enabled* and an error message rejecting the change is shown (see *Illustration 4.25*).

Illustration 4.25 Rejection of Hand On Mode on a Servo Drive while in State *Operation enabled*

4.4.10 Off Key

The [Off] key only functions when the device is in *Hand On* mode. In this case, *Off* mode is activated (see *Off* mode in *chapter 4.3.5.1 Servo Drive* and *chapter 4.3.5.2 SAB*). It is ignored in all other cases.

4.4.11 Auto On Key

The [Auto On] key hands over control to the fieldbus master. The device goes into remote control state and local control is not possible. This operation is only possible if the servo drive or SAB is not in state *Operation enabled*.

4.4.12 Reset Key

If the device is in *Hand On* mode, pressing the [Reset] key has the same effect as the *Reset* bit (bit 10) of the *Controlword*. It resets all alarms and errors, and changes the state of the device from *Fault* to its default (non-operational) state.

4.4.13 Up [▴] and Down [▾] Keys

The [▴] and [▾] keys move the cursor up and down in the navigation display.

Both keys have wrap-around navigation:

- When the top position is selected and the [▴] key is pressed again, the last position in the navigation display is selected.
- When the bottom position is selected and the [▾] key is pressed again, the first position in the navigation display is selected.

When a parameter value is being edited, use the [▴] and [▾] keys to set the new value: press the [▴] key to increment the edited value and the [▾] key to decrement it. Keeping the [▴] key pressed keeps incrementing the value by 1. Holding the key down speeds up incrementing to enable faster changes for larger values. This function also applies to the [▼] key.

4.4.14 Left [◂] and Right [▸] Keys

When editing a parameter, use the [◂] key to move the cursor 1 digit to the left, and the [▸] key to move the cursor 1 digit to the right. Both keys have wrap-around navigation: When the cursor is at the 1st digit (on the far right), pressing the [▸] key moves it to the last digit (on the far left). This function also applies to the [◂] key.

When the [◂] key is pressed while navigating the menu system, the menu screen moves to the previous subgroup or group. This function also applies to the [▸] key.

4.5 LCP-specific Parameters

LCP-specific parameters are defined for both the ISD 510 Servo Drive LCP (see *chapter 4.5.1 ISD 510 Servo Drive-specific LCP Parameters*) and for the SAB LCP (see *chapter 4.5.2 SAB-specific LCP Parameters*).

4.5.1 ISD 510 Servo Drive-specific LCP Parameters

Parameter number	Name	Unit	Description
00-20	Display line 1.1 small	–	Select a variable for display in line 1, left position.
00-21	Display line 1.2 small	–	Select a variable for display in line 1, middle position.
00-22	Display line 1.3 small	–	Select a variable for display in line 3, right position.
00-23	Display line 2 large	–	Select a variable for display in line 2.
00-24	Display line 3 large	–	Select a variable for display in line 3.
00-33	Unit for position readout	–	Selects the desired unit for position readout.
00-34	Unit for velocity readout	–	Selects the desired unit for velocity readout.
00-35	Unit for acceleration readout	–	Selects the desired unit for acceleration readout.
52-02	Hand On mode	–	Gets or sets the *Hand On* mode of operation: Position, Velocity, Jog, or Inertia measurement.
52-03	Hand On state	–	Sets the *Hand On* state: enabled or disabled.
52-11	Positioning type	–	Gets or sets the positioning type (absolute or relative) when running in *Hand On* mode for positioning.
52-63	Start Inertia Measurement	–	Starts the inertia measurement procedure.
52-64	Inertia Measurement Result	kg m^2	Shows the inertia measurement result.
52-70	Jog Mode Control	–	Enables or disables *Jog* in *Hand On* mode.
52-71	Jog speed	User units, as defined in *parameter group 00-3**.	Contains the commanded jog speed in user units (factor group has been applied).

Table 4.11 ISD 510 Servo Drive-specific LCP Parameters

4

4.5.2 SAB-specific LCP Parameters

Parameter number	Name	Unit	Description
00-20	Display line 1.1 small	–	Select a variable for display in line 1, left position.
00-21	Display line 1.2 small	–	Select a variable for display in line 1, middle position.
00-22	Display line 1.3 small	–	Select a variable for display in line 1, right position.
00-23	Display line 2 large	–	Select a variable for display in line 2.
00-24	Display line 3 large	–	Select a variable for display in line 3.
50-17	U_{AUX} Control	–	Enables or disables AUX line voltage.
50-18	UDC Control	–	Enables or disables UDC.
50-81	Position guide value reference	Degrees (°)	Scaled position guide value reference (0–360°).
51-64	AUX line 1 user limit	A	Sets a user limit for the current on auxiliary line 1. The current limit can be set in steps of 0.1 A. The value *0* disables the user current limitation. A warning is set when 90% of the user current limit is exceeded. Once 100% of the user current limit is exceeded, the line is switched off and the SAB enters an error state.
51-65	AUX line 2 user limit	A	Sets a user limit for the current on auxiliary line 2. The current limit can be set in steps of 0.1 A. The value *0* disables the user current limitation. A warning is set when 90% of the user current limit is exceeded. Once 100% of the user current limit is exceeded, the line is switched off and the SAB enters an error state.
54-10	Epl id assignment line	–	Selects the SAB line number for the ID assignment.
54-11	Drive index	–	Selects the index of the device in the selected line (1–32).
54-13	Epl id assignment start	–	Starts the manual ID assignment.

Table 4.12 SAB-specific LCP Parameters

5 Operation with ISD Toolbox

5.1 Overview

The ISD Toolbox is a standalone PC software designed by Danfoss. It is used for parameterization and diagnostics of the servo drives and the SAB. It is also possible to operate the devices in a non-productive environment. The ISD Toolbox contains several sub-tools for various functionalities.

The most important sub-tools are:

- *Scope* for visualization of the tracing functionality of the servo drives and SAB.
- *Parameter list* for reading/writing parameters.
- *Firmware update*
- *Drive control/SAB control* to operate the servo drives and/or SAB for testing purposes.
- *CAM editor* for designing CAM profiles for the servo drives.

5.2 ISD Toolbox Installation

5.2.1 System Requirements

To install the ISD Toolbox software, the PC must meet the following requirements:

- Supported hardware platforms: 32-bit, 64-bit.
- Supported operating systems: Microsoft® Windows XP Service Pack 3, Windows 7, Windows 8.1.
- .NET framework version: 3.5 Service Pack 1.
- Minimum hardware requirements: 512 MB RAM, Intel Pentium 4 with 2.6 GHz or equivalent, 20 MB hard disk space.
- Recommended hardware requirements: Minimum 1 GB RAM, Intel Core i5/i7 or compatible.

5.2.2 Installation

Administrator rights are required for installing the software with the Windows operating system. Contact your administrator if necessary.

1. Check that the system meets the system requirements as described in *chapter 5.2.1 System Requirements*.
2. Download the ISD Toolbox installation file (*www.drives.danfoss.com/services/pc-tools/*).
3. Right-click on the *.exe* file and select *Run as administrator*.
4. Follow the on-screen instructions to complete the installation process.

5.3 ISD Toolbox Communication

This chapter describes the Ethernet specific network interface settings needed by the ISD Toolbox. There are 2 basic communication methods: direct communication and indirect communication. Their particular network settings are described in the respective sections.

Read and perform the steps with care. Incorrect network configurations can lead to loss of connectivity of a network interface.

Firewall
Depending on the firewall settings and the fieldbus used, the messages sent and received by the ISD Toolbox may be blocked by the firewall on the ISD Toolbox host system. This may lead to a loss of communication and the inability to communicate with the devices on the fieldbus. Therefore, ensure that the ISD Toolbox is allowed to communicate through the firewall on the ISD Toolbox host system. Inappropriate changes to firewall settings may lead to security issues.

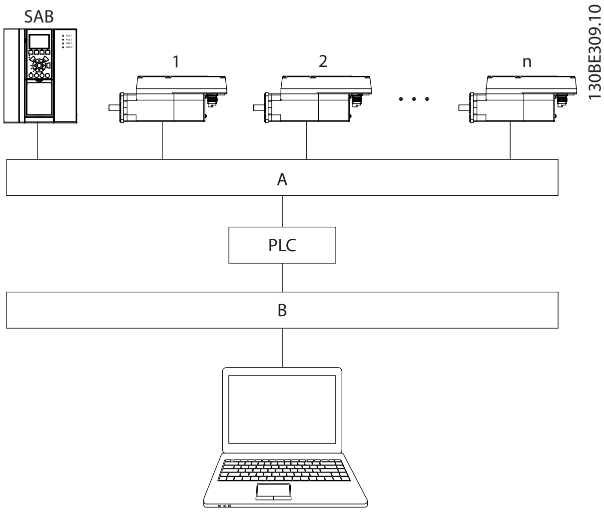
NOTICE

When using a dedicated network interface, the ISD Toolbox must be allowed to communicate specifically through this network interface.

Indirect communication
Communication between ISD 510 devices and the ISD Toolbox through a PLC is called indirect communication. Ethernet-based fieldbus communication (marked A in *Illustration 5.1*) takes place between the PLC and the ISD 510 devices. However there is non-fieldbus communication between the PLC and the ISD Toolbox host system (marked B in *Illustration 5.1*).

In the scenario in *Illustration 5.1*, the PLC has the master function and uses cyclic communication with the devices. Therefore, not all functionalities of the ISD Toolbox, for example the drive control, can be used.

5



A	Fieldbus
B	General-purpose network

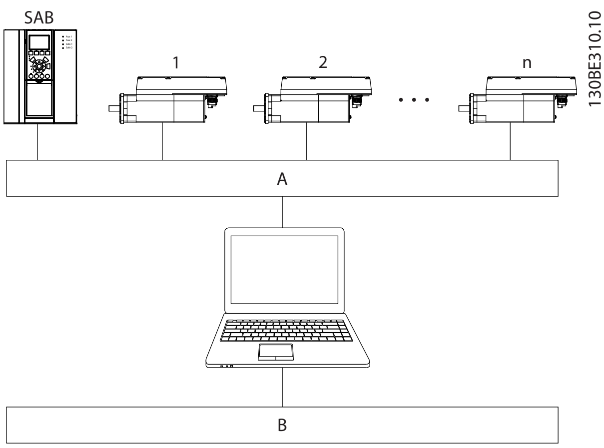
Illustration 5.1 Logical View of Indirect Ethernet-based Fieldbus Communication (Communication via PLC)

NOTICE

The logical view only shows the connectivity from a high-level software perspective and does not reflect the actual physical topology of the network.

Direct communication

For Ethernet-based fieldbus communication (direct communication), the ISD Toolbox must use a dedicated network interface on the ISD Toolbox host system. Do not use this network interface simultaneously for any other communication.



A	Fieldbus
B	General-purpose network

Illustration 5.2 Logical View of Direct Ethernet-based Fieldbus Communication

NOTICE

The logical view only shows the connectivity from a high-level software perspective and does not reflect the actual physical topology of the network.

5.3.1 Network Settings for Indirect Communication

Any network interface can be used to communicate through a PLC and a dedicated network interface is not needed.

When establishing the communication through a PLC, the ISD Toolbox configures a routing table using the selected *Network Address Translation (NAT)*. Adding a route to the Windows routing table requires administrator privileges. Therefore, administrator credentials may be requested when initializing the connection.

Carry out the following steps to enable indirect communication.

Disable IPv6 on the network interfaces used for communication on the PC:

1. Open the *Network and Sharing Center*.
2. Select *Change adapter settings*.
3. Right-click on the network interface used for fieldbus communication and select *Properties*.
4. If the *TCP/IPv6* is available for the network interface, disable it.

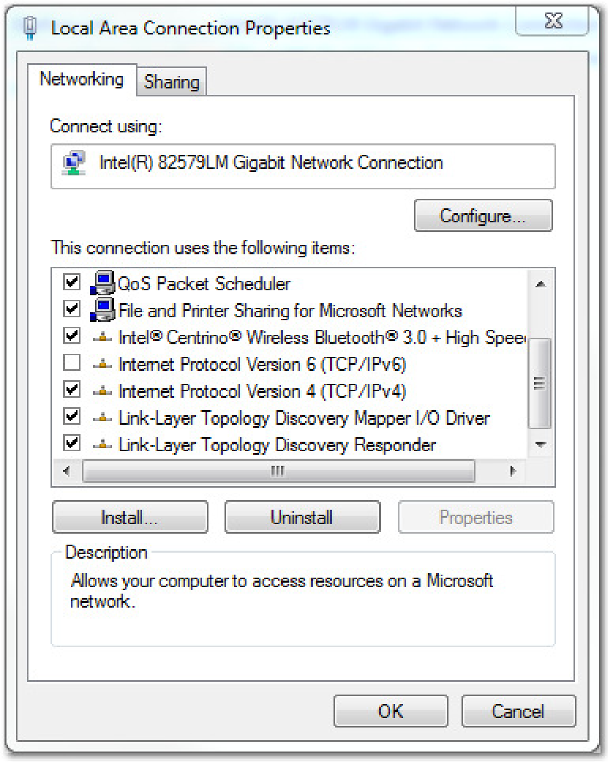


Illustration 5.3 Local Area Connection Properties

NOTICE

When observing the network packets via Wireshark®, checksum offloading often causes confusion as the network packets to be transmitted are handed over to Wireshark® before the checksums have been calculated. Wireshark® shows these empty checksums as invalid, even though the packets contain valid checksums when they leave the network hardware later.

Use 1 of these 2 methods to avoid this checksum offloading problem:

- **Turn off the checksum offloading in the network driver if possible.**
- **Turn off the checksum validation of the specific protocol in the Wireshark® preferences.**

Additional settings for indirect communication over EtherCAT®

Set the IP address of the EtherCAT® Master:

1. Open the *TwinCAT® System Manager*.
2. Select [I/O-Configuration → I/O Devices → Device1 (EtherCAT®)] and check the IP-address in the *Adapter* tab.
   The IP-address of the PLCs network adapter can not be a link-local address (so not in the range of 169.254.0.1 to 169.254.255.254).
3. If necessary, change the IP-address inside the *IPv4 Protocol* properties according to the given operating system. This can be done on the controller locally or via *Remote Desktop*.

Activate IP routing on the EtherCAT® Master:

NOTICE

The procedure described here can vary depending on the type of PLC and operating system installed.

1. Open the *TwinCAT® System Manager*.
2. Click on *Advanced Settings…* under [I/O-Configuration → I/O Devices → Device1 (EtherCAT®)] in the *EtherCAT* tab.
3. Select *EoE Support* in the *Advanced Settings* window.
4. Enable *Connect to TCP/IP Stack* in the *Windows Network* section.
5. Enable *IP Enable Router* in the *Windows IP Routing* section.
6. Reboot the PLC for the changes to take effect.

Set the IP address of the EtherCAT® slave (servo drive or SAB):

1. Open the *TwinCAT® System Manager*.
2. Click on *Advanced Settings…* under [I/O-Configuration → I/O Devices → Device1 (EtherCAT®) → Box 1 (VLT® Servo Access Box L1 → Drive 2 (VLT® Integrated Servo Drive ISD 510)] in the *EtherCAT* tab.
3. Select [Mailbox → EoE] in the *Advanced Settings* window.
4. Enable *Virtual Ethernet Port* and enter a valid IP Address.
5. Each slave in the configuration requires an IP-address. This address is reassigned with every transition from *INIT* to *Pre-Operational* state of the slave state machine. The IP communication of the slaves is deactivated per default.

NOTICE

The last number of the IP address is the ID that is used in the ISD Toolbox to identify the device.

5.3.2 Network Settings for Direct Communication with Ethernet POWERLINK®

Disable all network protocols except TCP/IPv4 on the network interface used for direct Ethernet POWERLINK® communication. This prevents other PC software or the operating system using this network interface for other tasks, such as file and printer sharing and network discovery. Disabling these protocols reduces the number of non-relevant packets sent over the network interface and thus reduces the overall network load.

How to disable all unused protocols on the network interface on the PC:

1. Open the *Network and Sharing Center*.
2. On the left, click on *Change adapter settings*.
3. Right-click on the network interface used for fieldbus communication and select *Properties*.
4. Uncheck all checkboxes except the one for *Internet Protocol Version 4 (TCP/IPv4)*.

5

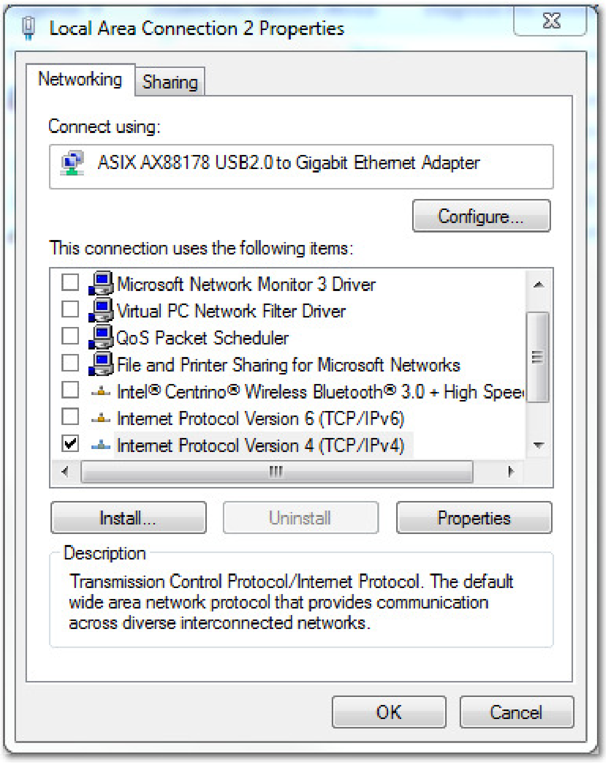


Illustration 5.4 Local Area Connection 2 Properties

Disable the *IPv4 Checksum offload* on the network interfaces as described in *chapter 5.3.1 Network Settings for Indirect Communication*.

How to set the correct Ethernet POWERLINK® master IP address:

1. Open the *Network and Sharing Center*.
2. On the left, click on *Change adapter settings*.
3. Right-click on the network interface used for fieldbus communication and select *Properties*.
4. Click on *Internet Protocol Version 4 (TCP/IPv4)* (the checkbox must be checked) and then click on *Properties*.
5. Select *Use the following IP address* and use 192.168.100.240 as the IP address and 255.255.255.0 as the subnet mask. Leave all other fields empty.

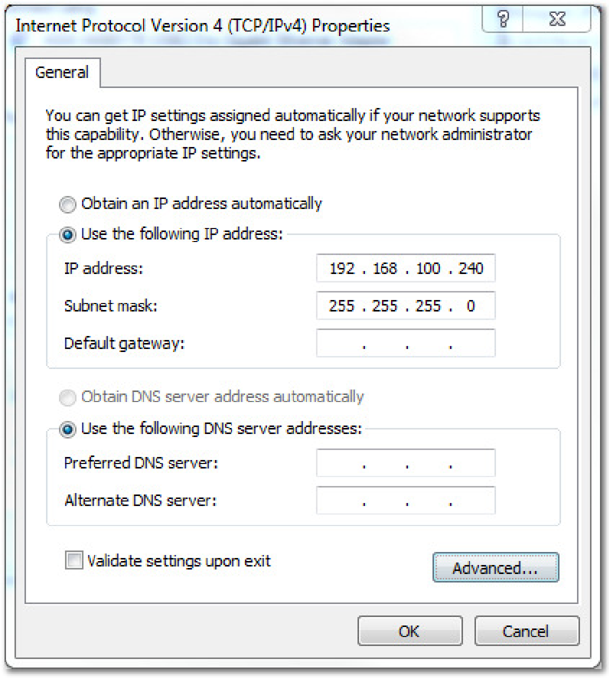


Illustration 5.5 Internet Protocol Version 4 (TCP/IPv4) Properties

5.3.3 Network Settings for Direct Communication with EtherCAT®

No EtherCAT®-specific network interface configuration needs to be performed on the ISD Toolbox host PC.

5.4 ISD Toolbox Commissioning

STEP 1: Open the main window

The *Main Window* is the basis for all ISD Toolbox functionalities. See *chapter 5.5.1 Main Window* for more information.

STEP 2: Connect to network

NOTICE

Pre-configure the appropriate communication settings to connect to a network. See *chapter 5.3 ISD Toolbox Communication* for further information.

1. In the *Main Window* toolbar, click on the *Connect to bus* icon to open the *Connect to Network* window.
2. Select the fieldbus type and the network interface to connect to.
3. Click on *OK* to connect.
4. Verify that the connection is successful by checking the status strip in the *Main Window*.

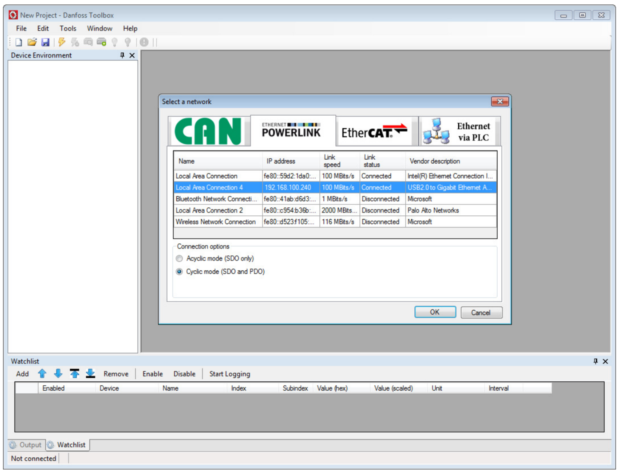


Illustration 5.6 Connect To Network Window (Ethernet POWERLINK®)

STEP 3: Scan for devices

1. After verifying that the ISD Toolbox is connected to the selected network, click on the *Scan for Devices* icon in the toolbar to trigger the device scan procedure.

NOTICE

If connected to an Ethernet POWERLINK® network in cyclic mode, select the scan range (minimum and maximum IDs) in the next window to reduce the time needed for scanning. In all other cases, the complete ID range is scanned.

2. When the scan is complete, a list of available devices is shown in the *Select Devices* window. Select which devices to add to the *Device Environment* and click on *OK*.
3. All selected devices appear in the *Device Environment* window and automatically go online (indicated by a glowing light bulb next to each device name).

The ISD Toolbox software can be used to optimize the control loop parameters. For this purpose, determine the inertia of the servo drive and the external load by using the inertia measurement within the drive control (see section *Inertia Measurement Mode tab* in *chapter 5.7.4 Drive Control (Servo Drive only)*).

Carry out the following steps to optimize the control loop parameter:

1. Use the drive control in profile position mode to:
    - 1a Create a step response (see section *Mode of operation controls – Profile Position Mode tab* in *chapter 5.7.4 Drive Control (Servo Drive only)*)
    - 1b Modify the control loop parameters.
2. Use the scope to visualize the result (see *chapter 5.7.3 Scope (Single and Multi-device for Servo Drive and SAB)*).
   Trace the following parameters:
    - Rotor N Act (Rotor actual velocity)
    - Nctrl N Set (Speed setpoint – input for speed controller)
    - Ictrl Iq Act (Actual current of torque component)
    - Ictrl Iq Set (Setpoint current of torque component)
    - Pctrl following error

To use the control loop parameters, enter them in the start-up parameter list of the PLC development environment. Alternatively, use them within the CAM profiles.

NOTICE

Any settings made within the ISD Toolbox software are overwritten when using a PLC.

5.5 Look and Feel

The ISD Toolbox is a multiple document interface program (MDI) – an environment that hosts multiple software tools in 1 single parent window. This allows simultaneous active interaction with, analysis, and commissioning of ISD 510 servo drives and Servo Access Boxes.
The parent window containing all the software tools is called *Main Window* and the software tools themselves are called *Sub-tools*, indicating that they can only be hosted within the ISD Toolbox environment.

5

5.5.1 Main Window

The main window contains:

- Menu and toolbar at the top of the window.
- Status strip at the bottom of the window.
- Workspace.

After start-up, the main window is shown with an empty device environment that is docked at the left side as default, however it can be moved to the right-hand side. The *Output* and *Watchlist* windows are docked at the bottom. The workspace is empty on start-up.
The *Device Environment*, *Output*, and *Watchlist* windows have a close button and a pin button to prevent the window from hiding. Both windows can be floating windows. The workspace hosts the sub-tools.

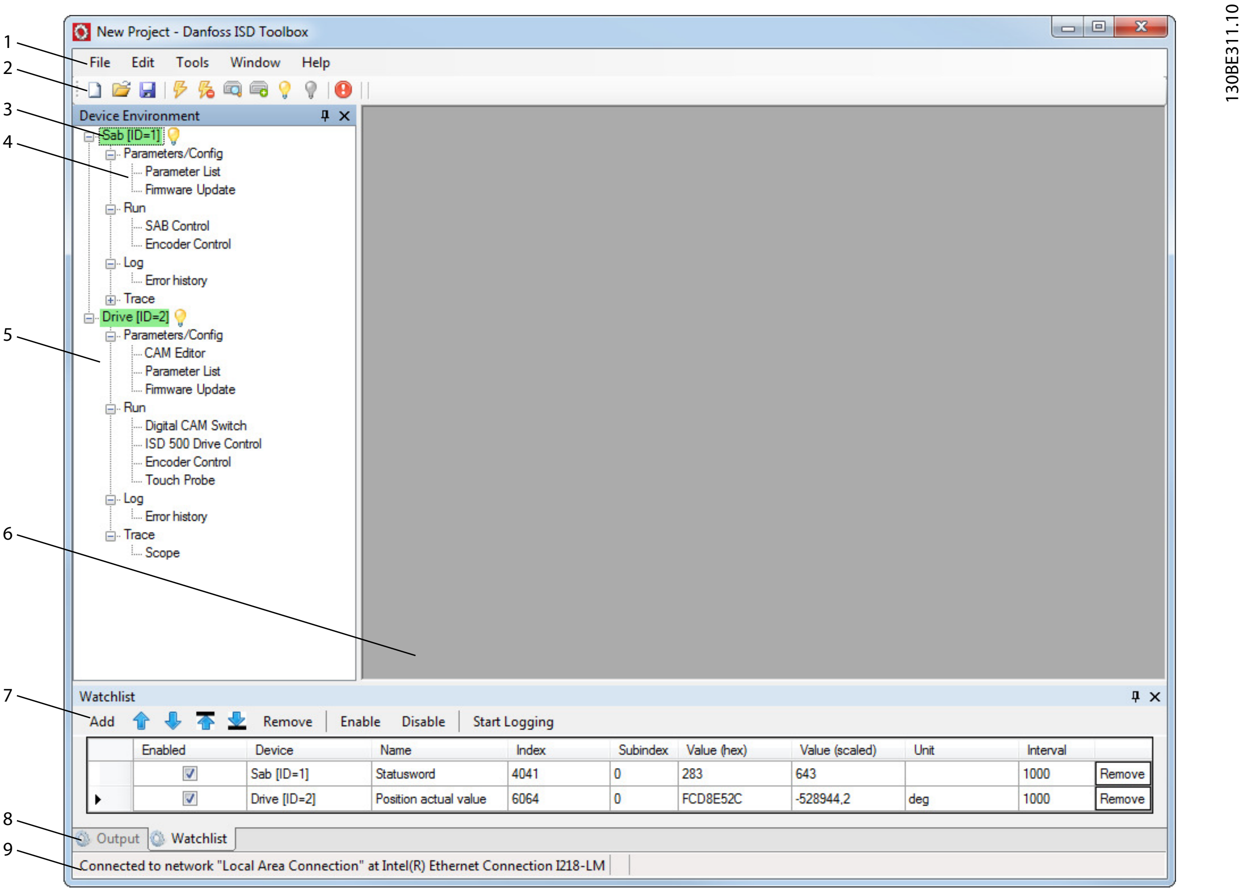


Illustration 5.7 Main Window

1	Menu bar	Contains the general functionalities for saving and loading projects, managing connections, showing and changing settings, managing open sub-tools, and showing help contents.
2	Tool bar	Contains shortcuts for saving and loading projects, connecting to and disconnecting from networks, automatic searching for online devices, or manually adding devices. Those functionalities can also be found inside the menu bar. Furthermore, a *Panic* button (shortcut key [F8]) is available in an emergency. It resets all devices on the bus and thereby disables their operation, however it does not provide any safety functionality. The behavior of the ISD servo drives is defined by the *Abort connection* option code. The SAB state is not directly affected by the reset command.

3	Online/offline status and state information	• *Online* devices are indicated by a glowing light bulb next to the device ID. - An online device is a logical device for which a physical device exists, which the ISD Toolbox is currently connected to. - The color indicates the state of the device and is device–specific. • *Offline* devices are indicated by a gray light bulb next to the device ID. - An offline device is a logical device without a corresponding physical device. An offline device can represent a saved device configuration or state, for example for offline analysis or troubleshooting. It also contains pre-configured parameter values to be written to a physical device.
4	Available sub-tools	Each sub-tool opens in a separate window within the workspace. It is possible to open multiple sub-tools that can communicate with a device simultaneously without affecting each other. Depending on the device type, different sub-tools are available. See *chapter 5.7 Sub-Tools* for further information. A sub-tool is opened by double-clicking the left mouse button on its name in the *Device Environment*, or by selecting the entry and pressing the *Enter* key on the keyboard.
5	Device environment	The *Device Environment* section of the *Main Window* lists all logical devices managed by the ISD Toolbox, visualizes their states, and serves as the user interface for accessing the device functionalities. The *Device Environment* window lists all available sub-tools for each added device. See *chapter 5.5.2 Device Environment Window* for further information.
6	Workspace	This is the space for hosting the sub-tools and its size depends on the *Main Window* size. The sub-tools can be maximized, minimized, horizontally or vertically aligned, or cascaded.
7	Watchlist window	Evaluates the parameter values of 1 or more devices by cyclically reading them from the devices. Allows parameter values to be logged and saved to a text file. It is also possible to modify/write values in the watchlist.
8	Output window	Shows operating information, warnings, and errors. Depending on the user settings, it shows messages of up to 3 different logging levels (high, medium, and low). Used for showing advanced error and warning information.
9	Status strip	Shows the communication state of the ISD Toolbox. If connected to a network, it shows the used hardware interface (for example, network adapter) and the network name.

Table 5.1 Legend to *Illustration 5.7*

5.5.2 Device Environment Window

The *Device Environment* window lists all available sub-tools for each added device. See *chapter 5.7 Sub-Tools* for further information on the sub-tools.

The *Device Environment* window makes a distinction between *online* and *offline* devices:

- A glowing light bulb next to the device ID indicates that the device is *online*.
- A gray light bulb next to the device ID indicates that the device is *offline*.

A sub-tool is opened by double-clicking the left mouse button on its name in the *Device Environment*, or by selecting the entry and pressing the [Enter] key on the keyboard.

Illustration 5.8 shows an example of the *Device Environment* window with 2 online ISD drives (IDs 1 and 2). The sub-tools of both devices (for example CAM Editor, Parameter List, Firmware Update, and Scope) are ordered by sub-tool category.

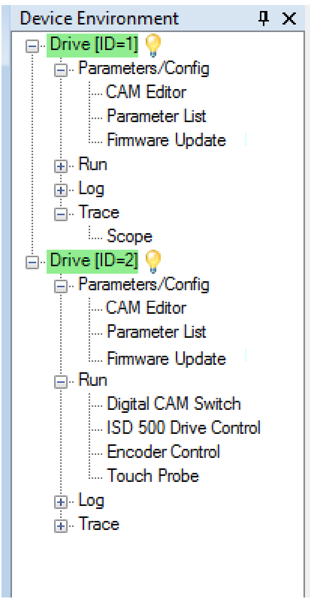


Illustration 5.8 Device Environment Window with 2 Online Devices

For each online device, the *Device Environment* visualizes the current state of the device. For this purpose, each root node entry has a background color. The color code and the available states for the device are listed in *Table 5.2* and *Table 5.3*.

CAN CiA DS402 state	Color
Not reachable/unknown state/offline mode	No color
Not ready to switch on	Gray
Switch on disabled	Light green
Ready to switch on	Turquoise
Switched on	Blue
Operation enabled	Green
Quick stop active	Yellow
Fault	Red

Table 5.2 ISD 510 Servo Drive Color Code

SAB state	Color
Not reachable/unknown state/offline mode	No color
Init	Gray
U_{AUX} disabled	Yellow
Standby	Light green
Power up	Blue
Operation enabled	Green
Fault	Red

Table 5.3 SAB Color Code

When right-clicking on a device in the *Device Environment* window, a context menu with functionalities and device-specific information is shown. In addition to other functionalities, the device information window can be opened. The *Device Environment* window can be hidden via menu [Window → Show/Hide device environment window], or by pressing the [F4] key.

5.5.2.1 Device Information Window

For each device, a *Device Information* window can be opened by using the device context-menu in the *Device Environment* window. The design of the *Device Information* window and its functionality depends on the fieldbus used.

For EtherCAT®

The *Device Information* window for EtherCAT® devices is shown in *Illustration 5.9*. The EtherCAT ®*Device Information* window shows the following information in read-only text boxes:

- Firmware version
- Device name
- Slave index
- Address
- Supported features (CoE, FoE, EoE, DC)

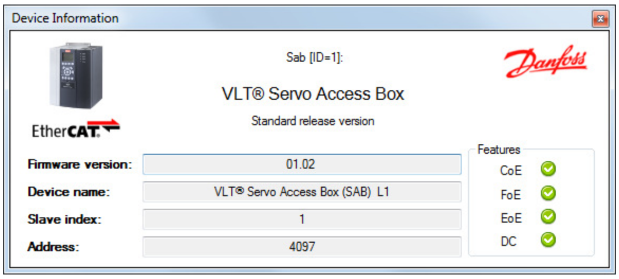


Illustration 5.9 Device Information Window – EtherCAT®

For Ethernet POWERLINK®

The *Device Information* window for Ethernet POWERLINK® devices contains the firmware version of the device and contains 2 tabs: *General Information* and *Service Mode*. The *General Information* tab is shown in *Illustration 5.10*.

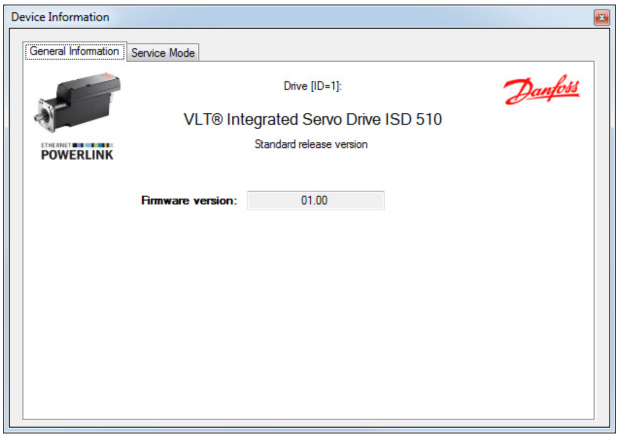


Illustration 5.10 Information Window – Ethernet POWERLINK® – General Information

The *Service Mode* tab contains controls for entering the Ethernet POWERLINK® specific service mode of the ISD 510 devices (servo drive and SAB), and for reading device information in *Service Mode*. *Illustration 5.11* shows the *Service Mode* tab page of the *Device Information* window.

To enter *Service Mode*, the device must be connected using direct, acyclic communication. First carry out a power cycle if the device has been connected to a PLC.

Clicking on the button *Enter Service Mode* changes the device state to service mode and enables the button *Acquire Device Info*. When clicking on the *Acquire Device Info* button, the device information is read from the device. While the device is in *Service Mode*, it is not possible to close the *Device Information* window or access any other control in the ISD Toolbox. This ensures the integrity of the device state.

Illustration 5.11 shows the *Service Mode* tab page of the *Device Information* window. The device is in service mode (indicated by the red text "Service Mode"), and the device information shown in the respective text boxes has already been read.

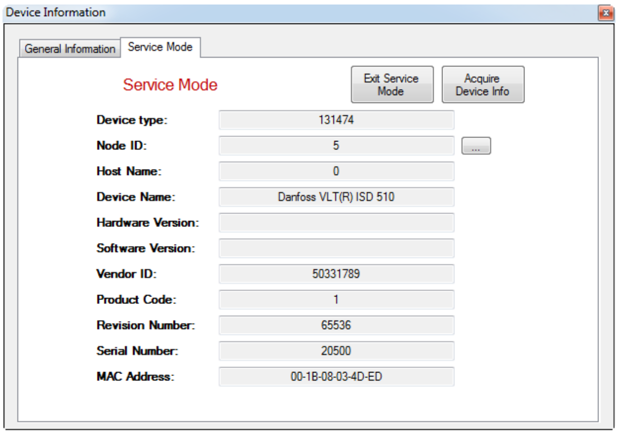


Illustration 5.11 Information Window – Ethernet POWERLINK® – Service Mode

The *Service Mode* tab consists of the following read-only text fields that are read when entering *Service Mode* and acquiring the device information:

- Device type
- Node ID
- Host Name
- Device Name
- Hardware Version
- Software Version
- Vendor ID
- Product Code
- Revision Number
- Serial Number
- MAC Address

While the device is in *Service Mode*, the *Node ID* can be changed by clicking on the button on the right of the *Node ID* field and typing in the desired node ID in the pop-up window. The ID is instantly applied on the device and a power cycle is not required.

5

5.5.3 Watchlist Window

Parameters that are added to the *Watchlist* window are continuously polled from the device using a specified read interval. All readable parameters of all configured devices can be used in the *Watchlist* window. All writable parameters can also be changed in the *Watchlist* window by writing the value into 1 of the columns *Value (hex)* or *Value (scaled)*.

To log the cyclically read values in a text file, click on the *Start Logging* button and then select the target log file path. The log file has a plain-text comma-separated format (.csv) and can be viewed without the need of any specialized software.

The *Interval* for polling the values are configurable individually for each parameter in multiples of the *Watchlist resolution* parameterized in the *Options* window (see *chapter 5.5.8 Options Window*).

The parameters can be added and moved up or down the list. It is also possible to remove, enable, or disable the update of each individual parameter and remove, enable, or disable all parameters of a specific device. Furthermore, it is possible to remove, enable, or disable all the parameters in the *Watchlist*.

The *Watchlist* window can be hidden or shown via menu [Window → Show/Hide watchlist], or by pressing the [F5] key.

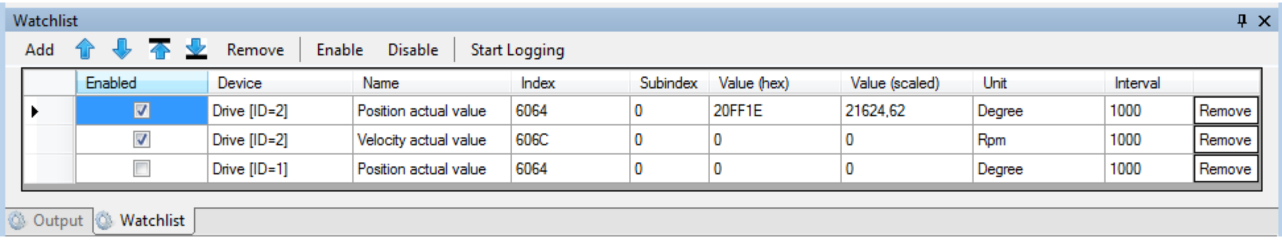


Illustration 5.12 Watchlist Window

5.5.4 Output Window

The *Output* window shows operating information, warnings, and errors from the Toolbox application itself, as well as warnings and errors from the connected devices. It shows detailed information regarding Toolbox failures, incorrect configurations, or missing software components and supports the following functions:

- Analysis of communication
- Tracking device states
- Searching for errors
- Testing and debugging devices

The maximum number of messages can be set by using the *Options* window (see *chapter 5.5.8 Options Window*). The *Output* window logging level has 3 possible settings that can be changed using the *Options* window:

- High: Shows basic events.
- Medium: Default setting.
- Low: Shows detailed communication and configuration events.

When a new message is shown, the *Output* window automatically scrolls to the bottom of the list in order to show the latest message.

The *Output* list entries are classified as *Messages*, *Warnings*, or *Errors*. Select whether an entry type should be shown by checking/unchecking the respective checkbox at the top of the *Output* window.

The timestamp of the message is generated when the message is shown. The content of the window can be saved to a plain-text file by clicking on the *Save log* button.

The *Output* window can be hidden or shown via menu [Window → Show/Hide output window] or by pressing the [F6] key.

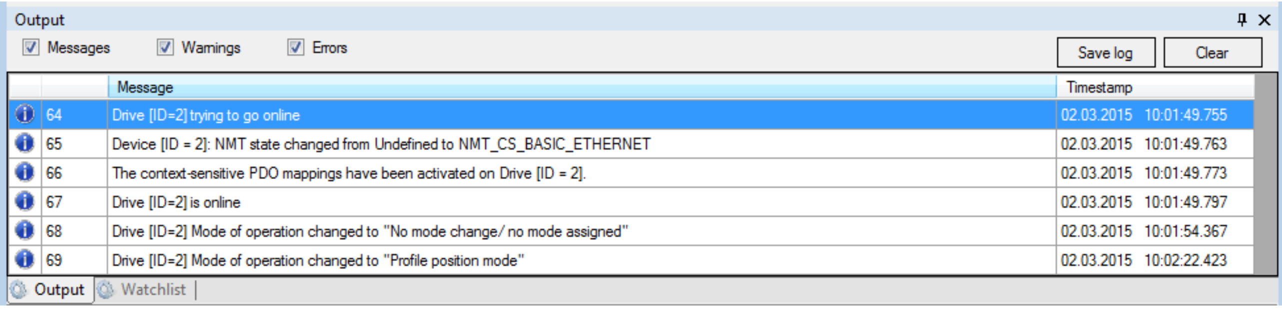


Illustration 5.13 Output Window

5.5.5 Project File

A project file (extension .isdproj) contains all devices that are part of the *Device Environment*. All parameters of each device, apart from state information, are saved within a project file that also contains information about open sub-tools, their window size, location, and the contents of the *Watchlist*.

Project files can be opened or saved using the commands in the menu bar or the icons in the toolbar.

5.5.6 Importing and Exporting Devices

It is possible to import and export the configuration and parameter values of devices. Exported devices can be reimported and shown in the *Device Environment* window.

The file extension of the ISD Toolbox exported device files is *.isddev*. Multiple devices can be exported to a single file.

5.5.7 Online Help

The context-sensitive help for the ISD Toolbox software is accessible via the menu bar. All the main subjects are accessible via the table of contents and there is a search function. Press the [F1] key from within the ISD Toolbox to open the corresponding help page.

5.5.8 Options Window

The *Options* menu entry opens the *Toolbox options* window that contains tabs with the following option categories:

- General
- Watchlist
- Default values
- Drive Units
- Fieldbus-specific
- Updates

The *Toolbox options* window also has a checkbox to specify whether the *Toolbox options* window should be shown on start-up.

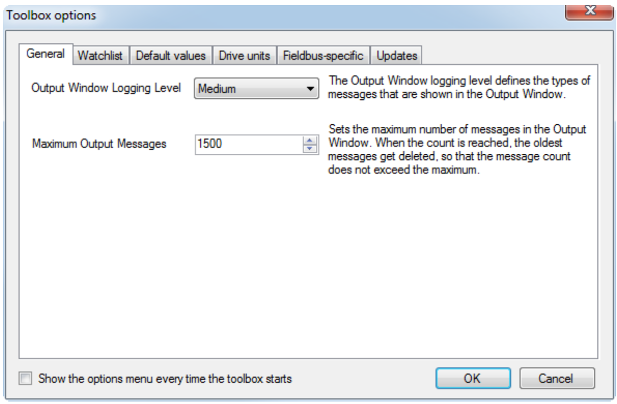


Illustration 5.14 Toolbox Options Window

The *General* tab contains the settings *Output Window Logging Level* and *Maximum Output Messages*. The *Output Window Logging Level* has 3 possible settings:

- High: Shows basic events.
- Medium: Default setting.
- Low: Shows detailed communication and configuration events.

The *Maximum Output Messages* option can be set to values of 500–2500. After the maximum message count is reached, the oldest message is deleted in order to show a new one.

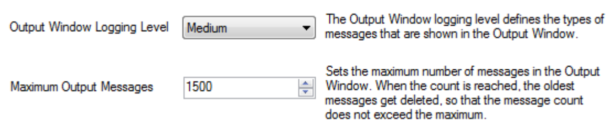
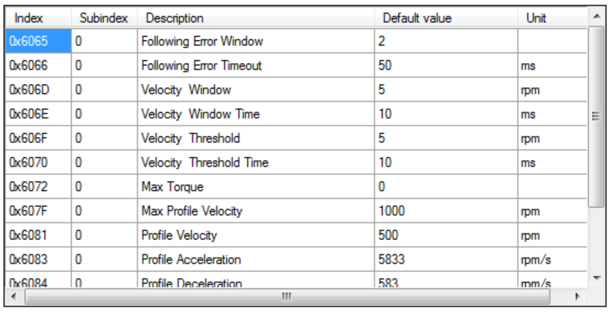


Illustration 5.15 Toolbox Options - General Tab

The *Watchlist* tab contains the *Watchlist resolution* field that sets the rate at which the *Watchlist* triggers its update procedure (see *chapter 5.5.3 Watchlist Window*). It is the minimum interval at which the *Watchlist* is updated. The update rate of each parameter in the *Watchlist* can only be a multiple of the *Watchlist* resolution.

Illustration 5.16 Toolbox Options - Watchlist Tab

The *Default values* tab consists of a data grid with the columns *Index*, *Subindex*, *Description*, *Default value*, and *Unit*. It contains multiple pre-defined parameters that are applied to devices that contain them. All default values are used as initial RxPDO values that are transmitted to the devices when using cyclic communication. It is not possible to add new parameters to the list.



Index	Subindex	Description	Default value	Unit
0x6065	0	Following Error Window	2	
0x6066	0	Following Error Timeout	50	ms
0x606D	0	Velocity Window	5	rpm
0x606E	0	Velocity Window Time	10	ms
0x606F	0	Velocity Threshold	5	rpm
0x6070	0	Velocity Threshold Time	10	ms
0x6072	0	Max Torque	0	
0x607F	0	Max Profile Velocity	1000	rpm
0x6081	0	Profile Velocity	500	rpm
0x6083	0	Profile Acceleration	5833	rpm/s
0x6084	0	Profile Deceleration	583	rpm/s

Illustration 5.17 Toolbox Options - Default Values Tab

The *Drive units* tab contains settings for the position, velocity, and acceleration units for drive parameters that are scaled according to the DS402 Factor Group. These settings are used for all drive devices but do not influence any parameters of the drives. They are only used for scaling the values within the ISD Toolbox.

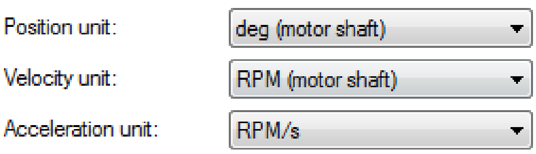


Illustration 5.18 Toolbox Options - Drive unit

The following units are available for scaling position, velocity, and acceleration:

- Position unit
  - Degree
  - Radian
  - Revolution
- Velocity unit
  - RPM
  - RPS
  - Degree/sec
  - Radian/sec
- Acceleration unit
  - RPM/sec
  - RPS/sec
  - Degree/sec^2
  - Radian/sec^2

The *Fieldbus-specific* tab contains fieldbus-specific settings, organized in the sub-tabs *Ethernet POWERLINK*, *EtherCAT*, and *Communication via PLC*. Settings are not required for *Ethernet POWERLINK*.

The *Communication via PLC* tab contains the *SDO init timeout* and *SDO transfer timeout* values in ms. Modify these 2 values if the connection to the PLC is slow or the SDO channels of the devices behind the PLC are filled by the PLC itself, and the SDO communication is therefore slow.

The default *SDO init timeout* is 40 ms and the default *SDO transfer timeout* is 700 ms.

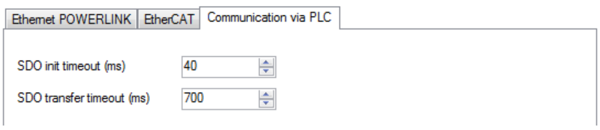


Illustration 5.19 Toolbox Options - Fieldbus-specific Tab

The *Updates* tab contains the functionality for searching for new firmware packages and updating the ISD Toolbox, its sub-tools, configuration, and online-help. This can be triggered by clicking on the *Check for Updates now* button.

The check for updates procedure can also be performed automatically when the ISD Toolbox starts. This option is disabled per default but can be enabled by checking the check-box *Automatically check for updates* when the ISD Toolbox starts (see *Illustration 5.20*).

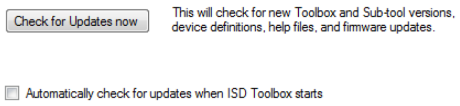


Illustration 5.20 Toolbox Options - Update Tab

5.6 Connection and Devices

5.6.1 Connect to Bus

Connecting to networks is done via menu [Edit → Connect to Bus] or by using the *Connect to Bus* shortcut in the toolbar. This opens a window for selecting the fieldbus type, network interface, and communication options (see *Illustration 5.21*).

Each available fieldbus type is listed as a tab in the *Select a network* window. After selecting a fieldbus type, the available network interfaces and communication options specific to the fieldbus type are shown.

NOTICE

The *CAN* fieldbus tab is not relevant for the ISD 510 servo system.

Select fieldbus type *Ethernet POWERLINK®* or *EtherCAT®* for direct communication to the devices and then select cyclic or acyclic communication.

By selecting the tab *Ethernet via PLC*, indirect communication via a PLC is selected. In this case, enter the IPv4 address of the PLC, and the IPv4 address of the internal subnet of the devices (NAT address).

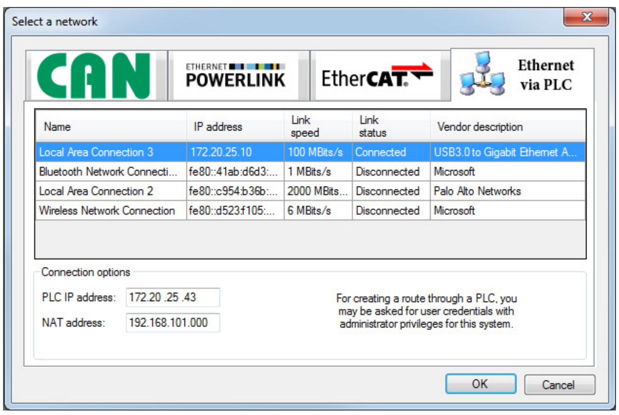


Illustration 5.21 Select a Network Window (Ethernet via PLC)

After successfully connecting to a network, a window opens where it is possible to go online with any offline devices that have been added to the *Device Environment*.

When connecting to an Ethernet POWERLINK® network, the ISD Toolbox performs checks on the network interface to ensure that communication can be established. The following preconditions must be fulfilled; otherwise an error is shown:

- The network interface has the IPv4 address 192.168.100.240.
- The link status of the network interface is *Connected*.

It is only possible to connect to 1 network at a time. If the ISD Toolbox connects to another network, the old connection is closed automatically before connecting to the new network.

5.6.2 Disconnect from Bus

The function *Disconnect from bus* means that all communication to the network is stopped. All devices that are online go offline.

5.6.3 Online/Offline Devices

An *Online Device* is a logical device for which a physical device, currently connected to the ISD Toolbox, exists.

An *Offline Device* is a logical device with no corresponding connected physical device. It can represent a saved device configuration or state (for example, for offline analysis or troubleshooting). It can also contain pre-configured parameter values to be written to a physical device.

5.6.4 Adding/Removing Devices

To add an offline device to the *Device Environment*, specify the device type, fieldbus type, device ID, and the major firmware version number. It is then possible to go online with the added device once it is connected to the ISD Toolbox.

The device ID has to be positive (>0), and the maximum ID is 239 for both Ethernet POWERLINK® and EtherCAT®.

If a device ID that has already been added to the *Device Environment* is selected, a warning sign appears next to the *Device ID* input field and a tooltip informs that a device with the specified ID is already existent in the *Device Environment*. However, an offline device (with a double node ID) can still be added to the *Device Environment*.

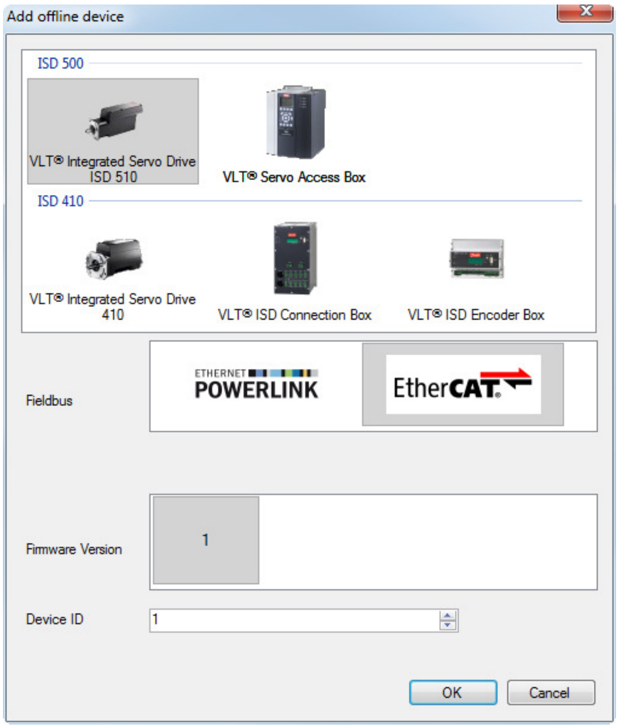


Illustration 5.22 Offline Device Window

All devices that are added to the project are shown inside the *Device Environment* window as root nodes of the tree shown (see *Illustration 5.8*). Each device is denoted by using its device type and node ID in brackets. The sub-tool categories and the sub-tools themselves are shown as child nodes of the respective device.

5.6.5 Scan for Devices

When connected to a network, the ISD Toolbox can scan for available devices on the network and add them to the *Device Environment*. The scanning procedure is triggered by using the menu [Edit → Scan for devices] or via the *Scan for Devices* icon in the toolbar.

Depending on the fieldbus, the scan procedure for the whole ID range can take some time. To limit the maximum time for the procedure, it is possible to scan only within a certain ID range (cyclic Ethernet POWERLINK®). This can be done via menu [Edit → Scan for devices → Scan in range]. This shows a window where the minimum and maximum IDs to search for can be specified. Use the menu [Edit → Scan for devices → Scan entire network] to search the entire network (CAN, EtherCAT®, acyclic Ethernet POWERLINK®, and Ethernet via PLC).

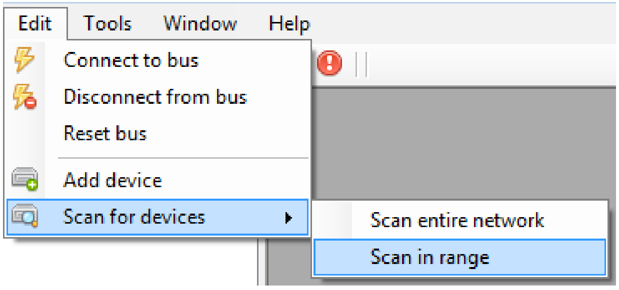


Illustration 5.23 Scan in Range Menu Item

When the scan procedure is triggered, a window indicating the scan progress and the IDs found on the fieldbus are shown. When the scan is completed, the *Select Devices* window shows all found devices and the desired devices can be added to the *Device Environment*. The scan procedure can be stopped at any time and the devices already found can be added to the *Device Environment*.

5.7 Sub-Tools

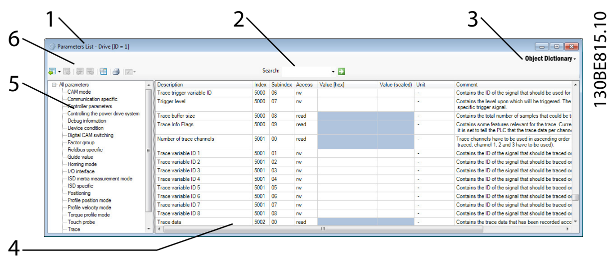
5.7.1 Parameter List (Servo Drive and SAB)

The *Parameter List* sub-tool shows all available parameters of a device. It is the main tool designed for browsing parameters and reading and writing parameter values on an online device. The *Parameter List* shows all parameters in table format containing both parameter information and parameter values.

There are 2 parameter views available in the *Parameter List*:

- Object dictionary view
- Parameter view

When the *Object dictionary view* is selected, the *Parameter List* shows the parameters with their unique index-sub-index identifier (see *Illustration 5.24*). When the *Parameter view* is selected, the *Parameter List* shows the parameters with their unique parameter number. The parameter groups tree at the left of the sub-tool window is used to filter the shown parameters according to their respective categories. Selecting the root node of the tree disables any filters and shows all parameters.



1	Title
2	Search field
3	Object dictionary or parameter view
4	Parameter table
5	Parameter groups
6	Toolbar

Illustration 5.24 Parameter List Sub-tool

The toolbar in the *Parameter List* sub-tool contains the following functionalities:

Read all parameters	Reads all parameters inside the selected group from the device and updates their values in the parameter table. Only available for online devices. The drop-down menu of this icon contains these functionalities: • *Read All and Export* Reads all parameters and writes them to a file. • *Import from File* Reads back a parameter file and puts the values into the parameter list. The values are not automatically written to the device.
Write all parameters	Writes the displayed values of all parameters inside the selected group to the device. Only available for online devices.
Read the selected parameters	Reads parameters selected in the parameter table from the device and updates their values in the parameter table. Only available for online devices.
Write the selected parameters	Writes the displayed values of the parameters selected in the parameter table to the device. Only available for online devices.
Get default value for single parameter	Reverts the values of the parameters selected in the parameter table to their default values and updates the parameter table. The default parameter values are not written to the device automatically.
Print	Offers the possibility to print the parameter list using the standard Windows print window.
Store/restore parameters (only available for Ethernet POWERLINK®)	Sends a request to the device to store the current parameter values to non-volatile memory, or to restore them from non-volatile memory. It does not automatically perform a read from the device. Only available for online devices.

Table 5.4 Parameter List Sub-tool Functionalities

In the *Search* field, a parameter can be searched for by name, object index, and sub-index (object dictionary view), or parameter number (parameter view). The *Search* field saves the last 10 search inputs.

The information about a value range is shown as a tooltip when the mouse moves over the value field.

If no value is read from the device, the default value of this parameter is shown in italics (only for values that have a default parameter).

Depending on the value type (numeric, text, a set of specific values) the input field provides a reasonable input or display method. If there are limits to a numerical value, they are applied to the input field either by making it impossible to enter wrong values or, if this is not possible, by keeping the focus inside the field until a correct value has been entered.

Reading and writing of single parameters is also available via the *Context* menu or by using the shortcuts *Ctrl+R* for reading and *Ctrl+W* for writing.

Reading or writing all parameters may take up to 3 minutes, due to the large number of parameters and the bandwidth limits of the fieldbus.

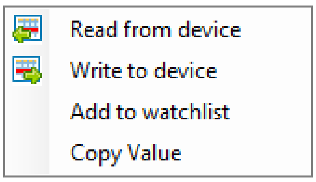


Illustration 5.25 Parameter Context Menu

5

5.7.2 Firmware Update (Single and Multi-device for Servo Drive and SAB)

5.7.2.1 Single Device Firmware Update

This sub-tool is used to update the firmware of a device over a network, for example direct Ethernet POWERLINK® or EtherCAT® connection or indirect connection via a PLC. The firmware can be updated by sending a firmware package to the device.

Illustration 5.26 shows the *Firmware Update* sub-tool. After clicking the *Open* button, the firmware package file with a *.bin* extension can be flashed. The name of the selected *.bin* file is then shown next to the button. Alternatively, drag and drop a firmware package file from Windows Explorer to the *Firmware Update* sub-tool to open it.

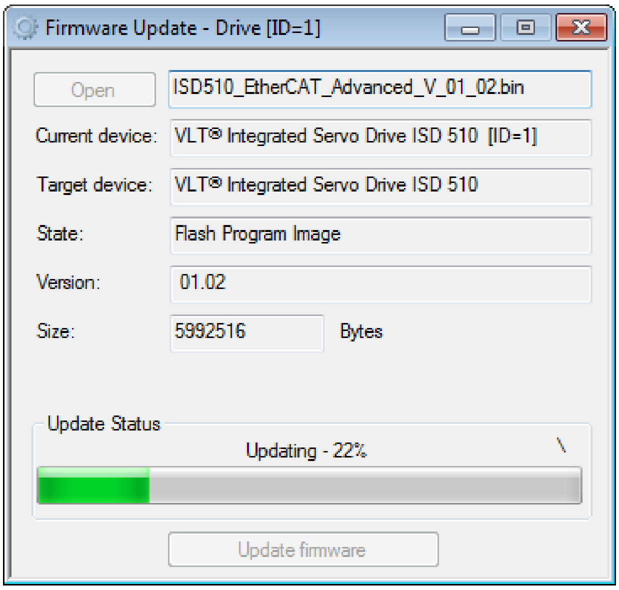


Illustration 5.26 Firmware Update Sub-tool

It may take some time until the message *Firmware update is complete* is shown. Do not power off during this time.

Field	Description
Current device	Shows the device to be updated.
Target device	Shows the device type defined in the firmware package file.
Version	Shows the target version of the loaded firmware package file.
Size	Contains the size of the firmware package file in bytes.

Field	Description	
Update Status	Shows the current state of the *Firmware Update* sub-tool. It contains a text field and a progress bar. The text field can show 1 of these states:	
	No image loaded	No firmware package file was loaded from the file system.
	Ready	A firmware package file was successfully loaded and matches the target device type.
	Initializing	The firmware update is starting. This state is typically short.
	Updating - X%	The firmware update procedure is being executed and is X% complete.
	Firmware update is complete	All packets have been sent to the device successfully.
	Firmware update failed	An error occurred when transferring the firmware package. The *Output* window contains further information.
Update firmware	Starts the firmware update procedure.	

Table 5.5 Firmware Update Sub-Tool

NOTICE

The device type specified in fields *Current device* and *Target device* must match, otherwise an error message is shown, and the *Update firmware* button remains disabled.

The *Firmware Update* sub-tool only allows updating new firmware when the device is not in operation. The firmware cannot be updated if 1 or more of the following conditions are true:

- The fieldbus is Ethernet POWERLINK® and cyclic communication is performed.
- The device is an ISD servo drive and it is in state *Operation Enabled*.
- The device is an SAB and it is in state *Operation Enabled*.

5.7.2.2 Multi-Device Firmware Update

The *Multi-Device Firmware Update* sub-tool shown in *Illustration 5.27* contains functionality for updating the firmware of multiple devices with the same firmware package.

Close all other tools before opening the *Multi-Device Firmware Update* sub-tool because no other functionalities can be used during the firmware update process.

Open the Multi-device Firmware Update sub-tool via menu [Multiple Device Functionalities → Firmware Update] for a list of online devices. The devices to update can be selected but must have the same device type: Servo drive or SAB.

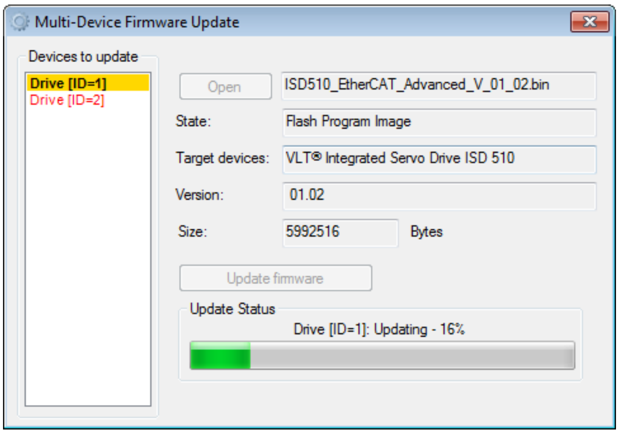


Illustration 5.27 Multi-Device Firmware Update Sub-tool

On the left, the sub-tool contains the list of all the selected devices. The status of the update of each device can be seen in *Table 5.6*:

Color	Status
Yellow background and bold font	Device is currently being updated.
White background and green font	Device was updated successfully.
White background and red font	Update of device failed.
White background and black regular font	Update procedure not yet started on this device.

Table 5.6 Update Status

The right side of the *Multi-Device Firmware Update* sub-tool is structured in the same way as the single device firmware update sub-tool.

While updating, a button for canceling the update process appears at the bottom of the sub-tool. If it is pressed, the current running update process is completed, however the remaining selected devices are not updated.

5.7.3 Scope (Single and Multi-device for Servo Drive and SAB)

The *Scope* sub-tool is a graphical user interface component for the tracing functionality of the servo drive. It is used for optimizing control loop parameters and debugging purposes. The design and usability is close to the industry-standard digital oscilloscopes.

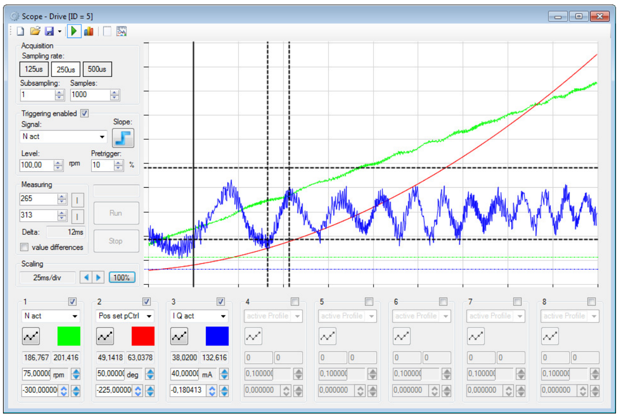
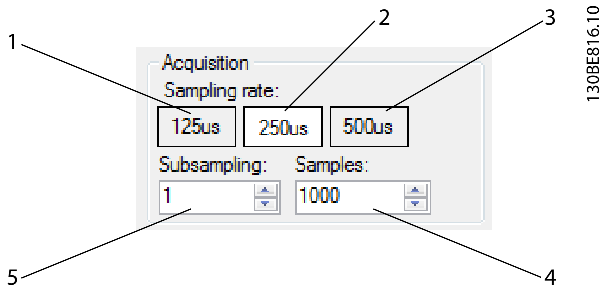


Illustration 5.28 Scope Sub-tool

5.7.3.1 Sampling

The scope sampling options are shown and controlled using the sampling control in the top-left corner of the *Scope* window. *Illustration 5.29* gives an overview of the sampling control.



1	Real-time task sampling
2	Fast task sampling
3	Slow task sampling
4	Number of samples per channel
5	Subsampling

Illustration 5.29 Sampling Control

Set the task level of the trace using the *Sampling rate* buttons:

- 125 µs (Real-time task sampling)
- 250 µs (Fast task sampling)
- 500 µs (Slow task sampling)

The ISD 510 devices have different sampling rates. The sampling rates of the tasks are read from the device itself when starting the *Scope*. The available sampling rates depend on the fieldbus cycle time.

Real-time task	100 µs, 125 µs
Fast task	200 µs, 250 µs
Slow task	400 µs, 500 µs

Table 5.7 Sampling Rates

In the example depicted in *Illustration 5.29*, the real-time task sampling is 8 kHz (125 µs between 2 samples), the fast task sampling is 4 kHz (250 µs between 2 samples), and the slow task sampling is 2 kHz (500 µs between 2 samples). These sampling rates are also shown for offline devices.

The *Subsampling* field controls the subsampling level of the trace, which is a factor applied to the sampling rate: for example, using a fast task sampling with 250 µs between 2 samples and subsampling 3, the resulting trace has 750 µs between 2 samples.

The field *Samples* controls the length of the trace in terms of the number of sample counts per channel. The maximum number allowed depends on the trace buffer of the device and the number of channels to trace. For example, if a device can trace a total maximum of 32000 samples and 4 channels are configured to be traced, then the maximum number of samples per channel is 8000.

5.7.3.2 Triggering

The triggering functionality of the devices is visualized by the *Triggering* control on the left side of the *Scope*. Specify here whether the trace should be triggered by a certain event, or if the trace should be started instantly.
Due to its real-time nature, the triggering functionality is entirely implemented on the devices – the *Scope* only controls and visualizes the triggering configuration on the devices and does not implement any triggering logic.
A trigger event is defined by a trace signal, whose value is exceeded in a positive or negative direction.
Illustration 5.30 shows an example of a trigger event that is defined by the trace signal *Cam pos set* exceeding the value of 180 degrees in a positive direction (rising slope).

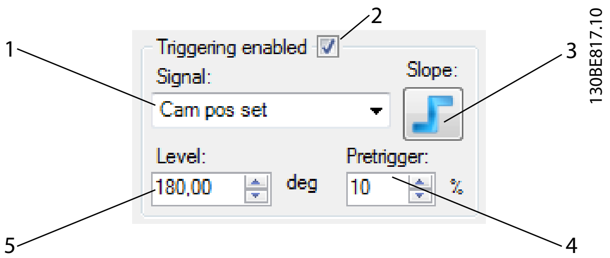


Illustration 5.30 Triggering Control

Callout	Field name	Description	Function
1	Signal	Drop-down list of trigger signals	Contains all available signals on the device and selects the signal to be used for triggering. The behavior of the *Signal* drop-down list is exactly the same as the behavior of the *Signal name* drop-down list in the *Signal chooser* control. It is not necessary to also trace the trigger signal.
2	Triggering enabled	Checkbox to enable/ disable triggering	Controls whether triggering should be used on the device. If triggering is disabled, then the trace is started instantly after activation (pressing the *Run* button). Also, all other fields of the *Triggering* control are disabled.
3	Slope	Trigger slope	Controls the direction from which the *Level* value should be reached in order for the trace to be triggered: a rising slope means that the value should be reached in a positive direction, and a falling slope means that the value should be reached in a negative direction. The default value of the button is rising.
4	Pretrigger	Pretrigger	Sets the number of samples to be traced before the trigger event occurs. The value is given as a percentage of the number of samples per channel (field *Samples*). The default value of the field is 10%.
5	Level	Trigger level	Shows the value at which the trace should be triggered (depending on the trigger slope configuration). If defined, the unit of the selected trace signal is shown on the right of the *Level* field. The trigger level field is a decimal value with 2 decimal places. The default value of the field is 0.

Table 5.8 Legend to *Illustration 5.30*

5.7.3.3 Trace Signals

The *Scope* sub-tool reflects the tracing functionality of the devices and supports tracing up to 8 signals at the same time. The channels are shown at the bottom of the *Scope* window and can be configured, enabled, or disabled. Each channel is represented by a signal chooser numbered 1–8. *Illustration 5.28* shows a trace with 3 enabled channels (1–3) and 5 disabled channels (4–8).

For online devices, the list of available signals and their definitions is automatically obtained by the ISD Toolbox from the respective device. For offline devices, the list included in the ISD Toolbox configuration is used.

The *Scope* sub-tool presents the trace signals by their names (see *chapter 9.2.3 Trace Signals* and *chapter 9.3.3 Trace Signals*). Each *Signal Chooser* shows the signal data and configures the graphical display, see *Illustration 5.31*.

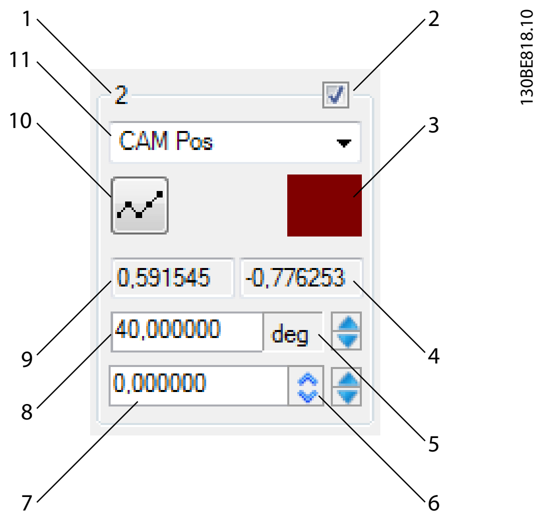


Illustration 5.31 Signal Chooser

Callout	Description	Function
1	Channel index	Shows the index of the channel that is controlled by the *Signal Chooser*. The signal channels are numbered 1–8 (see *Illustration 5.28*).
2	Channel activation checkbox	The checkbox controls whether the given channel is recorded on the device. By selecting and deselecting the channel activation checkboxes, the number of signals to be traced in the next trace is configured.
3	Channel color	Sets the color in which the traced data for the given channel is visualized. The color of a channel can be changed at any time.
4	Value at right cursor	The left and right cursor fields show the values of the trace data at the 2 cursors in the unit shown by signal unit. The 2 fields are updated dynamically while dragging. The value fields show the traced values with the precision of 6 decimal places.
5	Signal unit	The unit in which the traced data for the signal is shown.
6	Auto scale	Calculates the best scale and offset for showing the entire graph on the plot and performs them. Modifies the vertical scale and the vertical offset of this signal (see *chapter 5.7.3.8 Trace Visualization*).
7	Vertical offset	Both an input and a visualization control for the vertical offset of the signal. A vertical offset of 0 means that the vertical zero-point of the graph is located exactly in the vertical center of the plot area. The offset value is shown in the same way as the vertical scale and the values at the left and right cursors: with 6 decimal places and in the unit shown by *Signal unit*. The vertical offset can be modified in 3 ways: • Type in the desired offset in the *Vertical offset* field. • Use the up/down arrows on the right of the *Vertical offset*. • Auto-scale the trace using the *Auto scale* button (modifies the vertical scaling and the offset of this channel).

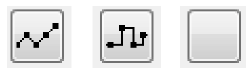
Callout	Description	Function
8	Vertical scale	Both an input and a visualization control for the units per vertical division, that is, the vertical scale of the signal. The scale value is shown in the same way as the values at the left and right cursors: with 6 decimal places and in the unit shown by *Signal unit*. The vertical scale can be modified in 3 ways: • Type in the desired scale (units per vertical division) in the field. • Use the up/down arrows on the right of the *Vertical scale* and *Signal unit* fields. • Auto-scale the trace using the *Auto scale* button (modifies the vertical scaling and the offset of this channel). The up/down arrows for controlling the vertical scale always change the scale factor by 2: clicking on the up arrow does a "vertical zoom-in", meaning that the *Vertical scale* is divided by 2; likewise, clicking on the down arrow does a "vertical zoom-out" and multiplies the *Vertical scale* by 2.
9	Value at left cursor	The left and right cursor fields show the values of the trace data at the 2 cursors in the unit shown by *Signal unit*. The 2 fields are updated dynamically while dragging. The value fields show the traced values to 6 decimal places.
10	Channel display type	3-state button that alters the graph visualization of the traced signals. The 3 possible states are: • Linear interpolation • Digital interpolation • Hidden line In linear interpolation mode, each 2 subsequent samples are connected by a straight line. This visualizes the traced data as a continuous signal. In digital interpolation mode, the traced data is visualized as a discrete signal: the value of the signal is instantly changed at each sample and remains the same until the next sample. In hidden line mode, the traced data for the channel is not visualized, although the data is available. The icon of the channel display type button changes according to the current display type.  Visualization Icons: Linear Interpolation (Left), Digital Interpolation (Middle), and Hide (Right)
11	Signal name	The drop-down list contains all available signals on the device and selects the signal to be traced on the given channel. The signals are shown in alphabetical order of their short names. When opening the drop-down list and moving the mouse cursor over a signal, the full name of the signal is shown in a tooltip.

Table 5.9 Legend to *Illustration 5.31*

5.7.3.4 Status

The *Scope* sub-tool has 5 states:

- Offline: Scope is running in offline mode. A trace cannot be initiated.
- Ready: Trace is not currently running and can be started.
- Waiting: A trace has been initiated and the Scope is waiting for a status update from the device.
- Acquiring: The device has started tracing data.
- Triggered: Triggering is enabled and the trigger point for the trace was reached.

5.7.3.5 Running a Trace

The *Scope* must be in state *Ready* in order to run a trace. Click on the *Run* button shown in *Illustration 5.33*.

5.7.3.6 Polling

When a trace is started with a triggering condition, the condition may not be met for a long time (for example, trigger on error). In this case, polling for data ready is not desired until the trace is done. Therefore, stop polling and manually check for data ready after a certain period of time.

Polling can be enabled or disabled on the *Scope* by using the tool strip button *Polling for data ready*. When polling is active, the button shows a green play icon; when polling is deactivated, the button shows a blue square stop icon (see *Illustration 5.32*). Polling is enabled by default.

Illustration 5.32 Polling for Data Ready Button States

5.7.3.7 Canceling a Trace

It is possible to stop a running trace by clicking on the *Stop* button if the *Scope* is in 1 of the states *Waiting*, *Acquiring*, or *Triggered*.

5.7.3.8 Trace Visualization

Trace data is visualized on a 2-dimensional chart. The horizontal axis represents time and the vertical axis represents signal values.

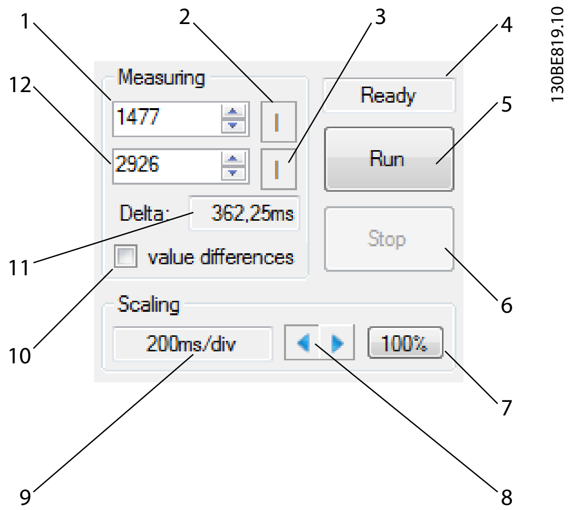
In order to easily compare and analyze the values of multiple signals at a given point in time, the data visualization for all traced signals is on the same chart, in which the signals share the same (horizontal) time axis. Each signal has its own vertical value scale and offset. It is possible to do a direct comparison between the values of 2 signals with the same unit by setting the same vertical scale and offset to both signals.

As the vertical axes of all signals are independent from each other, they can only be configured separately using the respective signal choosers. However, all traced signals share the same horizontal axis and it can therefore only be configured for all signals. The horizontal axis can be configured either numerically by using the *Scaling* control (see *Illustration 5.33*), or graphically by using the mouse to select the view range (see *Illustration 5.34*).

The main tools for evaluating sample values on a visualized trace are the 2 vertical cursors, shown as black dashed lines on the chart control (see *Illustration 5.28*). The positions of the vertical cursors can be set either numerically by using the *Measuring* control (see *Illustration 5.33*), or graphically by using the mouse to drag a cursor to the desired position. When changing the position of a cursor, the values of the fields *Value at left cursor* and *Value at right cursor* are updated for every signal to reflect the cursor positions. By clicking on the button next to the cursor position field, the corresponding cursor is set to the middle of the viewable screen. When selecting the *Value difference* display, the *Scope* automatically calculates the numerical difference between the values at the cursor positions.

There are 2 auxiliary horizontal cursors that can ease the comparison between signals; when moving a horizontal cursor, a tooltip appears next to it, showing the vertical position of the cursor for each signal (each signal has its own vertical scale and offset).

Illustration 5.33 depicts the *Measuring* and *Scaling* controls that are shown on the left side of the *Scope* sub-tool. When hovering over the *Horizontal scale* field, a tooltip appears that contains the frequency value corresponding to the ms/div value.



1	Sample index at left cursor.
2	Place left cursor in middle of view.
3	Place right cursor in middle of view.
4	Trace state.
5	Start a trace.
6	Cancel a running trace.
7	Increases/decreases the horizontal scale.
8	Horizontal scale adjustment. Adjusts the horizontal scaling so that all samples are shown within the view.
9	Horizontal scaling used in ms per division.
10	Enables/disables the display of the value differences between the cursors. The difference is shown in the *Signal Chooser* control.
11	Time delta between left and right cursor.
12	Sample index at right cursor.

Illustration 5.33 Trace Measuring and Horizontal Scaling

5

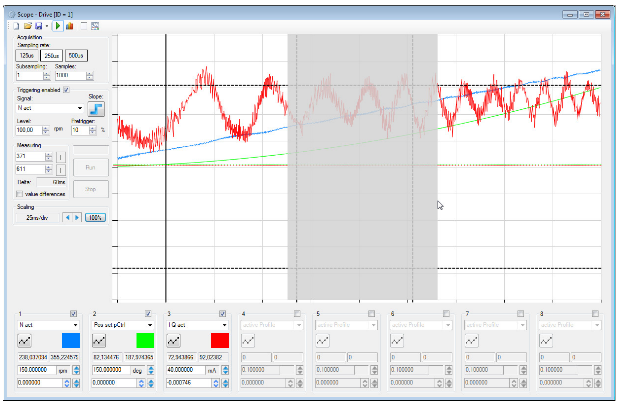


Illustration 5.34 Graphical Zoom In using the Mouse

The *Scope* can perform automatic scaling and offset of the trace signals. The scaling and offset behavior depends on the state of *Scope* in which the trace is shown. There are 3 different auto-scale modes in *Scope*:

- Zero-centred (Ctrl+1)
- Aligned and zero-centred (Ctrl+2)
- Maximized (Ctrl+3)

The *Maximized* (Ctrl+3) and *Zero-centred* modes perform the scaling and offset separately for each signal without any aligning between channels. The *Aligned and zero-centred* mode performs the same scale and offset on signals with the same unit.

The *Zero-centred* (Ctrl+1) auto-scale mode sets the vertical zero-point of each signal to be in the middle of the chart area and the global maximum of the signal to be at about 10% below the top of the chart.

The *Aligned and zero-centred* (Ctrl+2) auto-scale mode groups the channels by signal unit and performs the same scaling for every channel in a group. It sets the vertical zero-point of each signal to be in the middle of the chart area. For each group, it sets the signal with the highest amplitude to have a global maximum at most at about 10% below the top of the chart and a global minimum at least at about 10% above the bottom of the chart. All other signals in the same group are set the same scale.

The *Maximized* (Ctrl+3) auto-scale mode maximizes the usage of the *Scope* chart area by scaling each trace signal to have its global minimum at about 10% above the bottom of the chart and its global maximum at about 10% below the top of the chart. This way, each signal is zoomed to a maximum while all trace samples are within the visible vertical range.

Auto-scaling can be triggered by the following keyboard shortcuts:

- Ctrl+1 performs *Zero-centred auto-scaling*.
- Ctrl+2 performs *Aligned and zero-centred auto-scaling*.
- Ctrl+3 performs *Maximized auto-scaling*.

5.7.3.9 Saving and Loading Data

The *Scope* sub-tool can save trace data and trace settings in various ways:

- ISD trace files (.isdtrc)
    - Full measurement range
    - Cursor-selected measurement range
- Trace settings (.isdtrs)
- Pattern file (.dat)

A trace can be opened by double-clicking on the file itself (for example in Windows Explorer) or by selecting menu entry [File → Open → Scope trace] and then selecting the *.isdtrc* file.

ISD trace files (.isdtrc)
Scope can save and load trace data with the extension **.isdtrc*. The file contains the data of the recorded trace and the display information (scaling factors, offsets, and colors).

Measurement range (.isdtrc)
Sometimes, it is required to cut out unneeded data. In this case, *Scope* can save the trace data of a given range within the shown trace. To store only a subset of the trace, set the vertical cursors to the start and end of the range to be saved and then select the drop-down option *Save measurement range*. The format and information that is saved is the same as when saving the full range.

Trace settings (.isdtrs)
In order to save and restore a given trace configuration and visualization parameters, it is possible to save only these settings as an ISD trace settings file. The structure of the file is the same as the structure of the trace settings section of the *.isdtrc file* format but without the data itself. When opening an *.isdtrs* file with the *Scope*, it is automatically applied.

Pattern file (.dat)
To perform a pattern search with the drive in *Advanced CAM* mode (see *chapter 2.4.5.5 Advanced CAM*), it is necessary to first record a pre-defined pattern. Trace the signals *Pattern Sensor Act Even* and *Pattern Sensor Act Odd* and mark the pattern with the vertical cursors at the start and the end of the pattern to be found. Then save it via the drop-down option *Save Pattern As* in the drop-down menu of the *Save* button.

5.7.3.10 Online and Offline Mode

Scope is available in online and offline mode.
When operating in *Online* mode, the full *Scope* functionality is available, including configuring and performing traces on the device. *Offline* mode offers the possibility to open, view, and analyze saved traces.
When operating in *Online* mode, the *Scope* sub-tool uses the trace signal description directly from the device; when using *Offline* mode, *Scope* uses the local Toolbox configuration instead.
When going online with the device or starting *Scope* for a device that is already online, the trace signal description and sampling rates are automatically downloaded from the device and updated. Furthermore, the device is automatically checked for available trace data.
For *Offline* devices, the local ISD Toolbox configuration is used to show the sampling rates.

5.7.3.11 Reports, Document Exporting, and Printing

For documentation and printing purposes, the *Scope* offers a trace reporting functionality that generates a printable view of the trace. It also facilitates export to a PDF or spreadsheet file, and direct printing.

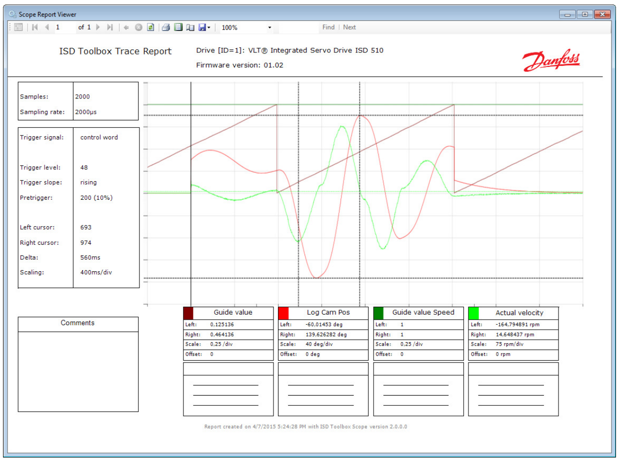


Illustration 5.35 Scope Report

5.7.3.12 Multi-device Scope

The *Multi-device scope* sub-tool itself looks similar to the single scope sub-tool, but has a list of devices inside the tool strip. While tracing, an additional window shows the status for all selected devices.
It is also possible to trace signals across different devices. Therefore every channel has an additional drop-down box where the device can be selected for each channel.
The result of all traces is shown in the same diagram (see *Illustration 5.36*).

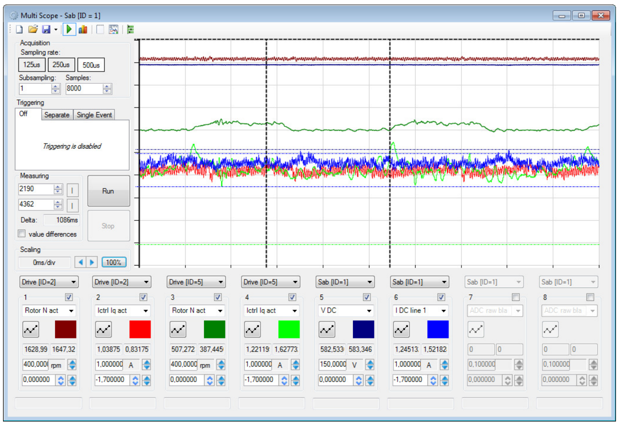
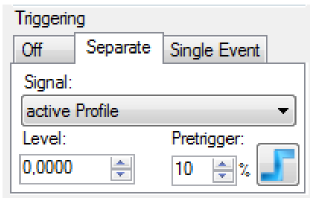


Illustration 5.36 Result of All Traces

It is possible to mix tracing across different device types, so it is possible to trace, for example, the power of an SAB together with the position of a servo drive.

There are different possibilities available for triggering (see *Illustration 5.37*):



Off	No triggering takes place. The devices involved all do an individual trace. They start immediately after the Run button has been pressed. The first sample of device 1 is not necessarily taken at exactly the same time as the first sample of device 2. There is a short delay between the devices.
Separate	All devices involved trigger individually but on the same signal. This signal must be available on all involved devices. Therefore, the *Scope* reduces the trigger signals to these signals that are available on all devices (also across device types!) that are involved in the tracing.
Single event	The trigger condition on 1 single device is configured. As soon as this event occurs, all other devices trigger too. The devices that listen to the triggering device take their first sample with a short delay as the trigger event must first be communicated. However, their traces are taken at exactly the same time.

Illustration 5.37 Multi-device Scope – Triggering

There are 2 different display modes to compensate the potential delays between the devices:

- Sample based: The first sample of every device is shown above each other.
- Time based: The samples are ordered in a timely correct way so it is possible that the trace signals do not overlap.

The display mode can be switched using the icon in the *Scope* toolbar.

5.7.4 Drive Control (Servo Drive only)

The *Drive Control* sub-tool contains the functionalities to operate the servo drives. It is only available when the ISD Toolbox is connected via direct connection and is operating as fieldbus master in cyclic mode. This sub-tool is not designed for productive use and does not replace the need for a PLC.

WARNING

UNINTENDED START

When using the drive control or working in the parameter list, the servo drive can start unintentionally which could result in death or serious injury.

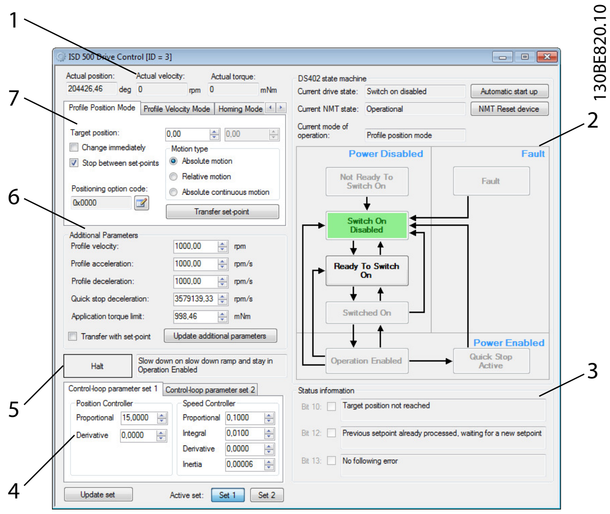
- **Ensure that nobody is in the vicinity of the servo drive(s) when working with the ISD Toolbox software.**
- **Ensure that the parameters are set in accordance with the capabilities of the machine.**

By using the *Drive Control* sub-tool, the drive can be operated in the following modes of operation:

- Profile position mode (see *chapter 2.4.1 Profile Position Mode*).
- Profile velocity mode (see *chapter 2.4.2 Profile Velocity Mode*).
- Profile torque mode (see *chapter 2.4.3 Profile Torque Mode*).
- Homing mode (see *chapter 2.4.4 Homing Mode*).
- ISD Inertia measurement mode (see *chapter 2.4.7 ISD Inertia Measurement Mode*).

The modes *Cyclic Synchronous Position Mode* and *Cyclic Synchronous Velocity Mode* are not supported by the ISD Toolbox because of the non-deterministic fieldbus behavior on a general-purpose personal computer.

The *Drive Control* sub-tool elements are shown in *Illustration 5.38* and described in the following sections:



1	Actual position, velocity, and torque
2	DS402 state machine control
3	Status information area
4	Control loop parameters
5	Halt control area
6	Additional parameters area
7	Mode of operation controls

Illustration 5.38 Drive Control Sub-tool

Position, velocity, and torque actual value

The *Actual position* (see *chapter 7.7.5 Parameter 50-03: Position Actual Value (0x6064)*), *Actual velocity* (see *chapter 7.11.3 Parameter 50-04: Velocity Actual Value (0x606C)*), and *Actual torque* value (see *chapter 7.12.5 Parameter 52-31: Torque Actual Value (0x6077)*) fields are cyclically updated from the servo drive if the ISD Toolbox is connected via cyclic communication. The fields are read-only values. The units of the values can be set in the *Options* window (see *chapter 5.5.8 Options Window*).

Mode of operation controls

The *Modes of Operation* control is a tab consisting of the supported modes of operation. Switching to a mode of operation is not done by only selecting its respective tab, but by explicitly using the mode-specific control.

Mode of operation controls – Profile Position Mode tab

The *Profile Position Mode* control is operated by setting a target position (set-point) and transferring it to the drive. Set the value of the *Target position* field using the numeric up-down buttons and click on the *Transfer set-point* button. The *Target position value* is shown in the position unit set in the *Options* window (see *chapter 5.5.8 Options Window*). *Illustration 5.39* shows the *Drive Control* sub-tool with an activated *Profile Position Mode* control. Profile velocity, acceleration, and deceleration can be set using the

Additional Parameters described in this chapter. The application torque limit must be >0 to enable movement.

Click on the *Transfer set-point* button to set the mode of operation of the servo drive to *Profile Position Mode* and transfer the set-point.

The *Change immediately* and *Stop between set-points* options are represented by the 2 checkboxes on the left of the control.

Set the desired motion type (absolute or relative) using the radio buttons *Absolute motion*, *Relative motion*, or *Absolute continuous motion* (see *Illustration 5.39*).

The options and motion type are transmitted to the servo drive together with the new set-point, when clicking on the *Transfer set-point* button.

The *Absolute continuous motion* type continuously moves between the 2 target positions that are given in the 2 numeric fields. Transfer another absolute or relative set-point to stop this sequence. The *Profile Position Mode* control relies on cyclic communication to send any commands to the servo drive.

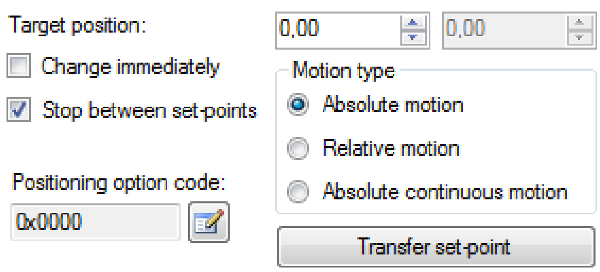


Illustration 5.39 Drive Control – Profile Position Mode

The *Positioning option code* text box shows the actual value of the positioning option code on the servo drive as a hexadecimal number with a leading *0x* for clarity (see *chapter 7.10.3 Parameter: Positioning Option Code (0x60F2)*). Click on the *Edit* button to the right of the text field to open the *Positioning Options* window (see *Illustration 5.40*).

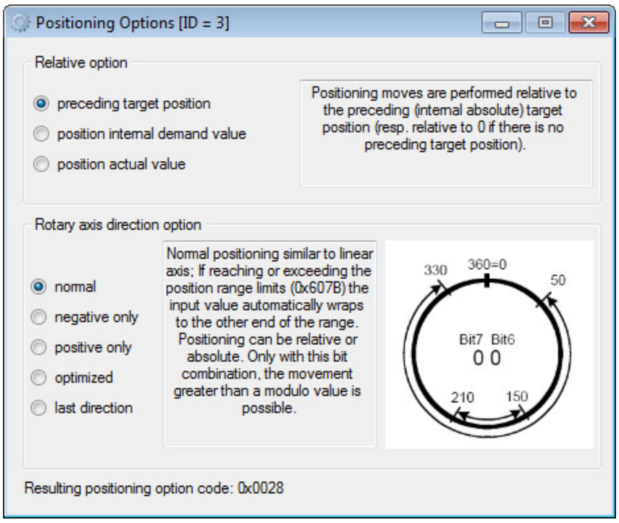


Illustration 5.40 Positioning Options Window

The *Positioning Options* window consists of the 2 options that make up the positioning option code of the servo drive:

- Relative option
- Rotary axis direction option

The possible values for each option are shown as radio buttons. When a radio button is selected, textual information on the selected value option is shown on the right of the option. For *Rotary axis direction option* the behavior of the option is also graphically visualized. The resulting positioning option code value is shown in hexadecimal format at the bottom of the window. It is updated every time an option changes. Whenever changing an option, it is immediately transmitted to the servo drive. Therefore, the *Positioning Options* window relies on cyclic communication with the servo drive.

Mode of operation controls - Profile velocity mode

The *Profile Velocity Mode* is operated by sending a target velocity to the servo drive. There are 2 methods:

- Set the value of the *Target velocity* numeric field and click on the *Transfer set-point* button.
- Use the *Target velocity* slider to set the value.

Set the *Additional Parameters* to applicable values to allow motion to take place.

The *Target velocity* value is shown in the velocity unit set in the *Options* window (see *chapter 5.5.8 Options Window*). *Illustration 5.41* shows the *Drive Control* sub-tool with an activated *Profile Velocity Mode* control.

The minimum and maximum values of both the *Target velocity* numeric field and the *Target velocity* slider are set to match the *Maximum profile velocity* parameter on the servo drive. The initial slider position is in the middle of the slider, representing *0*.

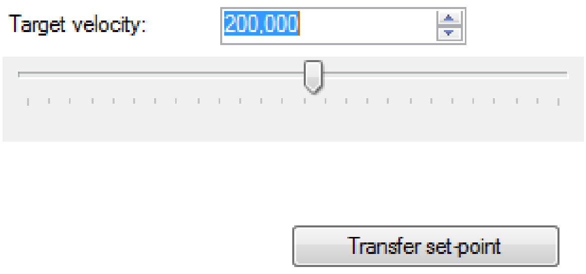


Illustration 5.41 Drive control – Profile Velocity Mode

The mode of operation of the servo drive is set to *Profile Velocity Mode* once the button *Transfer set-point* has been clicked or the *Target velocity* slider has been moved.

The *Profile Velocity Mode* control relies on cyclic communication to send any commands to the servo drive.

Mode of operation controls - Homing Mode Tab

To operate *Homing* mode, select a homing procedure and configure its parameters via the *Method* drop-down list and the parameters grid in the *Homing* mode control. *Illustration 5.42* shows the *Homing mode* control.

When the *Drive Control* is started, the supported homing modes are read from the servo drive and are filled in the *Method* drop-down list.

When selecting a homing method, its relevant parameters appear in the parameters grid and can be set before starting the homing procedure. All parameters are given in the units that are set using the *Options* window (see *chapter 5.5.8 Options Window*).

When a parameter value in the parameter grid is changed, the value is immediately transmitted to the servo drive. The homing procedure can be triggered by clicking on the *Start or continue homing* button. Set the *Additional Parameters* to applicable values to allow motion to take place. The *Homing* mode control relies on cyclic communication with the servo drive. The mode of operation of the servo drive is changed to *Homing* mode once the *Start or continue homing* button is clicked.

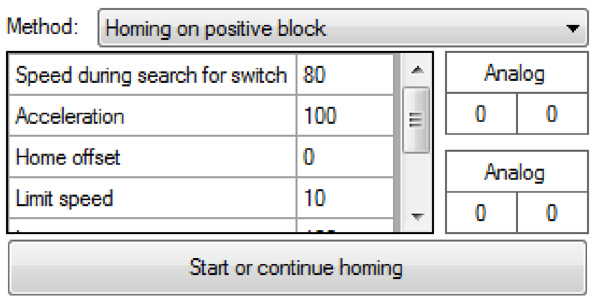


Illustration 5.42 Drive Control - Homing Mode

Mode of operation controls - Inertia Measurement Mode tab

The *ISD Inertia Measurement* mode can be used with the *Inertia Measurement* control. First set the *Max. measurement velocity* and *Acceleration torque* values according to the application requirements. Click on the *Start measurement* button in the inertia measurement control to start the measurement. *Illustration 5.43* shows the *Drive Control* sub-tool with the inertia measurement control activated.

After the measurement is completed, the measured inertia is shown in the respective field, in the unit kg m^2. When a measurement is started, it can be aborted by clicking on the button *Abort measurement*. If the inertia could not be measured, an error message is shown.

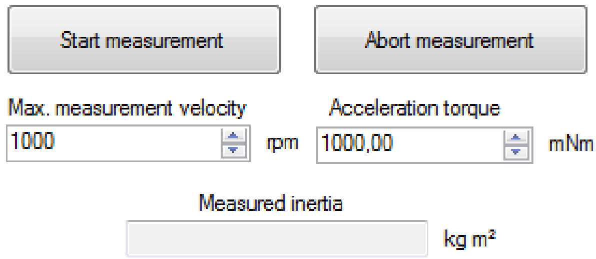


Illustration 5.43 Drive Control - ISD Inertia Measurement Mode

NOTICE

The *ISD Inertia Measurement* control is only supported in cyclic mode.

NOTICE

When the inertia measurement is carried out, ensure that the axis/mechanics connected to the drive can move freely.

Mode of operation controls - Torque Profile Mode tab

Send a target torque and the slope to the servo drive to operate the *Torque Profile*. There are 2 methods:

- Set the value of the *Target torque* field and click on the *Transfer set-point* button.
- Use the *Target torque* slider to set the value.

The *Target torque* value is shown in mNm. *Illustration 5.44* shows the *Drive Control* sub-tool with an activated *Torque Profile Mode* control.

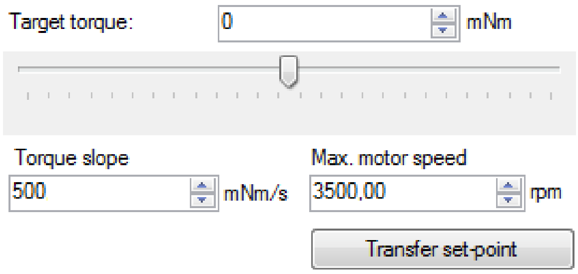


Illustration 5.44 Drive Control - Torque Profile Mode

The minimum and maximum values of both the *Target torque* numeric field and the *Target torque* slider are set to match the *Application torque limit* parameter on the servo drive, which is configurable in the field for *Additional Parameters*. The initial slider position is in the middle of the slider, representing *0*.

The mode of operation of the servo drive is set to *Profile Torque* mode once the button *Transfer set-point* is clicked or the *Target torque* slider is moved.

The parameters *Torque slope* and *Max. motor speed* are needed to operate the servo drive in *Torque Profile* mode. They are represented by the 2 respective numeric up/down fields in the *Torque Mode* control. The *Torque slope* field shows its parameter value as mNm/s, and the *Max. motor speed field* is shown in the velocity unit set in the *Options* window (see *chapter 5.5.8 Options Window*).

NOTICE

The *Torque Profile* mode control relies on cyclic communication to send any commands to the servo drive.

DS402 State Machine control

The *DS402 State Machine* control visualizes and controls the servo drive state machine and can be used to enable or disable the servo drive, and to reset an error. The control can only be used in cyclic mode (direct communication), as it sends the commands to the servo drive in the form of process data objects (PDO).

The DS402 state machine consists of the 7 DS402 states; every state is assigned a distinct color, has a list of navigable successor states, and a list of automatic transitions that can only be triggered by the drive firmware itself. *Table 5.2* in *chapter 5.5.2 Device Environment Window* lists all servo drive states along with their respective colors. The DS402 states are divided into 3 groups:

- Power disabled
- Power enabled
- Fault

The *DS402 state machine* control that is included in the *Drive control* sub-tool is shown in *Illustration 5.38*. The active state is highlighted with its defined state color. and the directly navigable successors of the active state are accessible (enabled). The states that cannot be directly entered from the current state are not accessible (disabled).

The button *Automatic Start up* is used to transfer the servo drive automatically to state *Operation Enabled*. If the state *Operation Enabled* cannot be reached within 2 s, the procedure stops automatically.

The button *NMT Reset device* resets the communication state machine of the servo drive. Afterwards, automatic traversing to the *NMT state Operation* takes place. As long as the *NMT state* of the servo drive is not *Operational*, the *DS402 state machine* control is disabled.

Additional Parameters area

The *Additional Parameters* area contains drive parameters that are needed for using the different modes of operation:

- Profile velocity (see *chapter 7.5.6 Parameter 52-12: Profile Velocity (0x6081)*)
- Profile acceleration (see *chapter 7.5.7 Parameter 50-11: Profile Acceleration (0x6083)*)
- Profile deceleration (see *chapter 7.5.8 Parameter 50-12: Profile Deceleration (0x6084)*)
- Quick stop deceleration (see *chapter 7.5.9 Parameter 50-13: Quick Stop Deceleration (0x6085)*)
- Torque limit (see *chapter 7.5.13 Parameters 52-15, 52-23, and 52-36: Application Torque Limit (0x2053)*)

The parameters are automatically read from the servo drive when the *Drive Control* sub-tool is started. The units in which the *Additional Parameters* fields are shown depend on the settings in the *Toolbox options* window (see *chapter 5.5.8 Options Window*).

The *Additional Parameters* can be transmitted to the servo drive by clicking on the button *Update additional parameters*.

If the *Transfer with set-point* checkbox is selected, the parameters are transmitted to the servo drive every time a mode-of-operation-specific set-point is sent to the servo drive (for example new *Target position* in *Profile Position Mode* or new *Target velocity* in *Profile Velocity Mode*). When the *Transfer with set-point* option is selected, the *Update additional parameters* button is disabled.

Halt control area

The *Halt* control consists of the *Halt* button and the halt option text. The *Halt* button is used to toggle between 2 possible states: pressed, or released. When the *Halt* button is pressed, the halt bit in the *Controlword* is set to *1*; otherwise, it is set to *0*. The halt option text is obtained by reading the halt option code (see *chapter 7.20.7 Parameter 50-47: Halt Option Code (0x605D)*) from the servo drive. The halt option code cannot be changed using the *Drive Control* sub-tool. Use the *Parameter List* sub-tool (see *chapter 5.7.1 Parameter List (Servo Drive and SAB)*) to change the halt option code.

Status information area

The *Status information* area shows the values of the Statusword bits 10, 12, and 13 (see *chapter 7.3.1 Parameter 16-03 Statusword (0x6041)*). Additionally, it shows the meaning of these bits, depending on the actual mode of operation (See *chapter 7.5.2 Parameter 52-01: Modes of Operation Display (0x6061)*).

For each of the 3 bits, a read-only check box (indicating if the bit value is *0* or *1*), and a read-only text box with the value meaning are shown.

NOTICE

The values of all 3 bits are obtained via the Statusword of the servo drive. Therefore, the *Status information* area depends on cyclic communication in order to show the actual servo drive status.

Control-loop parameter sets

The settings of the *Control-loop parameters* are shown and can be controlled in the *Control-loop parameter* area. The 2 *Control-loop parameter* sets on a servo drive are represented by the 2 tabs named *Control parameter set 1* and *Control parameter set 2*.

All *Control-loop parameter* fields have 4 decimal places, apart from *Speed Controller Inertia*, which has 5 decimal places.

The *Control-loop parameters* are retrieved from or transmitted to the servo drive using SDO communication.

Both parameter sets are read from the servo drive when the *Drive Control* sub-tool is started. Pressing the *Update set* button transmits the parameter set that is currently visible in the tab control to the servo drive. The active *Control-loop parameter* set is shown and can be changed by the 2 *Active Set* buttons (*Set 1* and *Set 2*).

NOTICE

By clicking on 1 of the 2 set buttons, the controlword of the servo drive is changed to match the selected set and transmitted to the servo drive as process data. Therefore, setting the active *Control-loop parameter* set relies on cyclic communication.

NOTICE

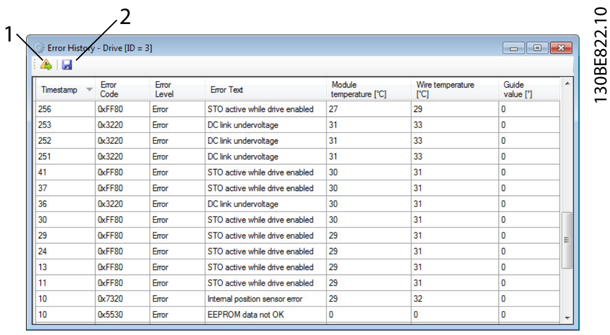
In a productive environment, add the *Control-loop parameters* in the *PLC development* environment.

5.7.5 Get Error History (Servo Drive and SAB)

The *Error History* sub-tool reads out and shows the error and warning history of the servo drive or SAB. It consists of a data grid that contains the following fields:

- ISD 510 servo drive:
  - Timestamp
  - Error Code
  - Error Level
  - Error Text
  - Module Temperature
  - Wire Temperature
  - Guide Value
- SAB
  - Timestamp
  - Error Code
  - Error Level
  - Error Text
  - Temperature Power Card
  - Temperature Control Card
  - Temperature SAB Card

The *Error History* sub-tool initially contains an empty data grid that is filled after clicking on the *Read error history* button (see *Illustration 5.45*).



Timestamp	Error Code	Error Level	Error Text	Module temperature [°C]	Wire temperature [°C]	Guide value [°]
256	0xFF80	Error	STO active while drive enabled	27	29	0
253	0x3220	Error	DC link undervoltage	31	33	0
252	0x3220	Error	DC link undervoltage	31	33	0
251	0x3220	Error	DC link undervoltage	31	33	0
41	0xFF80	Error	STO active while drive enabled	30	31	0
37	0xFF80	Error	STO active while drive enabled	30	31	0
36	0x3220	Error	DC link undervoltage	30	31	0
30	0xFF80	Error	STO active while drive enabled	30	31	0
29	0xFF80	Error	STO active while drive enabled	29	31	0
24	0xFF80	Error	STO active while drive enabled	29	31	0
13	0xFF80	Error	STO active while drive enabled	29	31	0
11	0xFF80	Error	STO active while drive enabled	29	31	0
10	0x7320	Error	Internal position sensor error	29	32	0
10	0x5530	Error	EEPROM data not OK	0	0	0

1	Read error history
2	Save error history

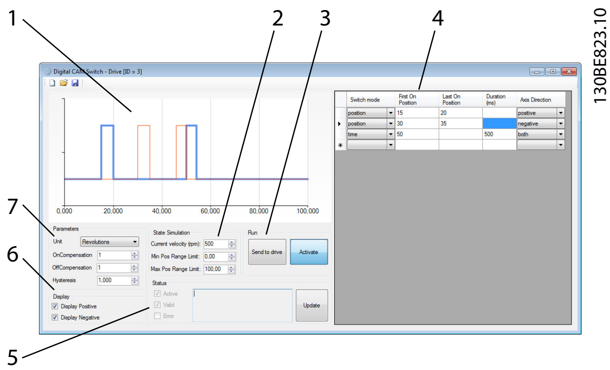
Illustration 5.45 Error History Sub-tool

The *Error History* table is read-only with entries sorted by their timestamp. It is not possible to manipulate its values.

The *Error History* can be saved to a plain-text file in a comma-separated format with a *.log* extension. The comma-separated values are ordered in the same way as in the *Error History* sub-tool. Click on the *Save error history* button to save the file. The *Save error history* button is disabled until the error history is read for the first time.

5.7.6 Digital CAM Switch (Servo Drive only)

The *Digital CAM Switch* sub-tool is used to create and visualize digital CAM switch configurations, import and export them to *.dcs* files, and transfer and activate them to ISD servo drives. *Illustration 5.46* gives an overview of the *Digital CAM Switch* sub-tool.



1	Plot area
2	Simulation controls
3	Run controls
4	Switch definitions table
5	Status area
6	Display area
7	Parameters area

Illustration 5.46 Digital CAM Switch Sub-tool

The *Parameters* area contains the global switch configuration. It consists of the parameters *Unit* (*User* or *Revolutions*), *On compensation*, *Off compensation*, and *Hysteresis*. The *On compensation* and *Off compensation* values are given in milliseconds. Depending on the selected unit, the *Hysteresis* value is interpreted either as user-defined units or as revolutions. The *Hysteresis* field contains 3 decimal places that are only evaluated if the selected unit is *Revolutions*. If the selected unit is *User*, then only the integer part of the *Hysteresis* value is taken into account.

The switch definitions table contains all switches that build up the configuration. Every switch entry consists of the fields *Switch mode* (position or time), *First On Position*, and *Axis Direction* (positive, negative, or both).

When using switch mode *position*, set the *Last On Position* parameter. When using the switch mode *time*, set the *Duration (ms)* parameter. Depending on the selected unit in the parameters area, the *First On Position* and *Last On Position* parameters are interpreted either as user-defined units (*User*) or as *Revolutions*.

The *State Simulation* and *Display* controls are used only for visualization purposes and have no effect on the digital CAM switch configuration. The horizontal plot area range is set according to the values in the simulation fields *Min Pos Range Limit* and *Max Pos Range Limit*. Depending on the field *Current velocity (rpm)*, the length of time switches in position units is calculated and visualized.

In the *Display* area, the checkboxes *Display Positive* and *Display Negative* control if the switches in positive and negative direction are visualized in the plot area.

The plot area graphically visualizes the digital CAM switch definition. The horizontal plot range is defined by the simulation parameters *Min Pos Range Limit* and *Max Pos Range Limit*. The vertical plot axis is discrete, with values indicating if the digital output is inactive (*0*) or active (*1*) at any position within the plot range, given the specified simulation velocity. Switches in positive direction are visualized by a light blue line and switches in negative range are visualized using a thin red line. For switches that are defined in both directions, both a positive and a negative visualization is made. By enabling or disabling the visualization in positive or negative direction using the checkboxes in the *Display* area, it is possible to visually isolate and observe the resulting digital output value when the servo drive is running in either a positive or negative direction.

The *Run* controls can only be used for online servo drives. The *Send to drive* button saves the digital CAM switch configuration to a temporary file and then transmits it to the servo drive using the file transfer protocol that is available on the current fieldbus. The *Activate* button sends an activation command to the servo drive.

The *Status* area can only be used for online servo drives. It shows the actual status of the digital CAM switch functionality on the servo drive. It contains the read-only checkboxes *Active*, *Valid*, and *Error*, corresponding to the active, valid, and error bits in the digital CAM switch parsing state object. If an error occurs, the corresponding error text is shown in the read-only text field in the *Status* area. The status can only be manually updated using an SDO transfer, which can be triggered by using the *Update* button.

The *Digital CAM Switch* sub-tool can save and load digital CAM switch configurations using the save and load buttons in the toolbar.

5

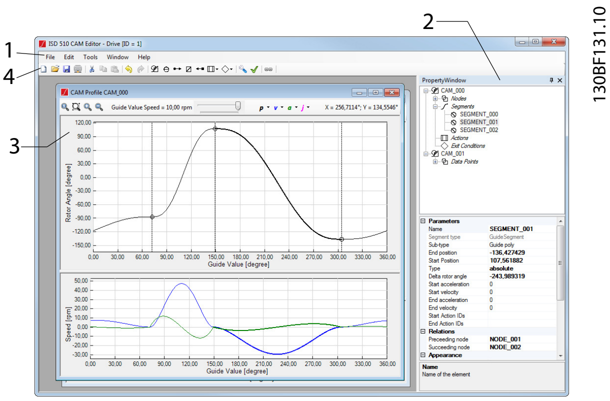
5.7.7 CAM Editor (Servo Drive only)

The ISD *CAM Editor* is a software tool for creating, editing, and visualizing CAM profiles, see *chapter 2.4.5 CAM Mode* for further information on CAM profiles. The ISD *CAM Editor* is a multiple document interface program (MDI). The parent window containing all CAM profiles is called *Main Window* and the windows containing the CAM profiles themselves are called *CAM Profile* windows.

The *CAM Editor* is implemented as an ISD Toolbox subtool, but it is not hosted as a dockable subtool inside the ISD Toolbox *Main Window*. Instead it is opened in a separate window. This allows the window to be maximized and used independently of the other ISD Toolbox subtools. In this way, it is possible to maximize the *CAM Editor* on 1 monitor and the ISD Toolbox on another for convenient editing and testing of CAM profiles.

Illustration 5.47 shows the ISD *CAM Editor* tool and its 4 main components:

- Menu bar
- Toolbar
- CAM Profile windows
- Property window



1	Menu bar
2	Property window
3	CAM Profile window
4	Toolbar

Illustration 5.47 CAM Editor

A CAM profile project can contain several CAM profiles.

5.7.7.1 Menu Bar

The menu bar organizes all general *CAM Editor* functionalities inside different groups. Shortcuts are given in brackets and characters for direct access are underlined.

File menu

The *File menu* contains entries for creating, saving, loading, and exporting CAM projects and CAM profiles. It also contains the list of recently opened *CAM Editor* projects.

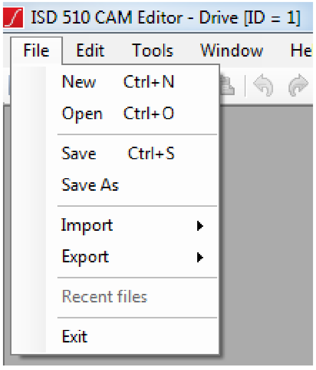


Illustration 5.48 CAM Editor File Menu

The *New* entry opens a new *CAM Editor* project. Therefore, it removes all opened CAM profiles, closes their respective CAM profile windows, and shows the *Add CAM Profile* wizard (see *chapter 5.7.7.4 Wizards*).
The *Open* entry opens a saved *CAM Editor* project. If there are any opened CAM profiles, they are closed automatically before opening the CAM project from the selected file.
The *Save* and *Save As* saves the *CAM Editor* project under the already entered file name or using a new file name.

The *Import* entry contains 2 subentries:

- *CAM profiles from other project:* CAM profiles can be imported from another *CAM Editor* project file. After selecting the project file, the specific CAM to be imported can be selected (see *Illustration 5.49*).
- *CAM profile from file:* CAM profiles can be imported from a CAM profile file (extension **.cam*). The imported profile is shown in a new CAM profile window.

The *Export* entry contains 4 subentries for exporting CAM profiles:

- *CAM profile to file:* Shows a save file dialog to select the file path to save the current CAM profile to. The current CAM profile is the one shown in the active CAM profile window.
- *CAM profile to FTP:* Shows the *Choose CAMs* dialog (see *Illustration 5.49*) to select which CAM profiles to export. For each selected CAM profile, an FTP Upload dialog (see *Illustration 5.50*) is shown for selecting the target FTP site (for example, a PLC).

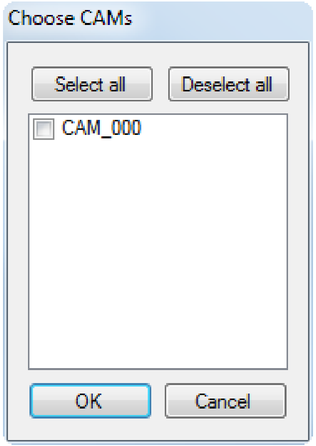


Illustration 5.49 Choose CAMs Dialog

- *Grid points:* Shows a *save file* dialog to select the file path to export the position (p), velocity (v), acceleration (a), and jerk (j) points of the current CAM profile to. The data is exported as a csv file, in which every row contains all 4 values data points (p, v, a, j). The data point delta (distance between 2 points) can be set via the form under menu [Tools → Options]. The current CAM profile is the one shown in the active CAM profile window.
- *CAM profile to drive [ID=id]:* Shows the *TFTP Upload* dialog (see *Illustration 5.51*) to transfer the current CAM profile to the drive for which the *CAM Editor* is opened. The *TFTP Upload* dialog contains a drop-down list for selecting which profile index the exported profile should be transferred to. It also contains a drop-down list for selecting the protocol for transferring the profile. The status text at the top of the dialog indicates the current transfer state.

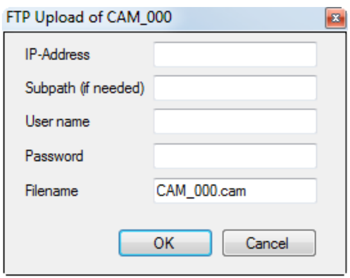


Illustration 5.50 FTP Upload Dialog

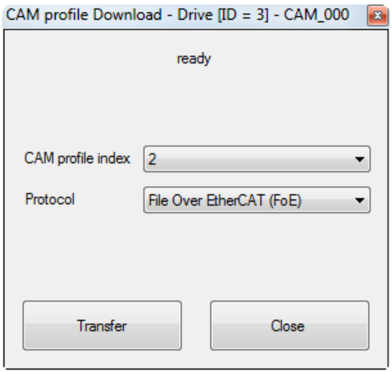


Illustration 5.51 TFTP Upload Dialog

The *Recent files* list contains the last opened or saved CAM projects and profiles.
The *Exit* menu entry closes the *CAM Editor*.

Edit menu
The *Edit* menu contains the following standard editing entries:

- Undo
- Redo
- Copy
- Paste
- Select All

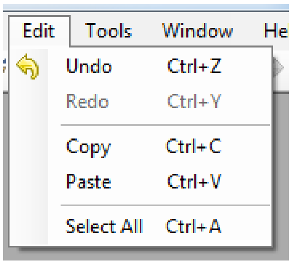


Illustration 5.52 CAM Editor Edit Menu

Tools menu
The *Tools* menu contains entries for adding CAM profile elements to a profile, and for adding comments, performing *Sanity Check*, and showing the *Options* dialog. The availability of certain entries depends on whether the basic or advanced CAM profile type is selected (see *Illustration 5.53*).

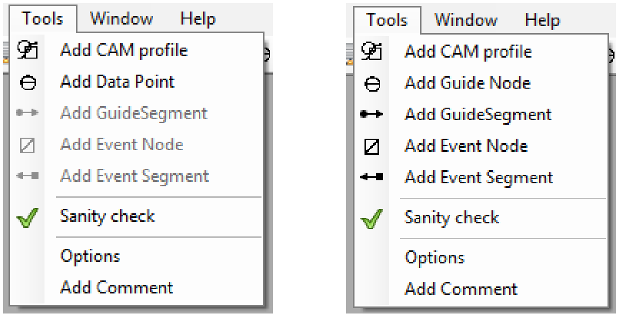


Illustration 5.53 CAM Editor Tools Menu for Basic CAM (left) and Advanced CAM (right)

All menu entries for adding CAM profile elements open wizards for creating these elements (see *chapter 5.7.7.4 Wizards*).
The *Sanity Check* entry performs a profile-specific sanity check on the current profile and shows the result in the *CAM Editor* output window.

The *Options* entry shows the *Options* dialog (see *Illustration 5.54*).
It contains:

- Color settings for the CAM profile window
- Default FTP upload settings
- AutoSave back-up interval
- Grid Point export options

The configured FTP upload settings are used as default entries for the *FTP Upload* dialog exporting a CAM profile to FTP (see *Illustration 5.50*).

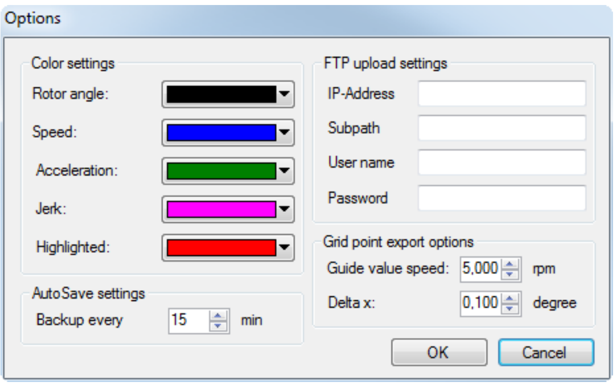


Illustration 5.54 CAM Editor Options Dialog

The *Add Comment* entry opens the *Add Comment* dialog. This contains a form with a text box and the buttons *OK* and *Cancel* to add, edit, and remove plain-text comments to/from the CAM project.

Window menu
The *Window* menu contains entries for managing the opened windows (see *Illustration 5.55*).

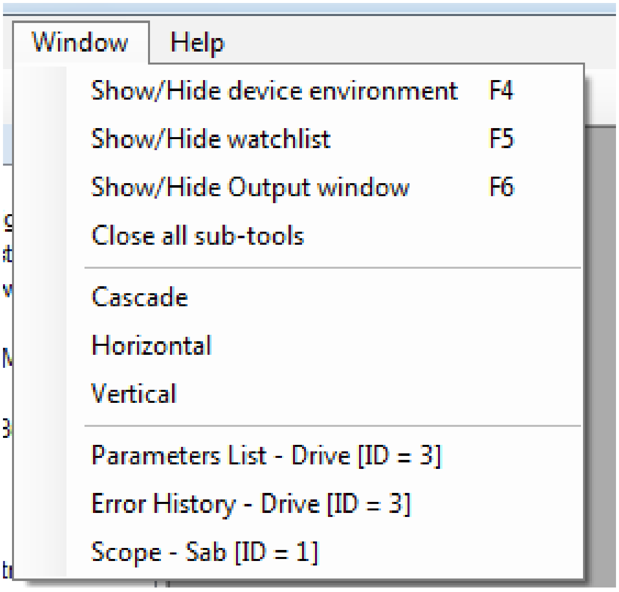


Illustration 5.55 CAM Editor Window Menu

The *Property Window* entry shows or hides the *CAM Editor Property Window*. After the *Property Window* is hidden, it is docked to the original position.

The *Adjust value ranges* entry sets all open CAM Profile windows to the same value ranges to enable easier comparison of CAM profiles.

The *Cascade*, *Horizontal*, and *Vertical* entries cascade, horizontal align, or vertical align all open windows.

All open CAM Profile windows are listed at the bottom of the *Window* menu. When a CAM profile window entry is clicked on, the associated CAM Profile window is shown on top of all other windows.

Help menu
The *Help* menu contains the entries shown in *Illustration 5.56*. The *Contents*, *Index*, and *Search* entries open the ISD Toolbox online help.
The *About* entry opens the *About CAM Editor* window, giving the *CAM Editor* version information.

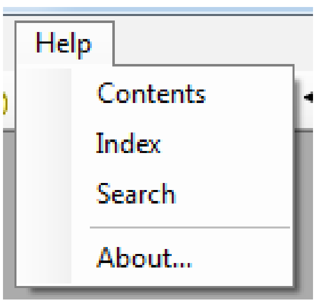
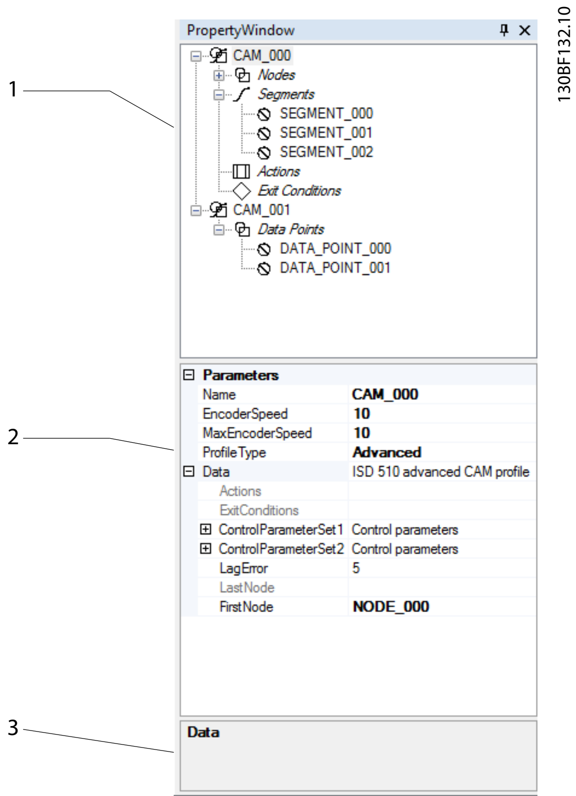


Illustration 5.56 CAM Editor Help Menu

5.7.7.2 Property Window

The *Property Window* shows all CAM profiles that are opened in the *CAM Editor* in the form of a project structure (profile tree). The properties of the selected CAM profile elements are shown below the project structure. At the bottom of the *Property Window*, the information and data area shows textual information for the currently selected CAM profile element. The *Property Window* is shown in *Illustration 5.57*.



1	Profile tree
2	Parameter list
3	Information and data area

Illustration 5.57 CAM Editor Property Window

The profile tree reflects the CAM profile structure. It consists of CAM profiles (basic and/or advanced) and their respective CAM profile elements:

- Basic CAM profile:
  - Data points
- Advanced CAM profile:
  - Nodes
  - Segments
  - Actions
  - Exit conditions

The parameter list contains all parameters of the selected CAM profile element. The properties of the selected element can be changed. The contents of the parameter list depend on the attributes of the selected CAM profile element. CAM profile elements that can be selected and edited are:

- CAM profiles
- Data points (basic CAM)
- Guide nodes and event nodes (advanced CAM)
- Guide segments and event segments (advanced CAM)
- Actions (advanced CAM)
- Exit conditions (advanced CAM)

5.7.7.3 Toolbar

The *CAM Editor* toolbar contains shortcuts for diverse editing, importing, and exporting tasks. *Illustration 5.58* shows the main menu and toolbar of the *CAM Editor* main window.

Illustration 5.58 CAM Editor Main Menu and Toolbar

The *CAM Editor* toolbar contains shortcuts to the Menu entries for easy access. All toolbar entries are named after their respective counterparts in the main menu. The functionality of all entries is the same as the respective functionality of the Menu entries.

5.7.7.4 Wizards

Add CAM Profile wizard

The *Add CAM Profile* wizard guides through creating a basic or an advanced CAM profile. It contains 4 steps and each step is on a separate screen:

1. Selecting the CAM profile name and type.
2. Selecting the 1st control parameter set.
3. Selecting the 2nd control parameter set.
4. Selecting the CAM settings.

Illustration 5.59 shows an example of the *Add CAM Profile* wizard.

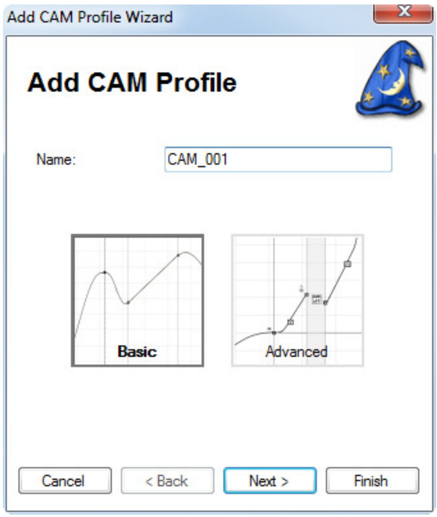


Illustration 5.59 Selecting the CAM Profile Name and Type

To select the CAM profile type, click on 1 of the buttons *Basic* or *Advanced*.
The 2nd and 3rd screens ask for values for the control parameter sets. The 4th screen asks for additional CAM settings. It is possible to deselect the optional elements in screens 2, 3, and 4.
Close the wizard by clicking on the *Finish* button.

Create Data Point wizard (basic CAM)

The *Create Data Point* wizard guides through creating a data point for a basic CAM profile. It contains 1 step, defining the position (guide value and rotor angle) and velocity of the data point. Click on *Finish* to place the data point at the specified position.
Illustration 5.60 shows an example of a "create" Wizard.

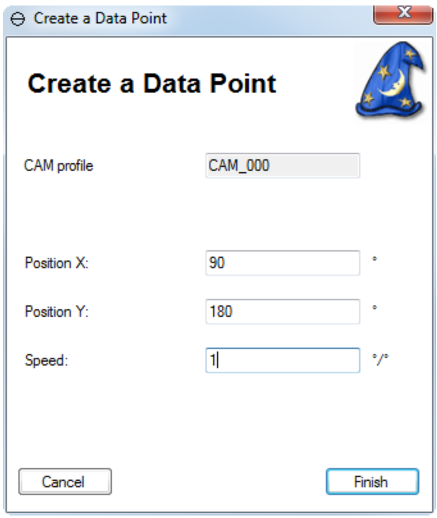


Illustration 5.60 Create Data Point Wizard

Create Guide Node wizard (advanced CAM)

The *Create Guide Node* wizard guides through creating guide nodes for advanced CAM profiles. It contains 1 step that defines the node ID and the guide value position (position X) of the guide node. Click on *Finish* to place the guide node at the specified guide value position.

Create a Guide Segment wizard (advanced CAM)

The *Create a Guide Segment* wizard guides through creating guide segments for advanced CAM profiles. This wizard handles the creation of all guide segment types. Common parameters, such as segment type, preceding and succeeding node, and the segment ID, can be selected on the 1st screen. After clicking on the *Next* button, the wizard continues depending on the selected guide segment type. For more details on those parameters, see *chapter 2.4.5.5 Advanced CAM*.

The last screen of the wizard is common for all segment types. It contains fields for setting the start action IDs, end action IDs, and for specifying whether the segment should be the default segment for its preceding node. The start action IDs must be specified as comma separated integer values.

Click on the *Finish* button to create the new segment with the selected segment type and parameters.

Create an Event Node wizard (advanced CAM)

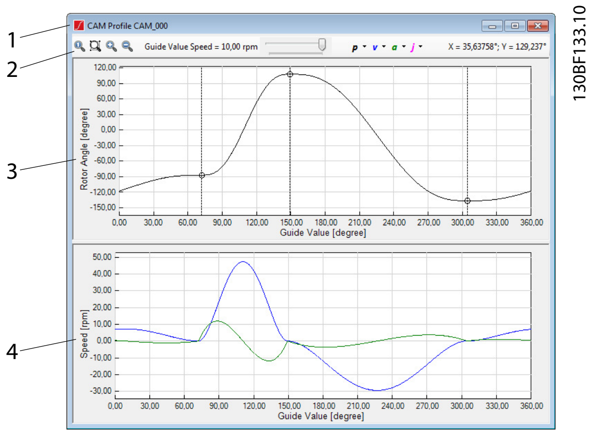
The *Create an Event Node wizard* guides the user through creating event nodes for advanced CAM profiles. It contains 1 step defining the ID, and the event segment container for the event node. Click on *Finish* to create the event node within the specified container.

Create an Event Segment wizard (advanced CAM)

The *Create an Event Segment* wizard guides the user through creating event segments for advanced CAM profiles. It contains 1 step for defining the ID, the segment subtype, the event segment container, and the preceding and succeeding event nodes of the segment. If a *Time poly* is selected, the wizard contains a 2nd step for specifying the specific properties of the *Time poly* (see *chapter 5.7.7.7 Editing Advanced CAM Profiles*).

5.7.7.5 CAM Profile Window Overview

The *CAM Profile* window is the graphical visualization and editing user interface for both basic and advanced CAM profiles. For every CAM profile, there is 1 *CAM Profile* window that visualizes all CAM elements in the profile. *Illustration 5.61* shows the *CAM Profile* window and its 4 basic elements.



1	Window title
2	Toolbar
3	Rotor angle plot area
4	Velocity, acceleration, and jerk plot area

Illustration 5.61 CAM Profile Window

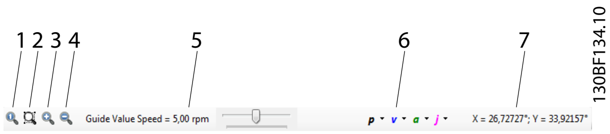
The horizontal scale and horizontal view offset of both graphs are always synchronized. This allows easy visual comparison between the rotor position and its derivatives (velocity/acceleration/jerk).

The vertical scales of the graphs are independent from each other and can be varied according to the visualization scenario.

The following sections describe the *CAM Profile* window elements.

Toolbar

The *CAM Profile* window toolbar contains visualization functionalities that do not affect the shown CAM profile itself. *Illustration 5.62* shows the *CAM Profile* window toolbar and its components.



1	Zoom to show the entire profile
2	Marquee zoom
3	Zoom in
4	Zoom out
5	Guide value speed simulation control
6	Unit selection: p = position v = velocity a = acceleration j = jerk
7	Mouse cursor position

Illustration 5.62 CAM Profile Window Toolbar

The button *Zoom to show the entire profile* calculates the best zoom to show the entire CAM profile and fill both graphs. The vertical zooms of the 2 graphs are calculated independently.

The button *Marquee zoom* enables a region of the rotor angle graph to be selected to zoom to. The marquee implementation follows the standard marquee functionality known in most existing editing programs.

The *Zoom in* and *Zoom out* buttons respectively perform zoom in and zoom out on both graphs.

The *Guide value velocity* track bar changes the guide value speed used to visualize the event segments with. The track bar can have values between 0 and the *Maximum guide value velocity* parameter of the CAM profile.

The *Unit selection* drop-down items allow changing the position, velocity, acceleration, and jerk units that are used for visualization. The *p*, *v*, *a*, and *j* symbols are shown with the color used for drawing the respective graph.

The *Mouse cursor position* area shows the value at which the mouse cursor is pointing (position/velocity/acceleration/jerk).

Rotor angle plot area

The rotor angle plot area is a 2-dimensional plot of real numbers to show the rotor angle (vertical axis) in relation to the guide value (horizontal axis). The guide value axis is given in degrees, and the rotor angle axis is given in the unit specified on the *CAM Editor* window toolbar. The horizontal axis is labeled as *Guide Value [degrees]* and the vertical axis is labelled as *Rotor Angle [unit]*, where *[unit]* denotes the user-defined position unit.

The rotor angle graph is rendered in the rotor angle color specified in the *Options* window. The default color is black. The rotor angle graph shows all CAM elements of the profile and can be used for graphical editing of basic CAM data points or advanced CAM nodes and segments.

Zoom in or zoom out is possible using the mouse wheel. While zooming, the zoomed area is centered on the point at which the mouse cursor is pointing. The maximum possible horizontal zoom out is 360°. The vertical zoom is not bounded. The horizontal zoom is always synchronized with the velocity, acceleration, and jerk graph.

By pressing the center mouse button while dragging the mouse, it is possible to move the center zoom point in order to explore the graph while zoomed in. The visible point area is between 0° and 360°.

The visual editing of CAM elements is performed by selecting and dragging elements, or by using the available context menus on the plot area. The visual editing of the CAM elements varies across the different profile types and element types (see *chapter 5.7.7.6 Editing Basic CAM Profiles* and *chapter 5.7.7.7 Editing Advanced CAM Profiles*).

It is possible to select 1 or more CAM elements via 1 of the following 2 methods:

- Left-click on the desired element. Press the [Ctrl] key on the keyboard and click on unselected or selected elements to add or remove them from the current selection.
- Use the mouse to draw a box around the CAM elements to select. To select an element, the box must entirely encase the element. To add further elements to the selection, press the [Ctrl] key on the keyboard and draw another box encasing those additional elements.

Velocity, acceleration, and jerk plot area

The velocity, acceleration, and jerk plot area is a 2-dimensional plot of real numbers to show the 3 derivatives of the rotor angle (vertical axis) in relation to the guide value (horizontal axis). All 3 graphs are visualized in a single plot area. The guide value axis is given in degrees and the velocity, acceleration, and jerk graphs are given in the units specified in the *CAM Profile* window toolbar. The horizontal axis is labeled as *Guide Value [degrees]* and the vertical axis is labelled as *Value [unit]*, where *Value* denotes the last activated graph (velocity, acceleration, or jerk) and *[unit]* denotes the user-defined position unit for the graph.

The 3 graphs are rendered in the respective color specified in the *Options* window. The default colors are:

- Velocity: blue
- Acceleration: green
- Jerk: magenta

The mouse wheel and the center mouse button have the same functionality as for the rotor angle plot area.

The plot area only visualizes the calculated velocity, acceleration, and jerk graphs. No nodes or data points are visualized in the plot area. It is not possible to select any CAM elements on the graph.

5.7.7.6 Editing Basic CAM Profiles

This section describes the specific visualization and editing functionalities for basic CAM profiles. Further information about basic CAM can be found in *chapter 2.4.5.4 Basic CAM*.

Data point and rotor movement visualization

When editing basic CAM profiles, it is possible to place, move, copy, and remove basic CAM data points from the profile. The data points are visualized as circles in the rotor angle plot area. Also, for every point, a dashed vertical line is shown at its guide value position. It is possible to select a data point by clicking on the respective circle or vertical dash line that represents it.
Illustration 5.63 shows a *CAM profile* window for a basic CAM with a selected basic CAM data point at the position (120, -10) that can be moved in any direction.

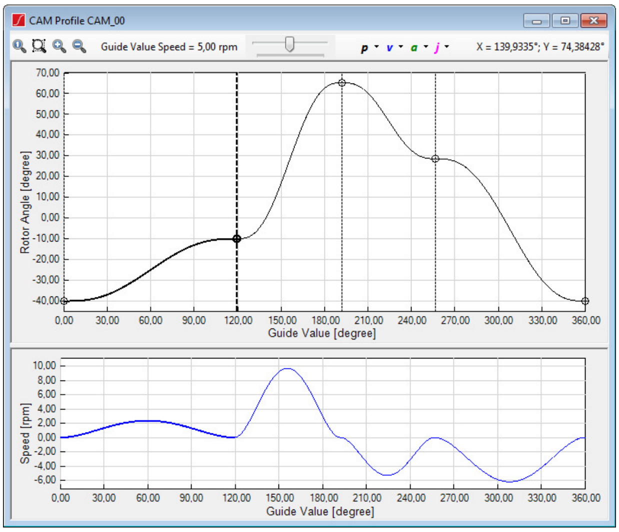


Illustration 5.63 Editing a Basic CAM Profile

It is possible to graphically move a basic CAM data point by dragging its circle. To move multiple basic CAM data points at once, select multiple points and drag 1 of them: the others are moved by the same offset as the dragged point.

The resulting positions between the basic CAM data points are visualized on the rotor angle plot area. Although the resulting graphs look like segments, they cannot be selected and are only updated after data point modifications.

A data point is always connected with its 2 neighboring data points, if existing. When moving a data point between 2 other data points, the graphs between the data points are recalculated and reconnected.

Whenever the profile is not a full profile, it can be visualized either as acyclic (profile starts with 1st data point and ends at last data point) or as cyclic (last node is virtually connected with first node).

Profile properties

Basic CAM profiles have the following properties:

- Profile Type
- Data
    - Control parameters
        - Control set 1
            - Position P
            - Position D
            - Speed P
            - Speed I
            - Speed D
            - Inertia
        - Control set 2 (same as Control set 1).
    - Following error
        - Window
        - Time
    - Scaling
        - Master scaling
            - Numerator
            - Denominator
        - Slave scaling
            - Numerator
            - Denominator
- Display
    - Guide value velocity
    - Maximum guide value velocity
    - Display as cyclic

These properties are all visible in the Property Window, when the CAM profile is selected in the profile tree. Whenever the value of a property has changed, this is reflected by the CAM profile graph.

Data point properties

The basic CAM data point has the following properties:

- Parameters:
    - Master position
    - Slave position
    - Velocity
    - Acceleration

These properties are all visible in the *Property Window*, when the basic CAM data point is selected in the profile tree or in the *CAM Profile* window. Whenever the value of a property has changed, this is reflected by the CAM profile graph.

Context menu

The context menus of the *CAM Editor* window in basic CAM mode can be used for adding and moving data points.

To create a new data point, open the context menu by right-clicking on any location of the *Rotor Angle* plot area, select the *Create Data Point entry*, and using the mouse cursor, select where to place the new data point by left-clicking on the plot area again.
Illustration 5.64 shows a scenario where there is a right-click on an empty location of the plot area, left-click on the only context-menu entry *Create Data Point*, and then left-clicking on a position to create the data point.
Illustration 5.65 shows the created data point.

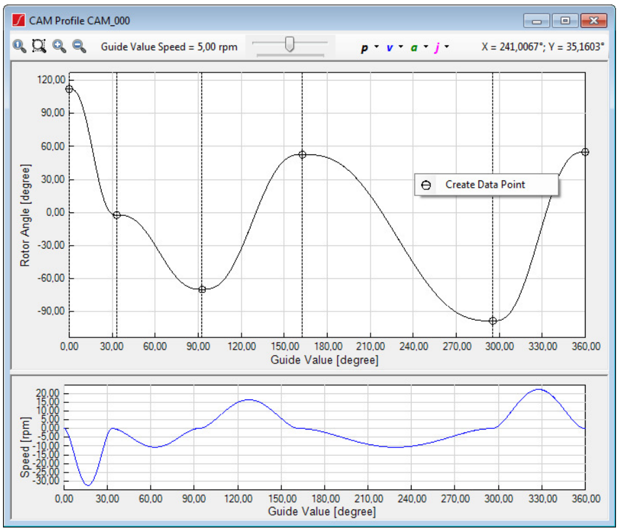
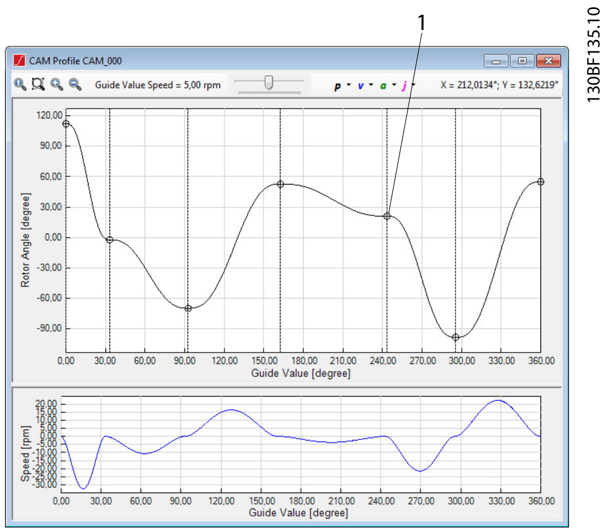


Illustration 5.64 Basic CAM Editing: Context Menu



1	Data point created

Illustration 5.65 Basic CAM Editing: Creating a Data Point

5

If 1 or more data points have been selected and there is a right-click on any location, or if there are no selected data points but a right-click on a data point, the extended basic CAM context menu is displayed, as shown in *Illustration 5.66*.

The extended basic CAM context menu contains the following elements:

- Create data point
- Move selection
    - This is only visible when multiple data points are selected.
    - It shows a window for typing in a delta X value and delta Y value for all selected points and moves the points as specified.

For each selected data point, a submenu that is named after the data point (for example, DATA_POINT_006 in *Illustration 5.66*) is shown. This contains the following entry:

- *Move…*: Has the same functionality as the *Move selection* element, however it is only applied to the selected data point.

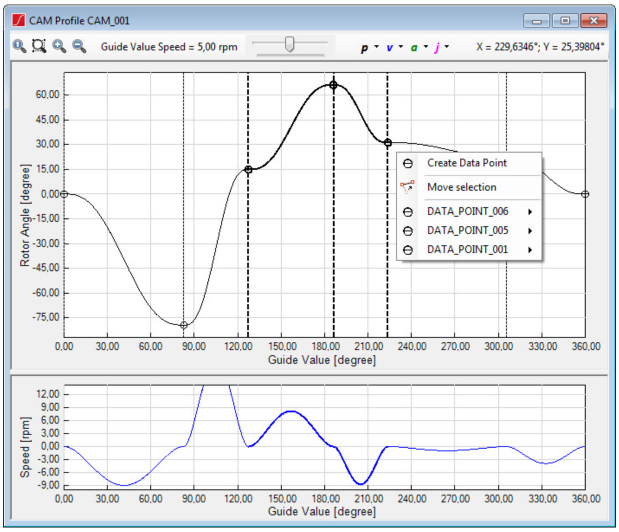


Illustration 5.66 Context Menu for Multi-Selection

Sanity check

The *Sanity check* functionality for basic CAM profiles examines if there are 2 data points with the same guide value (horizontal axis) and reports such cases.

5.7.7.7 Editing Advanced CAM Profiles

The advanced CAM visualization and editing are different to basic CAM mode. This section presents and describes the advanced CAM profile editing and visualization. Further information about advanced CAM can be found in *chapter 2.4.5.5 Advanced CAM*.

Advanced CAM guide nodes only contain guide values (master positions) and event nodes do not contain any positioning or timing information. All other data (rotor angle, velocity, acceleration, duration) is contained in the advanced CAM segments.

Node and segment visualization

An advanced CAM node is visualized as a dashed vertical line. It can be selected and moved horizontally in both directions. *Illustration 5.67* shows 3 *Guide nodes* and 3 *Guide polys* between them. The circles drawn on top of the nodes represent the edges of the *Guide poly*.

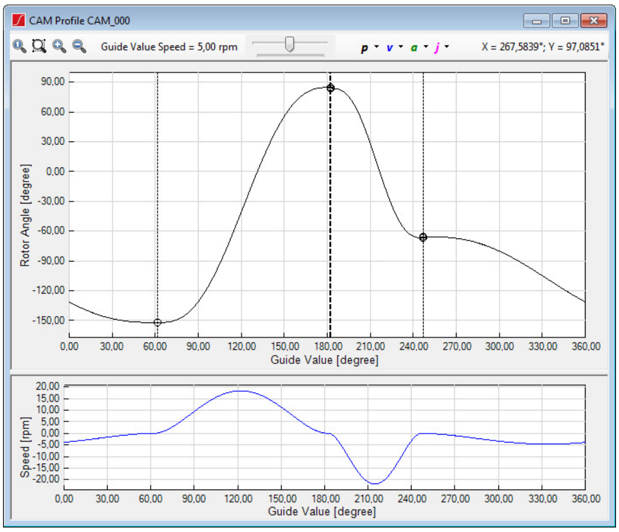


Illustration 5.67 Advanced CAM Mode: Guide Nodes and Guide Polys

Profile properties

Advanced CAM profiles have the following properties:

- Profile type
- Data
    - Control parameters
        - Control set 1
            - Position P
            - Position D
            - Speed P
            - Speed I
            - Speed D
            - Inertia
        - Control set 2 (same as control set 1).
    - Following error
        - Window
        - Time
    - Scaling
        - Master scaling

            - Numerator
            - Denominator
        - Slave scaling
            - Numerator
            - Denominator
    - Display
        - Guide value velocity
        - Maximum guide value velocity
    - First node
    - End nodes

Node properties

Advanced CAM guide nodes contain the following properties:

- Parameters:
    - ID
    - Node Type (read-only): always *Guide node*
    - Position
    - Signal
    - Actions (comma-separated action IDs)
- Relations
    - Default segment
    - Preceding segments (read-only)
    - Succeeding segments (read-only)
- Appearance
    - Visible

Advanced CAM event nodes contain the following properties:

- Parameters:
    - ID
    - Node type (read-only): always *Event node*
    - Signal
    - Actions IDs
- Relations
    - Related container (read-only)
    - Is starting node (read-only)
    - Default segment
    - Preceding segments (read-only)
    - Succeeding segments (read-only)
- Appearance
    - Visible

Context menu

The context menus when editing advanced CAM profiles are the same as the ones in basic CAM mode:

- When right-clicking on an empty area on the rotor angle plot, the entries for creating a new node or a new segment are shown.
- When right-clicking on a node or segment, entries for modification are shown.

Illustration 5.68 shows the context menu for creating a *Guide node* or a *Guide poly* that appears when right-clicking on an empty area on the rotor angle plot.

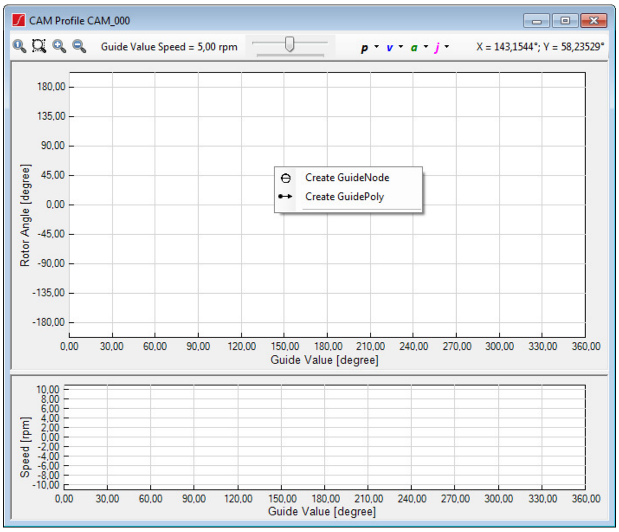


Illustration 5.68 Advanced CAM Editing: Context Menus for Creation

When using the context menu, it is possible to graphically create *Guide nodes* and *Guide polys* by clicking on the desired locations (guide value, rotor angle). The *Guide polys* can be further transformed to any other guide segment type.

Guide nodes are created in the same way as basic CAM data points. The difference between them is that *Guide nodes* only define a guide value position (x-coordinate), but do not map a rotor angle to it (y-coordinate). Therefore, only the x-coordinate of the mouse pointer is considered when graphically creating a guide node.

A *Guide poly* is graphically created by specifying its preceding and succeeding nodes, as well as its start and end rotor angles, via 1 of the following 2 methods:

- It can be created for 2 already existing guide nodes by clicking on the node at the desired rotor angle level.
- By using the left mouse button to plot the new *Guide poly* with the desired start and end rotor angles, which simultaneously creates the relevant guide nodes.

After selecting the *Create Guide Poly* item from the context menu, only 2 clicks are required to define a *Guide poly*: The

1st one for setting the preceding node and start rotor angle, and the 2nd one for setting the succeeding node and end rotor angle. To set the preceding or succeeding node, either left-click on an existing node to select it, or left-click on an empty area to create a new node and select it. In both cases, the start and end rotor angles depend on the y-coordinate of the mouse pointer, however the start and end velocity and acceleration are set to 0 as default.

It is possible to create 3 different types of *Guide poly* by using the right-mouse button when selecting the succeeding node:

- Create a 1st order *Guide poly* (P1, linear position, constant velocity, zero acceleration).
- Create a 2nd *Guide poly* with specified acceleration (P2, quadratic position, linear velocity, constant acceleration).
- Create a 2nd *Guide poly* with specified end position (P2, quadratic position, linear velocity, constant acceleration calculated from end position).

After right-clicking and selecting the desired type in the context menu (see *Illustration 5.69*), the succeeding node and end rotor angle are selected using the left mouse button.

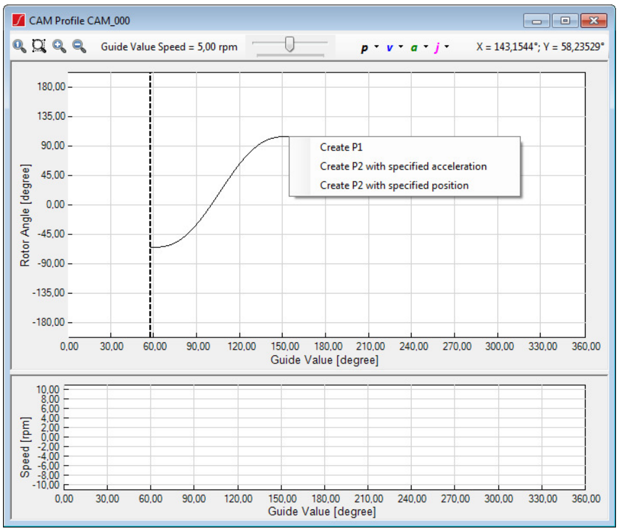


Illustration 5.69 Advanced CAM Editor: Create Special Guide Poly Types

The segment context menu offers functionalities for fast segment transformation. All functionalities contained in the segment context menu are also available in the segment properties list (see *Illustration 5.70*).
The context menu contains the entries *Preceding node* and *Succeeding node*, which are used to view and change the selected preceding and succeeding node of a segment.

It is possible to transform *Guide poly* and *Time polys* to 1st and 2nd order polynomials (P1 or P2) by using the *Transform to P1* or *Transform to P2* entry.

Using the *Default segment* entry, it is possible to specify if the segment is the default segment of its preceding node. Using the segment context menu, it is possible to change the subtype of a segment by right-clicking on a segment, opening the segment submenu, and selecting the desired target type under the entry *Change SubType* (see *Illustration 5.70*).

For guide segments, the *Change SubType to . . .* entry allows switching a segment between the following types:

- Move distance segment
- Event segment container
- Flying stop segment
- Guide poly
- Return segment

The transformation options of the guide segment context menu are shown in *Illustration 5.70*.

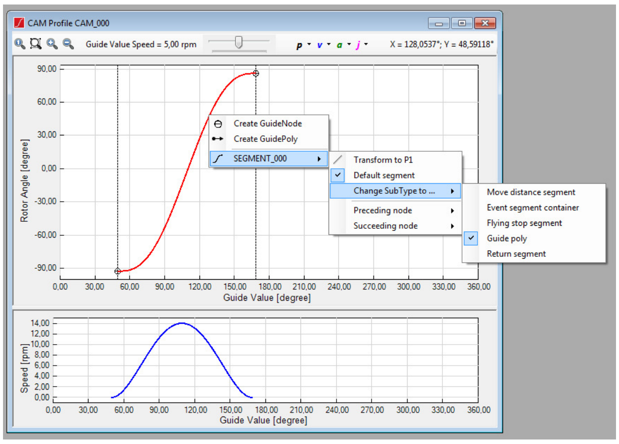


Illustration 5.70 Advanced CAM Editing: Segment Context Menu

When there is an existing *Event segment container*, the advanced *CAM Editor* context menu contains entries for creating an event node and creating a *Time poly* (see *Illustration 5.71*). The process for creating event nodes and event segments is similar to the process for creating guide nodes and guide segments. The main differences are:

- Event nodes do not contain any coordinates so the point in time they are passed fully depends on their predecessor event segments. Therefore, when selecting the option *Create event node* and placing an event node using the left mouse button, the event node is not instantly shown at the position on which was clicked. Instead, it is only shown after an event segment is set to have the event node as successor node.
- When creating a *Time poly*, its predecessor node has to already exist and a path must also exist

between the beginning of the event segment container and the selected predecessor node. This is necessary because event nodes do not have x-coordinates and therefore, the starting point of an event segment is calculated from its predecessors, which leads back to the beginning of the event segment container.

- When creating a *Time poly*, its duration parameter is set according to the end position selected and a succeeding event node is always created for the *Time poly*.

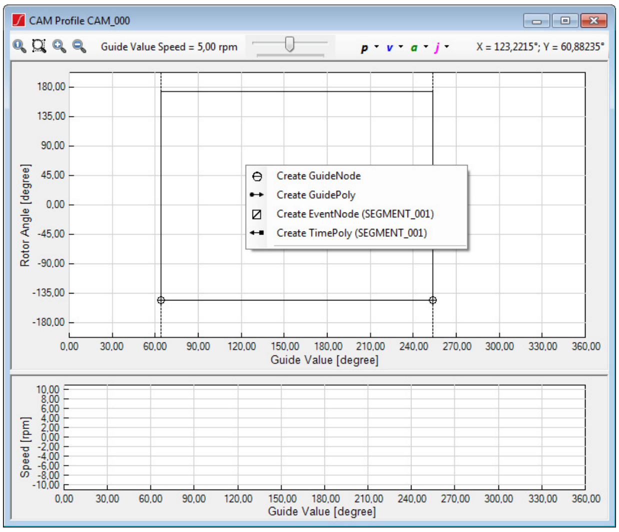


Illustration 5.71 Advanced CAM Editing: Event-Based Context Menu

For event segments, the *Change SubType to . . .* entry allows switching a segment between the following types:

- Time poly
- Velocity segment
- Torque segment
- Sync segment
- PWM off segment
- Friction segment

Illustration 5.72 presents the event segment context menu with its transformation options.

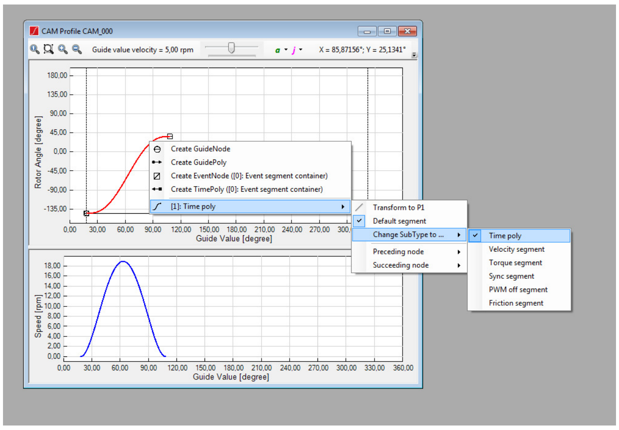


Illustration 5.72 Event Segment Context Menu

Common segment properties

This section lists the common properties of all guide segments and event segments. The properties are organized in the following property groups:

- Parameters
  - ID: Unique segment ID within the CAM profile.
  - Segment type: Read-only property for helping the user verify the segment type. The value is *Guide segment* or *Event segment*.
  - Sub-type: The sub-type of the segment (for example, *Guide poly* or *Move distance segment* for guide segments, and *Time poly* or *Velocity segment* for event segments).
  - Default: Specifies if the segment is the default segment for its preceding node.
- Relations
  - Preceding node: Contains the preceding node of the segment.
  - Succeeding node: Contains the succeeding node of the segment.
- Start
  - Start actions: Contains the list of actions that are performed at the beginning of the segment.
- End
  - End actions: Contains the list of actions that are performed at the end of the segment.
- Appearance
  - Visible: Indicates if the segment is shown inside the *CAM Profile* window.
- Information (all read-only)

- Resulting starting position: Contains the calculated starting position of the segment. For absolute segments, it equals the specified start position. For relative segments, it is calculated from the preceding segments.
- Minimum speed: Contains the calculated minimum speed that occurs inside the segment.
- Maximum speed: Contains the calculated maximum speed that occurs inside the segment.
- Minimum acceleration: Contains the calculated minimum acceleration that occurs inside the segment.
- Maximum acceleration: Contains the calculated maximum acceleration that occurs inside the segment.

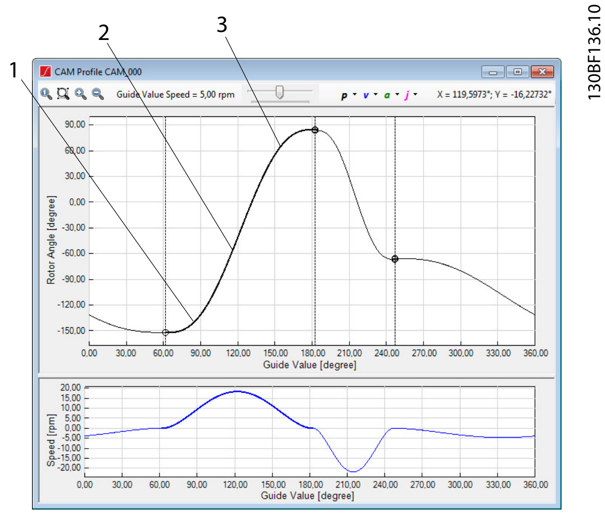
Guide poly

The *Guide poly* has the following properties:

- Parameters
    - Type: Defines if the segment is executed at an absolute slave position or relative to the previous position.
    - Start position: Specifies the axis position at the beginning of the segment. Describes the position at gear in (= motor side). If the segment type is *relative*, the start position attribute only modifies the logical CAM position. In *relative* segments, the property can be left blank (unspecified). If the start position is not specified in a *relative* segment, the logical CAM position from the previous segment is used as starting position.
    - End position: Specifies the axis position at the end of the segment. When changing this property, the value of the *Distance* property is automatically recalculated.
    - Distance: Specifies the distance between the start and end axis positions (meaning the angle to turn). Negative distance values define backward movements. When changing this property, the value of the *End position* property is automatically recalculated.
- Start
    - Start acceleration: Specifies the acceleration of the axis at the beginning of the segment. The acceleration is calculated as a ratio between the axis and the guide value (rev/rev^2). Parameterized jumps may occur in the acceleration when 2 succeeding segments have different *End acceleration* and *Start acceleration* values.
    - Start velocity: Specifies the velocity of the axis at the beginning of the segment. The velocity is calculated as a ratio between the axis and the guide value (rev/rev). The velocities of all segments that are connected by the same node should be the same to ensure smooth movement. Incorrect parameterization results in a jump in velocity.
- End
    - End acceleration: Specifies the acceleration at the end of the segment. For more information, see *Start acceleration*.
    - End velocity: Specifies the velocity at the end of the segment. For more information, see *Start velocity*.
- Information (all read-only)
    - Polynomial order: Contains the polynomial order that results from the specified start and end point, start and end velocity, and start and end acceleration.

When the position of the *Preceding node* or *Succeeding node* of a *Guide poly* is changed (graphically or numerically), the *Guide poly* is automatically recalculated and redrawn.
When any of the *Guide poly* properties are changed by using the *Property Window*, the *Guide poly* is automatically recalculated and redrawn.

It is possible to edit start position and end position of the *Guide poly* graphically by selecting it and then using vertical dragging as shown in *Illustration 5.73*.



1	By dragging the left side of the segment (near to its preceding guide node), only the start position of the segment is changed.
2	By dragging the center of the segment, both the start position and the end position of the segment are changed and the segment is thus moved up or down.
3	By dragging the right side of the segment (near to its succeeding guide node), only the end position of the segment is changed.

Illustration 5.73 Graphically Editing a Guide Poly

Move Distance segment

The *Move Distance* segment has the following properties:

- Parameters
  - Start position: Specifies the axis position at the beginning of the segment, relative to the end position of the previous segment. If the parameter is left blank (unspecified), the logical CAM position from the previous segment is used as the starting position.
- Start
  - Start acceleration: Specifies the acceleration of the axis at the beginning of the segment. It is the same as the start acceleration of *Guide poly*.
  - Start velocity: Specifies the velocity of the axis at the beginning of the segment. It is the same as the start velocity of *Guide poly*.
- End
  - End acceleration: Specifies the acceleration at the end of the segment. For more information, see *Start acceleration*.
  - End velocity: Specifies the velocity at the end of the segment. For more information, see *Start velocity*.

When the position of the preceding node or succeeding node of a move distance segment is changed (graphically or numerically), the segment is automatically recalculated and redrawn.
When any of the move distance segment properties are changed by using the *Property Window*, the segment is automatically recalculated and redrawn.

The move distance segment is calculated with the aid of the *Simulated angle* property. This can be used for testing the result of different angles sent to the drive. The default value is 0°.
The start position or simulated angle property values cannot be edited graphically. Use the *Property Window* to change them.

Return segment

The *Return segment* has the following properties:

- Parameters
  - Start position: Specifies the axis position at the beginning of the segment, relative to the end position of the previous segment. If the parameter is left blank (unspecified), the logical CAM position from the previous segment is used as the starting position.
  - Partition: Specifies the number of equivalent positions that can be used by the drive. The reference position is determined by the absolute position at the beginning of the segment and the partition. This parameter can be used for shaped plates when several equal, valid starting positions are allowed. The worst case movement is influenced by this parameter. Set the value to *0* to disable this feature.
  - Revolutions: Number of revolutions that are used when calculating valid positions, for example if there is a gear.
  - Offset: Desired end rotor position relative to the nearest physical position. The reference position is determined by the absolute position at the beginning of this segment and the partition.
- Information (all read-only)

5

- End rotor angle: Specifies the end rotor position relative to the nearest reference position.

When the position of the preceding node or succeeding node of a return segment is changed (graphically or numerically), the segment is automatically recalculated and redrawn.
When any of the return segment properties are changed using the *Property Window*, the segment is automatically recalculated and redrawn.

It is not possible to graphically edit the start position value or the specific partition, revolutions, and offset property values. Use the *Property Window* to set these parameters. The partition of the segment is shown near to its middle point.

Flying stop segment

The *Flying stop* segment has the following properties:

- Parameters
    - Start position: Specifies the axis position at the beginning of the segment, relative to the end position of the previous segment. If the parameter is left blank (unspecified), the logical CAM position from the previous segment is used as the starting position.
    - Maximum constant distance: Defines the maximum rotor angle the drive may be commanded to turn at constant speed.
    - Brake distance: Defines the rotor angle needed for braking after the constant turning phase.
    - Brake length: Guide value for the length of the deceleration phase of this segment. The segment must be long enough to run the maximum constant distance, and have enough guide value left for at least the brake length. If there is space left, the drive stays in standstill until the succeeding *Guide node* is reached.
- Start
    - Start velocity: Specifies the velocity of the axis at the beginning of the segment. It is the same as the start velocity of *Guide poly*.
- Information (all read-only)

- Brake point: Contains the calculated point (guide value, rotor angle) at which the drive starts braking.
- End rotor angle: Contains the calculated position at the end of this segment.

When the position of the *Preceding node* or *Succeeding node* of a *Flying stop* segment is changed (graphically or numerically), the segment is automatically recalculated and redrawn. The exact behavior depends on the moved node and direction:

- If the position of the preceding node is decreased (that is, the preceding node is moved to the left), the *Maximum constant distance* property is increased to match the increased segment length. In this case, the velocity, brake distance, and brake length properties are not changed. This results in a change of the resulting end position of the segment.
- If the position of the preceding node is increased (that is, the preceding node is moved to the right), there are 2 different cases:
    - If the maximum constant distance is >0, it is decreased to match the decreased segment length.
    - If the maximum constant distance is 0, the brake length is decreased to match the decreased segment length.
- If the position of the succeeding node is decreased (that is, the succeeding node is moved to the left), there are 2 different cases:
    - If the maximum constant distance is >0, it is decreased to match the decreased segment length.
    - If the maximum constant distance is 0, the brake length is decreased to match the decreased segment length.
- If the position of the succeeding node is increased (that is, the succeeding node is moved to the right), the *Maximum constant distance* property is increased to match the increased segment length. In this case, the velocity, brake distance, and brake length properties are not changed. This results in a change of the resulting end position of the segment.

When any of the *Flying stop* segment properties are changed using the *Property Window*, the segment is automatically recalculated and redrawn. The *Brake length* and *Maximum constant distance* parameters correlate and therefore affect each other when their values change.

Illustration 5.74 shows a *Flying stop* segment with length 90°, maximum constant distance 80°, brake distance 45°, and brake length 50°.

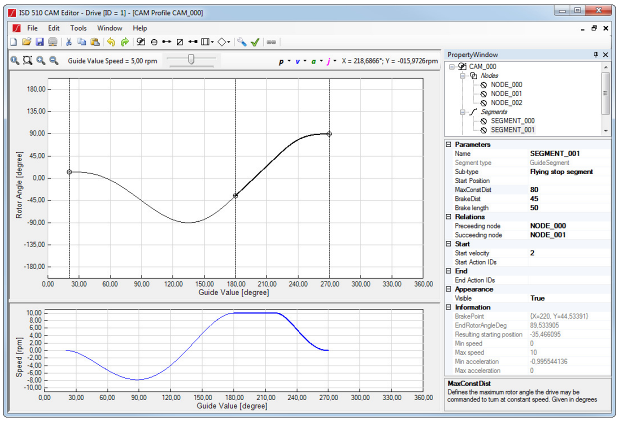


Illustration 5.74 Editing a Flying Stop Segment

Event Segment Container

The *Event Segment Container* does not contain specific properties. When changing the position of the *Preceding node* or *Succeeding node* of an *Event Segment Container* (graphically or numerically), the segment is automatically redrawn, together with all event nodes and segments it contains.

Common event segment properties

In addition to the common segment properties, there are properties for event segments. The properties listed here are common for all event segments. These properties are organized in the same groups as the base properties common to all segments.

- Parameters
    - Duration: Specifies the time given in ms from the beginning to the end of the segment.
- Relations
    - Related container (read-only): Contains the *Event Segment Container* that the event segment belongs to.
- End
    - Exit conditions: Defines if there is 1 or multiple exit conditions attached to this segment. If there are no exit conditions assigned to the segment, the duration attribute is the only exit condition. To define multiple exit conditions, list all exit IDs inside the property, separated by a comma. If there are multiple exit conditions, the segment is aborted as soon as 1 of them applies (logical OR).

Time poly

The *Time poly* has the following properties:

- Parameters
    - Type: Defines if the segment is executed at an absolute slave position or relative to the previous position.
    - Start position: Specifies the axis position at the beginning of the segment. Describes the position at gear in (= motor side). If the segment type is *relative*, the start position attribute only modifies the logical CAM position. In relative segments, the property can be left blank (unspecified). If the start position is not specified in a relative segment, the logical CAM position from the previous segment is used as the starting position.
    - End position: Specifies the axis position at the end of the segment. When changing this property, the value of the *Distance* property is automatically recalculated.
    - Distance: Specifies the distance between the start and end axis positions (meaning the angle to turn). Negative distance values define backward movements. When changing this property, the value of the *End position* property is automatically recalculated.
- Start
    - Start acceleration: Specifies the acceleration of the axis at the beginning of the segment. Parameterized jumps may occur in the acceleration when 2 succeeding segments have different *End acceleration* and *Start acceleration* values.
    - Start velocity: Specifies the velocity of the axis at the beginning of the segment. The velocities of all segments that are connected by the same node should be the same to ensure smooth movement. Incorrect parameterization results in a jump in velocity.
- End
    - End acceleration: Specifies the acceleration at the end of the segment. For more information, see *Start acceleration*.
    - End velocity: Specifies the velocity at the end of the segment. For more information, see *Start velocity*.
- Information (all read-only)

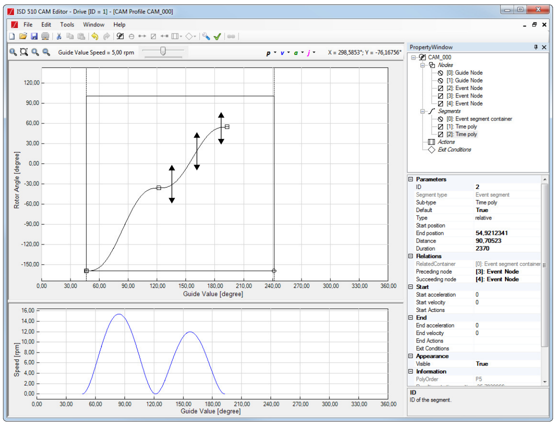
5

5

- Polynomial order: Contains the polynomial order that results from the specified start and end point, start and end velocity, and start and end acceleration.

Editing a *Time poly* is very similar to editing a *Guide poly*, with the exception that it is not possible to change the positions of the preceding or succeeding nodes. Instead, it is possible to set the duration of the *Time poly* that is used for calculating the succeeding node position, that is, for performing the P5 calculation.

It is possible to edit the *Time poly* start position, end position, or both. Do this in the same way as for the *Guide poly*; Drag the beginning, middle, or end of the segment vertically as shown in *Illustration 5.75*.



130BF137.10

Illustration 5.75 Editing a Time Poly

Velocity segment

The *Velocity segment* has the following properties:

- Parameters
    - Start position: Specifies the axis position at the beginning of the segment, relative to the end position of the previous segment. If the parameter is left blank (unspecified), the logical CAM position from the previous segment is used as the starting position.
    - Velocity: Specifies the velocity of the axis during this segment.
    - Acceleration: Specifies the acceleration of the axis when increasing the velocity. Parameterized jumps may occur in the acceleration when 2 succeeding segments have different *End acceleration* and *Start acceleration* values.
    - Deceleration: Specifies the deceleration of the axis when decreasing the velocity.
    - Torque limit: Specifies the maximum torque used during this segment. Given in mNm.

The velocity segment can only be edited using the *Property Window*. The duration of the segment is used for calculating the segment length. For fast visual identification, the letter *v* is shown in the middle of the segment. *Illustration 5.76* shows a velocity segment with a velocity of 200 RPM and a duration of 2500 ms.

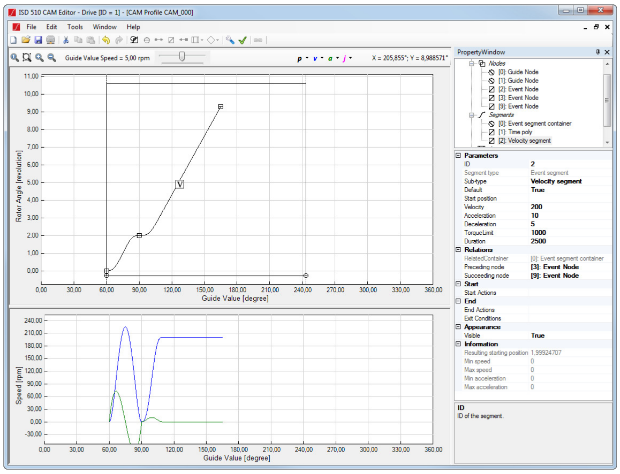


Illustration 5.76 Velocity Segment

Torque segment

The *Torque segment* has the following properties:

- Parameters
    - Start position: Specifies the axis position at the beginning of the segment, relative to the *End position* of the previous segment. If the parameter is left blank (unspecified), the logical CAM position from the previous segment is used as the starting position.
    - Torque: Specifies the target torque during this segment. The value is given in mNm.
    - Torque ramp: Specifies the rate of change of torque during this segment. The value is given in mNm per second.
    - Velocity limit: Specifies the maximum velocity used during this segment.

The torque segment can only be edited using the *Property Window*. The duration of the segment is used for calculating the segment length. The velocity limit of the segment is used for calculating and showing the velocity, and calculating the segment polynomial. For quick visual identification of the segment, the letter τ is shown in the middle of the segment. *Illustration 5.77* contains a torque segment with a velocity limit of 60 RPM and a duration of 2000 ms.

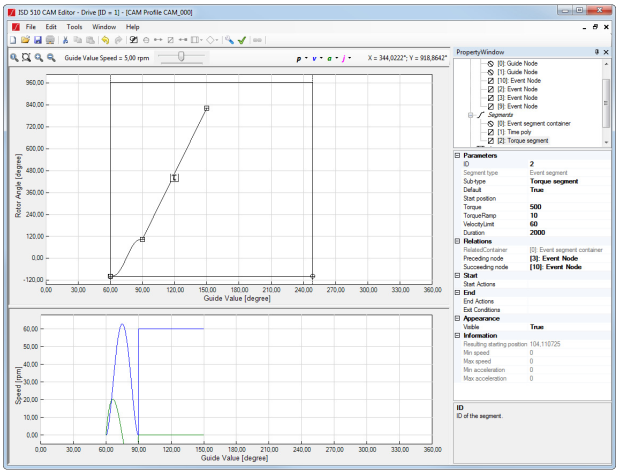


Illustration 5.77 Torque Segment

Sync segment

The *Sync segment* has the following properties:

- Parameters
    - Start position: Specifies the axis position at the beginning of the segment, relative to the *End position* of the previous segment. If the parameter is left blank (unspecified), the logical CAM position from the previous segment is used as the starting position.
    - Velocity ratio: Specifies the velocity of the axis during this segment. The velocity is calculated as a ratio between the axis and the guide value (rev/rev). The velocities of all segments that are connected by the same node should be the same to ensure smooth movement. Incorrect parameterization results in a jump in velocity.
    - Acceleration: Specifies the acceleration of the axis when increasing the velocity. Parameterized jumps may occur in the acceleration when 2 succeeding segments have different *End acceleration* and *Start acceleration* values.
    - Deceleration: Specifies the deceleration of the axis when decreasing the velocity.
    - Torque limit: Specifies the maximum torque used during this segment. The value is given in mNm.

The sync segment is shown as a *P0* polynomial (constant position, zero velocity). For fast visual identification, sync is shown in the middle of the segment (see *Illustration 5.78*).

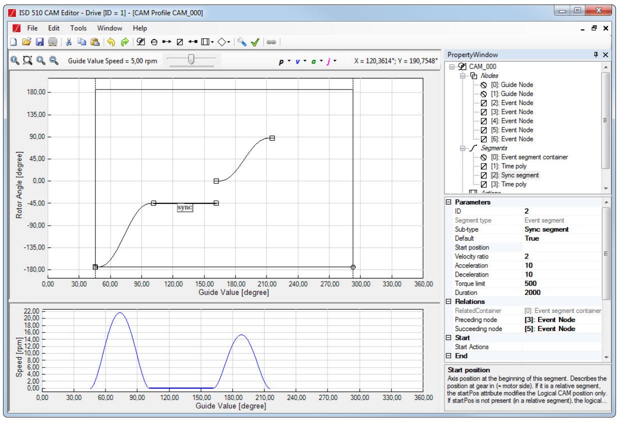


Illustration 5.78 Sync Segment

PWM off segment

The *PWM off segment* does not contain specific properties. The duration of the segment is used for calculating the segment length.
The *PWM off segment* is not shown as a curve. Instead it is denoted by a grayed-out area between its preceding and succeeding nodes. This visualization style implies the undefined behavior of the segment.

For quick visual identification, “pwm off” is shown in the middle of the segment (see *Illustration 5.79*).

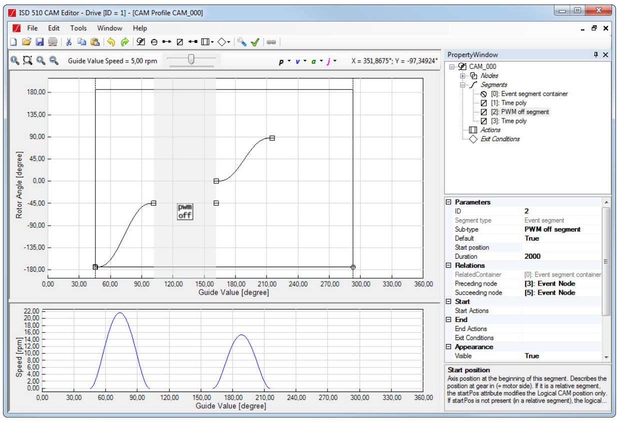


Illustration 5.79 PWM Off Segment

Friction segment

The *Friction segment* has the following properties:

- Parameters
    - Start position: Specifies the axis position at the beginning of the segment, relative to the end position of the previous segment. If the parameter is left blank (unspecified), the logical CAM position from the previous segment is used as the starting position.
    - Velocity high: Velocity of the axis during this segment.

- Velocity low: Velocity of the axis during this segment.
- Acceleration: Specifies the acceleration of the axis when increasing the velocity. Parameterized jumps may occur in the acceleration when 2 succeeding segments have different *End acceleration* and *Start acceleration* values.
- Deceleration: Specifies the deceleration of the axis when decreasing the velocity.
- Do compensation: Specifies if friction compensation should take place. If *True* is selected, the measured friction is compensated automatically by the servo drive. If *False* is selected, the value can be used for diagnostics.
- Guide value: Specifies the guide value offset for starting the measurement.
- Timeout: Specifies the timeout in ms for reaching the guide value offset and starting the measurement.

The friction segment can only be edited using the *Property Window*. The duration of the segment is used for calculating the segment length. The highest value of *Velocity low* and *Velocity high* is used for calculating the segment polynomial. For fast visual identification, F_t is shown in the middle of the segment. *Illustration 5.80* shows a friction segment with the following settings:

- Velocity high: 120 RPM
- Velocity low: 60 RPM
- Duration: 2075 ms

As the *Velocity high* value is higher than the *Velocity low* value, it is used for calculating and visualizing the segment.

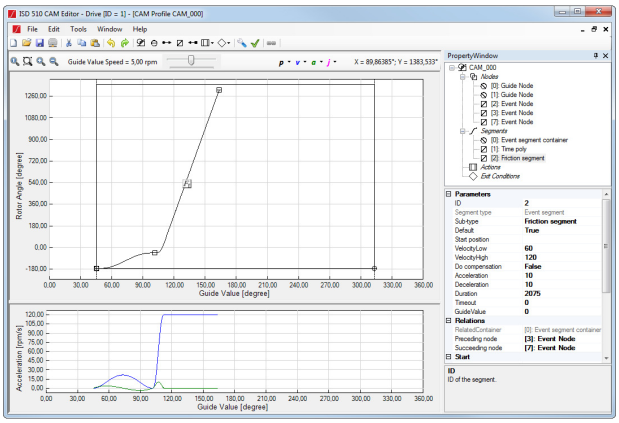


Illustration 5.80 Friction Segment

Event segment path visualization

The event nodes do not have fixed guide value positions. They are calculated by the preceding segments. Whenever an *Event node* has 2 succeeding event segments, either 1 of those may be activated when a CAM profile is activated. This in turn affects the positions of all following event nodes and segments. This behavior may result in complex profiles that are difficult to calculate and visualize. To help create complex event-based behavior, the advanced *CAM Editor* computes an active segment path by using the default segments for all nodes.

The default segment path is iteratively computed as follows:

- The 1st event node of the parent *Event segment container* is selected and its default segment is marked as active.
- In the marked segment, the default segment of the succeeding event node is marked as active.

All iterated nodes and marked segments are shown in the visualized path and are visualized as black curves. Segments and nodes that are not marked as active are visualized as gray curves (see *Illustration 5.81*).

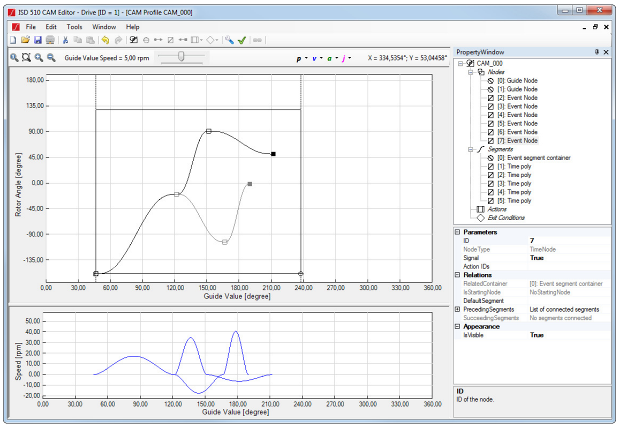


Illustration 5.81 Visualization of 2 Paths within an Event Segment Container

A change to a default segment of an event node results in a modification of the entire active path shown. This behavior is shown in *Illustration 5.82* and *Illustration 5.83* where the default segment of the 2nd node is switched. This triggers a new computation of the following path and a relocation of all succeeding elements due to the different duration and end position of the newly selected default segment.

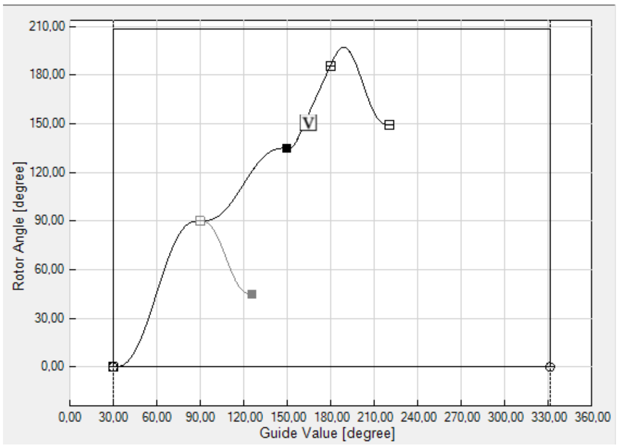


Illustration 5.82 Following Event Segments Depending on the Selected Path 1

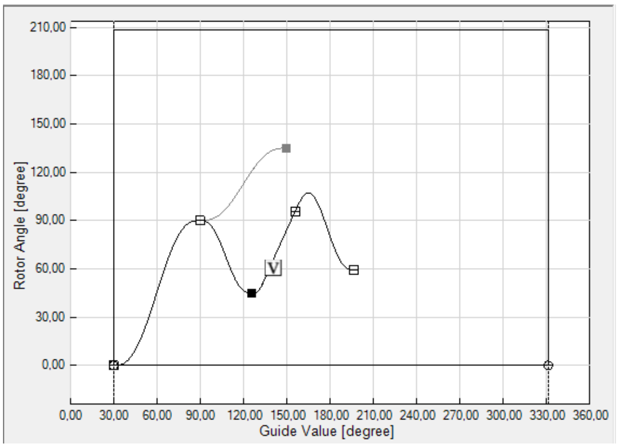


Illustration 5.83 Following Event Segments Depending on the Selected Path 2

Sanity check

The *Sanity check* for advanced CAM profiles helps to identify potential problems and risks in a profile. The sanity check identifies and shows problems in 2 categories:

- Warning: Shown to create awareness of certain behavior.
- Error: Prevent the export of the CAM profile.

The following situations result in a warning:

- Jump in position: Shown whenever 2 succeeding segments, connected by a node, define different *End position* and *Start position*.
- Jump in velocity: Shown whenever 2 succeeding segments, connected by a node, define different *End velocity* and *Start velocity*.
- Jump in acceleration: Shown whenever 2 succeeding segments, connected by a node, define different *End acceleration* and *Start acceleration*.

The following situations result in an error:

- No data or incomplete data: <2 nodes or <1 segment have been specified in the CAM profile.
- No starting node: No guide node with ID *0* is specified in the CAM profile.

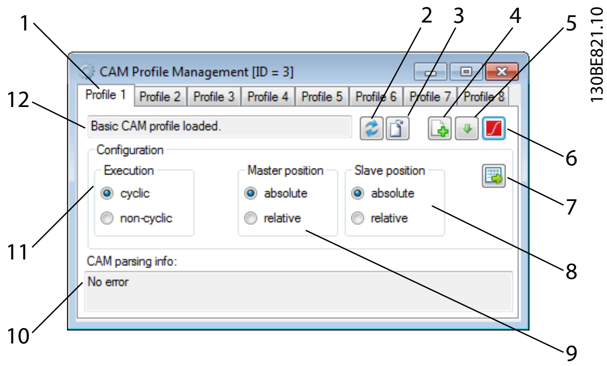
5.7.7.8 Standalone Emulation of the CAM Editor

The ISD Toolbox is loaded and the *CAM Editor* is automatically shown and maximized when a *.cam* or a *.camproj* is double-clicked in Microsoft® Windows.

An offline drive device is automatically added to the device environment of the ISD Toolbox, rather than attempting to connect to an existing physical device. Therefore, it cannot be used to transfer CAM profiles to a servo drive.

5.7.8 CAM Profile Management

There are multiple tasks for managing CAM profiles on the servo drive: sending CAM profile files to the servo drive, triggering the parsing procedure in the servo drive, and configuring the profiles. These tasks are done by using the sub-tool *CAM Profile Management*. The *CAM Profile Management* sub-tool contains a tab control with a separate tab for every CAM profile slot on the servo drive (1–8), see *Illustration 5.84*.



1	Profile tabs (1–8)	Provides information on the profiles.
2	Refresh button	Reads the state for the selected profile from the servo drive and updates the user interface.
3	Parse button	Triggers the parsing of the selected profile on the servo drive.
4	Send File button	Shows an *Open File* window for selecting a *.cam* file and transmits it to the servo drive. The parsing of the profile is not started automatically and must be triggered by using the *Parse* button.

5

5	Get File from Drive button	If a *.cam* file exists for the selected profile, this function transferss it from the servo drive and shows a *Save File* window to select the location to save the file to.
6	Open in CAM Editor button	If a *.cam* file exists for the selected profile, this function transfers it from the servo drive to a temporary location and opens it with the *CAM Editor*.
7	Send Configu-ration button	Sends the configuration (*Execution*, *Master Position*, and *Slave Position*) to the servo drive.
8	Slave Position configu-ration	Contains 2 radio buttons that specify if the selected profile should be executed with *slave absolute* or *slave relative* position. When the sub-tool is opened, the option is read from the servo drive and the radio buttons are selected accordingly.
9	Master Position configu-ration	Contains 2 radio buttons that specify if the selected profile should be executed with *master absolute* or *master relative* position. When the sub-tool is opened, the option is read from the servo drive and the radio buttons are selected accordingly.
10	CAM Parsing Info	Visualizes the download info and parsing info for the selected profile index: • After a download procedure, shows its result. • After a parse procedure on the servo drive, shows text corresponding to the parsing state status code on the servo drive. • When clicking on the *Refresh* button, the ISD Toolbox reads the parsing state status code from the servo drive and updates the text. • When reading the CAM profile from the servo drive and saving it as a *.cam* file, shows the location of the saved file.
11	Execution area	Contains 2 radio buttons that specify if the selected profile should be executed in cyclic or non-cyclic mode.
12	Loading info	Text field showing the loading state for the selected profile.

Illustration 5.84 CAM Profile Management

5.7.9 Touch Probe (Servo Drive only)

The *Touch Probe* sub-tool configures the settings of the touch probe functionality of the servo drive. There are 2 touch probe channels available that can be used in parallel. See *chapter 2.5.2 ISD Touch Probe* for more information on the touch probe functionality.

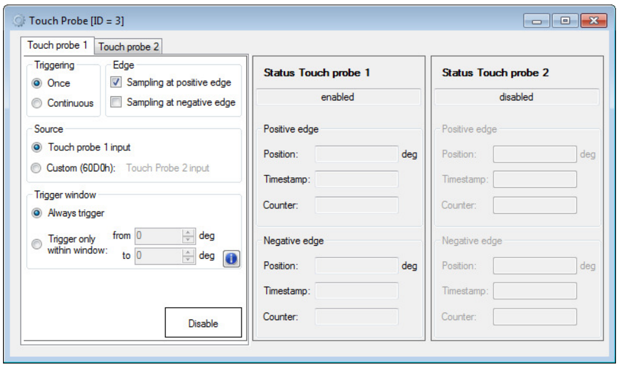


Illustration 5.85 Touch Probe Sub-tool

5.7.10 SAB Control (SAB only)

The *SAB Control* sub-tool visualizes and controls the SAB state machine and therefore can be used to enable or disable the UDC and U_{AUX} voltage on the lines, and to reset an error. The sub-tool can only be used in cyclic mode (direct communication) as it sends the commands to the SAB in the form of process data objects (PDO).

The *SAB Control* consists of the 6 SAB states; every state is assigned a distinct color and has a list of navigable successor states and a list of automatic transitions that can only be triggered by the SAB firmware itself. The SAB states, along with their respective colors, are described in *chapter 5.5.2 Device Environment Window*.

Illustration 5.86 depicts the *SAB Control* sub-tool containing the defined SAB states. The active state is highlighted with its defined state color. The directly navigable successors of the active state are accessible (enabled) but the others are not accessible (disabled).

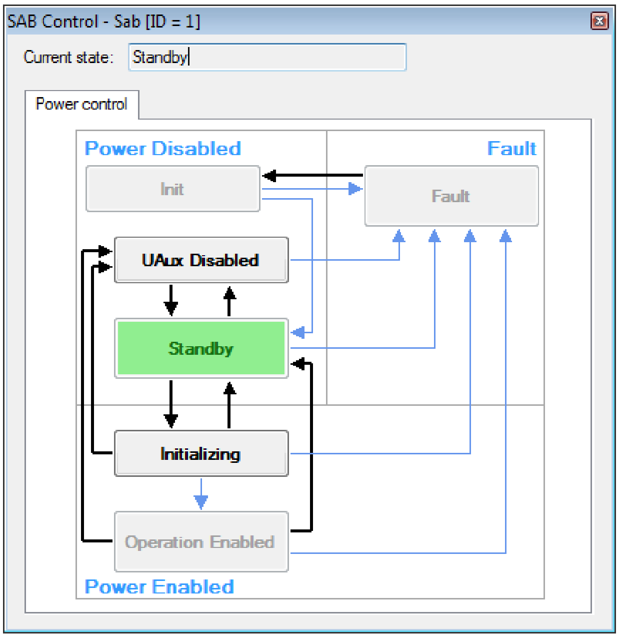


Illustration 5.86 SAB Control

At the top of the *SAB Control* window, a text field shows the current state of the SAB.

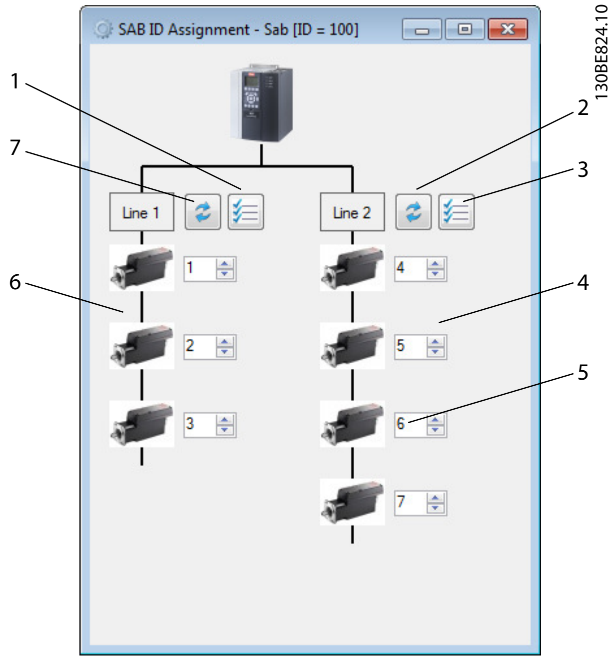
The available *manual* transitions to navigable successors are shown as black arrows and the possible *automatic* transitions are visualized as thin light blue arrows.

In order to transition from a state to a successor, click on the desired successor state. Only valid successors are accessible in the *SAB Control* window, preventing an invalid transition command being sent to the SAB.

5.7.11 SAB ID Assignment via Ethernet POWERLINK® (SAB only)

The *SAB ID Assignment* sub-tool controls the Ethernet POWERLINK®-specific SAB functionality to set the IDs of the servo drives on the network and to visualize the topology of the network. ID assignment is not required for EtherCAT® (see *chapter 6.1.1 EtherCAT®*).

The *SAB ID Assignment* sub-tool graphically shows the SAB and its 2 lines. Initially, both lines are empty. After clicking on the *Refresh* button for 1 of the 2 lines, the connected drives and their corresponding IDs on the line are determined and shown. The lines are shown vertically and each servo drive is shown together with its Ethernet POWERLINK® ID.
Illustration 5.87 visualizes the *SAB ID Assignment* sub-tool, showing a topology with 3 drives on line 1 and 4 drives on line 2.



1	Auto-assign line 1
2	Refresh line 2
3	Auto-assign line 2
4	Servo drives on line 2
5	Servo drive ID
6	Servo drives on line 1
7	Refresh line 1

Illustration 5.87 SAB ID Assignment

Each servo drive is visualized with a corresponding drive image and a numeric field showing its ID.

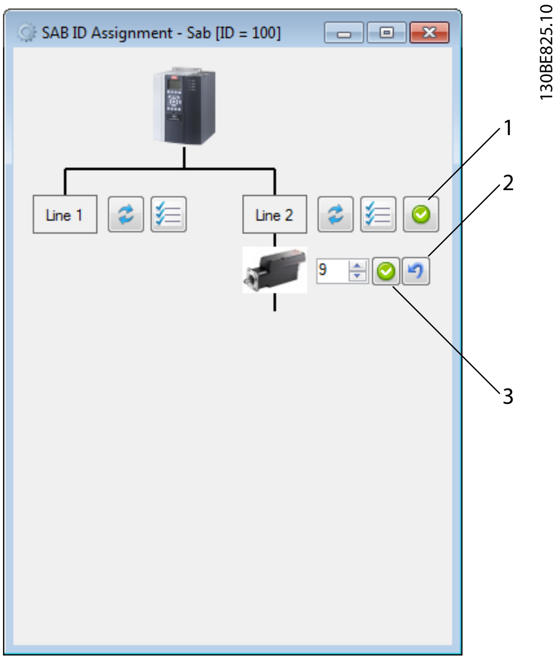
The *Auto-assign line* buttons for both lines perform an automatic assignment of all servo drive IDs on the respective line. Clicking on an *Auto-assign* button opens a separate window that prompts for the starting ID. Afterwards the automatic ID assignment procedure is initiated.

It is also possible to set the device IDs of specific servo drives on any line manually by setting the desired ID in the text field next to each servo drive.

When changing the value of a drive ID field, 2 additional buttons appear next to the ID field (see *Illustration 5.88*). These are:

- *Apply changed ID*: Applies the set ID on the servo drive.
- *Revert to changed ID*: Undoes the changes to the ID field and reverts the value back to the actual ID of the servo drive.

If at least 1 ID field is manually changed on a line, then the button *Apply all changes* appears next to the respective Auto-assign line button. This button applies all ID changes to the line.



1	Apply all changes to line 2
2	Revert to original ID
3	Apply changed ID

Illustration 5.88 SAB ID Assignment - Change Drive ID

NOTICE

After an ID is applied using the *SAB ID Assignment* sub-tool, the ID is instantly set on the servo drive. If the servo drive is already added to the *Device Environment* using its old ID, remove it and add it again with its new ID.

6 Programming

6.1 ID Assignment

6.1.1 EtherCAT®

EtherCAT® needs no special ID assignment (IP address). Special ID assignment is only required when using indirect communication via the ISD Toolbox software (see *chapter 5.3 ISD Toolbox Communication* for further information).

6.1.2 Ethernet POWERLINK®

6.1.2.1 Single Device ID Assignment

When assigning an ID to a single device, use the *Device Information* window in the ISD Toolbox (see *chapter 5.5.2.1 Device Information Window*). Setting an ID to a device can also be done via the LCP (see *chapter 4 Local Control Panel (LCP) Operation*).

Setting the Node ID directly on a servo drive or on the SAB
All IP-related parameters are located in *parameter group 12-0* IP Settings*. According to the Ethernet POWERLINK® standard, the IP address is fixed to 192.168.100.xxx. The last number is the value in *parameter 12-60 Node ID*. For *parameter 12-02 Subnet Mask*, the IP address is fixed to 255.255.255.0 and cannot be changed.

Attach the LCP to the servo drive or SAB for which the *Node ID* should be changed. Change the value in *parameter 12-60 Node ID* to select the desired IP address.

Setting the Node ID for a single servo drive via the SAB
It is also possible to change the *Node ID* of a servo drive when the LCP is connected to the SAB. This functionality is contained in *parameter group 54-** ID Assignment* on the SAB in sub-group *54-1* Manual*.

1. Attach the LCP to the SAB that is connected to the servo drive for which the *Node ID* should be changed.
2. Configure the parameters:
    - 2a *54-10 EPL ID assignment line*
    - 2b *54-11 Drive index* (position of the servo drive in the line)
    - 2c *51-12 EPL ID assignment assign ID*
3. Set *parameter 54-13 EPL ID assignment start* to *[1] start*.

6.1.2.2 Multiple Device ID Assignment

When assigning IDs to several devices (for example, when setting up a new network), use the ISD Toolbox subtool *SAB ID assignment* (see *chapter 5.7.11 SAB ID Assignment via Ethernet POWERLINK® (SAB only)*). Setting the IDs of all the servo drives connected to an SAB at the same time can also be done via the LCP when it is connected to the SAB (see *chapter 3 Servo Access Box (SAB) Operation*).

Setting the Node IDs of all servo drives on an SAB line
The automatic SAB ID assignment is used for automatically setting the *Node IDs* on all servo drives for a specified SAB line. This functionality is contained in *parameter group 54-** ID Assignment* on the SAB in sub-group *54-0* Automatic*.

1. Attach the LCP to the SAB that is connected to the servo drives for which the *Node ID* should be changed.
2. Configure the parameters:
    - 2a *54-02 EPL ID assignment line*
    - 2b *54-03 EPL ID assignment start ID*
3. Set *parameter 54-04 EPL ID assignment start* to *[1] start*.

6.2 Basic Programming

The libraries provided for the ISD 510 servo system can be used in TwinCAT® V2 and in the Automation Studio™ (Version 3.0.90 and 4.x, supported platform SG4) environment to easily integrate the functionality without the need of special motion runtime on the controller. The provided function blocks conform to the PLCopen® standard. Knowledge of the underlying fieldbus communication and/or the CANopen® CiA DS 402 profile is not necessary.

The library contains:

- Function blocks for controlling and monitoring the servo drive and the SAB.
- Function blocks for all available motion commands of the servo drive.
- Function blocks and structures for creating *Basic CAM* profiles.
- Function blocks and structures for creating *Labeling CAM* profiles.

6

6

6.3 TwinCAT®

6.3.1 Programming with TwinCAT®

6.3.1.1 ISD Deliverables

To integrate the servo drive and the SAB into a TwinCAT® project, the following files are required:

- Library for the ISD 510 servo system: *Danfoss_VLT_ISD_510.lib*
- ESI file (EtherCAT® Slave Information) for the ISD 510 servo drive: *Danfoss ISD 500.xml*
- ESI file (EtherCAT® Slave Information) for the SAB: *Danfoss SAB.xml*

6.3.1.2 Creating a TwinCAT® Project

Information on how to install TwinCAT® can be found in detail in the Beckhoff Information System (*infosys.beckhoff.com*). Open the information system and select [TwinCAT 2 → TwinCAT Quick Start → Installation].

Information on how to create a new project in TwinCAT® can be found in detail in the Beckhoff Information System (*infosys.beckhoff.com*). Open the information system and select [TwinCAT 2 → TwinCAT Quick Start or TwinCAT 2 → TX1200 TwinCAT PLC → TwinCAT PLC Control].

How to include the ISD 510 library into a TwinCAT® project:

1. In the *Resources* tab of TwinCAT® PLC Control, open the *Library Manager*.
2. In the upper left area of the *Library Manager* window, right-click and select *Additional Library ....*
3. Select the *Danfoss_VLT_ISD_510.lib* file (according to the location on the hard drive).
4. Click on *Open*. Now the libraries are integrated into the TwinCAT® PLC control project.

Inside the library, the POUs are organized into folders:

- BasCam_51x
  - Contains POUs for the creation of basic CAMs.
- ISD_51x
  - Contains POUs defined by PLCopen® (Name starting with MC_) and POUs defined by Danfoss (name starting with DD_). The POUs defined by Danfoss provide additional functionality for the axis.
  - It is possible to combine POUs defined by PLCopen® with POUs defined by Danfoss.
  - The names of the POUs that target the servo drive all end with _ISD51x.
- Intern_51x
  - Contains POUs that are needed internally for the libraries.
  - Do not use these POUs in an application.
- LabCam_51x
  - Contains POUs for the creation of labeling CAMs.
- SAB_51x
  - Contains POUs defined by Danfoss (Name starting with DD_) and provide the functionality for the SAB.
  - The names of the POUs that target the SAB all end with _SAB.

When integrating the ISD 510 library, some standard libraries are integrated automatically, unless they are already part of the project.

NOTICE

Do not remove these libraries otherwise the ISD libraries will not work.

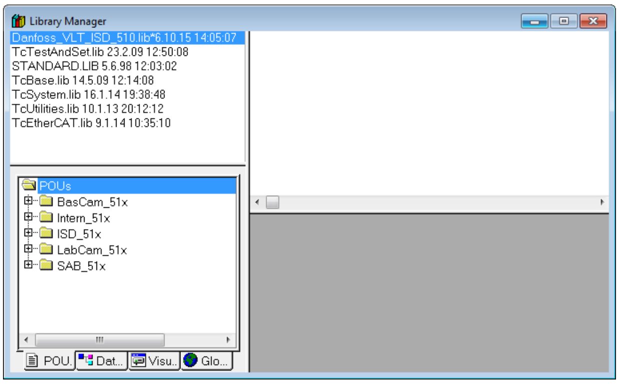


Illustration 6.1 Library Manager after Including the ISD 51x Library

Inside the library, the following lists of constants are defined:

- AxisErrorCodes
    - Constants for error codes of the axis.
    - Error codes can be read using the function block *MC_ReadAxisError_ISD51x* and/or *DD_ReadAxisWarning_ISD51x*.
- AxisTraceSignals
    - Constants for the trace signals of the axis.
    - Intended to be used with the function block *DD_Trace_ISD51x*.
- BasCam_51x
    - Constants for the creation of basic CAMs.
- CamParsingErrors
    - Constants for parsing problems of a CAM.
    - Error reason is returned by function block *MC_CamTableSelect_ISD51x*.
- Danfoss_VLT_ISD510
    - Contains the version information of the library.
- FB_ErrorConstants
    - Constants for errors inside POUs.
    - The reason is given in an output *ErrorInfo.ErrorID* that is available in all POUs.
- Intern_51x
    - Constants that are needed internally for the library.
    - They are not intended to be used in an application.
- LabCam_51x
    - Constants for the creation of labeling CAMs.
- SabErrorCodes
    - Constants for error codes of the SAB.
    - Error codes can be read using the function block *DD_ReadSabError_SAB* and/or *DD_ReadSabWarning_SAB*.
- SabTraceSignals
    - Constants for the trace signals of the SAB.
    - Intended to be used with the function block *DD_Trace_SAB*.
- SdoAbortCodes
    - Constants for errors concerning reading and writing of parameters.
    - The reason is given in an output *AbortCode* that is available in several POUs.

Instantiating AXIS_REF_ISD51x

Inside the folder *ISD_51x* in library *Danfoss_VLT_ISD_510*, there is a function block called *AXIS_REF_ISD51x*. Create 1 instance of this function block for every servo drive that has to be controlled or monitored. Each instance of *AXIS_REF_ISD51x* is the logical representation of 1 physical servo drive.

Instantiating SAB_REF

Inside the folder *SAB_51x* in library *Danfoss_VLT_ISD_510*, there is a function block called *SAB_REF*. Create 1 instance of this function block for every SAB that has to be controlled or monitored.

Each instance of *SAB_REF* is the logical representation of 1 physical SAB.

NOTICE

When compiling the library, check that the option *Replace constants* under [Project → Options... → Build] is activated.

Afterwards, save and compile the project to update the automatically generated variable information for the *TwinCAT® System Manager*.

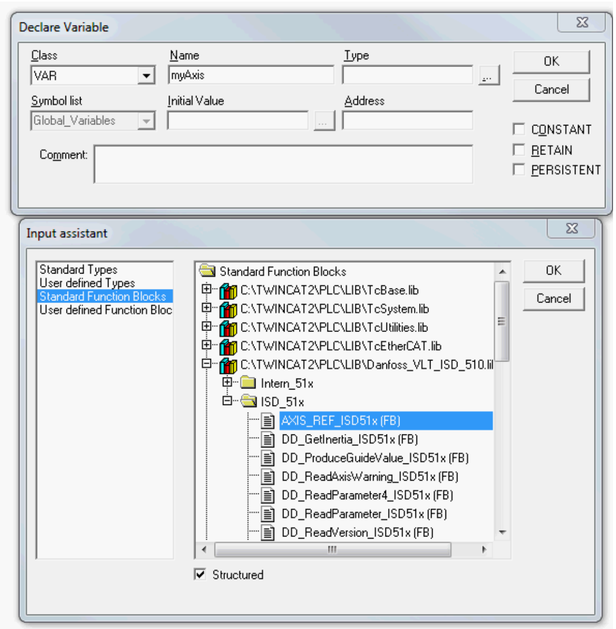


Illustration 6.2 Instantiation of AXIS_REF_ISD51x

Append a PLC Project into TwinCAT® System Manager

To create a link between the *TwinCAT® PLC Control* project and the *TwinCAT® System Manager*, connect the saved project, especially the inputs and outputs, to the *TwinCAT® System Manager*:

1. To add the project information to the *TwinCAT® System Manager*, right-click on *PLC-Configuration* and select *Append PLC project…*.
2. In the *Insert IEC 1131 Project* window, select the project information file according to the location on the hard drive. The file has the same name as the PLC project, but with the file extension *.tpy*.
3. Click on *Open*.

Import fieldbus device and add to TwinCAT®

The next step is to import the servo drive and the SAB into the *TwinCAT® System Manager* software:

1. Copy the ESI file *Danfoss ISD 500.xml* into the folder *TwinCAT Installation Folder\Io\EtherCAT* on the hard drive. This only has to be done once per project. The *TwinCAT® System Manager* automatically searches for ESI files at this location on the hard drive during start-up.
2. To add an EtherCAT® master, right-click on [I/O-Configuration → I/O Devices] and select *Append Device…*.
3. In the following window, select [EtherCAT → EtherCAT] (see *Illustration 6.3*).
4. Click on *OK*.
5. Select *Device 1 (EtherCAT®)* and select the correct *Network Adapter* on the right side of the window in the *Adapter* tab.
6. To add an SAB, right-click on *Device1 (EtherCAT®)* and select *Append Box…*.
7. In the *Insert EtherCAT Device* window, select [Danfoss GmbH → VLT® ISD Series → VLT® Servo Access Box L1] for Line 1 of the SAB (and/or VLT® Servo Access Box L2 for Line 2 of the SAB).
8. Click on *OK*.
9. To add a servo drive to line 1 of the SAB, right-click on *Box 1 (VLT® Servo Access Box L1)* and select *Append Box...*.
10. In the *Insert EtherCAT Device* window, select [Danfoss GmbH → VLT® ISD Series → VLT® ISD 510 Integrated Servo Drive].
11. Click on *OK*.
12. Answer the question if the servo drive is used as an NC axis with *No*. If the servo drive is to be used as an NC axis, see *chapter 6.3.1.3 Configuration as a TwinCAT® NC Axis*.

NOTICE

Add 1 entry to the EtherCAT® master of the *TwinCAT® System Manager* for each physical servo drive and SAB. Add the servo drive to the correct SAB line.

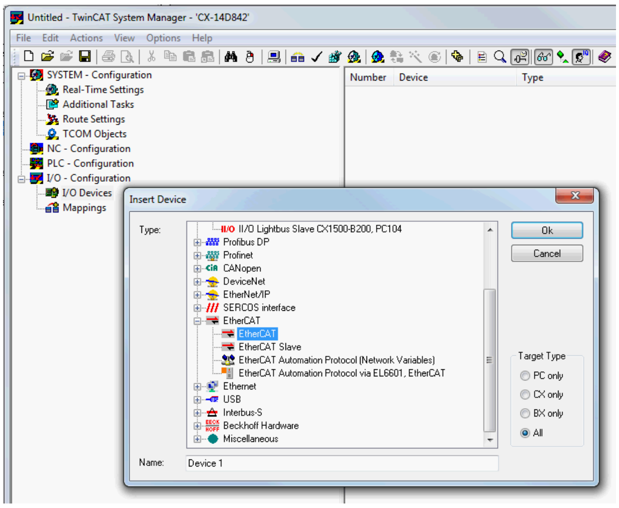


Illustration 6.3 Add an EtherCAT® Master to the Project

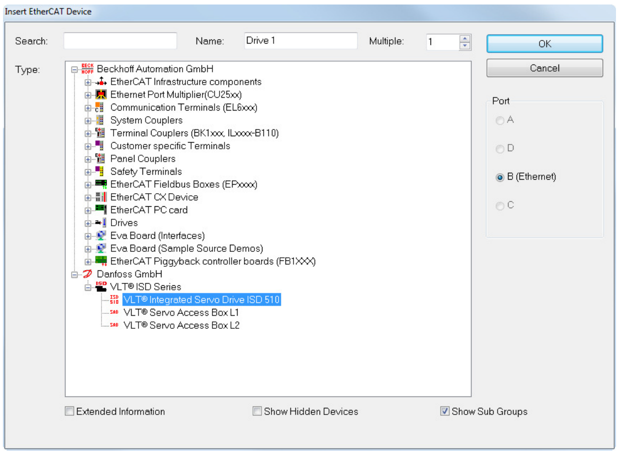


Illustration 6.4 Add an ISD 510 Servo Drive to the Project

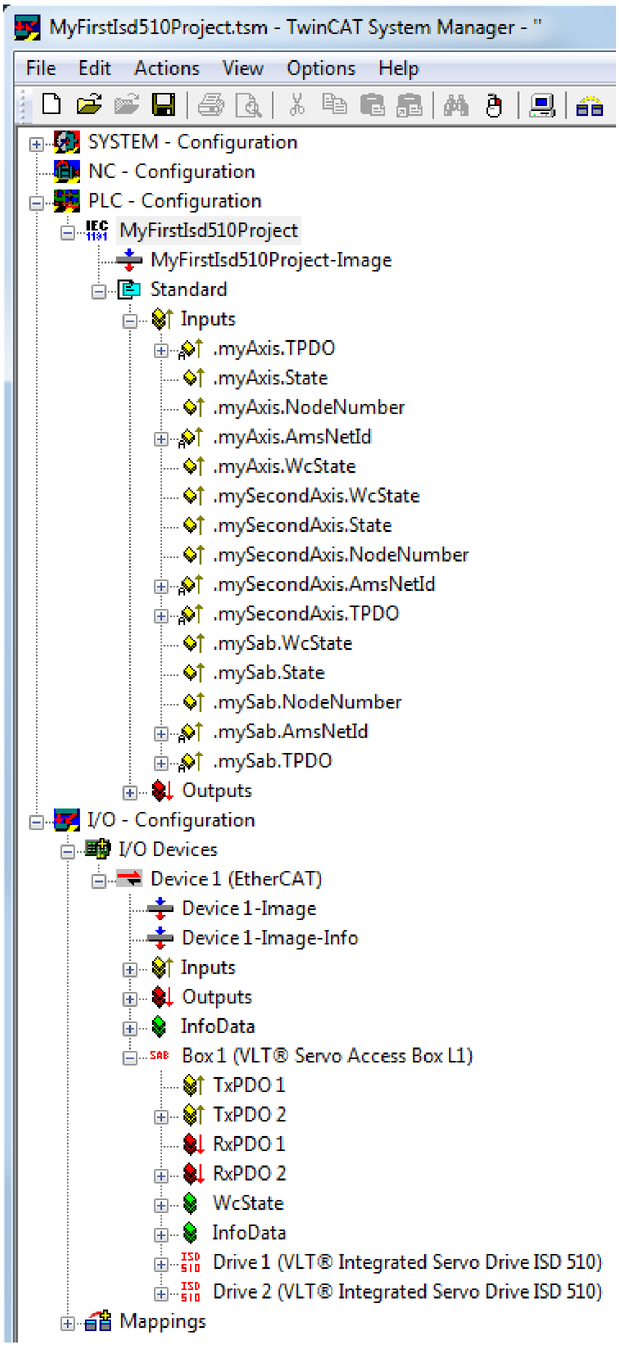


Illustration 6.5 TwinCAT® System Manager after Appending the PLC Project and Adding an SAB and 2 Servo Drives

I/O configuration and I/O mapping

When connecting >1 servo drive, connect port C (X2) of the previous servo drive to port A (X1) of the next servo drive. Also make the port assignment for the SAB. If the hardware set-up is already present, the *TwinCAT® System Manager Scan devices* function can be used to automatically add the connected devices to the configuration in the correct order.

Configure the servo drive, so that the PDO mapping matches the requirements of the library, via the *TwinCAT® System Manager*.

1. Click on the ISD servo drive entry.
2. Select the *Slots* tab on the right side of the window.
3. Remove the current PDO configuration by selecting the entry *Module 1 (CSV PDO)* in the *Slot* box.
4. Click on *X*.
5. Select *Library PDO* in the *Module* box.
6. Click on <.

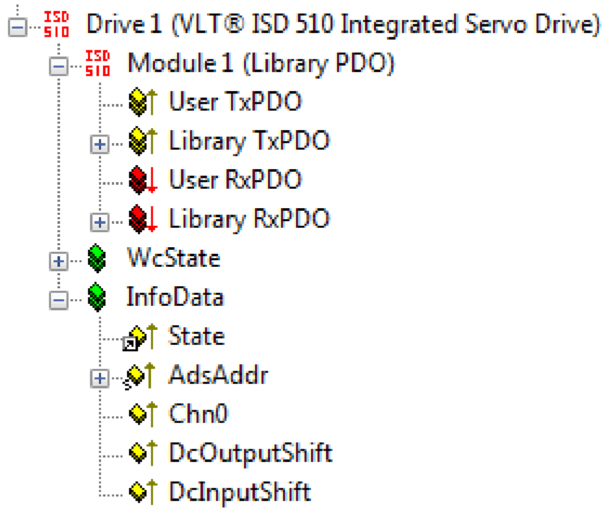


Illustration 6.6 ISD 510 Servo Drive with Correct I/O Configuration

Attach the input and output variables of the PLC program to the physical inputs and outputs of the device. Use the *TwinCAT® System Manager* for this so that the library has access to all necessary objects.

1. Select *Library TxPDO* via menu [I/O-Configuration → I/O Devices → Device1 (EtherCAT®) → Box 1 (VLT® Servo Access Box L1) → Drive 2 (VLT® ISD 510 Integrated Servo Drive) → Module 1 (Library PDO) → Library TxPDO].
2. Select all entries *Lib pdo tx1* to *Lib pdo tx9* on the right side of the window (see *Illustration 6.7*).
3. Right-click and select *Change Multi Link…*.
4. In the *Attach Variable 36.0 Byte(s) (Input)* window, select [PLC-Configuration → MyFirstIsd510Project → Standard → .myAxis.TPDO].
   Ensure that the *Matching Size* option is selected in the *Attach Variable* window.
5. Click on *OK*.
6. Click on library *RxPDO* via menu [I/O-Configuration → I/O Devices → Device1 (EtherCAT®) → Box1 (VLT® Servo Access Box L1) → Drive2 (VLT® ISD 510 Integrated Servo Drive) → Module1 (Library PDO) → Library RxPDO].

7. Select all entries *Lib pdo rx1* to *Lib pdo rx9* on the right side of the window.
8. Right-click and select *Change Multi Link…*.
9. In the *Attach Variable 36.0 Byte(s) (Output)* window, select [PLC-Configuration → MyFirstIsd510Project → Standard → .myAxis.RPDO].
10. Click on *OK*.
11. Right-click on *WcState* via [I/O-Configuration → I/O Devices → Device1 (EtherCAT®) → Box1 (VLT® Servo Access Box L1) → Drive2 (VLT® ISD 510 Integrated Servo Drive) → WcState] and select *Change Link…*.
12. In the *Attach Variable State (Input)* window, select [PLC-Configuration → MyFirstIsd510Project → Standard → .myAxis.WcState].
13. Click on *OK*.
14. Right-click on *State* via [I/O-Configuration → I/O Devices → Device1 (EtherCAT®) → Box1 (VLT® Servo Access Box L1) → Drive2 (VLT® ISD 510 Integrated Servo Drive) → InfoData] and select *Change Link…*.
15. In the *Attach Variable State (Input)* window, select [PLC-Configuration → MyFirstIsd510Project → Standard → .myAxis.State.
16. Click on *OK*.
17. Right-click on *netId* via [I/O-Configuration → I/O Devices → Device1 (EtherCAT®) → Box1 (VLT® Servo Access Box L1) → Drive2 (VLT® ISD 510 Integrated Servo Drive) → InfoData → AdsAddr] and select *Change Link…*.
18. In the *Attach Variable netId (Input)* window, select [PLC-Configuration → MyFirstIsd510Project → Standard → .myAxis.AmsNetId].
19. Click on *OK*.
20. Right-click on *port* via [I/O-Configuration → I/O Devices → Device1 (EtherCAT®) → Box1 (VLT® Servo Access Box L1) → Drive2 (VLT® ISD 510 Integrated Servo Drive) → InfoData → AdsAddr] and select *Change Link…*.
21. In the *Attach Variable port (Input)* window, select [PLC-Configuration → MyFirstIsd510Project → Standard → .myAxis.NodeNumber].
22. Click on *OK*.

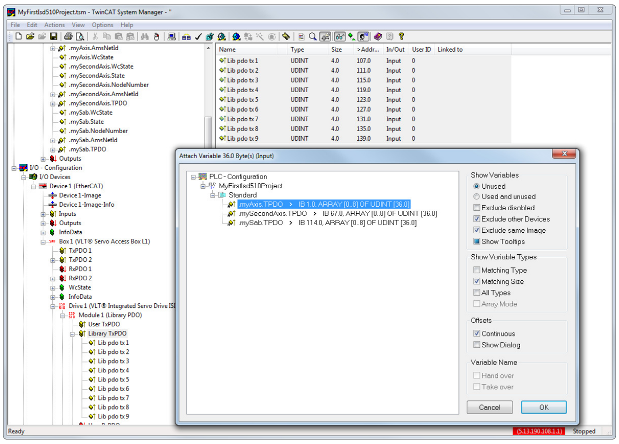


Illustration 6.7 Attaching Inputs and Outputs to the Physical Data Points

NOTICE

Repeat the steps 2–22 for Box 1 (VLT® Servo Access Box L1) and the instance *mySAB*.

To transfer the mappings back to the PLC program, select *Activate Configuration…* in menu item *Actions*.
After a rebuild in *TwinCAT® PLC Control*, the TwinCAT® configuration is according to *Illustration 6.8* (here *myAxis* and *mySecondAxis* are instances of *AXIS_REF_ISD51x* and *mySAB* is an instance of *SAB_REF*). The concrete addresses can be different.

TwinCAT_Configuration

```
(* Generated automatically by TwinCAT - (read only) *)
VAR_CONFIG
    .myAxis.WcState AT %IX52.0 : BOOL;
    .myAxis.State AT %IB40 : UINT;
    .myAxis.NodeNumber AT %IB42 : T_AmsPort;
    .myAxis.AmsNetId AT %IB44 : T_AmsNetIdArr;
    .myAxis.TPDO AT %IB1 : ARRAY [0..8] OF UDINT;
    .myAxis.RPDO AT %QB0 : ARRAY [0..8] OF UDINT;
    .mySecondAxis.WcState AT %IX56.0 : BOOL;
    .mySecondAxis.State AT %IB57 : UINT;
    .mySecondAxis.NodeNumber AT %IB59 : T_AmsPort;
    .mySecondAxis.AmsNetId AT %IB61 : T_AmsNetIdArr;
    .mySecondAxis.TPDO AT %IB67 : ARRAY [0..8] OF UDINT;
    .mySecondAxis.RPDO AT %QB36 : ARRAY [0..8] OF UDINT;
    .mySab.WcState AT %IX103.0 : BOOL;
    .mySab.State AT %IB104 : UINT;
    .mySab.NodeNumber AT %IB106 : T_AmsPort;
    .mySab.AmsNetId AT %IB108 : T_AmsNetIdArr;
    .mySab.TPDO AT %IB114 : ARRAY [0..8] OF UDINT;
    .mySab.RPDO AT %QB72 : ARRAY [0..8] OF UDINT;
END_VAR
```

Illustration 6.8 TwinCAT® Configuration: I/O Mapping of 2 Servo Drives and 1 SAB

NOTICE

Put the SAB to a separate SYNC unit to avoid interruptions in communication to the SAB if the U_{AUX} supply to the servo drives is switched off due to an error.

Cycle time settings

The minimum cycle time is 400 µs. The ISD 510 devices can run EtherCAT® cycle times in multiples of 400 µs or 500 µs. The devices are automatically parameterized by the PLC on start-up, depending on the EtherCAT® configuration of the physical interface. To access the system base time, select [SYSTEM-Configuration → Real-Time Settings] in the *TwinCAT® System Manager*. Multiples of this base time can then be used as EtherCAT® cycle times.

NOTICE

Set the task cycle time of the PLC program to be the same as the EtherCAT® cycle time. Otherwise data can get lost and performance is reduced.

Set the PLC cycle time in *TwinCAT® PLC Control*:

1. Double-click *Task configuration* in the *Resources* tab.
2. Ensure that the PLC cycle time is the same as the EtherCAT® cycle time.

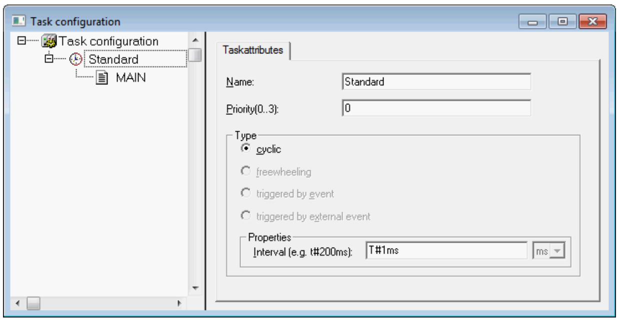


Illustration 6.9 Task Configuration to Parameterize PLC Cycle Time

NOTICE

After changing the task cycle time in *TwinCAT® PLC Control*, carry out a *ReScan* of the PLC configuration inside the *TwinCAT® System Manager* to update the settings. Afterwards, activate the configuration in the PLC.

6.3.1.3 Configuration as a TwinCAT® NC Axis

The servo drives can be used with the built-in NC functionality of TwinCAT®. Everything related to the SAB must be done as described in *chapter 6.3.1.2 Creating a TwinCAT® Project*.

1. In addition to the *Danfoss_VLT_ISD_510.lib* file, include the *TcMC2.lib* file (the *Danfoss_VLT_ISD_510.lib* file is still needed for the SAB to be operated).
2. Create 1 instance of *AXIS_REF* (instead of *AXIS_REF_ISD51x*) for each servo drive that is used as an NC axis.
3. Append the PLC project into the *TwinCAT® System Manager*, import the devices, and add them to TwinCAT® as described in *chapter 6.3.1.2 Creating a TwinCAT® Project*, however in the last step, answer the question if the servo drive is used as an NC axis with *Yes*. Then an NC task is created automatically.

In the *TwinCAT® System Manager*, select a different *I/O Configuration* for the servo drives used as NC axes.

1. Depending on the mode of operation to be used, select either the slot *CSP PDO* or *CSV PDO*. Per default, *CSV PDO* is mapped and pre-selected. Map the following variables if the servo drive is required to work with *CSP PDO*:
   - 1a In the *Settings* Tab of the NC Axis, select [NC-Configuration → NC-Task 1 SAF → Axes → Axis 1]. Click on the *Link To (all Types)…* button and select the desired servo drive.
2. In the same tab, select the preferred *Unit*.
3. Depending on the selected *Unit*, adjust the *Scaling Factor* for the axis encoder via menu [NC-Configuration → NC-Task 1 SAF → Axes → Axis 1 → Axis 1_Enc] in the *Parameter* tab.
   Example: When the unit *Degrees* is selected, the scaling factor is $360°/2^{20}$ = 0.00034332275390625.
4. Set the *Reference Velocity* in the *Parameter* tab via menu [NC-Configuration → NC-Task 1 SAF → Axes → Axis 1 → Axis 1_Enc.
5. Set the *Output Scaling Factor* (Velocity) to *125*.
6. Test the functionality and the configuration in the *Online* tab of the axis.

6.3.1.4 Connecting to the PLC

Information on how to connect to the PLC can be found in detail in the Beckhoff Information System (*infosys.beckhoff.com*). Open the information system and go to [TwinCAT 2 → TwinCAT System Manager → Operation → Controls → Choose Target System].

6

6.4 Automation Studio™

6.4.1 Programming with Automation Studio™

6.4.1.1 Requirements

The following files are required to integrate the VLT® Integrated Servo Drive ISD® 510 and the VLT® Servo Access Box into an Automation Studio™ project:

- Package of libraries for the ISD 510 servo system: *Danfoss_VLT_ISD_510.zip*
- XDD file (XML Device Description) for the servo drive: *0x0300008D_ISD510.xdd*
- XDD file (XML Device Description) for the SAB: *0x0300008D_SAB.xdd*

6.4.1.2 Creating an Automation Studio™ Project

NOTICE

The procedures described in this chapter apply to Automation Studio™ Version 3.0.90. Refer to the Automation Studio™ Help for the corresponding steps with V4.x.

Information on how to install Automation Studio™ can be found in detail in the Automation Studio™ help. Open the *B&R Help Explorer* and go to [Automation software → Software Installation → Automation Studio].

Information on how to create a new project in Automation Studio™ can be found in detail in the Automation Studio™ help. Open the *B&R Help Explorer* and go to [Automation Software → Getting Started → Creating programs with Automation Studio → First project with X20 CPU].

How to include the ISD 510 libraries into an Automation Studio™ project:

1. In the *Logical View*, open the menu entry [File → Import…].
2. In the next window, select the *Danfoss_VLT_ISD_510.zip* file (according to the location on the hard drive).
3. Click on *Open*.
4. Assign the libraries to the CPU in the next window.
5. Click on *Finish*. Now the libraries are integrated into the Automation Studio™ project.

A new folder containing the ISD libraries is created during integration:

- ISD_51x
    - Contains program organization units (POUs) defined by PLCopen® (name starting with *MC_*) and POUs defined by Danfoss (name starting with *DD_*). The Danfoss POUs provide additional functionality for the servo drive.
    - It is possible to combine POUs defined by PLCopen® with POUs defined by Danfoss.
    - The names of the POUs that target the servo drive all end with *_ISD51x*.
- SAB_51x
    - Contains POUs defined by Danfoss (name starting with *DD_*) and provide the functionality for the SAB.
    - The names of the POUs that target the SAB all end with *_SAB*.
- BasCam_51x
    - Contains POUs for the creation of basic CAMs.
- LabCam_51x
    - Contains POUs for the creation of labeling CAMs.
- Intern_51x
    - Contains POUs that are needed internally for the libraries.
    - Do not use these POUs in an application.

When integrating the ISD_51x package, some standard libraries are integrated automatically, unless they are already part of the project.

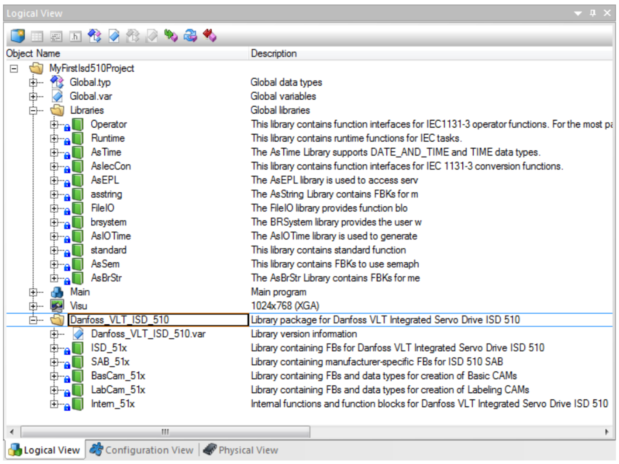


Illustration 6.10 Standard Libraries

NOTICE

Do not remove these libraries otherwise the ISD libraries will not work.

Inside the library, the following lists of constants are defined:

- AxisErrorCodes
    - Constants for error codes of the axis.
    - Error codes can be read using the function block *MC_ReadAxisError_ISD51x* and/or *DD_ReadAxisWarning_ISD51x*.
- AxisTraceSignals
    - Constants for the trace signals of the axis.
    - Intended to be used with the function block *DD_Trace_ISD51x*.
- BasCam_51x
    - Constants for the creation of basic CAMs.
- CamParsingErrors
    - Constants for parsing problems of a CAM.
    - Error reason is returned by function block *MC_CamTableSelect_ISD51x*.
- Danfoss_VLT_ISD510
    - Contains the version information of the library.
- FB_ErrorConstants
    - Constants for errors inside POUs.
    - The reason is given in an output *ErrorInfo.ErrorID* that is available in all POUs.
- Intern_ISD51x
    - Constants which are needed internally for the library.
    - They are not intended to be used in an application.
- LabCam_51x
    - Constants for the creation of labeling CAMs.
- SabErrorCodes
    - Constants for error codes of the SAB.
    - Error codes can be read using the function block *DD_ReadSabError_SAB* and/or *DD_ReadSabWarning_SAB*.
- SabTraceSignals
    - Constants for the trace signals of the SAB.
    - Intended to be used with the function block *DD_Trace_SAB*.
- SdoAbortCodes
    - Constants for errors concerning reading and writing of parameters.
    - The reason is given in an output *AbortCode* that is available in several POUs.

Instantiating AXIS_REF_ISD51x

Inside the library *ISD_51x*, there is a function block called *AXIS_REF_ISD51x*. Create 1 instance of this function block for every servo drive that has to be controlled or monitored. To create a link to the physical servo drive, link each instance to 1 physical servo drive. To do this (in the *Logical View*), initialize each instance with its node number and the slot name (for example, *IF3*) it is connected to.
Each instance of *AXIS_REF_ISD51x* is the logical representation of 1 physical servo drive.

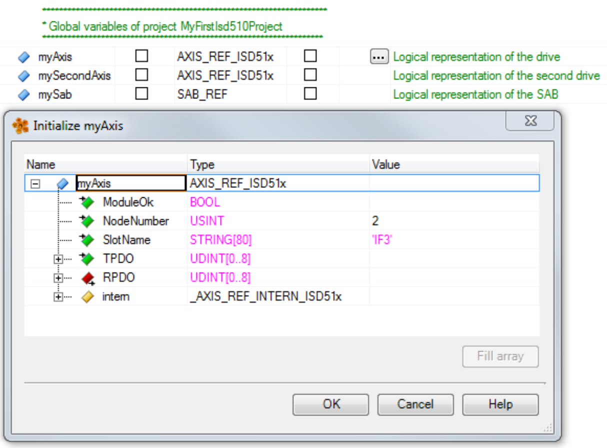


Illustration 6.11 Instantiation of AXIS_REF and Setting of Initial Values

Instantiating SAB_REF

Inside the library *SAB_51x*, there is a function block called *SAB_REF*. Create 1 instance of this function block for every SAB that has to be controlled or monitored. To create a link to the physical SAB, link each instance to 1 physical SAB. To do this (in the *Logical View*), initialize each instance with its node number and the slot name (for example, *IF3*) it is connected to.
Each instance of *SAB_REF* is the logical representation of 1 physical SAB.

Import fieldbus device and add to Physical View

The next step is to import the ISD 510 servo drive into Automation Studio™:

1. Select the menu entry [Tools → Import Fieldbus Device…].
2. Select the XDD file *0x0300008D_ISD510.xdd* from its location on the hard drive.
   This import only needs to be done once per project. The device is then known to Automation Studio™.
3. The ISD 510 servo drive can now be added to the Ethernet POWERLINK® interface of the controller in the *Physical View*:
   - 3a Right-click on the controller in the *Physical View* and select [Open → POWERLINK].
   - 3b Right-click on the interface and select *Insert…*.
   - 3c In the *Select controller module* window, select the ISD 510 in the group *POWERLINK Devices*.
   - 3d Click on *Next*.
   - 3e In the next window, enter the node number of the servo drive.

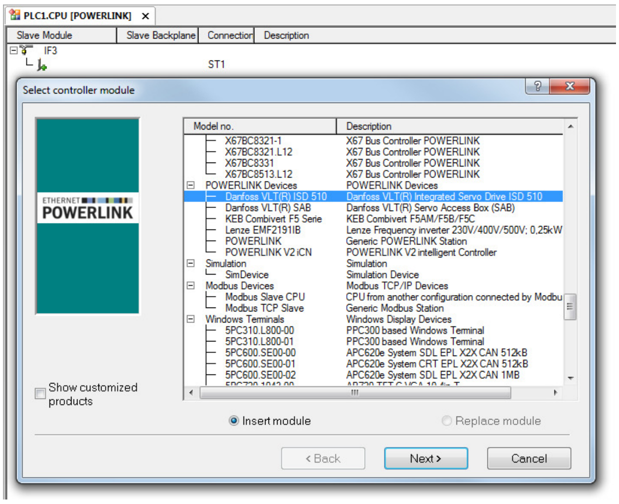


Illustration 6.12 Add an ISD 510 Servo Drive to the Project

For each physical servo drive, add 1 entry to the *Physical View* of Automation Studio™.

The next step is to import the Servo Access Box into Automation Studio™:

1. Select the menu entry [Tools → Import Fieldbus Device…].
2. Select the XDD file *0x0300008D_SAB.xdd* from its location on the hard drive. This import only needs to be done once per project. The device is then known to Automation Studio™.
3. The SAB can now be added to the Ethernet POWERLINK® interface of the controller in the *Physical View*:
   - 3a Right-click on the controller in the *Physical View* and select [Open → POWERLINK].
   - 3b Right-click on the interface and select *Insert…*.
   - 3c In the *Select controller module* window, select the SAB in the group *POWERLINK Devices*.
   - 3d Click on *Next*.
   - 3e In the next window, enter the node number of the SAB.

For each physical SAB, add 1 entry to the *Physical View* of Automation Studio™.

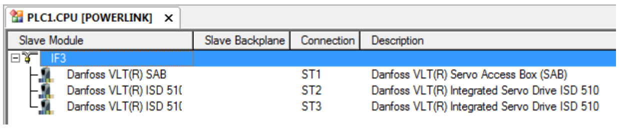


Illustration 6.13 1 SAB and 2 ISD 510 Servo Drives Added to the Ethernet POWERLINK® Interface

I/O configuration and I/O mapping

The *I/O Configuration* of the servo drive has to be parameterized in a way that the library has access to all necessary objects:

1. Right-click on the entry of the ISD 510 and select *Open I/O Configuration*.
2. In the *Channels* section, change the *Cyclic transmission* of the following objects:
   - 2a All sub-indexes of object *0x5050* (Lib pdo rx_I5050 ARRAY[]) to *Write*.
   - 2b All sub-indexes of object *0x5051* (Lib pdo tx_I5051 ARRAY[]) to *Read*.

The *I/O Configuration* of the SAB has to be parameterized in a way that the library has access to all necessary objects:

1. Right-click on the entry of the SAB and select *Open I/O Configuration*.
2. In the *Channels* section, change the *Cyclic transmission* of the following objects:
   - 2a All sub-indexes of object *0x5050* (Lib pdo rx_I5050 ARRAY[]) to *Write*.
   - 2b All sub-indexes of object *0x5051* (Lib pdo tx_I5051 ARRAY[]) to *Read*.

These settings configure the cyclic communication with the device. These parameters are required for the library to work.

NOTICE

It is possible to use copy & paste to apply the same *I/O Configuration* to multiple devices of the same type.

NOTICE

Set *Module supervised* to *off* for the servo drives and the SAB. The parameter is found in the *I/O Configuration* of the device.

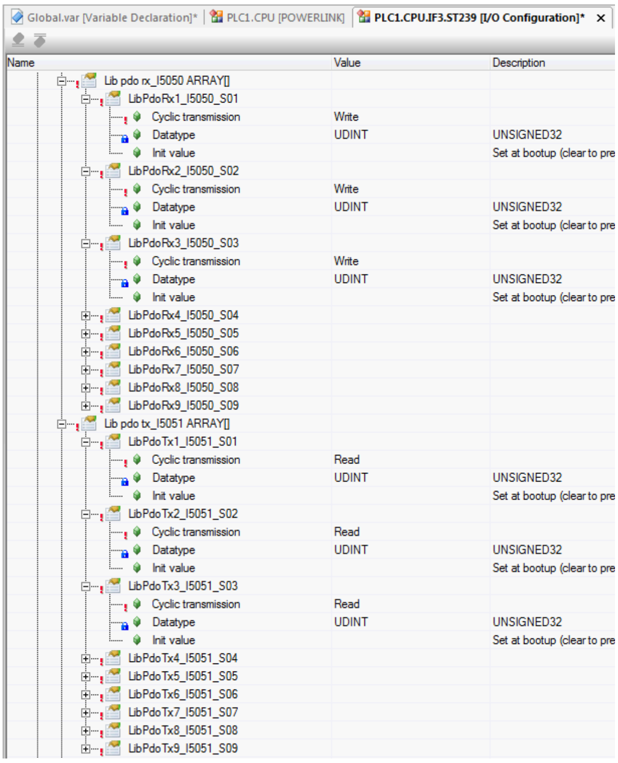


Illustration 6.14 I/O Configuration of an ISD 510 Device

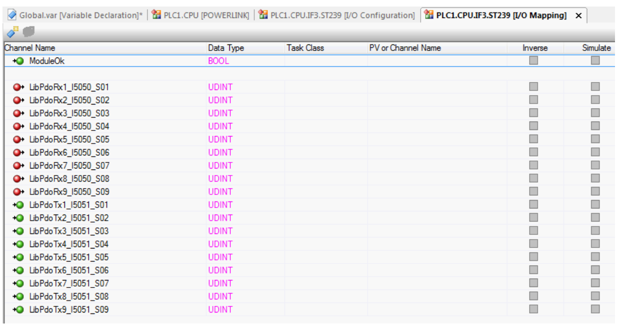


Illustration 6.15 I/O Mapping after Successful Configuration

Map the inputs and outputs of the instance of the *AXIS_REF_ISD51x* function block and the physical data points of the servo drive according to *Illustration 6.16* (here *myAxis* is an instance of *AXIS_REF_ISD51x*):

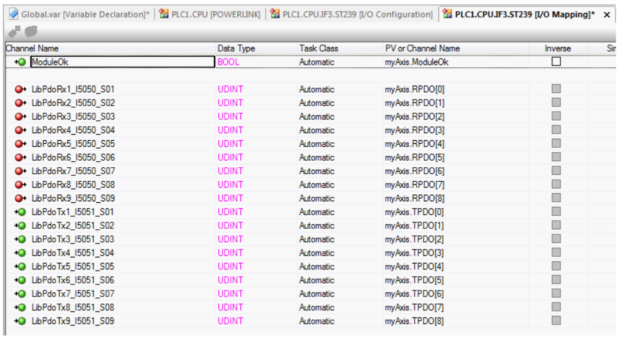


Illustration 6.16 I/O Mapping of an ISD 510 Servo Drive

Map the inputs and outputs of the instance of the *SAB_REF* function block and the physical data points of the SAB accordingly.

Cycle time settings

The minimum cycle time is 400 µs. The ISD 510 devices can run Ethernet POWERLINK® cycle times in multiples of 400 µs and multiples of 500 µs. The devices are automatically parameterized by the PLC on start-up, depending on the Ethernet POWERLINK® configuration of the physical interface. The Ethernet POWERLINK® configuration can be accessed by right-clicking [CPU → Open IF3 POWERLINK Configuration] in the *Physical View*.

NOTICE

Ensure that the task cycle times of the PLC program and Ethernet POWERLINK® are the same. Otherwise, data could be lost and performance reduced.

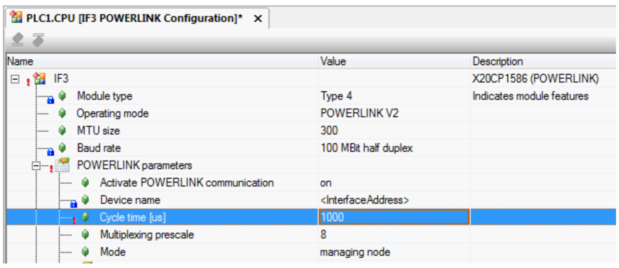


Illustration 6.17 Ethernet POWERLINK® Configuration Window to Parameterize Ethernet POWERLINK® Cycle Time

Set the PLC cycle time in Automation Studio™:

1. Right-click [CPU → Open Software Configuration] in the *Physical View*.
2. Ensure that the PLC cycle time is the same as the Ethernet POWERLINK® cycle time.

6.4.1.3 Connecting to the PLC

Information on how to connect to the PLC can be found in detail in the Automation Studio™ Help. Open the *B&R Help Explorer* and go to [Automation Software → Getting Started → Creating programs with Automation Studio → First project with X20 CPU → Configure online connection].

6.5 Function Block Descriptions

6.5.1 Overview PLCopen®

PLCopen® is a vendor- and product-independent worldwide association. Its mission is to be the leading association resolving topics related to control programming to support the use of international standards in this field. Function blocks of the PLC library or package are compliant with the following standard version: Technical Specification Function blocks for motion control (Formerly Part 1 and Part 2), Version 2.0, March 17, 2011.

6.5.1.1 Naming Conventions

Function blocks
The naming of the function blocks consist of a prefix, the function-specific name, and a postfix, which is described in PLCopen® as *Supplier ID*. For function blocks defined by PLCopen®, the prefix *MC_* is used. For ISD-specific function blocks, the prefix *DD_* is used. For function blocks for a servo drive, *_ISD51x* is used as Supplier ID and for SAB, *_SAB* is used.

Example:

- MC_Power_ISD51x = Function block defined by PLCopen, targeting the servo drive.
- DD_ReadVersion_SAB = Function block defined by Danfoss, targeting the SAB.

Enumerations
For enumerations defined by PLCopen®, the prefix *MC_* is used. For enumerations defined by Danfoss, the prefix *DD_* is used.
_ISD51x is used as *Supplier ID* for enumerations.

Internal
The library needs some POUs that are used internally, however they cannot be used in an application program. All internal POUs, enumerations, and constants start with a leading underscore _ and are sorted into the *Intern_51x* folder, and/or into the *Intern_51x* library inside the package.

6.5.1.2 Structure of Library/Package

All elements, such as function blocks, data types, and enumerations, are sorted into folders in the library, and/or into libraries inside the package, depending on the capabilities of the development environment.

See *chapter 6.3 TwinCAT®* and *chapter 6.4 Automation Studio ™* for further information.

The constants are structured in lists of constants:

- AxisErrorCodes/SabErrorCodes: Constants for all the alarms and warnings of the servo drive (see *chapter 9.2.2 Error Codes*) and SAB (see *chapter 9.3.2 Warnings and Alarms*).
- AxisTraceSignals/SabTraceSignals: Constants for trace signals of the servo drive and SAB (see *chapter 2.7.2 Trace*).
- FB_ErrorConstants: Constants for all the function block errors. Value provided in the function block output *ErrorInfo.ErrorID*.
- CamParsingErrors: Constants for specifying an error after the CAM parsing (see *chapter 2.4.5 CAM Mode*).

6.5.1.3 PLCopen® State Machine

The state diagram in *Illustration 6.18* defines the behavior of the axis at a high level when multiple motion control function blocks are activated simultaneously. This combination of motion profiles is useful in building a more complicated profile, or to handle exceptions within a program.

The basic rule is that motion commands are always taken sequentially, even if the PLC is capable of real parallel processing. These commands act on the axis state diagram.

The axis is always in 1 of the defined states. Any motion command that causes a transition changes the state of the axis and, as a result of that, modifies the way the current motion is computed.

The state diagram is an abstraction layer of what the real state of the axis is, compared to the image of the I/O points within a cyclic (PLC) program.

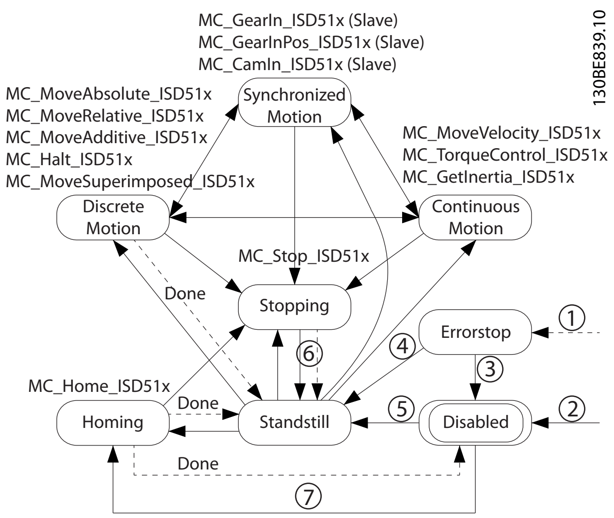
A change of state is reflected immediately when issuing the corresponding motion command. The response time depends on the specific functionality.

The diagram is focused on a single axis. The multiple axis function blocks, for example *MC_CamIn_ISD51x* or *MC_GearIn_ISD51x*, can be looked at, from a state diagram point of view, as multiple single-axes all in specific states. For example, the CAM-master can be in the state *ContinuousMotion*. The corresponding slave is in the state *SynchronizedMotion*. Connecting a slave axis to a master axis has no influence on the master axis.

Arrows within the state diagram show the possible state transitions between the states. State transitions due to an issued command are shown as full arrows. Dashed arrows show state transitions, which occur when a command of an axis has terminated, and system-related transitions (for example, error-related). The motion commands that transit the axis to the corresponding motion state are listed above the states. These motion commands may also be issued when the axis is already in the corresponding motion state.

PLCopen® function blocks that are not listed in the state diagram do not affect or change the state of the axis. The PLCopen® state information and the transition between the states is handled and kept inside the PLC library. The servo drive has no knowledge about the PLCopen® state machine.

The PLCopen® state changes as soon as a function block is activated and the preconditions are checked. That does not necessarily mean that the axis already changed the mode of operation. The new PLCopen® state is already active during the sending of parameters before the actual activation of the functionality (for example, in the case of *Homing*).



Note	Description
1	From any state. An error in the axis occurred.
2	From any state. MC_Power.Enable = FALSE and there is no error in the axis.
3	MC_Reset AND MC_Power.Status = FALSE
4	MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE (not valid for ISD 510)
5	MC_Power.Enable = TRUE AND MC_Power.Status = TRUE
6	MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE
7	Possible for *Homing* modes that require no motion. After *Homing* is done, the state changes back to the original state (for example, Disabled → Homing → Disabled or Standstill → Homing → Standstill).

Illustration 6.18 PLCopen State Machine

Disabled
The state *Disabled* describes the initial state of the axis. In this state, the movement of the axis is not influenced by the function blocks. Power is off and there is no error in the axis.

If the *MC_Power_ISD51x* function block is called with *Enable* := *TRUE* while in state *Disabled*, the state changes to *Standstill*.

Calling up *MC_Power_ISD51x* when *Enable* is *FALSE* in any state except *ErrorStop* transfers the axis to the state *Disabled*, either directly or via any other state. Any ongoing motion commands on the axis are aborted (*CommandAborted*).

ErrorStop
ErrorStop is valid as highest priority and applicable if there is an error. The power of the axis is always disabled and cannot be changed via *MC_Power_ISD51x*. While the error is pending, the state remains *ErrorStop* and, if possible, the axis stops. No further motion command can be accepted until a reset is carried out from the *ErrorStop* state.

The transition to *ErrorStop* refers to errors from the axis and axis control, and not from the function block instances. These axis errors may also be reflected in the output of the function blocks *function block instances errors*.

Standstill
Power is on, there is no error in the axis, and there are no motion commands active on the axis.

Homing
The axis is currently executing a homing procedure. The state *Homing* is left automatically as soon as the procedure is completed (with an error or successfully). No other motion commands except *MC_Stop_ISD51x* can be issued when the axis is in state *Homing*. This state can only be entered out of states *Disabled* (depending on the homing method) or *Standstill*.

Stopping
The axis changes to state *Stopping* when function block *MC_Stop_ISD51x* is called when *Execute* is *TRUE*. It can be used as a kind of emergency stop functionality, or in exception situations. No other motion function block can take over control of the axis if the input *Execute* is still *TRUE* and the servo drive has not reached velocity *0*. If both conditions are met, the axis changes to state *Standstill*.
For more information on this state, see *chapter 6.5.5.2 MC_Stop_ISD51x*.

Discrete motion
The axis processes a motion command that leads to a specific position, that is, it is running position controlled. The axis leaves the state *Discrete Motion* and automatically changes to state *Standstill* as soon as the specific target is reached.

Synchronized motion
The axis is processing a motion related to a master axis (guide value). The axis only leaves this state when the currently ongoing (synchronized) motion is aborted by another (non-synchronized) motion command.

6

Continuous motion
The axis is processing an endlessly ongoing motion command. The axis only leaves this state, when the currently ongoing (continuous) motion is aborted by another (non-continuous) motion command.

6.5.2 General Input/Output Behavior

6.5.2.1 Function Blocks with Execute Input

The parameters are used with the rising edge of the *Execute* input. To modify any parameter, change the input parameter(s) and trigger the *Execute* input again. Do not change references or in/out variables while *Busy* is *TRUE*. If the functionality needs an abort first (for example, internal closing of a file or abort of an SDO transfer), the abort is processed first and the functionality with the new parameters is started afterwards. This also includes checks for validity of the inputs (for example, when retriggering with invalid inputs, the abort is done first and afterwards an error is signaled at the function block).

Retrigger the *Execute* input to use the explicit *Abort* input (see *chapter 6.5.4.24 DD_Trace_ISD51x*).

Output exclusivity
The outputs *Busy*, *Done*, *Error*, and *CommandAborted* are mutually exclusive: only 1 of them can be *TRUE* on 1 instance of a function block at the same time. If *Execute* is *TRUE*, 1 of these outputs must be *TRUE*. Only 1 of the outputs *Active*, *Error*, *Done*, and *CommandAborted* is set at the same time.

The outputs *Done*, *Error*, *ErrorInfo*, and *CommandAborted* are reset with the falling edge of *Execute*. However, the falling edge of *Execute* does not stop or even influence the execution of the actual function block. Even if *Execute* was reset before the function block completed, 1 of the outputs *Done*, *Error*, or *CommandAborted* is set for exactly 1 cycle. If an instance of a function block receives a new rising edge at the *Execute* input before it finished (as a series of commands on the same instance), the function block will not return any feedback, such as *Done* or *CommandAborted* for the previous action.

Done
The *Done* output is set when the commanded action has been completed successfully (see case 3 in *Illustration 6.19*. If an error occurs during the execution of the functionality, the *Error* output is set, while the other outputs are *FALSE* (see case 2 in *Illustration 6.19*).

With multiple function blocks working at the same axis in a sequence, the following applies: When 1 movement of an axis is interrupted with another movement on the same axis without having reached the final goal, *Inxxx* of the 1st function block is not set, but *CommandAborted* is set (see case 1 in *Illustration 6.19*).

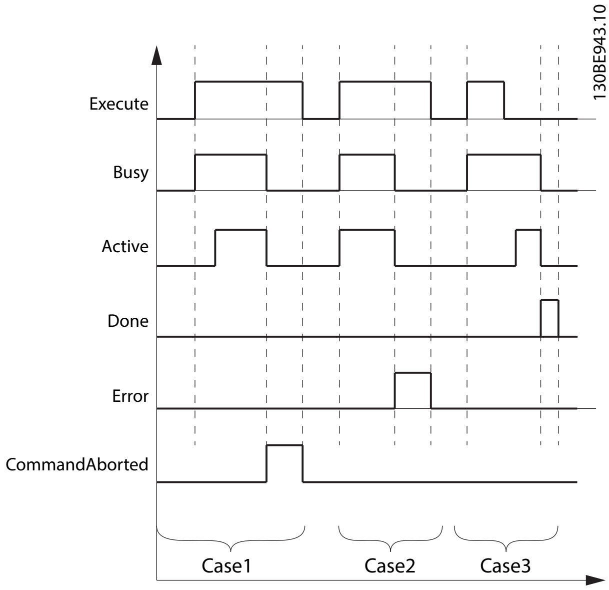


Illustration 6.19 Behavior of the Execute/Done Style Function Blocks

Busy
The output *Busy* reflects that the function block is not finished and new output values can be expected. *Busy* is set at the rising edge of *Execute* and reset when 1 of the outputs *Done*, *CommandAborted*, or *Error* is set. The function block must be kept in the active loop of the application program for at least as long as *Busy* is *TRUE*, because the outputs may still change.

Inxxx
The outputs *InVelocity*, *InGear*, *InTorque*, and *InSync* (from now on referred to as Inxxx) behave differently to the *Done* output.
If the function block is active, Inxxx is *TRUE* when the set or actual value (refer to *DD_ValueSourceSettings_ISD51x* for this selection) equals the commanded value. It is *FALSE* when, later, they are unequal. For example, the *InVelocity* output is *TRUE* when the set or actual velocity (see *DD_ValueSourceSettings_ISD51x*) is equal to the commanded velocity. This is similar for *InGear*, *InTorque*, and *InSync* outputs in the applicable function blocks.

With multiple function blocks working on the same axis in a sequence, the following applies: When 1 movement of an axis is interrupted with another movement on the same axis without having reached the final goal, *Done* of the 1st function block is not set, but *CommandAborted* is set (see case 1 in *Illustration 6.20*).

If an error occurs during the execution of the functionality, the *Error* output is set, while the other outputs are *FALSE* (see case 2 in *Illustration 6.20*).

Inxxx is updated even if *Execute* is *FALSE* if the function block has control of the axis (*Active* and *Busy* are *TRUE* (see case 3 in *Illustration 6.20*).

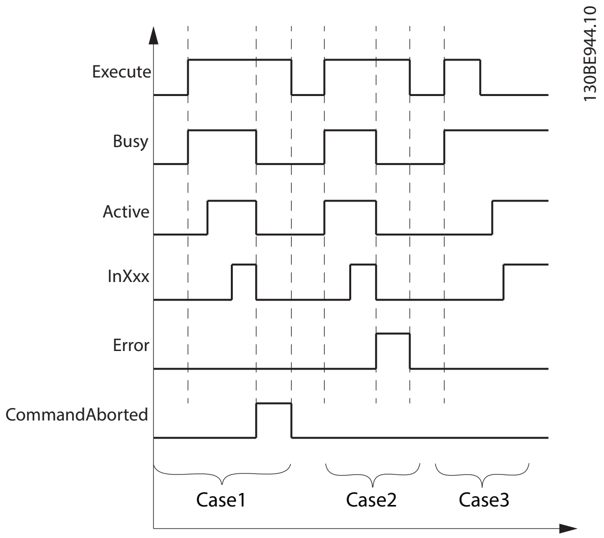


Illustration 6.20 Behavior of the Execute/Inxxx Style Function Blocks

CommandAborted
CommandAborted is *TRUE* when a commanded motion is interrupted by another motion command. The reset-behavior of *CommandAborted* is like that of the output *Done*. When *CommandAborted* becomes *TRUE*, the other output signals are reset (see case 1 in *Illustration 6.19* and *Illustration 6.20*).

6.5.2.2 Function Blocks with Enable Input

The parameters are used with the rising edge of the *Enable* input and can be modified continuously.

Output exclusivity
The outputs *Valid* and *Error* are mutually exclusive, meaning that only 1 of them can be *TRUE* on 1 instance of a function block at the same time. The outputs *Valid*, *Busy*, *Error*, and *ErrorInfo* are reset with the falling edge of *Enable* as soon as possible, depending on the cycle times of the PLC program and the point where *Enable* is reset.

Busy
The output *Busy* reflects that the function block is working and new output values can be expected. *Busy* is set at the rising edge of *Enable* and stays set while the function block is performing an action. Keep the function block in the active loop of the application program for at least as long as *Busy* is *TRUE*, because the outputs may still change.

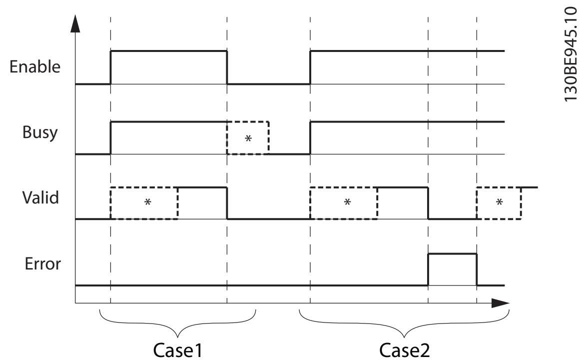


Illustration 6.21 Behavior of the Enable Style Function Blocks
**** Can take some time, depending on the functionality and the internal state***

Valid
The *Valid* output is *TRUE* if a valid output value is available, and the *Enable* input is *TRUE* (see case 1 in *Illustration 6.21*). The relevant output value can be refreshed if the input *Enable* is *TRUE*. If there is a function block error, the output is not valid (*Valid* is set to *FALSE*). When the error condition disappears, the values reappear and the output *Valid* is set again (see case 2 in *Illustration 6.21*).

6.5.2.3 Error Indication

All function blocks have 2 outputs that deal with errors that can occur while executing functionality. These outputs are defined as:

- Error: Rising edge of *Error* informs that an error occurred during the execution of the function block.
- ErrorInfo: Structure of *DD_ERROR_ISD51x*, which consists of the elements detailed in *Table 6.1*.

Variable name	Data type	Default value	Description
ErrorID	WORD	ISD51x_ERR_NO_ERROR	Contains unique error identification. The error constant definitions are listed in the constants list *FB_ErrorConstants*.
InstanceID	UDINT	0	Contains unique identifier of the function block instance that caused the error. Can be used for central error handling within a PLC project.

6

Variable name	Data type	Default value	Description
NodeID	UINT	0	Contains the node number of the device targeted by that function block.

Table 6.1 DD_ERROR_ISD51x

Instance errors do not always result in an axis error (bringing the axis to state *ErrorStop*).

6.5.2.4 Technical Units in the PLC library

The ISD 510 servo drives provide parameters (the factor group) to set the length, velocity, and acceleration units for application relevant units (see *chapter 7.4 Factor Group Objects*). There are 2 methods to write these parameters:

- Program them using a *WriteParameter* function block.
- Use the built-in functionality of the PLC development environment that writes parameters when the device starts up (recommended).

There is no scaling within the function blocks, so all parameters that are handed over to the inputs of the function blocks are directly sent to the servo drive to prevent rounding errors inside the servo drive. Therefore, consider the settings of the servo drive factor group parameters.

By using the servo drive *Velocity* or *Acceleration* factor parameters, the velocity and acceleration units are no longer derivatives of the length unit.

Unit	Data size	Data type
Position	INTEGER32	DINT
Distance	INTEGER32	DINT
Velocity	INTEGER32	DINT
Acceleration	Unsigned INTEGER32	UDINT
Deceleration	Unsigned INTEGER32	UDINT
Torque (for limiting issues)	Unsigned INTEGER16	UINT
Torque (for target torque)	INTEGER16	INT

Table 6.2 Data Types used for Physical Inputs and Outputs

Position versus distance
Position is a value defined within a coordinate system. *Distance* is a relative measure related to technical units and is the difference between 2 positions.

Sign rules
Acceleration and *Deceleration* are always positive values. *Position* and *Distance* can be both positive and negative. *Velocity* can be positive and negative, or can be positive value only, for example, if the direction is determined by other means.

Positive or negative direction is always related to the positive or negative direction parameterized in the servo drive (see *chapter 7.7.8 Parameters 51-02, 52-04, and 52-49: Application Settings (0x2016)*).

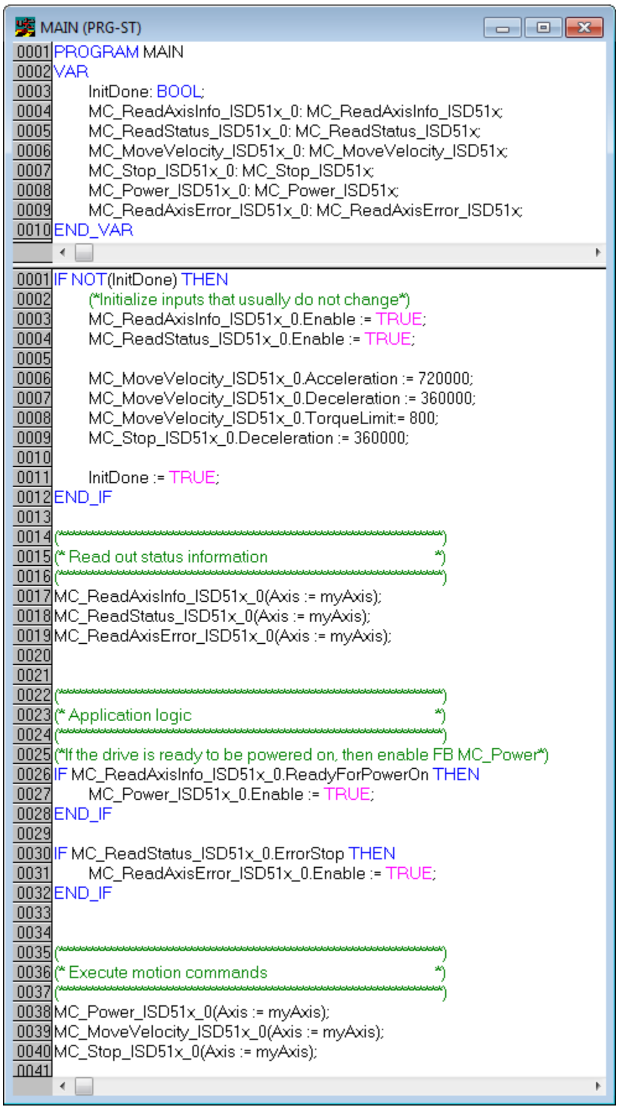
6.5.3 Programming Guidelines

Recommendations for implementation:

- Initialize parameters that usually do not change only once at the beginning of the program. In Automation Studio™, use the *_INIT* section.
- Call up function blocks that provide status or error information with *Enable* input at the beginning of the program.
- Use 1 instance of the function block *MC_Power_ISD51x* for every axis to control its power stage. Call up this function block in every PLC cycle.
- Use 1 instance of the function block *DD_Power_SAB* for every SAB to control the DC-link voltage on the output lines. Call up this function block in every PLC cycle.
- Call up function blocks that execute (motion) commands at the end of the program.
- Do not use any POUs from the library (folder) *Intern_51x*.
- Do not change the reference to the axis on a function block when it is busy.

Illustration 6.22 shows sample code for TwinCAT®.

6



```
PROGRAM MAIN
VAR
    InitDone: BOOL;
    MC_ReadAxisInfo_ISD51x_0: MC_ReadAxisInfo_ISD51x;
    MC_ReadStatus_ISD51x_0: MC_ReadStatus_ISD51x;
    MC_MoveVelocity_ISD51x_0: MC_MoveVelocity_ISD51x;
    MC_Stop_ISD51x_0: MC_Stop_ISD51x;
    MC_Power_ISD51x_0: MC_Power_ISD51x;
    MC_ReadAxisError_ISD51x_0: MC_ReadAxisError_ISD51x;
END_VAR

IF NOT(InitDone) THEN
    (*Initialize inputs that usually do not change*)
    MC_ReadAxisInfo_ISD51x_0.Enable := TRUE;
    MC_ReadStatus_ISD51x_0.Enable := TRUE;

    MC_MoveVelocity_ISD51x_0.Acceleration := 720000;
    MC_MoveVelocity_ISD51x_0.Deceleration := 360000;
    MC_MoveVelocity_ISD51x_0.TorqueLimit:= 800;
    MC_Stop_ISD51x_0.Deceleration := 360000;

    InitDone := TRUE;
END_IF

(**********************************************************)
(* Read out status information                           *)
(**********************************************************)
MC_ReadAxisInfo_ISD51x_0(Axis := myAxis);
MC_ReadStatus_ISD51x_0(Axis := myAxis);
MC_ReadAxisError_ISD51x_0(Axis := myAxis);


(**********************************************************)
(* Application logic                                      *)
(**********************************************************)
(*If the drive is ready to be powered on, then enable FB MC_Power*)
IF MC_ReadAxisInfo_ISD51x_0.ReadyForPowerOn THEN
    MC_Power_ISD51x_0.Enable := TRUE;
END_IF

IF MC_ReadStatus_ISD51x_0.ErrorStop THEN
    MC_ReadAxisError_ISD51x_0.Enable := TRUE;
END_IF


(**********************************************************)
(* Execute motion commands                                *)
(**********************************************************)
MC_Power_ISD51x_0(Axis := myAxis);
MC_MoveVelocity_ISD51x_0(Axis := myAxis);
MC_Stop_ISD51x_0(Axis := myAxis);
```

Illustration 6.22 Sample Code for TwinCAT®

6.5.4 Drive – Administrative

6.5.4.1 AXIS_REF_ISD51x

This function block shows the state of an ISD 510 servo drive. It handles the PDO communication and the internal state. To use the servo drive-related function blocks, instantiate 1 function block for each servo drive used. Also, connect the inputs and outputs to the objects that are mapped in the development environment for synchronous communication. See *chapter 6.3.1.2 Creating a TwinCAT® Project* and *chapter 6.4.1.2 Creating an Automation Studio ™ Project*.

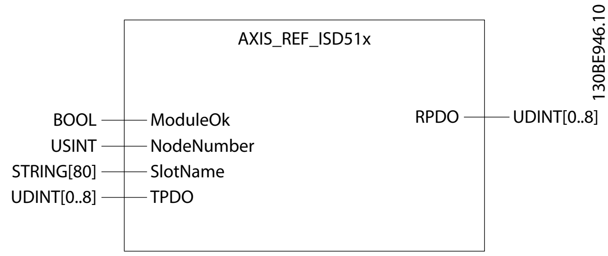


Illustration 6.23 AXIS_REF_ISD51x in Automation Studio™

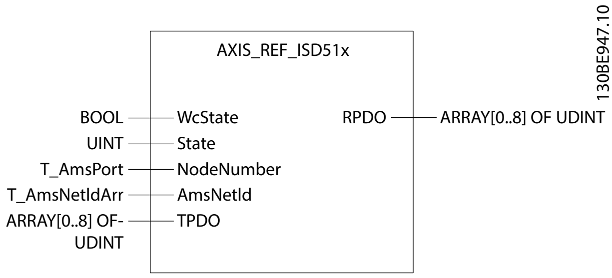


Illustration 6.24 AXIS_REF_ISD51x in TwinCAT®

6.5.4.2 MC_Power_ISD51x

This function block controls the power stage (*On* or *Off*). The *Enable* input enables the power stage in the servo drive and not the function block itself.

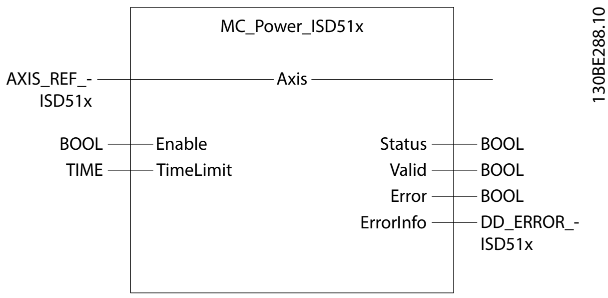


Illustration 6.25 MC_Power_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	*Chapter 6.5.4.1 AXIS_REF_ISD51x*		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	If this input is *TRUE*, power is being enabled.
TimeLimit	TIME	t#0ms	Time after which an error is signaled, if *Status* has not changed to *TRUE* while *Enable* is *TRUE*. Set the value to *0* to disable the time limit.
VAR_OUTPUT			
Status	BOOL		Effective state of the power stage.
Valid	BOOL		If *TRUE*, the function block has a valid set of outputs.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.3 MC_Power_ISD51x

If the *MC_Power_ISD51x* function block is called up and the variable *Enable* is *TRUE* while in PLCopen state *Disabled*, the axis state changes to *Standstill*.

Error is set to *TRUE* if the *Enable* input is *TRUE* for the time specified in the input *TimeLimit*, while the *Status* remains *FALSE*. It indicates a hardware problem with the power stage. If power fails (also during operation), it generates a transition to the *ErrorStop* state.

Only 1 *MC_Power_ISD51x* function block can be issued per axis.

The *Enable* input in this function block is not an *Enable* input as described in *chapter 6.5.2.2 Function Blocks with Enable Input*. Therefore, the general rules for the *Enable* input do not apply here. This function block is implicitly enabled. The *Enable* input of this function block controls the power stage of the servo drive. All outputs are always updated (so *Valid* can be *TRUE*, even if *Enable* is *FALSE*).

The input *TimeLimit* represents the maximal duration of functionality. If the *TimeLimit* is exceeded during switching on the servo drive, an *Error* is signaled on the outputs. However, the functionality according to the *Enable* input is continued. This means that the function block still tries to enable the servo drive, if *Enable* is set to *TRUE*, and/or to disable the servo drive if *Enable* is set to *FALSE*. If the servo drive starts reacting again, the *Error* output can change to *FALSE* again without a new rising edge of *Enable*. Set the value to *0* to disable the limiting functionality.

The command is transferred immediately, but it can take some time until the axis is powered up and the output *Status* becomes *TRUE*.

6.5.4.3 MC_Reset_ISD51x

This function block commands the transition from the state *ErrorStop* to *Disabled* by resetting all internal axis-related errors. It does not affect the output of the function block instances.
The command is transferred and executed immediately.

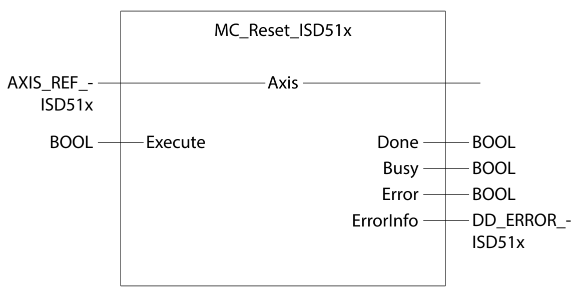


Illustration 6.26 MC_Reset_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Resets all internal axis-related errors.
VAR_OUTPUT			
Done	BOOL		Error was reset and state *Disabled* reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.4 MC_Reset_ISD51x

6.5.4.4 MC_ReadStatus_ISD51x

This function block returns the detailed status of the state diagram (see *Illustration 6.18*) of the selected axis.
The output data is available immediately.

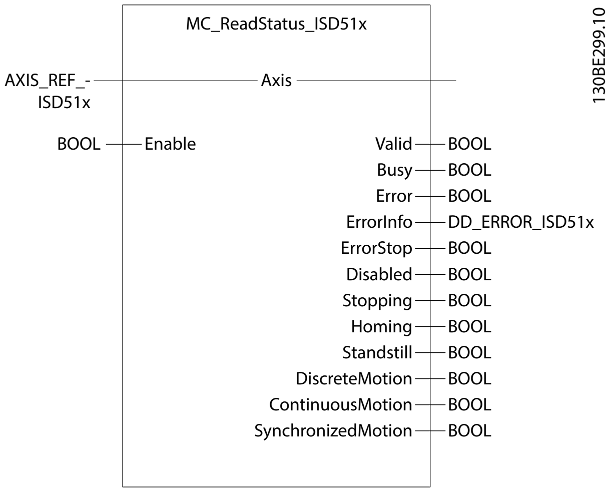


Illustration 6.27 MC_ReadStatus_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameters continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid set of outputs.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
ErrorStop	BOOL		*TRUE* if the axis is in state *ErrorStop*.
Disabled	BOOL		*TRUE* if the axis is in state *Disabled*.
Stopping	BOOL		*TRUE* if the axis is in state *Stopping*.
Homing	BOOL		*TRUE* if the axis is in state *Homing*.
Standstill	BOOL		*TRUE* if the axis is in state *StandStill*.
Discrete-Motion	BOOL		*TRUE* if the axis is in state *DiscreteMotion*.
Continuous-Motion	BOOL		*TRUE* if the axis is in state *ContinuousMotion*.
Synchronized-Motion	BOOL		*TRUE* if the axis is in state *SynchronizedMotion*.

Table 6.5 MC_ReadStatus_ISD51x

6.5.4.5 MC_ReadAxisError_ISD51x

This function block presents general axis errors that are unrelated to the function blocks (for example, overtemperature on the axis). The output *AxisErrorID* gives the last error that occurred in the axis (see *chapter 9.2.2 Error Codes*).
The output data must be read from the device and is therefore not immediately available.

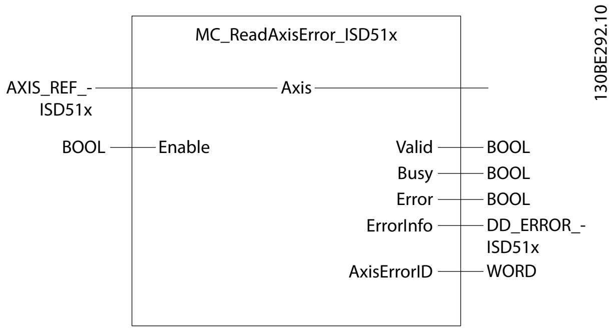


Illustration 6.28 MC_ReadAxisError_ISD51x

6

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameters continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		A valid output is available at the function block.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AxisErrorID	WORD		The value of the axis error. Available in the list of constants *AxisErrorCodes*.

Table 6.6 MC_ReadAxisError_ISD51x

6.5.4.6 DD_ReadAxisWarning_ISD51x

This function block presents general axis warnings that are unrelated to the function blocks (for example, warning of high temperature on the axis). The output *AxisWarningID* gives the last warning that occurred in the axis (see *chapter 9.2.2 Error Codes*).
The output data needs to be read from the device and is therefore not immediately available.

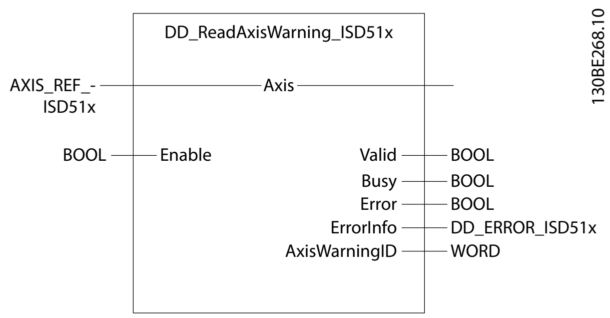


Illustration 6.29 DD_ReadAxisWarning_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the axis information continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AxisWarningID	WORD		The value of the axis warning. Available in the list of constants: *AxisErrorCodes*.

Table 6.7 DD_ReadAxisWarning_ISD51x

6.5.4.7 DD_ReadVersion_ISD51x

This function block reads the firmware version and the serial number of the servo drive. *Done* is *TRUE* when the data-outputs are valid. The version number consists of a major, a minor, and beta version number (see *chapter 7.22.4 Parameters 15-40, 15-41, and 15-43: Version log (0x4000)*).
The output data needs to be read from the device and is therefore not immediately available.

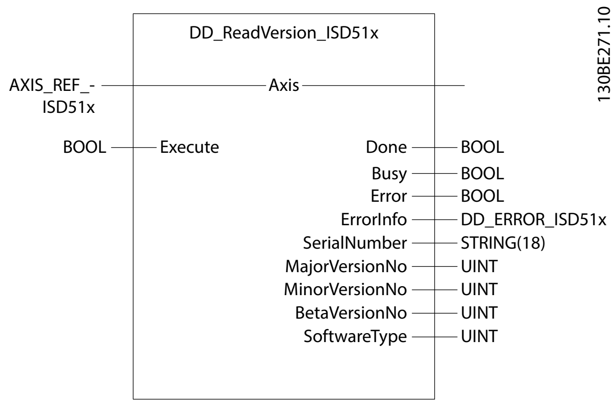


Illustration 6.30 DD_ReadVersion_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Read the information at rising edge.
VAR_OUTPUT			
Done	BOOL		The values have successfully been read from the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
SerialNumber	STRING[18]		Serial number of the axis.
MajorVersionNo	UINT		Major firmware version number.
MinorVersionNo	UINT		Minor firmware version number.
BetaVersionNo	UINT		Beta firmware version number.
SoftwareType	UINT		Loaded software type.

Table 6.8 DD_ReadVersion_ISD51x

6.5.4.8 DD_UpdateFirmware_ISD51x

This function block updates the firmware of the axis. Only update the firmware when the servo drive is in an unpowered state. Carry out a power cycle to use the updated firmware. For more details on the firmware update, see *chapter 2.2.1 Firmware Update*.

After the update process, check the firmware version (see *chapter 6.5.4.7 DD_ReadVersion_ISD51x*).

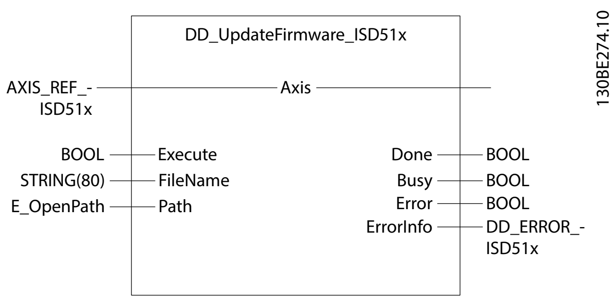


Illustration 6.31 DD_UpdateFirmware_ISD51x for TwinCAT®
(See *Table 6.9* for other available development environments.

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Starts the firmware update at a rising edge.
FileName	STRING [80]	''	Filename of the firmware file on the PLC.
pDevice	UDINT	0	Automation Studio™ only: Pointer to the device name on which the firmware file is located.
Path	E_OpenPath	PATH_GENERIC	TwinCAT® only: The variable of this type selects generic or 1 of the TwinCAT® system paths on the target device.
VAR_OUTPUT			
Done	BOOL		The firmware file has successfully been transferred. Power cycle the axis to enable the new firmware.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.9 DD_UpdateFirmware_ISD51x

6

6.5.4.9 MC_ReadAxisInfo_ISD51x

This function block returns detailed information related to an axis, such as modes and certain status information. The output data is available immediately.

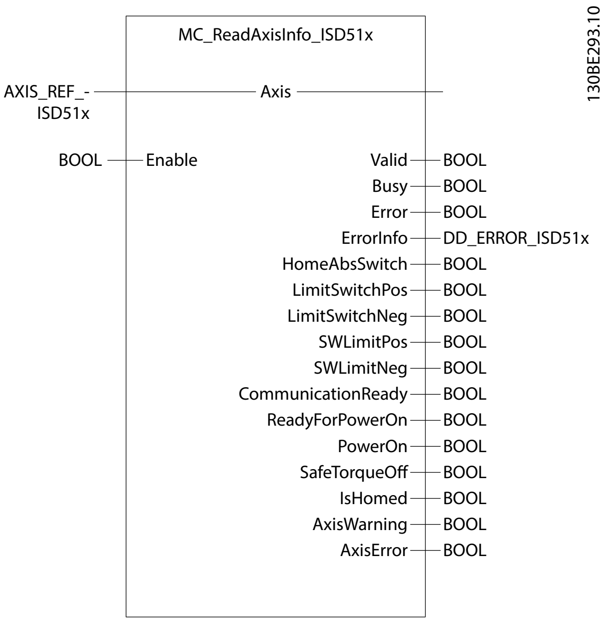


Illustration 6.32 MC_ReadAxisInfo_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the axis information continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid set of outputs.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
HomeAbsSwitch	BOOL		Digital home switch input is active.
LimitSwitch-Pos	BOOL		Positive hardware end switch is active.
LimitSwitch-Neg	BOOL		Negative hardware end switch is active.
SWLimitPos	BOOL		Positive software limit is active.
SWLimitNeg	BOOL		Negative software limit is active.
Communica-tionReady	BOOL		Network is initialized and ready for communication.
ReadyFor-PowerOn	BOOL		Drive is ready to be enabled (power on).
PowerOn	BOOL		*TRUE* shows that the power stage is switched on.
SafeTorqueOff	BOOL		*TRUE*: STO is activated = Safety voltage is missing.
IsHomed	BOOL		The absolute reference position is known for the axis. Axis is homed.
AxisWarning	BOOL		Warning(s) on the axis is/are present.
AxisError	BOOL		Error(s) on the axis is/are present.

Table 6.10 MC_ReadAxisInfo_ISD51x

6.5.4.10 MC_ReadMotionState_ISD51x

This function block returns the detailed status of the axis related to the motion currently in progress. See *chapter 7.22.3.1 Parameter 51-70: Constant Velocity Window (0x2030)* and *chapter 7.22.3.2 Parameter 51-71: Constant Velocity Window Time (0x2031)* for settings of the detailed evaluation of the information.
The output data is available immediately.

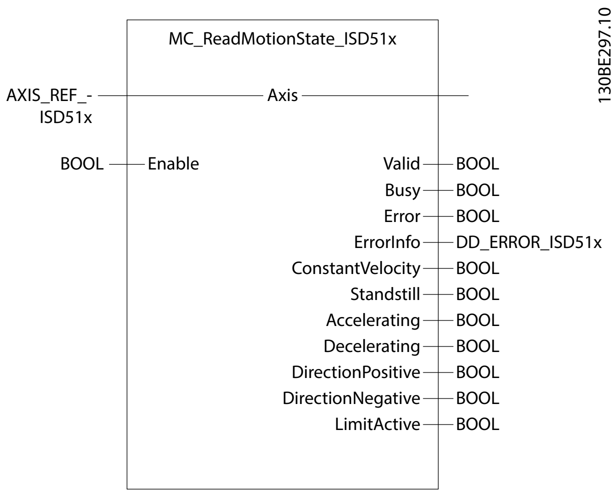


Illustration 6.33 MC_ReadMotionState_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameters continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid set of outputs.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
ConstantVelocity	BOOL		Velocity is constant. Velocity may be 0. For the actual value, a window is applicable, see *chapter 7.22.3.1 Parameter 51-70: Constant Velocity Window (0x2030)* and *chapter 7.22.3.2 Parameter 51-71: Constant Velocity Window Time (0x2031)*.
Standstill	BOOL		Velocity is constant and value *0*.
Accelerating	BOOL		Increasing the absolute value of the velocity.
Decelerating	BOOL		Decreasing the absolute value of the velocity.
Direction-Positive	BOOL		The position is increasing.
Direction-Negative	BOOL		The position is decreasing.
LimitActive	BOOL		An internal limit is active.

Table 6.11 MC_ReadMotionState_ISD51x

6.5.4.11 MC_ReadActualPosition_ISD51x

This function block provides the value of the actual position if *Enable* is set (see *chapter 7.7.5 Parameter 50-03: Position Actual Value (0x6064)*). *Valid* is *TRUE* when the data-output *Position* is valid. If *Enable* is reset, the data loses its validity and all outputs are reset, regardless of whether new data is available.
The output *Position* is a signed value.
The output data is available immediately.

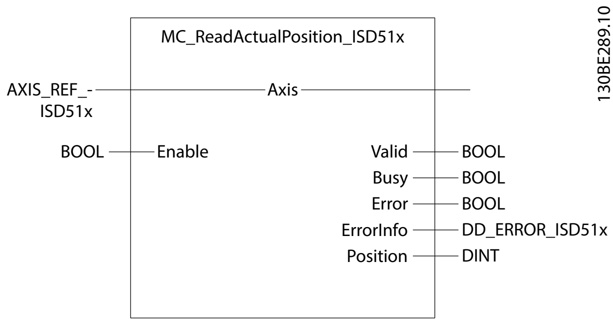


Illustration 6.34 MC_ReadActualPosition_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Position	DINT		New absolute position [user-defined position unit].

Table 6.12 MC_ReadActualPosition_ISD51x

6.5.4.12 MC_ReadActualVelocity_ISD51x

This function block provides the value of the actual velocity if *Enable* is set (see *chapter 7.11.3 Parameter 50-04: Velocity Actual Value (0x606C)*). *Valid* is *TRUE* when the data-output *Velocity* is valid. If *Enable* is reset, the data loses its validity and all outputs are reset, regardless of whether new data is available.
The output *Velocity* is a signed value.
The output data is available immediately.

6

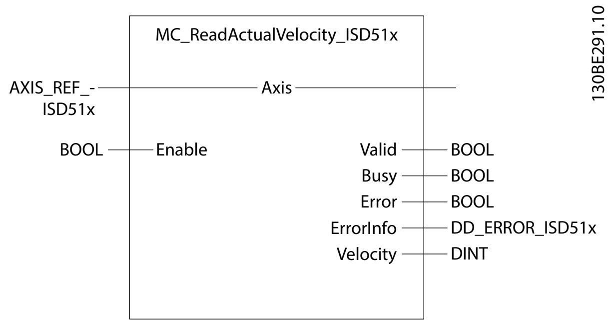


Illustration 6.35 MC_ReadActualVelocity_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Velocity	DINT		The value of the actual velocity [user-defined velocity unit].

Table 6.13 MC_ReadActualVelocity_ISD51x

6.5.4.13 MC_ReadActualTorque_ISD51x

This function block provides the value of the actual torque if *Enable* is set (see *chapter 7.12.5 Parameter 52-31: Torque Actual Value (0x6077)*). *Valid* is *TRUE* when the data-output *Torque* is valid. If *Enable* is reset, the data loses its validity, all outputs are reset, and new data is available.
The output *Torque* is a signed value.
The output data is available immediately.

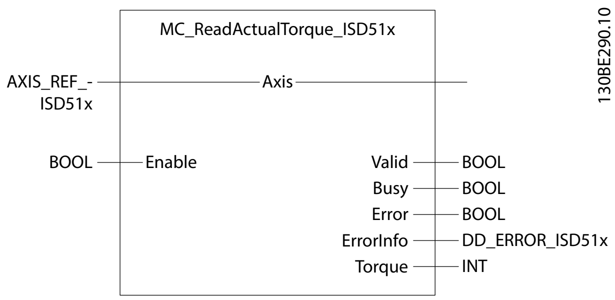


Illustration 6.36 MC_ReadActualTorque_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Torque	INT		The value of the actual torque or force [per thousand of rated torque].

Table 6.14 MC_ReadActualTorque_ISD51x

6.5.4.14 MC_ReadDigitalInput_ISD51x

This function block provides the value of the specified input. It is not guaranteed that pulses shorter than the fieldbus cycle time on the digital signal can be seen on the function block.
The output data is available immediately.

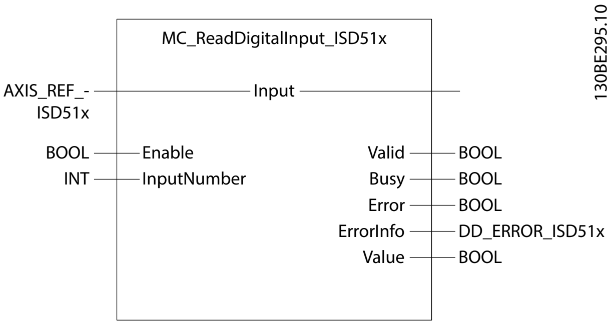


Illustration 6.37 MC_ReadDigitalInput_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Input	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the selected input signal continuously while enabled.
InputNumber	INT	1	Selects the input. Value range: 1;2
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Value	BOOL		The value of the selected input signal

Table 6.15 MC_ReadDigitalInput_ISD51x

6.5.4.15 DD_ReadAnalogInput_ISD51x

This function block reads the value of the analog input. The output data needs to be read from the device and is therefore not immediately available.

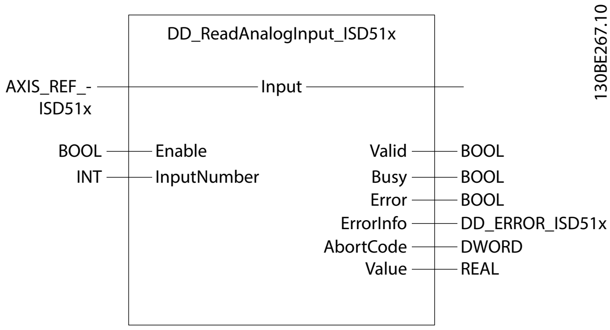


Illustration 6.38 DD_ReadAnalogInput_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Input	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the selected input signal continuously while enabled.
InputNumber	INT	1	Selects the input. Value range: 1;2
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Value	REAL		The value of the selected input signal
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.

Table 6.16 DD_ReadAnalogInput_ISD51x

6.5.4.16 MC_ReadDigitalOutput_ISD51x

This function block provides the value of the digital output. It is not guaranteed that short pulses on the digital signal can be seen on the function block.
The output data is available immediately.

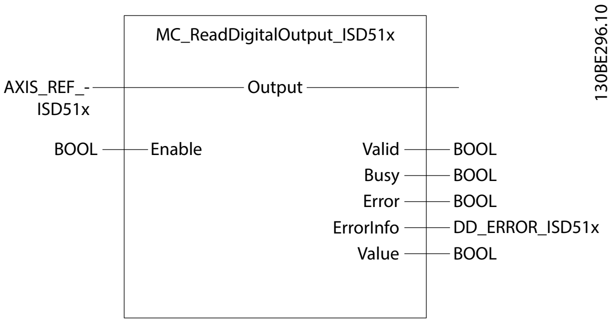


Illustration 6.39 MC_ReadDigitalOutput_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Output	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the output signal continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Value	BOOL		The value of the output signal.

Table 6.17 MC_ReadDigitalOutput_ISD51x

6.5.4.17 DD_WriteDigitalOutput_ISD51x

This function block writes a value to the digital output of the axis. If *Enable* is *TRUE*, the input *Value* is evaluated. If the input *Enable* is *FALSE*, the last value is used. This function block only works if the usage of the digital output is configured accordingly (see *chapter 7.21.5 Parameter 52-05: Digital Output Configuration (0x2FFF)*).

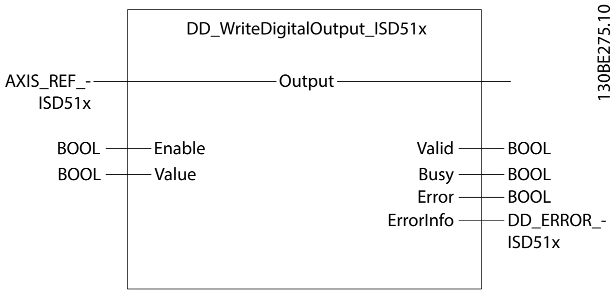


Illustration 6.40 DD_WriteDigitalOutput_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Output	AXIS_REF_ISD51x		Reference to the axis/signal output. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Write the value of the digital output.
Value	BOOL	FALSE	Value of the output: *TRUE* = set *FALSE* = clear
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.18 DD_WriteDigitalOutput_ISD51x

6.5.4.18 MC_ReadParameter_ISD51x and MC_ReadBoolParameter_ISD51x

This function block returns the value of the specified parameter. The time taken until valid data is available at the output depends on several factors, for example:

- PLC system
- Cycle times
- Amount of acyclic data requested

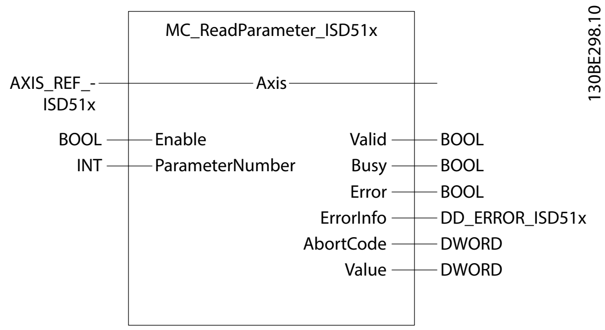


Illustration 6.41 MC_ReadParameter_ISD51x

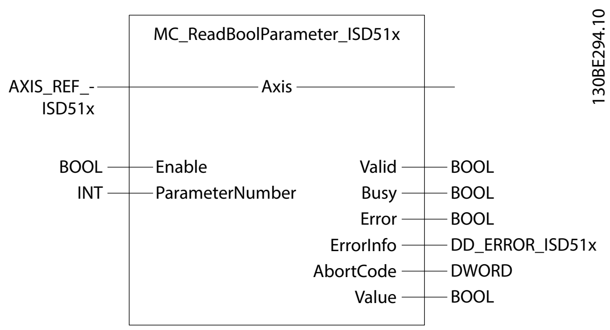


Illustration 6.42 MC_ReadBoolParameter_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
Parameter-Number	INT	0	Number of the parameter. See *Table 6.20*. All other numbers are not allowed and lead to an error.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: *SdoAbortCodes*.
Value	DWORD/BOOL		Value of the specified parameter in the data type, as specified in *Table 6.20*.

Table 6.19 MC_ReadParameter_ISD51x and MC_ReadBoolParameter_ISD51x

Para-meter Number	Name	Data type	R/W	Description
1	Comman-dedPosition	DINT	R	Commanded position
2	SWLimitPos	DINT	R/W	Positive software limit switch position
3	SWLimitNeg	DINT	R/W	Negative software limit switch position
4	EnableLi-mitPos	BOOL	R	Enable positive software limit switch
5	EnableLi-mitNeg	BOOL	R	Enable negative software limit switch
7	MaxPosi-tionLag	UDINT	R/W	Maximum position lag
8	MaxVelocity-System	REAL	R	Maximum allowed velocity of the axis in the motion system
9	MaxVeloci-tyAppl	UDINT	R/W	Maximal allowed velocity of the axis in the application
10	ActualVe-locity	DINT	R	Actual velocity
11	Comman-dedVelocity	DINT	R	Commanded velocity
13	MaxAcceler-ationAppl	UDINT	R/W	Maximal allowed acceleration of the axis in the application
15	MaxDeceler-ationAppl	UDINT	R/W	Maximal allowed deceleration of the axis in the application

Table 6.20 Parameters for MC_ReadParameter_ISD51x, MC_ReadBoolParameter_ISD51x and MC_WriteParameter_ISD51x

6.5.4.19 DD_ReadParameter4_ISD51x

This function block asynchronously reads general objects of up to 4 bytes from the object dictionary of the axis if the *Enable* input is set. For the index and sub-index of the parameters, see *chapter 7 Servo Drive Parameter Description*. The output data is read from the device and is therefore not immediately available.

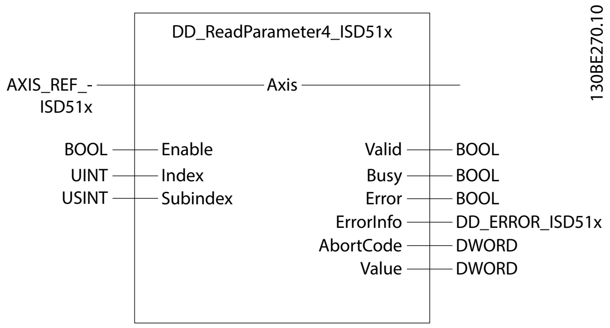


Illustration 6.43 DD_ReadParameter4_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
Index	UINT	0	Index of the object to be read.
Subindex	USINT	0	Sub-index of the object to be read.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.
Value	DWORD		Value of the specified parameter.

Table 6.21 DD_ReadParameter4_ISD51x

6.5.4.20 DD_ReadParameter_ISD51x

This function block asynchronously reads general objects from the object dictionary of the axis if the *Enable* input is set. For the index and sub-index of the parameters, see *chapter 7 Servo Drive Parameter Description*.
The output data is read from the device and is therefore not immediately available.

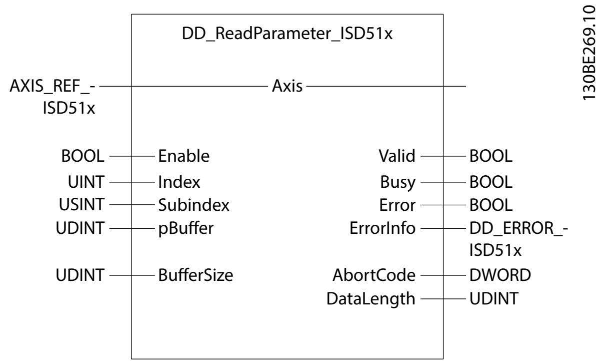


Illustration 6.44 DD_ReadParameter_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
Index	UINT	0	Index of the object to be read.
Subindex	USINT	0	Sub-index of the object to be read.
pBuffer	UDINT	0	Pointer to a buffer where the read data will be placed (use ADR() function).
BufferSize	UDINT	0	Maximum size of the read data (size of the provided buffer given in byte; use *SIZEOF*() function)
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.
DataLength	UDINT		Length of the read data [Byte].

Table 6.22 DD_ReadParameter_ISD51x

6.5.4.21 MC_WriteParameter_ISD51x

This function block modifies the value of the specified parameter.
It can take some time to write the data to the axis.

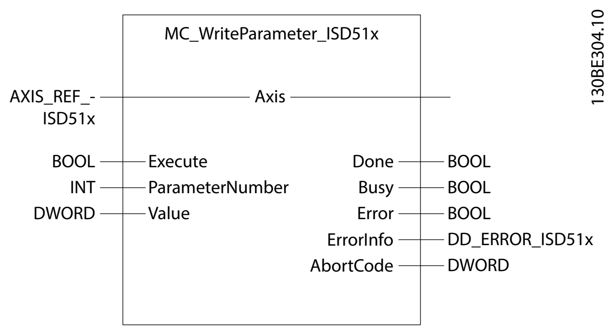


Illustration 6.45 MC_WriteParameter_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Write the value of the parameter at rising edge.
Parameter-Number	INT	0	Number of the parameter. See *Table 6.20*. All other numbers are not allowed and lead to an error.
Value	DWORD	0	New value of the specified parameter.
VAR_OUTPUT			
Done	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.

6

Variable name	Data type	Default value	Description
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.

Table 6.23 MC_WriteParameter_ISD51x

6.5.4.22 DD_WriteParameter_ISD51x

This function block asynchronously writes general objects to the object dictionary of the axis. For the index and sub-index of the parameters, see *chapter 7 Servo Drive Parameter Description*.
The data is written to the device asynchronously and is therefore not immediately available in the axis.

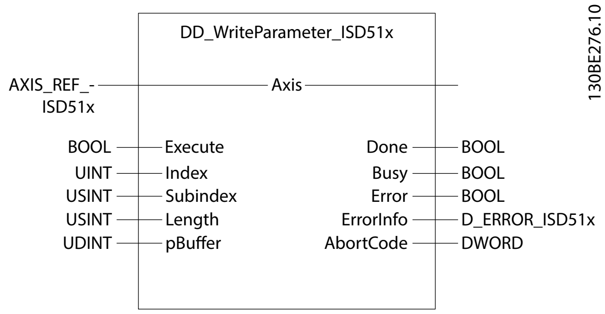


Illustration 6.46 DD_WriteParameter_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Write the value of the parameter at rising edge.
Index	UINT	0	Index of the object to be written.
Subindex	USINT	0	Sub-index of the object to be written.
Length	USINT	0	Length of the data to be written [Byte].
pBuffer	UDINT	0	Pointer to the buffer that contains the data to be written (use ADR() function).
VAR_OUTPUT			
Done	BOOL		The value has successfully been written to the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.

Table 6.24 DD_WriteParameter_ISD51x

6.5.4.23 DD_WriteParameter4_ISD51x

This function block asynchronously writes general objects of up to 4 bytes to the object dictionary of the axis. For the index and sub-index of the parameters, see *chapter 7 Servo Drive Parameter Description*.
The data is written to the device asynchronously and is therefore not immediately available in the axis.

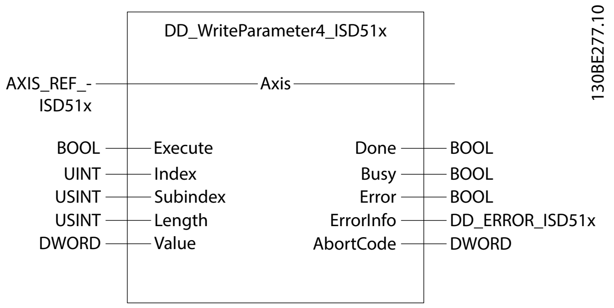


Illustration 6.47 DD_WriteParameter4_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Write the value of the parameter at rising edge.
Index	UINT	0	Index of the object to be written.
Subindex	USINT	0	Sub-index of the object to be written.
Length	USINT	0	Length of the data to be written [Byte].
Value	DWORD	0	New value of the specified parameter.
VAR_OUTPUT			
Done	BOOL		The value has successfully been written to the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.

Table 6.25 DD_WriteParameter4_ISD51x

6.5.4.24 DD_Trace_ISD51x

This function block is used to carry out a real-time trace within the axis using the settings given in the input variables. When the settings are sent and the trace started, the function block automatically polls the axis until the data has been recorded and then uploads the data automatically. Information about the status of the tracing can be monitored using the *Status* output (while *Busy* is *TRUE*).

Inside the axis, the data is sampled over time, meaning that there is an adjustable time difference between the samples (use inputs *SamplingRate* and *SubSampling*). For general tracing capabilities, refer to *chapter 2.7.2 Trace*.

The *Abort* input is used to abort the current functionality. The output *CommandAborted* is used to signal a successful aborting procedure (see *Illustration 6.48*). The abort of the functionality can take some time.

The behavior of output *CommandAborted* is similar to the *Done* output, only for a successful aborting procedure.

If there is an error during aborting, the *Error* output is set to *TRUE* and the error reason is indicated at output *ErrorInfo*. The output *CommandAborted* stays as *FALSE* in this case (see *Illustration 6.49*).

For function blocks with *Execute* as *Abort* input, the *Abort* input (and any other inputs of the function block) is only evaluated at a rising edge of *Execute*.

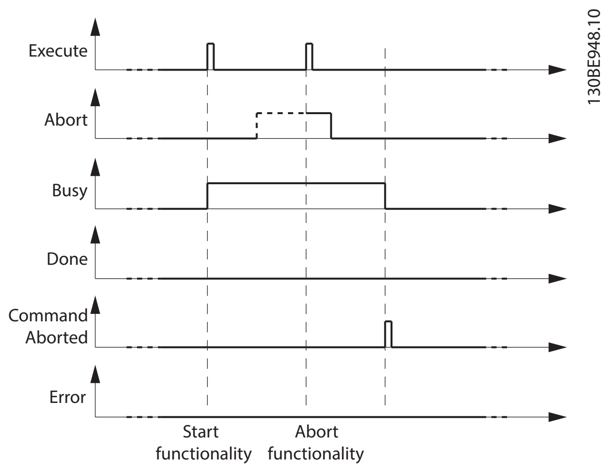


Illustration 6.48 Behavior of Successful Abort of Functionality

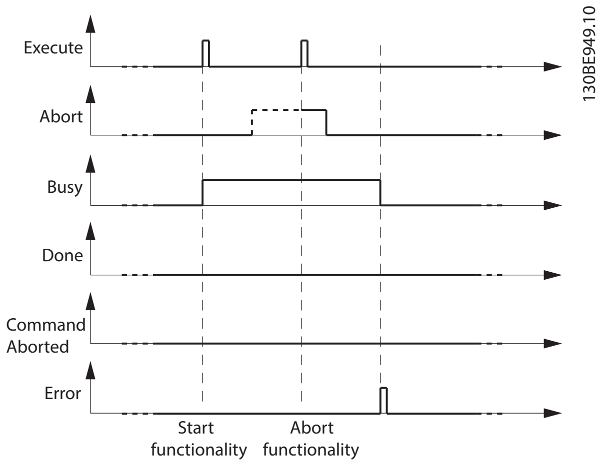


Illustration 6.49 Behavior When an Error Occurs During Abort of Functionality

A value of *0* is not allowed for the inputs *pTraceBuffer*, *TraceBufferSize*, *SampleCount*, and *SubSampling*. For the input *SignalIDs*, at least the 1st element of the array must be >0, otherwise the function block signals an error.
For the input *SampleCount*, the value, multiplied by the number of valid *SignalIDs*, must not exceed the *TraceBufferSize* and must not exceed the maximum available trace buffer size of the servo drive (see *chapter 2.7.2 Trace*).

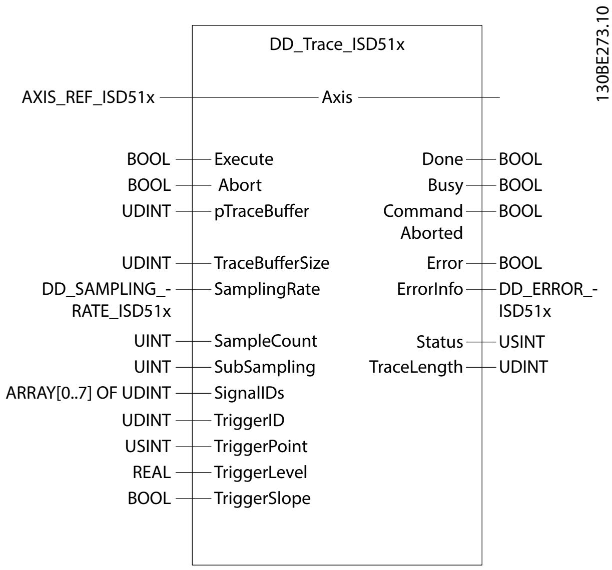


Illustration 6.50 DD_Trace_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.

Variable name	Data type	Default value	Description
VAR_INPUT			
Execute	BOOL	FALSE	Starts the trace functionality at rising edge and keeps on polling until the data is available.
Abort	BOOL	FALSE	Abort the ongoing trace. New values are only evaluated on a rising edge of Execute.
pTraceBuffer	UDINT	0	Reference to a buffer where the read trace data will be placed; Use ADR() function
TraceBufferSize	UDINT	0	Size of the trace buffer; use SIZEOF() function. Size of the provided buffer given in Byte.
SamplingRate	DD_SAMPLING_RATE_ISD51x	ddFastTask_ISD51x	Sampling rate of the trace.
SampleCount	UINT	4000	Number of samples to be traced per channel.
SubSampling	UINT	1	Multiplier to adjust time difference between trace samples.
SignalIDs	ARRAY[0..7] of UDINT	[0, 0, 0, 0, 0, 0, 0, 0]	IDs of the signals to be traced. Available in the list of constants: *AxisTraceSignals*.
TriggerID	UDINT	0	ID of the signal used for triggering. Available in the list of constants: *AxisTraceSignals*. Set the value to *0* for instant tracing.
TriggerPoint	USINT	10	Amount of pre-trigger history [in percentage]. Value range: 0–100
TriggerLevel	REAL	0.0	Level at which the device triggers (in trigger signal units).
TriggerSlope	BOOL	*TRUE*	*TRUE*: Triggers on rising slope. *FALSE*: Triggers on falling slope.
VAR_OUTPUT			
Done	BOOL		The trace has successfully been recorded and read from the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.

Variable name	Data type	Default value	Description
CommandAborted	BOOL		Trace has been aborted successfully.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Status	USINT		Holds the state of the tracing process (valid while *Busy* is *TRUE*): 0 = Configuring the trace. 1 = Trace configured and started. 2 = Waiting for trigger. 3 = Triggered; Waiting for completion of the trace. 4 = Uploading trace data. 5 = Trace data received successfully.
TraceLength	UDINT		Length of trace data that has been uploaded [Byte]. 1 sample (REAL) is 4 Bytes long

Table 6.26 DD_Trace_ISD51x

Name	Comment
ddRealTimeTask_ISD51x	Fastest possibility to record the samples. Sample time is either 100 µs or 125 µs, depending on the fieldbus cycle time.
ddFastTask_ISD51x	Sample time is either 200 µs or 250 µs, depending on the fieldbus cycle time.
ddSlowTask_ISD51x	Sample time is either 400 µs or 500 µs, depending on the fieldbus cycle time.

Table 6.27 Enumeration DD_SAMPLING_RATE_ISD51x

6.5.4.25 DD_BrakeHandling_ISD51x

This function block overwrites the status of the brake. The brake is controlled automatically by the axis, however it can be force lifted or engaged with this function block (for example, to move the shaft of the axis manually). The brake is a holding brake and is not intended to be used in normal operation.
The manipulation of the brake can take some time and is not applied to the axis immediately.

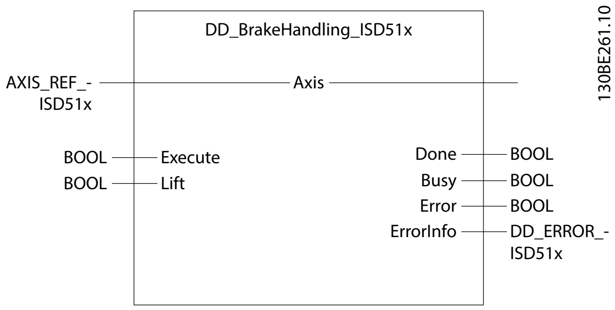


Illustration 6.51 DD_BrakeHandling_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Overwrites the brake state at rising edge.
Lift	BOOL	FALSE	*TRUE*: Overwrite the brake automatic and lift it. *FALSE*: Overwrite the brake automatic and release it.
VAR_OUTPUT			
Done	BOOL		The brake has successfully been modified.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.28 DD_BrakeHandling_ISD51x

6.5.4.26 DD_SelectControlParamSet_ISD51x

This function block selects the used control parameter set of the axis. If *Enable* is *TRUE*, the input *Set* is evaluated. If the input *Enable* is *FALSE*, the last value is used. This function block only works if the control is not overwritten by CAM mode (see *chapter 6.5.6.2 MC_CamIn_ISD51x*).
The manipulation of the control parameter set takes effect with the next network communication cycle.

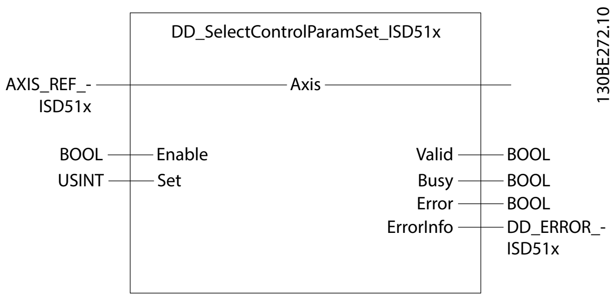


Illustration 6.52 DD_SelectControlParamSet_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Selects the control parameter set.
Set	USINT	1	Number of the control parameter set to be used. Values: 1 or 2
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.29 DD_SelectControlParamSet_ISD51x

6.5.4.27 MC_TouchProbe_ISD51x

This function block is used to record the axis position at a trigger event. This functionality is intended for single shot operation: The 1st event after the rising edge of *Execute* is recorded, however the events after that are ignored. 1 function block instance should represent exactly 1 trigger input.
The activation of the touch probe functionality can take some time.

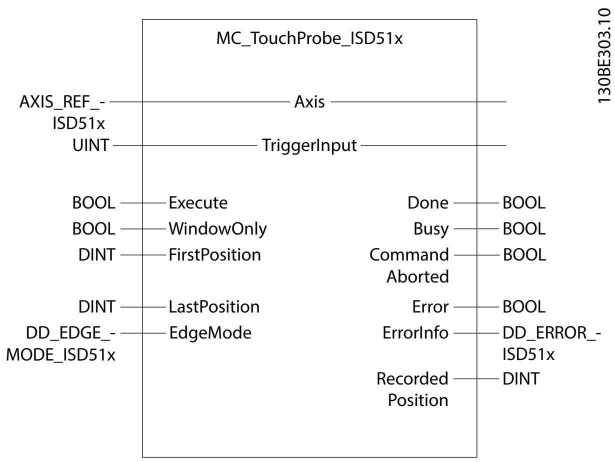


Illustration 6.53 MC_TouchProbe_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
TriggerInput	UINT		Reference to the trigger signal source.
VAR_INPUT			
Execute	BOOL	FALSE	Starts touch probe recording at rising edge.
WindowOnly	BOOL	FALSE	If *TRUE*, only trigger events within the defined window are accepted.
FirstPosition	DINT	0	Start position from where (positive direction) trigger events are accepted. The value itself is included in the window [user-defined position unit].
LastPosition	DINT	0	Stop position of the window. The value itself is included in the window [user-defined position unit].
EdgeMode	DD_EDGE_MODE_ISD51x	ddPositiveEdge_ISD51x	Indicates which input events trigger the axis.
VAR_OUTPUT			
Done	BOOL		Trigger event has been recorded.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandAborted	BOOL		Command is aborted by another command (*MC_AbortTrigger_ISD51x*).

Variable name	Data type	Default value	Description
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
RecordedPosition	DINT		Position where trigger event occurred [user-defined position unit].

Table 6.30 MC_TouchProbe_ISD51x

Enumeration *DD_EDGE_MODE_ISD51x* defines the edge types for the digital input.

Name	Description
ddPositiveEdge_ISD51x	Only positive edges on the digital input are used as events.
ddNegativeEdge_ISD51x	Only negative edges on the digital input are used as events.
ddBothEdges_ISD51x	Positive and negative edges on the digital input are used as events.

Table 6.31 Enumeration DD_EDGE_MODE_ISD51x

6.5.4.28 MC_AbortTrigger_ISD51x

This function block is used to abort function blocks that are connected to trigger events, for example, *MC_TouchProbe_ISD51x*.
The deactivation of the functionality can take some time.

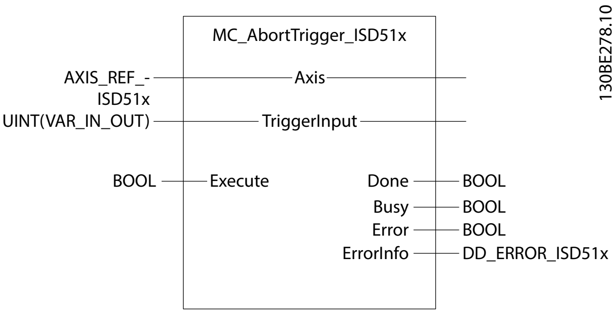


Illustration 6.54 MC_AbortTrigger_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
TriggerInput	UINT		Reference to the trigger signal source.
VAR_INPUT			
Execute	BOOL	FALSE	Starts touch probe recording at rising edge.
VAR_OUTPUT			
Done	BOOL		Trigger event has been recorded.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.32 MC_AbortTrigger_ISD51x

6.5.4.29 DD_PrepareDigCamSwitch_ISD51x

This function block prepares the axis with the digital CAM switches by sending the digital CAM switches information. It writes the digital CAM switches information to the servo drive but does not activate the digital CAM switches.

If the *Done* output is *TRUE*, the digital CAM switches information is valid and ready for processing. Use function block *DD_DigitalCamSwitch_ISD51x* to activate the digital CAM.
The sending and parsing of the CAM switches information can take some time.

If >1 axis needs to process the same digital CAM switches, execute this function block for every axis.

To generate a digital CAM switches file, see the description in *chapter 2.5.1 Digital CAM Switch*, or use the ISD Toolbox *chapter 5.7.6 Digital CAM Switch (Servo Drive only)*.

6

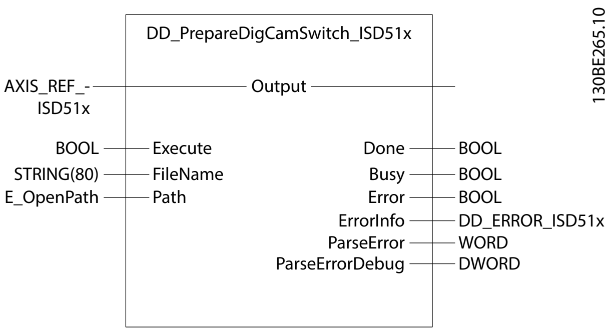


Illustration 6.55 DD_PrepareDigCamSwitch_ISD51x for TwinCAT®
(See *Table 6.33* for other available development environments).

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Output	AXIS_REF_ISD51x		Reference to the axis/ signal output. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start transfer of digital CAM switches data at rising edge.
FileName	STRING[80]	"	Filename of the digital CAM switches file on the PLC.
pDevice	UDINT	0	Automation Studio™ only: Pointer to the device name on which the digital CAM switches file is located.
Path	E_OpenPath	PATH_GENERIC	TwinCAT® only: The variable of this type selects generic or 1 of the TwinCAT® system paths on the target device.
VAR_OUTPUT			
Done	BOOL		CAM profile and configuration have been downloaded; Parsing was successful.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
ParseError	WORD		Detailed information on the type of error if there is a CAM parsing failure.
ParseErrorDebug	DWORD		Depending on the cause given in the output *ParseError*, additional debug information is given here. Available in the list of constants: *CamParsingErrors*.

Table 6.33 DD_PrepareDigCamSwitch_ISD51x

6.5.4.30 DD_DigitalCamSwitch_ISD51x

This function block activates the digital CAM switches functionality on the axis. It commands the digital output of the servo drive to switch in a similar way as a mechanical CAM controlled switch connected to an axis. Forward and backward movements are allowed.

This function block only works if the usage of the digital output is configured accordingly (see *chapter 7.21.5 Parameter 52-05: Digital Output Configuration (0x2FFF)*).
The activation of the digital CAM switches functionality takes effect immediately.

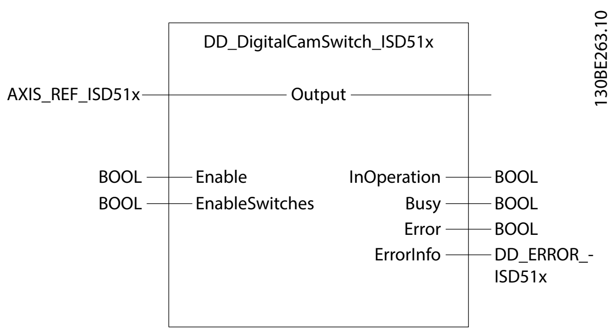


Illustration 6.56 DD_DigitalCamSwitch_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Output	AXIS_REF_ISD51x		Reference to the axis/signal output. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Control the digital CAM switching functionality.
Enable-Switches	BOOL	FALSE	Enables/disables the digital CAM switching functionality.
VAR_OUTPUT			
InOperation	BOOL		The digital CAM switching functionality is enabled.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.34 DD_DigitalCamSwitch_ISD51x

6.5.4.31 DD_ProduceGuideValue_ISD51x

This function block simulates a guide value inside the PLC. It must be called in every cycle to update the guide values. If *Enable* is *TRUE*, the guide values are updated and the current input values are used. Only linear ramps are used for the guide value calculation. For the inputs *GuideAcceleration* and *GuideDeceleration*, a value of *0* is not allowed and leads to an error.

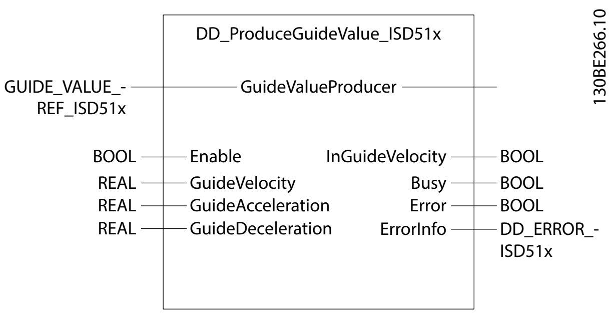


Illustration 6.57 DD_ProduceGuideValue_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
GuideValue-Producer	GUIDE_VALUE_REF_ISD51x		Reference to the guide value producer. See *Table 6.54*.
VAR_INPUT			
Enable	BOOL	FALSE	Updates the guide value producer if *Enable* is *TRUE*. If *Enable* is *FALSE*, the guide value stops immediately.
GuideVelocity	REAL	0.0	Velocity of the guide value [rps].
GuideAccel-eration	REAL	0.0	Acceleration value used while increasing the velocity of the guide value. Only positive values are allowed [rps/s].
GuideDecel-eration	REAL	0.0	Deceleration value used while decreasing the velocity of the guide value. Only positive values are allowed [rps/s].
VAR_OUTPUT			
InGuideVe-locity	BOOL		Commanded guide velocity reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.35 DD_ProduceGuideValue_ISD51x

6.5.5 Drive – Motion

6.5.5.1 MC_Home_ISD51x

This function block commands the axis to set its position to the input value given. It enables the execution of different homing modes. Depending on the selected *Mode*, several input parameters must be set (see *Table 6.38*). Also, the preconditions of this mode must be met. For detailed descriptions of the *Homing* modes, see *chapter 2.4.4 Homing Mode*.

The time limit is supervised by the PLC. Use function block *MC_Stop_ISD51x* (*chapter 6.5.5.2 MC_Stop_ISD51x*) to abort an active homing procedure.
It can take some time until the homing procedure starts.

6

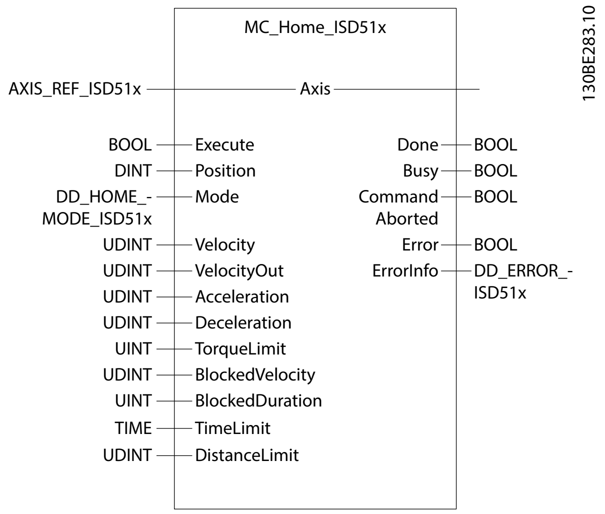


Illustration 6.58 MC_Home_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD 51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Position	DINT	0	Absolute position set [user-defined position unit].
Mode	DD_HOME_MODE_ISD51x	ddDirect_ISD51x	Defines the method used for homing. Depending on this selection, the appropriate input variables are used.
Velocity	UDINT	0	Value of the speed during search for switch/block [user-defined velocity unit].
VelocityOut	UDINT	0	Value of speed during search for edge of switch [user-defined velocity unit].
Acceleration	UDINT	0	Value of the acceleration [user-defined acceleration unit].
Deceleration	UDINT	0	Value of the deceleration [user-defined acceleration unit].
TorqueLimit	UINT	0xFFFF	Maximum torque used for this motion [per thousand of rated torque].
BlockedVelocity	UDINT	0	Axis assumes that it is blocked when the actual speed falls below the limit given here [user-defined velocity unit].
BlockedDuration	UINT	5	Axis assumes that it is blocked when the actual speed falls below the *BlockedVelocity* for the duration given here [ms].
TimeLimit	TIME	t#0ms	Timeout after which an error is signaled if the homing procedure has not been completed. The homing procedure is aborted automatically. Set the value to *0* to disable the time limit.
DistanceLimit	UDINT	0	Maximal distance in which the limit switch must be reached. Otherwise, the homing procedure is aborted with an error. Set the value to *0* to disable the distance limit.
VAR_OUTPUT			
Done	BOOL		Reference known and set successfully.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandAborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.36 MC_Home_ISD51x

The homing modes can be selected from the enumerations in *Table 6.37*.

Name	Corresponding homing mode
ddAbsolute_ISD51x	Homing on actual position.
ddPosBlock_ISD51x	Homing on positive block.
ddNegBlock_ISD51x	Homing on negative block.
ddNegLimSwitch_ISD51x	Homing on negative limit switch.
ddPosLimSwitch_ISD51x	Homing on positive limit switch.
ddPosHomeSwitch_ISD51x	Homing on positive home switch.
ddNegHomeSwitch_ISD51x	Homing on negative home switch.
ddDirect_ISD51x	Homing on current position.

Table 6.37 Enumeration DD_HOME_MODE_ISD51x

Ensure that the physical inputs of the servo drive are configured appropriately (see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*).

For the inputs *Velocity*, *VelocityOut*, *Acceleration*, *Deceleration*, and *BlockedVelocity*, a value of *0* is not allowed and leads to an error. This only applies to inputs that are required for the selected homing mode. The value of *BlockedVelocity* must be smaller than the value of *Velocity*.

	ddAbsolute_ISD51x	ddPosBlock_ISD51x or ddNegBlock_ISD51x	ddPosLimSwitch_ISD51x or ddNegLimSwitch_ISD51x	ddPosHomeSwitch_ISD51x or ddNegHomeSwitch_ISD51x	ddDirect_ISD51x
Required PLCopen® state before start	**Standstill or disabled**	**Standstill**	**Standstill**	**Standstill**	**Standstill or disabled**
Position	–	X	X	X	X
Velocity	–	X	X	X	–
VelocityOut	–	–	X	X	–
Acceleration	–	X	X	X	–
Deceleration	–	X	X	X	–
TorqueLimit	–	X	X	X	–
BlockedVelocity	–	X	–	–	–
BlockedDuration	–	X	–	–	–
TimeLimit	–	X	X	X	–
DistanceLimit	–	X	X	X	–

Table 6.38 Inputs marked with X must have a valid value

6.5.5.2 MC_Stop_ISD51x

This function block commands a controlled motion stop and transfers the axis to the state *Stopping*. It aborts any ongoing motion. When the axis is in state *Stopping*, no other function block can perform any motion on the same axis. After the axis has reached the velocity *0*, the *Done* output is set to *TRUE* immediately (see *chapter 7.20.2 Parameter 50-42: Target Reached Option Code (0x2054)*). The axis remains in the state *Stopping* for as long as *Execute* is still *TRUE*, or until the target velocity *0* has been reached. As soon as *Done* is set to *TRUE* and *Execute* is *FALSE*, the axis changes to state *Standstill*.

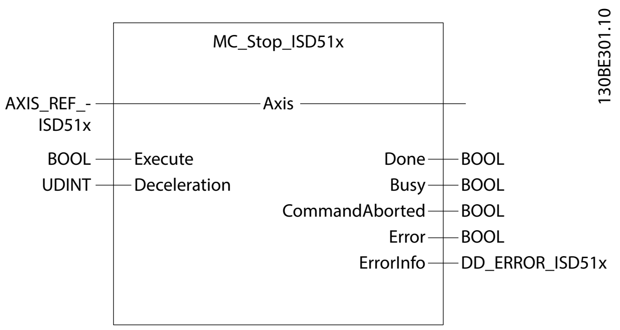


Illustration 6.59 MC_Stop_ISD51x

This function block is primarily intended for emergency stop functionality or exception situations. Calling this function block in state *Standstill* changes the state to *Stopping* and back to *Standstill* when *Execute* is *FALSE*. The state remains as *Stopping* for as long as the input *Execute* is *TRUE*. The *Done* output is set when the stop ramp is finished.

If this function block is aborted (by setting *MC_Power.Enable* to *FALSE*), or if an error appears in the axis, the blocking of the axis by this function block is released immediately. In this case, it is not necessary to set *Execute* to *FALSE* first to release the axis.

A value of *0* is not allowed for the *Deceleration* input.
The command is transferred and executed immediately.

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD 51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Deceleration	UDINT	0	Value of the deceleration [user-defined acceleration unit]. Only values >0 are allowed.
VAR_OUTPUT			
Done	BOOL		When target velocity *0* is reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandA-borted	BOOL		Command is aborted by switching off power (only possibility to abort).
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.39 MC_Stop_ISD51x

6.5.5.3 MC_Halt_ISD51x

This function block commands a controlled motion stop. The axis moves to state *DiscreteMotion*, until the velocity is *0* (see *chapter 7.20.2 Parameter 50-42: Target Reached Option Code (0x2054)*). With the *Done* output set, the state transfers to *Standstill*. This function block is used to stop the axis under normal operation conditions. Another motion command can be set during deceleration of the axis, which is executed immediately and aborts *MC_Halt_ISD51x*.

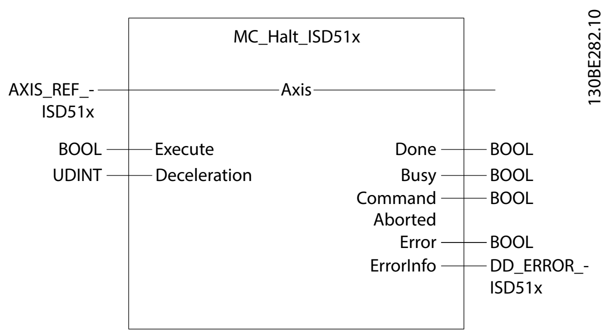


Illustration 6.60 MC_Halt_ISD51x

The value *0* is not allowed for the *Deceleration* input.
The command is transferred and executed immediately.

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD 51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Deceleration	UDINT	0	Value of the deceleration [user-defined acceleration unit]. Only values >0 are allowed.
VAR_OUTPUT			
Done	BOOL		When target velocity *0* is reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Command Aborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.40 MC_Halt_ISD51x

6.5.5.4 MC_MoveAbsolute_ISD51x

This function block commands a controlled motion to a specified absolute position. If no further actions are pending, this action completes with velocity *0*. The *Direction mcShortestWay_ISD51x* uses a trajectory that takes the shortest route. The direction is based on the current position when the command is issued.

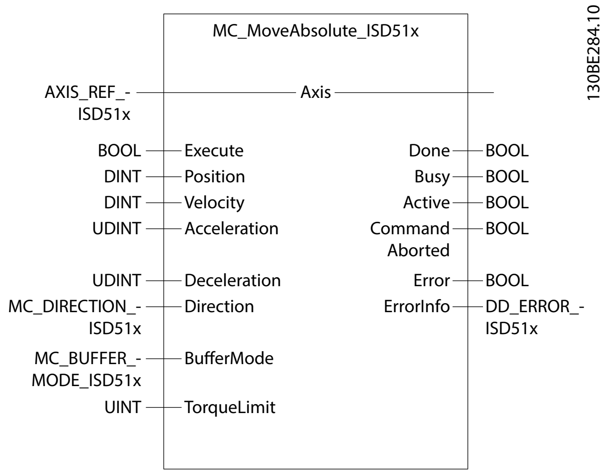


Illustration 6.61 MC_MoveAbsolute_ISD51x

Only positive values are allowed for the inputs *Velocity*, *Acceleration*, and *Deceleration*.
The command is transferred immediately and, if in aborting buffer mode, is also executed immediately.

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Position	DINT	0	Commanded position for motion [user-defined position unit]. The value can be positive or negative.
Velocity	DINT	0	Value of maximum velocity, that is not necessarily reached [user-defined velocity unit]. Only values >0 are allowed.
Acceleration	UDINT	0	Value of acceleration (increasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
Deceleration	UDINT	0	Value of the deceleration (decreasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
Direction	MC_DIRECTION_ISD51x	mcShortestWay_ISD51x	Influences the trajectory calculation. See *Table 6.42* for available values. The following directions are supported: • mcCurrentDirection_ISD51x • mcNegativeDirection_ISD51x • mcPositiveDirection_ISD51x • mcShortestWay_ISD51x
BufferMode	MC_BUFFER_MODE_ISD51x	mcAborting_ISD51x	Defines the chronological sequence of the function block. See *Table 6.43* for available values.
TorqueLimit	UINT	0xFFFF	Maximum torque used during this motion [per thousand of rated torque].
VAR_OUTPUT			
Done	BOOL		Commanded position finally reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Active	BOOL		The function block has control on the axis.
Command Aborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.41 MC_MoveAbsolute_ISD51x

Enumeration *MC_DIRECTION_ISD51x* defines the direction used to reach the target of a periodic axis.

Name	Description
mcCurrentDir-ection_ISD51x	Movement in direction of the last known positioning direction.
mcNegativeDir-ection_ISD51x	Movement only in negative direction; If the target position is higher than the actual position, the axis moves over the position wrap around.
mcPositiveDir-ection_ISD51x	Movement only in positive direction; If the target position is lower than the actual position, the axis moves over the position wrap around.
mcShor-testWay_ISD51x	Movement with the shortest distance to the target position.
mcLinearAxis_ISD51x	Normal movement similar to a linear axis.

Table 6.42 Enumeration MC_DIRECTION_ISD51x

Enumeration *MC_BUFFER_MODE_ISD51x* defines the chronological sequence of the function block.

Name	Description
mcAborting_ISD51x	Actual positioning process is aborted and replaced with a new one.
mcBuffered_ISD51x	Actual positioning process is continued and the next follows.

Table 6.43 Enumeration MC_BUFFER_MODE_ISD51x

The buffer mode itself is described in more detail in *chapter 2.4.1 Profile Position Mode*. It is only possible to have 1 buffered position. Any attempt to make a 2nd buffered command leads to an error.

6

MC_MoveAbsolute with BufferMode = Aborting

The potentially ongoing positioning is aborted immediately. The new target position is the value of the *Position* input of *MC_MoveAbsolute_ISD51x*. The trajectory parameters (*Velocity*, *Acceleration*, and *Deceleration*) are used immediately.

MC_MoveAbsolute with BufferMode = Buffered

The potentially ongoing positioning is finished first. The original target position is reached and the velocity in this target position is *0*. The new target position is the value of the position input of *MC_MoveAbsolute_ISD51x*. This means that it is the exact same end position as with *BufferMode = mcAborting_ISD51x*.

The trajectory parameters (*Velocity*, *Acceleration*, and *Deceleration*) are only used for the 2nd movement.

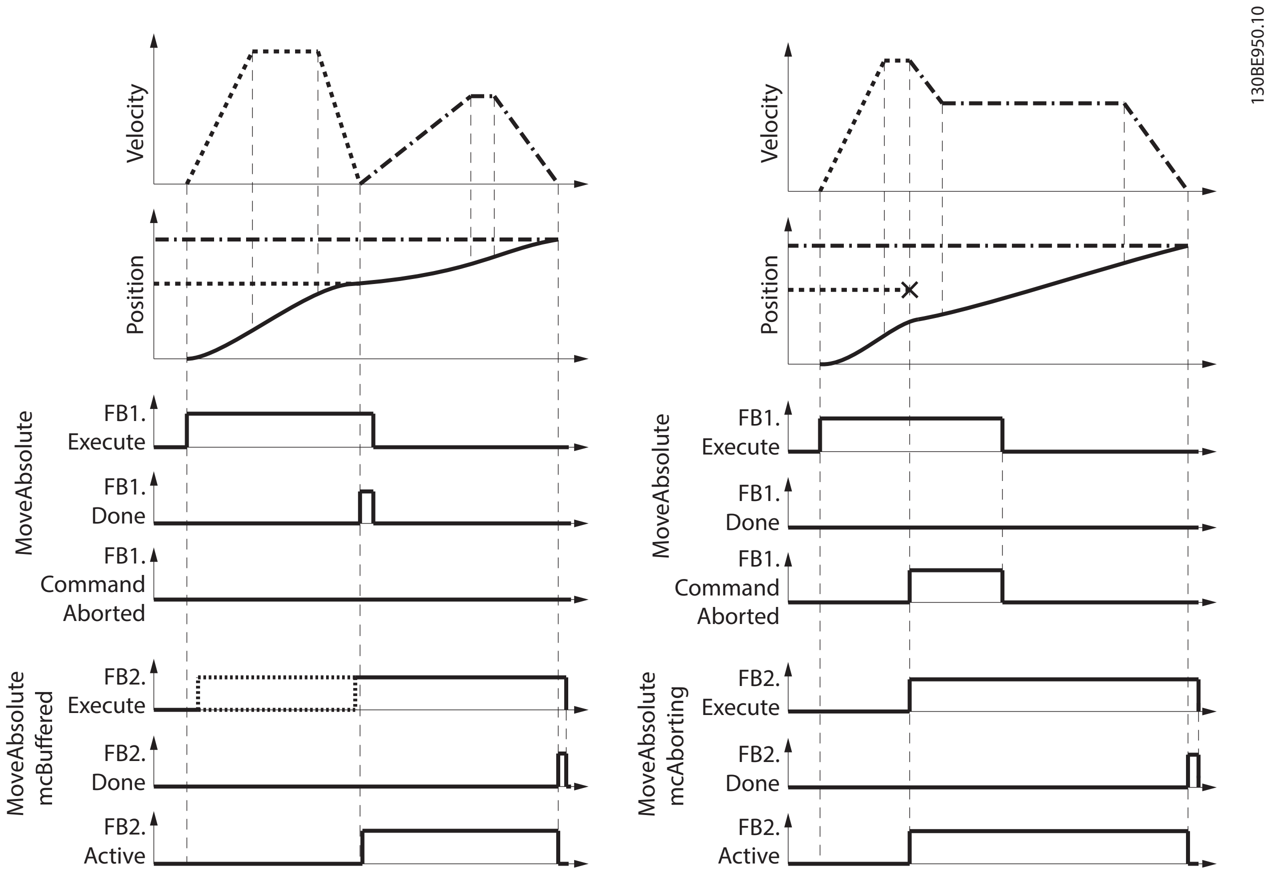


Illustration 6.62 Buffered versus Aborting with MC_MoveAbsolute

6.5.5.5 MC_MoveRelative_ISD51x

This function block commands a controlled motion of a specified distance relative to the set position at the time of execution. This action completes with velocity *0* if no further actions are pending. Only positive values are allowed for the inputs *Velocity*, *Acceleration*, and *Deceleration*.
The command is transferred immediately and, in aborting buffer mode, is also executed immediately.

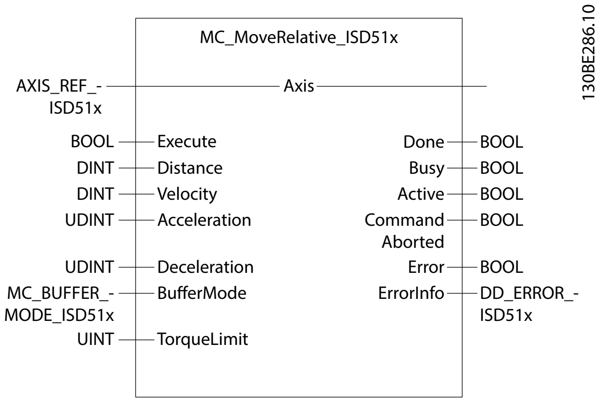


Illustration 6.63 MC_MoveRelative_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Distance	DINT	0	Commanded position for motion [user-defined position unit]. Can be a positive or negative value.
Velocity	DINT	0	Value of the maximum velocity (not necessarily reached) [user-defined velocity unit]. Only values >0 allowed.
Acceleration	UDINT	0	Value of the acceleration (increasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
Deceleration	UDINT	0	Value of the deceleration (decreasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
BufferMode	MC_BUFFER_MODE_ISD51x	mcAborting_ISD51x	Defines the chronological sequence of the function block. See *Table 6.43* for available values.
TorqueLimit	UINT	0xFFFF	Maximum torque used during this motion [per thousand of rated torque].
VAR_OUTPUT			
Done	BOOL		Commanded distance reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Active	BOOL		The function block has control on the axis.
CommandAborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.44 MC_MoveRelative_ISD51x

The buffer mode itself is described in more detail in *chapter 2.4.1 Profile Position Mode*. It is only possible to have 1 buffered position. Trying to command a 2^{nd} buffered command leads to an error.

MC_MoveRelative with BufferMode = Aborting
The potentially ongoing positioning is aborted immediately. The new target position is the actual position of the axis at the point of the rising edge of *Execute* of the function block, plus the value of the *Distance* input of *MC_MoveRelative_ISD51x*. The trajectory parameters (*Velocity*, *Acceleration*, and *Deceleration*) are used immediately.

If the function block is activated in the axis state *ContinuousMotion*, the specified relative distance is added to the set position at the point of the rising edge of *Execute*. This applies for both buffer modes *Buffered* and *Aborting*.

6

MC_MoveRelative with BufferMode = Buffered

The potentially ongoing positioning is finished first. The original target position is reached and the velocity in this target position is *0*. The new target position is the set position of the axis at the point of the rising edge of *Execute* of the function block, plus the value of the *Distance* input of *MC_MoveRelative_ISD51x*. This means that it is the exact same end position as with *BufferMode = mcAborting_ISD51x*.

The trajectory parameters (*Velocity*, *Acceleration*, and *Deceleration*) are only used for the 2nd movement.
If the function block is activated in the axis state *ContinuousMotion*, the specified relative distance is added to the set position at the point of the rising edge of *Execute*. This applies both for buffer modes *Buffered* and *Aborting*.

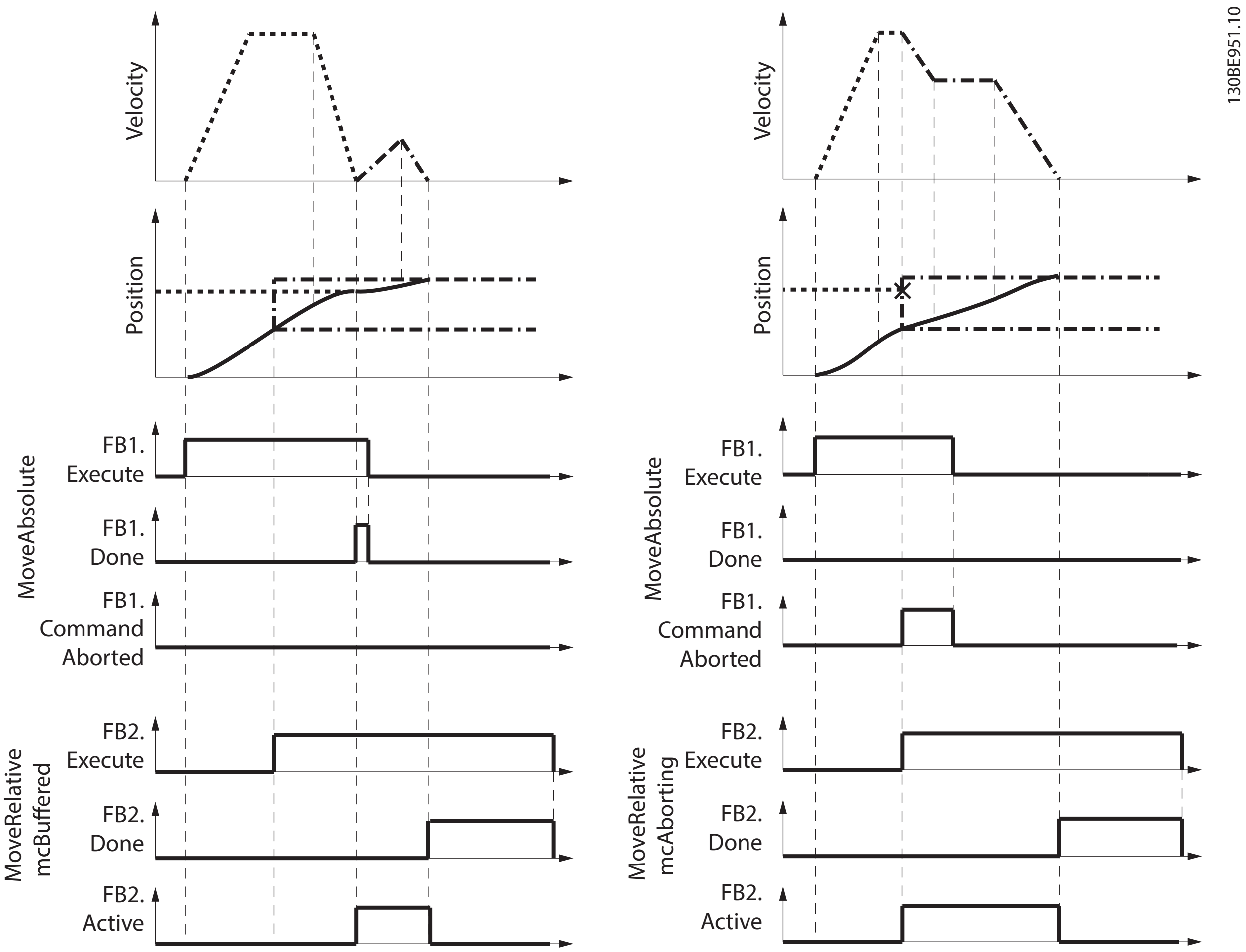


Illustration 6.64 Buffered versus Aborting with MC_MoveRelative

6.5.5.6 MC_MoveAdditive_ISD51x

This function block commands a controlled motion of a specified relative distance in addition to the most recent commanded position in the axis state *DiscreteMotion*. The most recent commanded position can also be the result of a previous *MC_MoveAdditive_ISD51x* that was aborted. If the function block is activated in the axis state *Continuous-Motion*, the specified relative distance is added to the set position at the time of execution.

Only positive values are allowed for the inputs *Velocity*, *Acceleration*, and *Deceleration*.
The command is transferred immediately and, if aborting buffer mode, also executed immediately.

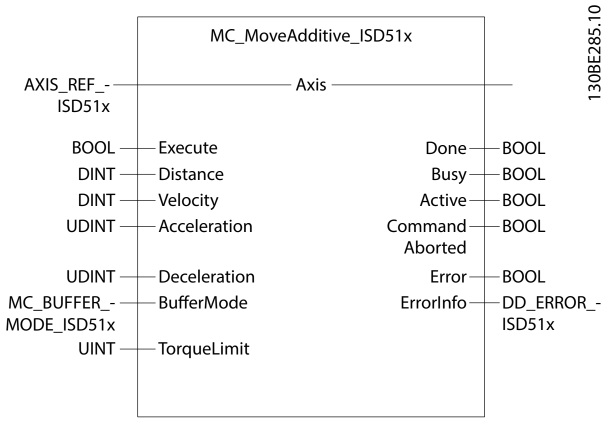


Illustration 6.65 MC_MoveAdditive_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Distance	DINT	0	Relative distance for the motion [user-defined position unit]. Can be a positive or negative value.
Velocity	DINT	0	Value of the maximum velocity (not necessarily reached) [user-defined velocity unit]. Only values >0 are allowed.
Acceleration	UDINT	0	Value of the acceleration (increasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
Deceleration	UDINT	0	Value of the deceleration (decreasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
BufferMode	MC_BUFFER_MODE_ISD51x	mcAborting_ISD51x	Defines the chronological sequence of the function block.
TorqueLimit	UINT	0xFFFF	Maximum torque used during this motion [per thousand of rated torque].
VAR_OUTPUT			
Done	BOOL		Commanded distance reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Active	BOOL		The function block has control on the axis.
CommandAborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.45 MC_MoveAdditive_ISD51x

The buffer mode itself is described in more detail in *chapter 2.4.1 Profile Position Mode*. It is only possible to have 1 buffered position. Trying to command a 2nd buffered command leads to an error.

MC_MoveAdditive with BufferMode = Aborting
The potentially ongoing positioning is aborted immediately. The original target position is not necessarily reached. The new target position is the (aborted) target position plus the value of the *Distance* input of *MC_MoveAdditive_ISD51x*. The trajectory parameters (*Velocity*, *Acceleration*, and *Deceleration*) are used immediately.
If the function block is activated in the axis state *ContinuousMotion*, the specified relative distance is added to the set position at the point of the rising edge of *Execute*. This applies for buffer mode *Buffered* and *Aborting*.

MC_MoveAdditive with BufferMode = Buffered

The potentially ongoing positioning is finished first. The original target position is reached and the velocity in this target position is *0*. The new target position is the (old) target position plus the value of the *Distance* input of *MC_MoveAdditive_ISD51x*. This means that it is the exact same end position as with *BufferMode = mcAborting_ISD51x*.

The trajectory parameters (*Velocity*, *Acceleration*, and *Deceleration*) are only used for the 2nd movement. If the function block is activated in the axis state *ContinuousMotion*, the specified relative distance is added to the set position at the point of the rising edge of *Execute*. This applies for buffer mode *Buffered* and *Aborting*.

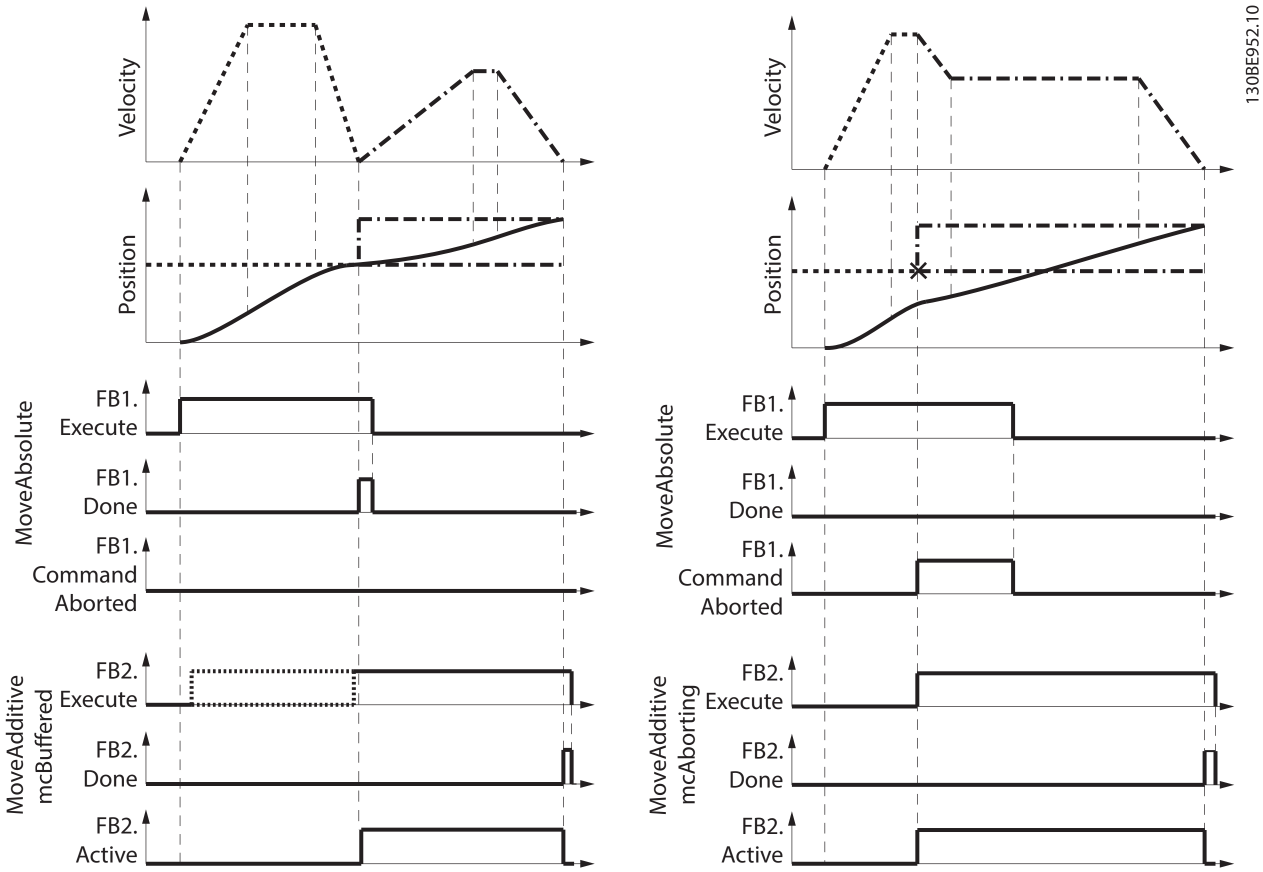


Illustration 6.66 Buffered versus Aborting with MC_MoveAdditive

6.5.5.7 MC_MoveVelocity_ISD51x

This function block commands a never ending controlled motion at a specified velocity. To stop the motion, interrupt the function block by issuing a new command with another function block. The signal *InVelocity* is reset when the function block is aborted by another function block. Only positive values are allowed for the inputs *Acceleration* and *Deceleration*.
The command is transferred and executed immediately.

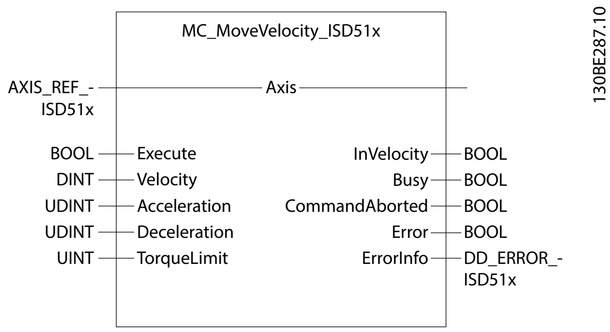


Illustration 6.67 MC_MoveVelocity_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Velocity	DINT	0	Value of the target velocity [user-defined velocity unit].
Acceleration	UDINT	0	Value of the acceleration (increasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
Deceleration	UDINT	0	Value of the deceleration (decreasing energy of the motor) [user-defined acceleration unit]. Only values >0 are allowed.
TorqueLimit	UINT	0xFFFF	Maximum torque used during this motion [per thousand of rated torque].
VAR_OUTPUT			
InVelocity	BOOL		Commanded velocity reached.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandA-borted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.46 MC_MoveVelocity_ISD51x

6.5.5.8 MC_TorqueControl_ISD51x

This function block continuously exerts a torque or force of the specified magnitude. This magnitude is reached using a defined ramp (*TorqueRamp*), and the function block sets the *InTorque* output if the commanded torque level is reached. If there is no external load, force is applicable. Positive torque is in the positive direction of velocity and the movement is limited by velocity. Only positive values are allowed for the inputs *TorqueRamp* and *Velocity*.
The command is transferred and executed immediately.

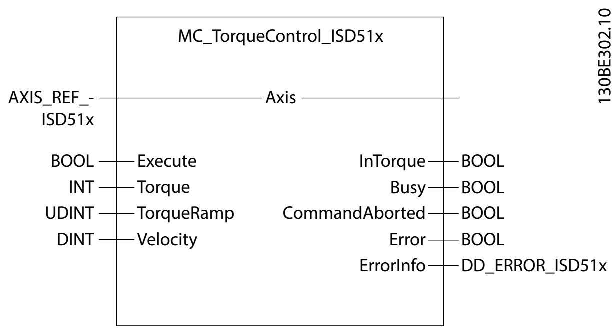


Illustration 6.68 MC_TorqueControl_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the motion at rising edge.
Torque	INT	0	Value of the target torque [user-defined velocity unit].
TorqueRamp	UDINT	0	The maximum time derivative of the set value of the torque or force [in technical units per s]. Only values >0 are allowed.

Variable name	Data type	Default value	Description
Velocity	DINT	0	Value of the maximum velocity (not necessarily reached) [user-defined velocity unit]. Only values >0 are allowed.
VAR_OUTPUT			
InTorque	BOOL		Setpoint value of torque or force equals the commanded value.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandAborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.47 MC_TorqueControl_ISD51x

6.5.5.9 MC_GearIn_ISD51x

This function block commands a synchronized motion with the master. The slave ramps up to the ratio of the master velocity and locks in when this is reached. Any lost distance during synchronization is not caught up. The gearing ratio can be changed while *MC_GearIn_ISD51x* is running, using a new rising edge of the *Execute* input.

After being *InGear*, the servo drive is running position-controlled. Only positive values are allowed for the inputs *Acceleration* and *Deceleration*. The value *0* is not allowed for the inputs *RatioNumerator* and *RatioDenominator*.
The command is transferred and executed immediately.

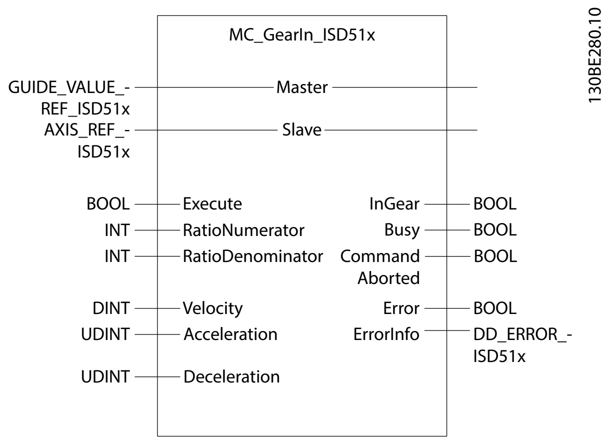


Illustration 6.69 MC_GearIn_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Master	GUIDE_VALUE_REF_ISD51x		Reference to the master axis. See *Table 6.54*.
Slave	AXIS_REF_ISD51x		Reference to the slave axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the gearing process at rising edge.
RatioNumerator	INT	1	Gear ratio numerator.
RatioDenominator	INT	1	Gear ratio denominator.
Velocity	DINT	0	Maximum velocity used during gearing in procedure [user-defined velocity unit].
Acceleration	UDINT	0	Acceleration used for gearing in [user-defined acceleration unit].
Deceleration	UDINT	0	Deceleration used for gearing in [user-defined acceleration unit].
VAR_OUTPUT			
InGear	BOOL		Is *TRUE* if the set value equals the commanded value.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandAborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.48 MC_GearIn_ISD51x

6.5.5.10 MC_GearInPos_ISD51x

This function block commands a synchronized motion with the master. It commands a gear ratio between the position of the slave and the master axes from the synchronization point onwards. Only values >0 are allowed for the inputs *Velocity*, *Acceleration*, and *Deceleration*. The value *0* is not allowed for the inputs *RatioNumerator* and *RatioDenominator*.
The command is transferred and executed immediately.

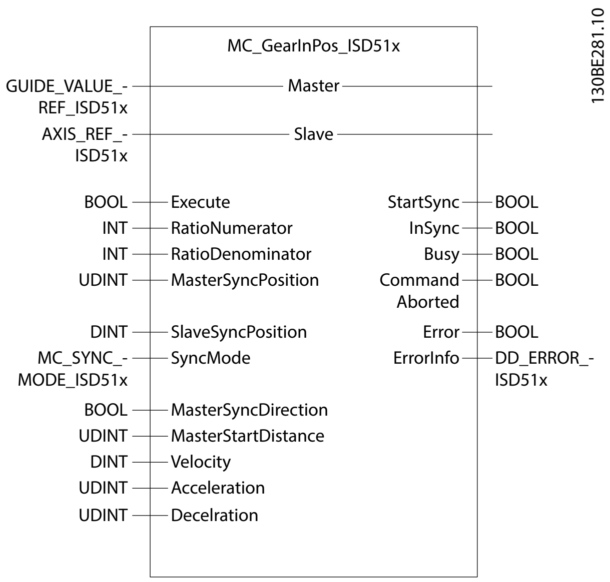


Illustration 6.70 MC_GearInPos_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Master	GUIDE_VALUE_REF_ISD51x		Reference to the master axis. See *Table 6.54*.
Slave	AXIS_REF_ISD51x		Reference to the slave axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start the gearing process at rising edge.
RatioNumerator	INT	1	Gear ratio numerator.
RatioDenominator	INT	1	Gear ratio denominator.
MasterSyncPosition	UDINT	0	The position of the master where the slave is in sync with the master.
SlaveSyncPosition	DINT	0	Slave position [user-defined position unit] at which the axes are running in sync.
SyncMode	MC_SYNC_MODE_ISD51x	mcShortest_ISD51x	Defines the mode for synchronizing. See *Table 6.50*.
MasterSyncDirection	BOOL	FALSE	*FALSE* = Master start distance is in the positive direction of the guide value. *TRUE* = Master start distance is in the negative direction of the guide value.
MasterStartDistance	UDINT	0	Master distance for gear in procedure (when the slave axis is starting to be synchronized).
Velocity	DINT	0	Value of the maximum velocity (not necessarily reached) during the synchronization process [user-defined velocity unit]. Only values >0 are allowed.
Acceleration	UDINT	0	Value of the acceleration (increasing energy of the motor) during synchronization [user-defined acceleration unit]. Only values >0 are allowed.
Deceleration	UDINT	0	Value of the deceleration (decreasing energy of the motor) during synchronization [user-defined acceleration unit]. Only values >0 are allowed.
VAR_OUTPUT			
InSync	BOOL		Is *TRUE* if the set value equals the commanded value.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandAborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.49 MC_GearInPos_ISD51x

6

Enumeration *MC_SYNC_MODE_ISD51x* defines the synchronization types for the gear in process.

Name	Description
mcShortest_ISD51x	Synchronization is done to have the shortest way for the slave.
mcCatchUp_ISD51x	Synchronization is done to catch up to the master axis (synchronization to current master cycle).
mcSlowDown_ISD51x	Synchronization is done to slow down to the master axis (synchronization 1 master cycle later).

Table 6.50 Enumeration MC_SYNC_MODE_ISD51x

6.5.5.11 DD_GetInertia_ISD51x

This function block is used to determine the inertia of the servo drive system. Only activate the function block when the axis is in state *Standstill*. Only values >0 are allowed for the inputs *VelocityLimit* and *TorqueLimit*.
The command is transferred and executed immediately.

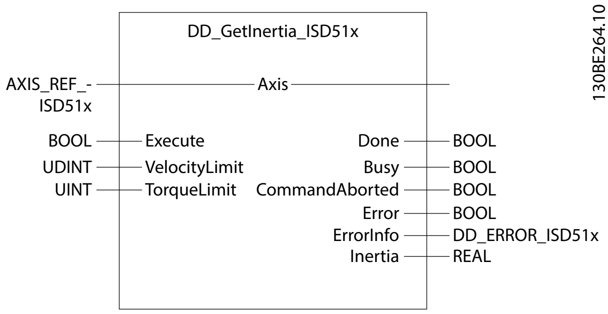


Illustration 6.71 DD_GetInertia_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Starts the inertia measurement at rising edge.
VelocityLimit	UDINT	0	Maximum velocity used during the measurement [user-defined velocity unit]. Only values >0 are allowed.
TorqueLimit	UINT	0xFFFF	Maximum torque used during this motion [mNm (Millinewtonmeter)].
VAR_OUTPUT			
Done	BOOL		The inertia measurement has completed successfully and the value is available.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandAborted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Inertia	REAL		Measured inertia [kg m²].

Table 6.51 DD_GetInertia_ISD51x

6.5.6 Drive – CAM Operation

6.5.6.1 MC_CamTableSelect_ISD51x

This function block selects the CAM tables by setting the connections to the relevant tables. When the *Done* output is set, the axis internal buffer defined by *CamTableID* is valid and ready for use in a *MC_CamIn_ISD51x* function block. The sending and parsing of the CAM can take some time.

The function block sends the information to the axis. Execute this function block for every axis if >1 axis must follow the same CAM. This function block only writes the CAM information to the servo drive and does not activate the CAM. Use function block *MC_CamIn_ISD51x* to activate the CAM. If the CAM profile contains a pattern alignment, the pattern file must be given to input *PatternFile*. If there is no pattern alignment inside the CAM, use an empty string.

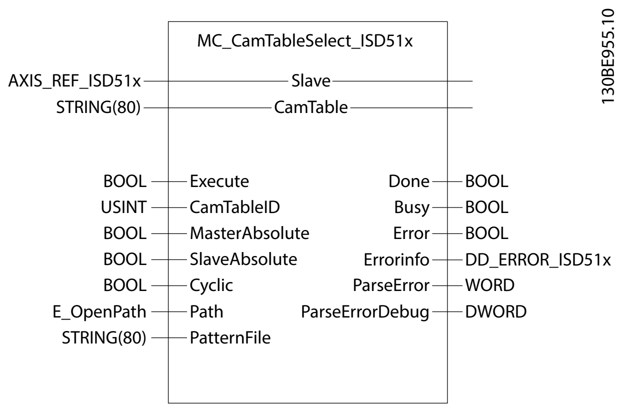


Illustration 6.72 MC_CamTableSelect_ISD51x in TwinCAT®

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Slave	AXIS_REF_ISD51x		Reference to the slave axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
CamTable	STRING[80]		Reference to CAM description (file name of CAM profile on the PLC).
VAR_INPUT			
Execute	BOOL	FALSE	Selection at rising edge
CamTableID	USINT	0	Identifier of CAM table to be used in the *MC_CamIn_ISD51x* function block. Numbers 1–8 are available CAM buffers.
MasterAbsolute	BOOL	FALSE	*TRUE* = absolute coordinates. *FALSE* = relative coordinates.
SlaveAbsolute	BOOL	FALSE	*TRUE* = absolute coordinates. *FALSE* = relative coordinates.
Cyclic	BOOL	TRUE	*TRUE* = cyclic. *FALSE* = non-cyclic.
pDevice	UDINT	0	Automation Studio™ only: Pointer to the device name on which the CAM file is located.
Path	E_OpenPath	PATH_GENERIC	TwinCAT® only: The variable of this type selects generic or 1 of the TwinCAT® system paths on the target device.
PatternFile	STRING[80]	“	Name of the pattern file on the PLC in case the CAM profile contains a pattern alignment.
VAR_OUTPUT			
Done	BOOL		CAM profile and configuration have been downloaded; Parsing was successful.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
ParseError	WORD		Detailed information on the type of error if there is a CAM parsing failure.
ParseErrorDebug	DWORD		Depending on the cause given in the output *ParseError*, additional debug information is given here. Available in the list of constants *CamParsingErrors*.

Table 6.52 MC_CamTableSelect_ISD51x

6.5.6.2 MC_CamIn_ISD51x

This function block engages the CAM. The command is transferred immediately and, if *ChangeImmediate* is *TRUE*, it is also executed immediately.

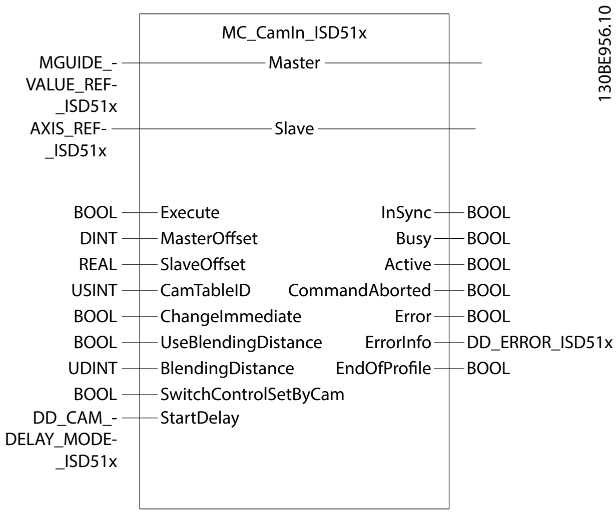


Illustration 6.73 MC_CamIn_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Master	GUIDE_VALUE_REF_ISD51x		Reference to the master axis. See *Table 6.54*.
Slave	AXIS_REF_ISD51x		Reference to the slave axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Start at rising edge.
MasterOffset	DINT	0	Offset of the master shaft to CAM [guide value unit].

6

6

Variable name	Data type	Default value	Description
SlaveOffset	REAL	0	Offset of the slave shaft [revolutions].
CamTableID	USINT	0	Identifier of the CAM table to be used; linked to the input of *MC_CamTable-Select_ISD51x*.
ChangeIm-mediate	BOOL	FALSE	*TRUE* = Abort the currently running CAM immediately. *FALSE* = Let the currently running CAM finish first.
UseBlending-Distance	BOOL	FALSE	*FALSE* = Automatically blend to the beginning of the new CAM. *TRUE* = Use *Blending-Distance* as minimum length for blending to the new CAM.
Blending-Distance	UDINT	0	Used in the direction of the master, minimum length used for blending to the new CAM.
SwitchCon-trolSetByCam	BOOL	TRUE	*TRUE* = Control parameter set selection is handled by the CAM itself *FALSE* = Control parameter set selection via function block *DD_SelectControl-ParamSet_ISD51x* (see *chapter 6.5.4.26 DD_Select-ControlParamSet_ISD51x*).
StartDelay	DD_CAM_DELAY_MODE_ISD51x	ddNoDelay_ISD51x	Influences the activation behavior of a CAM. Only valid for Master relative CAMs. See *Table 6.55*.
VAR_OUTPUT			
InSync	BOOL		Is *TRUE* if the slave follows the commanded CAM profile.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Active	BOOL		The function block has control on the axis.
CommandA-borted	BOOL		Command is aborted by another command.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
EndOfProfile	BOOL		Pulsed output signaling the cyclic end of the CAM profile. It is shown for 1 PLC cycle each time the end of the CAM profile is reached. In reverse direction, the *EndOfProfile* is also shown at the end of the CAM profile (in this case the 1st point of the CAM profile).

Table 6.53 MC_CamIn_ISD51x

The structure *GUIDE_VALUE_REF_ISD51x* is described in *Table 6.54*.

Variable name	Data type	Default value	Description
PositionGui-deValue	UDINT	0	Value of the position guide value. 1 cycle is scaled from 0–16#FFFF FFFF
VelocityGui-deValue	REAL	0	Velocity of the guide value [rps].

Table 6.54 GUIDE_VALUE_REF_ISD51x

Enumeration *DD_CAM_DELAY_MODE_ISD51x* defines the activation behavior of a CAM (only available with master relative CAMs).

Name	Description
ddNoDelay_ISD51x	CAM activation without delay.
ddInput1_ISD51x	Delayed activation of CAM: Processing of CAM is delayed until digital input 1 is on.
ddInput2_ISD51x	Delayed activation of CAM: Processing of CAM is delayed until digital input 2 is on.
ddAnyInput_ISD51x	Delayed activation of CAM: Processing of CAM is delayed until either digital input 1 or 2 is on.

Table 6.55 Enumeration DD_CAM_DELAY_MODE_ISD51x

6.5.6.3 DD_CamScaling_ISD51x

This function block modifies the parameters relating to the slave and master scaling. These parameters apply to 1 slave axis only. Activate this function block for every slave axis that needs manipulation. The manipulation of the factors can take some time because asynchronous communication is used.

The use of the parameters inside the axis takes effect immediately. The numerator and the denominator for 1 factor are used by the servo drive at the same time.

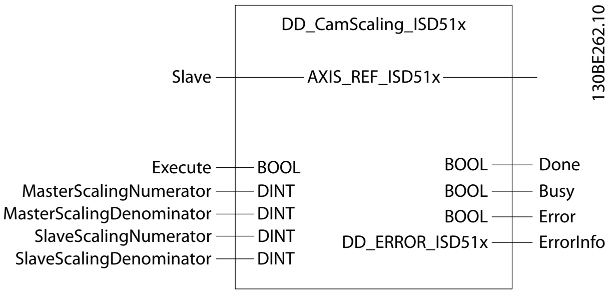


Illustration 6.74 DD_CamScaling_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Slave	AXIS_REF_ISD51x		Reference to the slave axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Starts the parameter transfer to the servo drive at rising edge.
MasterScalingNumerator	DINT	1	Numerator for the CAM master scaling. From the slave point of view, the master overall profile is multiplied by this factor.
MasterScalingDenominator	DINT	1	Denominator for the CAM master scaling. From the slave point of view, the master overall profile is divided by this factor.
SlaveScaling-Numerator	DINT	1	Numerator for the CAM slave scaling. The overall slave profile is multiplied by this factor.
SlaveScaling-Denominator	DINT	1	Denominator for the CAM slave scaling. The overall slave profile is divided by this factor.
VAR_OUTPUT			
Done	BOOL		The parameters have been set.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.56 DD_CamScaling_ISD51x

6.5.6.4 DD_SetFollowSegment_ISD51x

This function block is used for advanced CAMs only. It instructs the node to use the specified segment when it is passed.

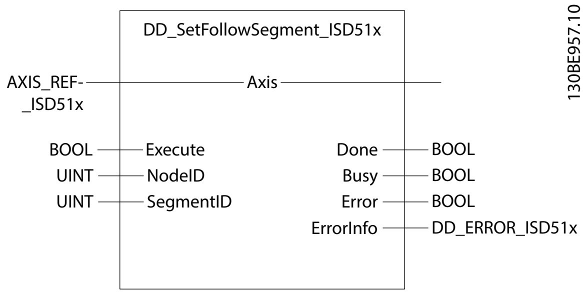


Illustration 6.75 DD_SetFollowSegment_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Sends the switching command to enable/ disable the node notification at rising edge.
NodeID	UINT	0	ID of the node where the following segment should be switched.
SegmentID	UINT	0	ID of the segment that should be used from now on after the node with NodeID.
VAR_OUTPUT			
Done	BOOL		The command has been set.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.57 DD_SetFollowSegment_ISD51x

6

6.5.6.5 DD_SetSegmentParameter_ISD51x

This function block is used for advanced CAMs only. It sends the angle value to the segment with the given *SegmentID* and is used with *MoveDistanceSegments* and *FlyingStopSegments* (see *chapter 5.7.7.7 Editing Advanced CAM Profiles* for further information). It must be sent before the specified segment is active. Sending it when the specified segment is active leads to an error (use *DD_ReadCAMinfo* to read the error). The timing is not checked inside the function block.

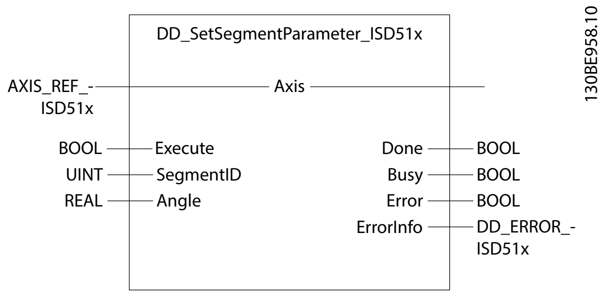


Illustration 6.76 DD_SetSegmentParameter_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Sends the angle value.
SegmentID	UINT	0	ID of the segment that should use this parameter.
Angle	REAL	0	Angle parameter to be used by the given segment. Angle must be given in (slave) revolutions.
VAR_OUTPUT			
Done	BOOL		The parameter has been set.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.58 DD_SetSegmentParameter_ISD51x

6.5.6.6 DD_RotationStop_ISD51x

This function block is used for basic and advanced CAMs. It stops the execution of the CAM for 1 master cycle.

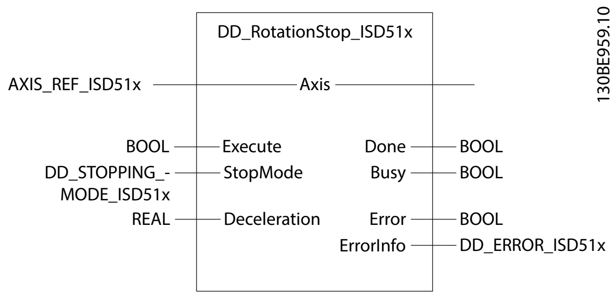


Illustration 6.77 DD_RotationStop_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Stops the processing of the CAM for this master cycle.
StopMode	DD_STOPPING_MODE_ISD51x	ddCoasting_ISD51x	Specifies the way of stopping the servo drive for this master cycle. See *Table 6.60*.
Deceleration	REAL	0	Deceleration value used for the ramping procedure. Only positive values are allowed [rps/s].
VAR_OUTPUT			
Done	BOOL		The command has been sent.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.59 DD_RotationStop_ISD51x

Name	Corresponding stopping mode
ddCoasting_ ISD51x	Coasting and stay in state *Operation.enabled*
ddRamp_ ISD51x	Slow down on specified ramp and stay in state *Operation enabled*. The deceleration value must be specified for this mode.
ddCurrentLimit_ ISD51x	Slow down on current limit and stay in state *Operation enabled*.

Table 6.60 Enumeration DD_STOPPING_MODE_ISD51x

6.5.6.7 DD_NodeNotification_ISD51x

This function block is used for advanced and basic CAMs. It enables/disables the sending of a notification when a certain node within the CAM is passed. Also, information about the following segment of this node is transmitted. The information can be read using function block *DD_ReadCamInfo_ISD51x* (see *chapter 6.5.6.9 DD_ReadCamInfo_ISD51x*).

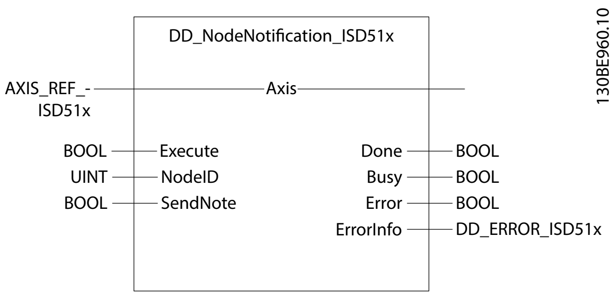


Illustration 6.78 DD_NodeNotification_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Sends the command to enable/disable the node notification.
NodeID	UINT	0	ID of the node that should/should not send a notification when it is passed.
SendNote	BOOL	FALSE	*FALSE*: No notification is sent. *TRUE*: Notification is sent together with information of the following segment.
VAR_OUTPUT			
Done	BOOL		The command has been sent.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.61 DD_NodeNotification_ISD51x

6.5.6.8 DD_GoToSetpoint_ISD51x

This function block is used for advanced and basic CAMs. It commands a movement to the setpoint of the CAM while the guide value velocity is *0*. This is used, for example, when starting up a CAM and the axis position is not on the CAM profile.

The required movement is then calculated by the axis itself, based on the direction option code (see *Table 6.42* for available values) over the specified time. The guide value velocity must stay at *0* until this movement is finished.

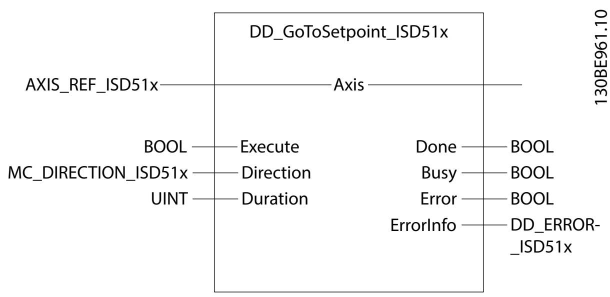


Illustration 6.79 DD_GoToSetpoint_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Execute	BOOL	FALSE	Starts the motion at rising edge.
Direction	MC_DIRECTION_ISD51x	mcShortestWay_ISD51x	Direction of motion. See *Table 6.42*.
Duration	UINT	0	Duration of this movement [ms].

Variable name	Data type	Default value	Description
VAR_OUTPUT			
Done	BOOL		The command has been sent.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.62 DD_GoToSetpoint_ISD51x

6.5.6.9 DD_ReadCamInfo_ISD51x

This function block is used for basic and advanced CAMs. It provides the information that the axis sends out during the processing of the CAM. The meaning of the status code and the parameters can be found in *chapter 2.4.5.5 Advanced CAM*.

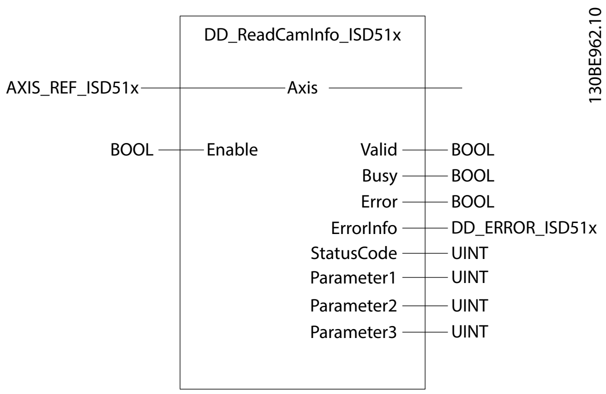


Illustration 6.80 DD_ReadCamInfo_ISD51x

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Axis	AXIS_REF_ISD51x		Reference to the axis. See *chapter 6.5.4.1 AXIS_REF_ISD51x*.
VAR_INPUT			
Enable	BOOL	FALSE	Read the information continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid set of outputs.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
StatusCode	UINT		The status code differentiates what kind of information it is.
Parameter1	UINT		Parameter 1: Meaning depends on the *StatusCode*.
Parameter2	UINT		Parameter 2: Meaning depends on the *StatusCode*.
Parameter3	UINT		Parameter 3: Meaning depends on the *StatusCode*.

Table 6.63 DD_ReadCamInfo_ISD51x

6.5.7 Drive – CAM Creation

6.5.7.1 Basic CAM

Use these function blocks and the corresponding structure to generate and modify a basic CAM inside the PLC. The information inside the structure can also be stored in a file (see *Table 6.70*), which can be sent to the servo drive (as described in *chapter 6.5.6.1 MC_CamTableSelect_ISD51x*). Use *DD_LoadBasicCamFromFile_ISD51X* (see *Table 6.69*) to restore the information from the file to the structure.

The structure of the basic CAM is built up as shown in *Table 6.64*.

DD_BASIC_CAM_ISD51x

Variable name	Data type	Default value	Description
MasterScaling	DD_FACTOR_ISD51x		Parameters for master scaling. See *Table 6.66*.
SlaveScaling	DD_FACTOR_ISD51x		Parameters for slave scaling. See *Table 6.66*.
ControlParameterSet1	DD_CONTROL_PARAMETER_ISD51x		Parameters for control parameter set 1. See *Table 6.67*.
ControlParameterSet2	DD_CONTROL_PARAMETER_ISD51x		Parameters for control parameter set 2. See *Table 6.67*.
FollowingError	DD_FOLLOWING_ERROR_ISD51x		Parameters for following error set-up. See *Table 6.68*.
DataPoints	ARRAY of DD_DATA_POINT_ISD51x		Array of data points for the basic CAM. See *Table 6.65*.
NumberOfDataPoints	UINT	0	Gives the number of used data points inside the array.

Table 6.64 DD_BASIC_CAM_ISD51x

The basic CAM consists of header information, such as master and slave scaling, control parameter sets 1 and 2, and error parameters. The parameters are optional within the file. If all the subelements of 1 structure are *0*, the element is not created within the CAM file.

The data points, which define the functionality of the CAM, are kept in an array of data points. The number of data points used must be written to the variable *NumberOfDataPoints*. Each data point contains the elements shown in *Table 6.65*.

DD_DATA_POINT_ISD51x

Variable name	Data type	Default value	Description
MasterPosition	REAL	0	Master position for this data point. Given in revolutions of guide value. Value range: 0–1.
SlavePosition	REAL	0	Axis position for this data point. Given in revolutions of rotor position. This is the position at gear in (motor side).
Velocity	REAL	0	Velocity of the axis in this data point. The velocity must be given as a factor between the velocity of the axis in relation to the velocity of the guide value.
Acceleration	REAL	0	Acceleration of the axis in this data point. The acceleration must be given as a factor between the acceleration of the axis in relation to the velocity of the guide value.

Table 6.65 DD_DATA_POINT_ISD51x

For this structure, the optional elements (*Velocity* and *Acceleration*) are always written to the file as they do not change the behavior.

DD_FACTOR_ISD51x

Variable name	Data type	Default value	Description
Numerator	DINT	0	Numerator part of the factor. 2 negative values result in a positive factor.
Denominator	DINT	0	Denominator part of the factor. 2 negative values result in a positive factor.

Table 6.66 DD_FACTOR_ISD51x

DD_CONTROL_PARAMETER_ISD51x

Variable name	Data type	Default value	Description
SpeedP	REAL	0	Proportional part of the speed controller.
SpeedI	REAL	0	Integral part of the speed controller.
SpeedD	REAL	0	Differential part of the speed controller.
Inertia	REAL	0	Inertia used for feed-forward calculations. [kg m^2]
PositionP	REAL	0	Proportional part of the position controller.
PositionD	REAL	0	Differential part of the position controller.

Table 6.67 DD_CONTROL_PARAMETER_ISD51x

6

6

DD_FOLLOWING_ERROR _ISD51x

Variable name	Data type	Default value	Description
WindowRev	REAL	0	Following error window [rev].
TimeOut	UINT	0	Following error timeout [ms].

Table 6.68 DD_FOLLOWING_ERROR _ISD51x

DD_LoadBasicCamFromFile_ISD51x

This function block is used to fill the CAM structure out of an existing basic CAM file. The validity of the XML-structure is not checked by this function block.

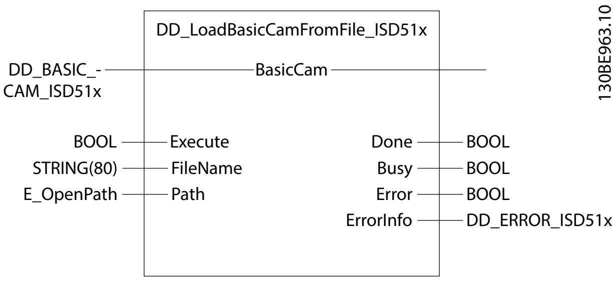


Illustration 6.81 DD_LoadBasicCamFromFile_ISD51x in TwinCAT®

Variable name	Data type	Default value	Description
VAR_IN_OUT			
BasicCam	DD_BASIC_CAM_ISD51x		Structure filled with the information of the file. This structure is cleared after activation. See *chapter 6.5.7.1 Basic CAM*.
VAR_INPUT			
Execute	BOOL	FALSE	Fills the CAM structure with the file information.
FileName	STRING[80]	‘’	Filename of the CAM profile file on the PLC.
pDevice	UDINT	0	Automation Studio™ only: Pointer to the device name on which the file is located.
Path	E_OpenPath	PATH_GENERIC	TwinCAT® only: The variable of this type selects generic or 1 of the TwinCAT® system paths on the target device.
VAR_OUTPUT			
Done	BOOL		CAM profile has been read and structure can now be used.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.69 DD_LoadBasicCamFromFile_ISD51x

DD_SaveBasicCamToFile_ISD51x

This function block is used to save a CAM structure into a CAM file. If the file already exists, it is overwritten. The optional elements in the CAM (for example, Master/slave scaling, or the control parameter sets) are not saved in the file if all subelements of the element are *0*.

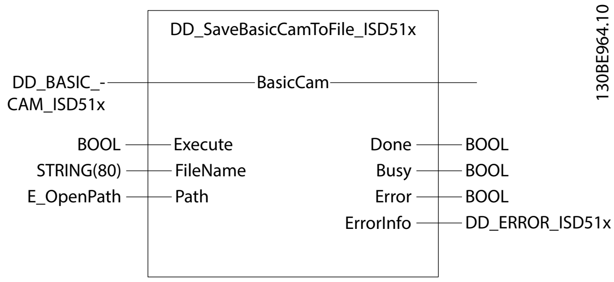


Illustration 6.82 DD_SaveBasicCamToFile_ISD51x in TwinCAT®

Variable name	Data type	Default value	Description
VAR_IN_OUT			
BasicCam	DD_BASIC_CAM_ISD51x		Content of this structure is written to a file. Do not change the content of the structure while this function block is *Busy*.
VAR_INPUT			
Execute	BOOL	FALSE	Write information from structure to the file.
FileName	STRING[80]	‘’	File name of the CAM profile file on the PLC.
pDevice	UDINT	0	Automation Studio™ only: Pointer to the device name on which the file is located.
Path	E_OpenPath	PATH_GENERIC	TwinCAT® only: The variable of this type selects generic or 1 of the TwinCAT® system paths on the target device.
VAR_OUTPUT			

Variable name	Data type	Default value	Description
Done	BOOL		CAM profile has been written to the file.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.70 DD_SaveBasicCamToFile_ISD51x

6.5.8 SAB

6.5.8.1 SAB_REF

Represents the state of an ISD 510 Servo Access Box. It handles the PDO communication and the internal state. To use the SAB-related function blocks, instantiate 1 function block for each SAB used. Also, connect the inputs and outputs to the objects that are mapped in the development environment for synchronous communication (see *chapter 6.3.1.2 Creating a TwinCAT® Project* and *chapter 6.4.1.2 Creating an Automation Studio ™ Project*).

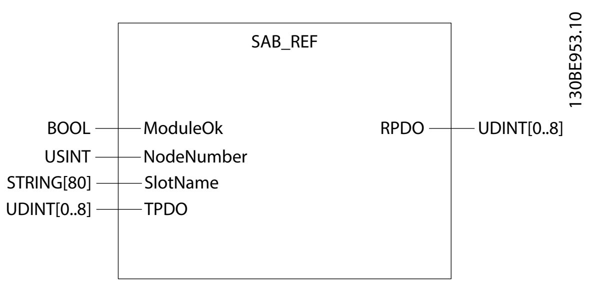


Illustration 6.83 SAB_REF in Automation Studio™

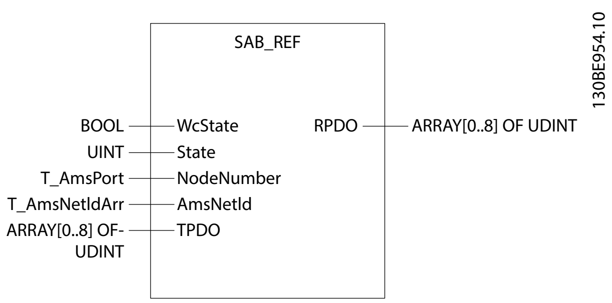


Illustration 6.84 SAB_REF in TwinCAT®

6.5.8.2 DD_Power_SAB

Controls the power for the output lines (*On* or *Off*). The *Enable* input enables the DC-link voltage on the output lines and not the function block itself.

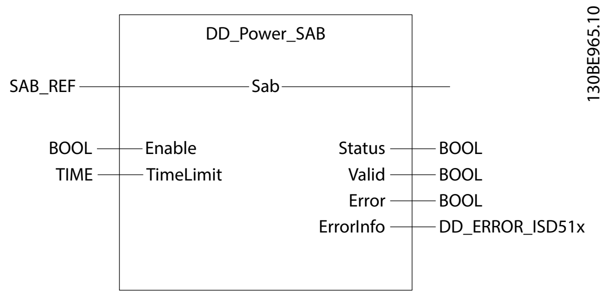


Illustration 6.85 DD_Power_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	If this input is true, DC-link voltage for the lines is enabled.
TimeLimit	TIME	t#0ms	Timeout after which an error is signaled if the *Status* has not changed to *TRUE* while *Enable* is *TRUE*. Set the value to *0* to disable the time limit.
VAR_OUTPUT			
Status	BOOL		Effective state of the DC-link voltage on the output lines.
Valid	BOOL		If *TRUE*, the function block has a valid set of outputs.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.71 DD_Power_SAB

If the *DD_Power_SAB* function block is called when *Enable* = *TRUE* while in *Standby* state, the SAB state changes to *Operation enabled*.

The error variable is set to *TRUE* if the *Enable* input is *TRUE* for the time specified in the input *TimeLimit*, while the *Status* remains *FALSE*. It indicates a hardware problem with the power stage. If power fails, also during operation, it generates a transition to the *Fault* state.

Only issue 1 function block *DD_Power_SAB* per SAB.

The *Enable* input in this function block is not an *Enable* input as described in *chapter 6.5.2.2 Function Blocks with*

Enable Input. Therefore, the general rules for the *Enable* input do not apply here. This function block is implicitly enabled. The *Enable* input of this function block controls the DC-link voltage on the output lines of the SAB. All outputs are always updated (so *Valid* can be *TRUE*, even if *Enable* is *FALSE*).

The input *TimeLimit* represents the maximal duration of functionality. If the *TimeLimit* is exceeded during switching on the SAB, an *Error* is signaled at the outputs. However, the functionality according to the *Enable* input is continued. If *Enable* is *TRUE*, the function block continues to attempt to enable the DC-link voltage of the SAB. If *Enable* is *FALSE*, the function block continues to attempt to disable the DC-link voltage of the SAB. If the SAB starts reacting again, the *Error* output can change to *FALSE* again without a new rising edge of *Enable*. Set the value to *0* to disable the time limit functionality.
The command is transferred immediately, but it can take some time until the SAB has enabled the DC-link voltage and the output *Status* becomes *TRUE*.

6

6.5.8.3 DD_Reset_SAB

This function block makes the transition from the state *Fault* to *Standby* by resetting all internal SAB-related errors. It does not affect the output of the function block instances.
The command is transferred and executed immediately.

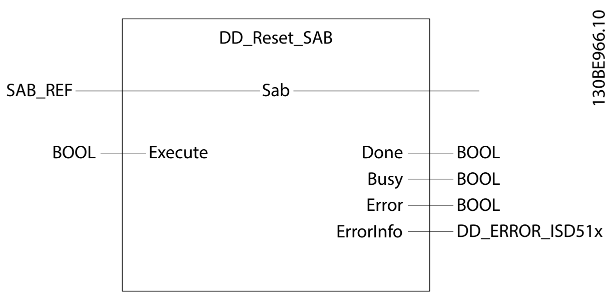


Illustration 6.86 DD_Reset_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Execute	BOOL	FALSE	Resets all internal SAB-related errors.
VAR_OUTPUT			
Done	BOOL		*Standby* state is reached
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.72 DD_Reset_SAB

6.5.8.4 DD_ReadSabInfo_SAB

This function block returns detailed information about the SAB.
The output data is available immediately.

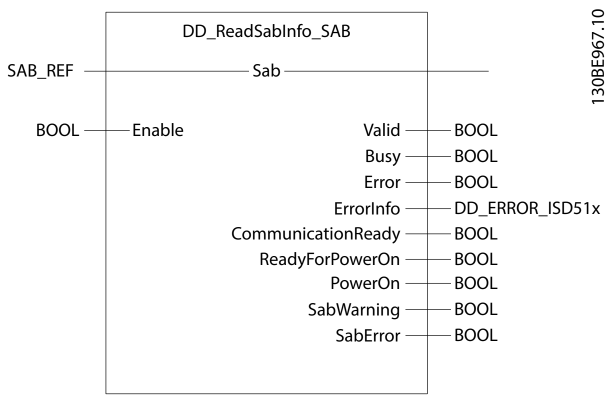


Illustration 6.87 DD_ReadSabInfo_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the SAB information continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid set of outputs.

Variable name	Data type	Default value	Description
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Communica-tionReady	BOOL		*Network* is initialized and ready for communication.
ReadyFor-PowerOn	BOOL		SAB is ready to enable the power on the lines (mains voltage is applied on the SAB).
PowerOn	BOOL		*TRUE* shows that the power is switched on at the output lines.
SabWarning	BOOL		Warning(s) present on the SAB.
SabError	BOOL		Error(s) present on the SAB.

Table 6.73 DD_ReadSabInfo_SAB

6.5.8.5 DD_ReadSabError_SAB

This function block presents general SAB errors unrelated to the function blocks (for example, overtemperature in the SAB). The output *SabErrorID* gives the last error that occurred in the SAB (see *chapter 9.3.2 Warnings and Alarms*).
The output data needs to be read from the device and is therefore not immediately available.

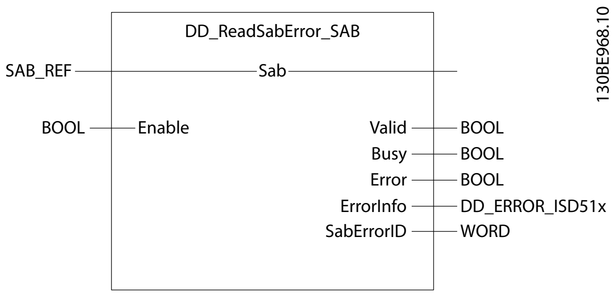


Illustration 6.88 DD_ReadSabError_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the SAB information continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid set of outputs.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
SabErrorID	WORD		The value of the SAB error. Available in the list of constants: SabErrorCodes.

Table 6.74 DD_ReadSabError_SAB

6.5.8.6 DD_ReadSabWarning_SAB

This function block presents general SAB warnings unrelated to the function blocks (for example, high temperature in the SAB). The output *SabWarningID* gives the last warning that occurred in the SAB (see *chapter 9.3.2 Warnings and Alarms*).
The output data needs to be read from the device and is therefore not immediately available.

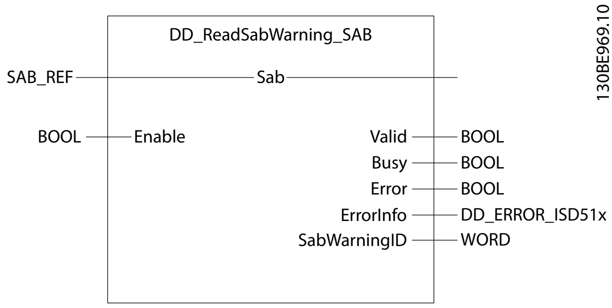


Illustration 6.89 DD_ReadSabWarning_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the SAB information continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid set of outputs.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
SabWarningID	WORD		The value of the SAB warning. Available in the list of constants: SabErrorCodes.

Table 6.75 DD_ReadSabWarning_SAB

6.5.8.7 DD_ReadVersion_SAB

This function block reads the firmware version and the serial number of the SAB. *Done* is *TRUE* when the data-outputs are valid. The version number consists of a major, a minor, and beta version number (see *chapter 8.29 Object 0x400A: Communication Settings*).
The output data needs to be read from the device and is therefore not immediately available.

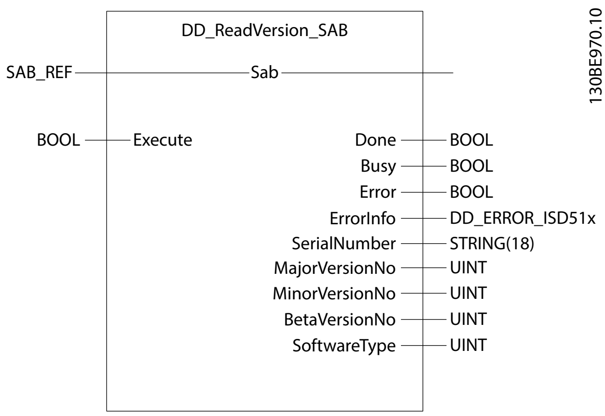


Illustration 6.90 DD_ReadVersion_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Execute	BOOL	FALSE	Read the information at rising edge.
VAR_OUTPUT			
Done	BOOL		The values have successfully been read from the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
SerialNumber	STRING[18]		Serial number of the SAB.
MajorVersionNo	UINT		Major firmware version number.
MinorVersionNo	UINT		Minor firmware version number.
BetaVersionNo	UINT		Beta firmware version number.
SoftwareType	UINT		Loaded software type.

Table 6.76 DD_ReadVersion_SAB

6.5.8.8 DD_UpdateFirmware_SAB

This function block updates the firmware of the SAB, which can only be carried out in *Standby* state. Carry out a power cycle to use the SAB after updating. For more details on the firmware update, see *chapter 3.8 Firmware Update*.

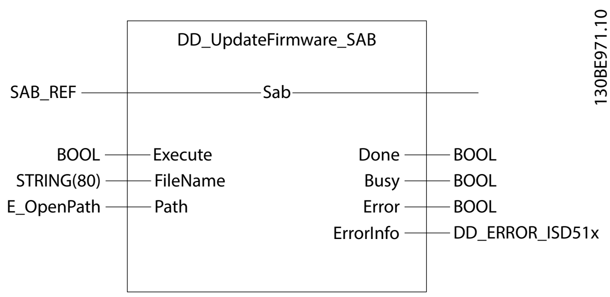


Illustration 6.91 DD_UpdateFirmware_SAB in TwinCAT®

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Execute	BOOL	FALSE	Starts the firmware update at a rising edge.
FileName	STRING[80]	''	File name of the firmware file on the PLC.
pDevice	UDINT	0	Automation Studio™ only: Pointer to the device name on which the firmware file is located.
Path	E_OpenPath	PATH_GENERIC	TwinCAT® only: The variable of this type selects generic or 1 of the TwinCAT® system paths on the target device.
VAR_OUTPUT			
Done	BOOL		The firmware file has successfully been transferred. Carry out a power cycle on the SAB to enable the new firmware.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.77 DD_UpdateFirmware_SAB

6.5.8.9 DD_ReadDcLinkPower_SAB

This function block provides the value of the actual DC-link power (see *chapter 8.4 Object 0x2001: DC-link Related Values*) if *Enable* is set. *Valid* is *TRUE* when the data-output *Power* is valid. If *Enable* is reset, the data loses its validity and all outputs are reset, regardless of whether new data is available.
The output data is available immediately.

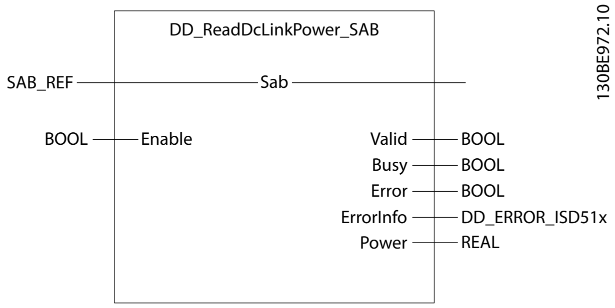


Illustration 6.92 DD_ReadDcLinkPower_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Power	REAL		Value of the actual DC-link current [Ampere].

Table 6.78 DD_ReadDcLinkPower_SAB

6.5.8.10 DD_ReadDcLinkVoltage_SAB

This function block provides the value of the actual DC-link voltage (see *chapter 8.4 Object 0x2001: DC-link Related Values*) if *Enable* is set. *Valid* is *TRUE* when the data-output *Voltage* is valid. If *Enable* is reset, the data loses its validity and all outputs are reset, regardless of whether new data is available.
The output data is available immediately.

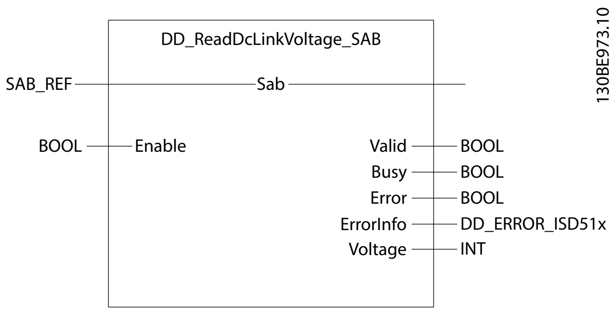


Illustration 6.93 DD_ReadDcLinkVoltage_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Voltage	INT		Value of the actual DC-link voltage [V].

Table 6.79 DD_ReadDcLinkVoltage_SAB

6.5.8.11 DD_ReadParameter4_SAB

This function block asynchronously reads general objects of up to 4 bytes from the object dictionary of the SAB if the *Enable* input is set. For the index and the sub-index of the parameters, see *chapter 8 SAB Parameter Description*. The output data needs to be read from the device and is therefore not immediately available.

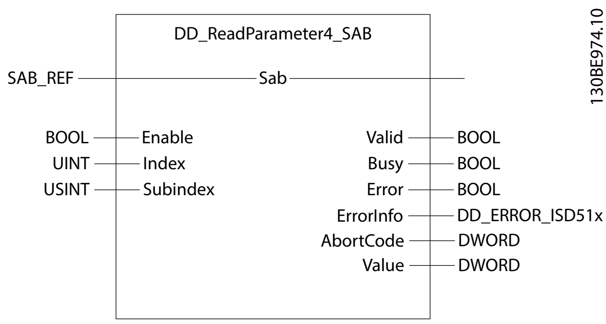


Illustration 6.94 DD_ReadParameter4_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
Index	UINT	0	Index of the object to be read.
Subindex	USINT	0	Sub-index of the object to be read.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.
Value	DWORD		Value of the specified parameter.

Table 6.80 DD_ReadParameter4_SAB

6.5.8.12 DD_ReadParameter_SAB

This function block asynchronously reads general objects from the object dictionary of the SAB if the *Enable* input is set. For the index and the sub-index of the parameters, see *chapter 8 SAB Parameter Description*. The output data needs to be read from the device and is therefore not available immediately.

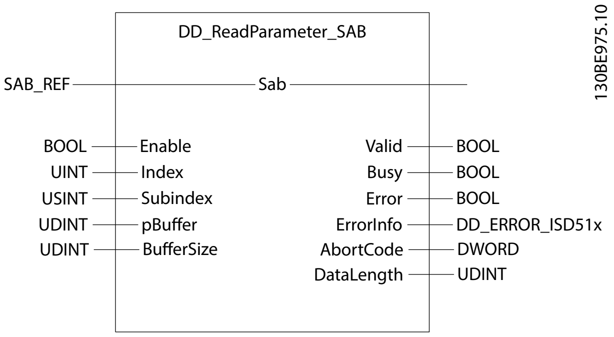


Illustration 6.95 DD_ReadParameter_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled
Index	UINT	0	Index of the object to be read.
Subindex	USINT	0	Sub-index of the object to be read.
pBuffer	UDINT	0	Pointer to a buffer where the read data will be placed (use ADR() function).
BufferSize	UDINT	0	Maximum size of the read data (size of the provided buffer given in byte; use SIZEOF() function)
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.
DataLength	UDINT		Length of the read data [Byte].

Table 6.81 DD_ReadParameter_SAB

6.5.8.13 DD_WriteParameter4_SAB

This function block asynchronously writes general objects of up to 4 bytes to the object dictionary of the SAB. For the index and the sub-index of the parameters, see *chapter 8 SAB Parameter Description*.
The data needs to be written to the device asynchronously and is therefore not available in the SAB immediately.

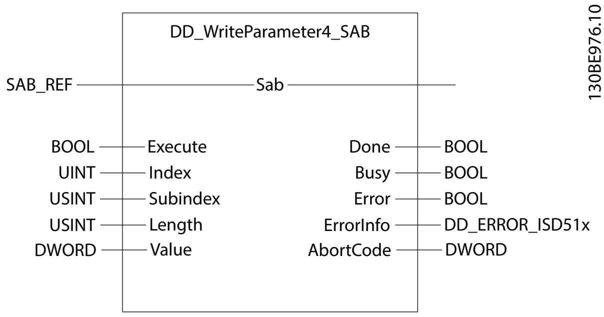


Illustration 6.96 DD_WriteParameter4_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Execute	BOOL	FALSE	Write the value of the parameter at rising edge.
Index	UINT	0	Index of the object to be written.
Subindex	USINT	0	Sub-index of the object to be written.
Length	USINT	0	Length of the data to be written [Byte].
Value	DWORD	0	New value of the specified parameter.
VAR_OUTPUT			
Done	BOOL		The value has been successfully written to the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.

Table 6.82 DD_WriteParameter4_SAB

6.5.8.14 DD_WriteParameter_SAB

This function block asynchronously writes general objects to the object dictionary of the SAB. For the index and the sub-index of the parameters, see *chapter 8 SAB Parameter Description*. The data needs to be written to the device asynchronously and is therefore not immediately available in the SAB.

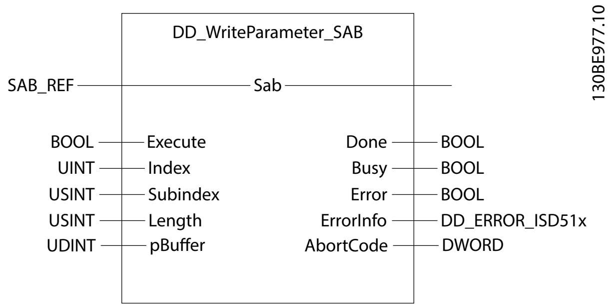


Illustration 6.97 DD_WriteParameter_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Execute	BOOL	FALSE	Write the value of the parameter at rising edge.
Index	UINT	0	Index of the object to be written.
Subindex	USINT	0	Sub-index of the object to be written.
Length	USINT	0	Length of the data to be written [Byte].
pBuffer	UDINT	0	Pointer to the buffer that contains the data to be written (use ADR() function).
VAR_OUTPUT			
Done	BOOL		The value has successfully been written to the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
AbortCode	DWORD		SDO abort code if there is an error. Available in the list of constants: SdoAbortCodes.

Table 6.83 DD_WriteParameter_SAB

6.5.8.15 DD_Trace_SAB

This function block is used to start a trace in the SAB with the settings given in the input variables. The behavior of this function block is the same as described for the function block *DD_Trace_ISD51x* (see *chapter 6.5.4.24 DD_Trace_ISD51x*).

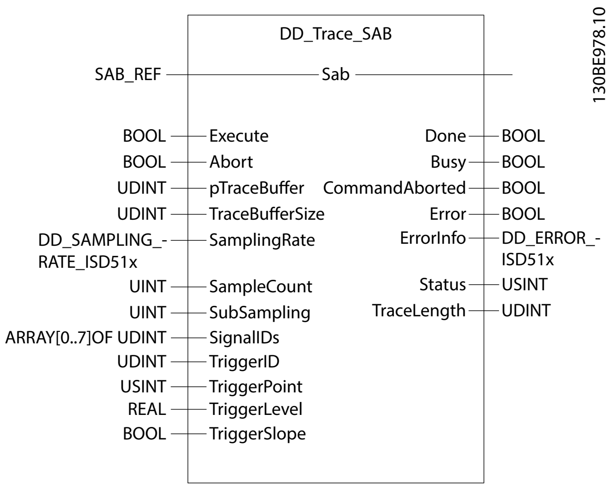


Illustration 6.98 DD_Trace_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Execute	BOOL	FALSE	Starts the trace functionality at rising edge and keeps on polling until the data is available.
Abort	BOOL	FALSE	Abort the ongoing trace. New values are only evaluated on rising edge of *Execute*.
pTraceBuffer	UDINT	0	Reference to a buffer where the read trace data will be placed; Use ADR() function.
TraceBufferSize	UDINT	0	Size of the trace buffer; use SIZEOF() function. Size of the provided buffer [Byte].

Variable name	Data type	Default value	Description
SamplingRate	DD_SAMPLING_RATE_ISD51x	ddFastTask_ISD51x	Sampling rate of the trace.
SampleCount	UINT	4000	Number of samples to be traced per channel.
SubSampling	UINT	1	Multiplier to adjust time difference between trace samples.
SignalIDs	ARRAY[0..7] of UDINT	[0, 0, 0, 0, 0, 0, 0, 0]	IDs of the signals to be traced. Available in the list of constants: *SabTraceSignals*.
TriggerID	UDINT	0	ID of the signal used for triggering. Available in the list of constants: SabTraceSignals. Set to *0* for instant tracing.
TriggerPoint	USINT	10	Amount of pre-trigger history [percentage]. Value range: 0–100
TriggerLevel	REAL	0.0	Level at which is triggered (in trigger signal units).
TriggerSlope	BOOL	TRUE	*TRUE*: Triggers on rising slope. *FALSE*: Triggers on falling slope.
VAR_OUTPUT			
Done	BOOL		The trace has successfully been recorded and read from the device.
Busy	BOOL		The function block is not finished and new output values are to be expected.
CommandAborted	BOOL		Trace has been aborted successfully.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
Status	USINT		Holds the state of the tracing process (valid while *Busy* is *TRUE*): 0 = Configuring the trace. 1 = Trace configured and started. 2 = Waiting for trigger. 3 = Triggered; Waiting for completion of the trace. 4 = Uploading trace data. 5 = Trace data received successfully.
TraceLength	UDINT		Length of trace data that has been uploaded [Byte]. 1 sample (REAL) is 4 Bytes long.

Table 6.84 DD_Trace_SAB

6.5.8.16 DD_SimulateGuideValue_SAB

This function block is used to simulate a guide value without a physical encoder connected. It can be conserved as virtual master axis. Linear ramps are used for the calculation of the position guide value. The simulated value can be read by using the function blocks *DD_ReadPosGuideValueRef_SAB* (see *chapter 6.5.8.17 DD_ReadPosGuideValueRef_SAB*) and *DD_ReadVelGuideValueRef_SAB* (see *chapter 6.5.8.18 DD_ReadVelGuideValueRef_SAB*).

Object *0x2063: Guide value reference option code* must be set accordingly to enable the simulation functionality (see *chapter 8.18 Object 0x2063: Guide Value Reference Option Code*).

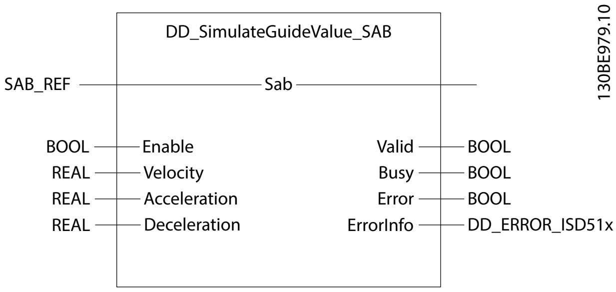


Illustration 6.99 DD_SimulateGuideValue_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	If *Enable* is *TRUE*, the guide value is simulated by the SAB.
Velocity	REAL	0.0	Velocity of the guide value [rps].
Acceleration	REAL	0.0	Acceleration used for the guide value calculation [rps/s].
Deceleration	REAL	0.0	Deceleration used for the guide value calculation [rps/s].
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.

Table 6.85 DD_SimulateGuideValue_SAB

6.5.8.17 DD_ReadPosGuideValueRef_SAB

This function block provides the value of the position guide value reference if *Enable* is set (see *chapter 8.17 Object 0x2062: Position Guide Value Reference*). *Valid* is *TRUE* when the data-output *PosGuideValue* is valid. If *Enable* is reset, the data loses its validity and all outputs are reset, regardless of whether new data is available. The output data is available immediately.

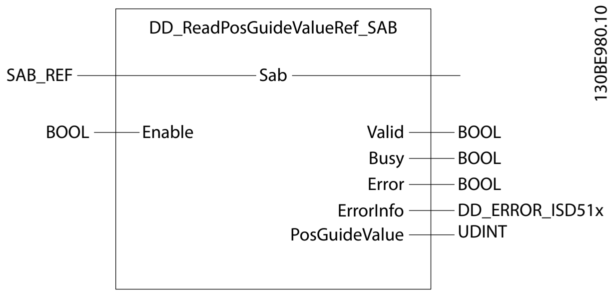


Illustration 6.100 DD_ReadPosGuideValueRef_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.

Variable name	Data type	Default value	Description
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
PosGuideValue	UDINT		Value of the position guide value reference [position guide value unit].

Table 6.86 DD_ReadPosGuideValueRef_SAB

6.5.8.18 DD_ReadVelGuideValueRef_SAB

This function block provides the value of the velocity guide value reference if *Enable* is set (see *chapter 8.19 Object 0x2065: Velocity Guide Value Reference*). *Valid* is *TRUE* when the data-output *VelGuideValue* is valid. If *Enable* is reset, the data loses its validity and all outputs are reset, regardless of whether new data is available.
The output data is available immediately.

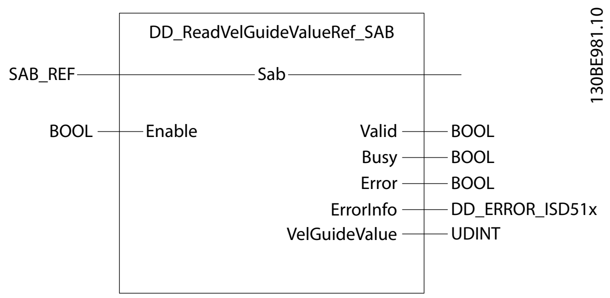


Illustration 6.101 DD_ReadVelGuideValueRef_SAB

Variable name	Data type	Default value	Description
VAR_IN_OUT			
Sab	SAB_REF		Reference to the SAB. See *chapter 6.5.8.1 SAB_REF*.
VAR_INPUT			
Enable	BOOL	FALSE	Get the value of the parameter continuously while enabled.
VAR_OUTPUT			
Valid	BOOL		The function block has a valid output.
Busy	BOOL		The function block is not finished and new output values are to be expected.
Error	BOOL		An error has occurred within the function block.
ErrorInfo	DD_ERROR_ISD51x		Error identification and instance identifier. See *chapter 6.5.2.3 Error Indication*.
VelGuideValue	UDINT		Value of the velocity guide value reference. [velocity guide value unit].

Table 6.87 DD_ReadVelGuideValueRef_SAB

6.6 Simple Programming Template

TwinCAT®
A basic sample PLC application for starting up the ISD 510 servo system with 1 SAB and 2 axes is provided. The project *ISD_System_SampleProject* can be downloaded from the Danfoss website.

Automation Studio™
Detailed information on how to open the sample project within the ISD package in Automation Studio™ can be found in the Automation Studio™ Help. Open the B&R Help Explorer and go to [Programming → Examples → Adding sample programs] and follow the instructions for library samples.

7 Servo Drive Parameter Description

7.1 Overview

Parameters are organized in various parameter groups for easy selection of correct parameters.

7.2 Controlword Object

7.2.1 Parameter 16-00 Controlword (0x6040)

This object indicates the received command controlling the state machine. It is structured as defined in *Table 7.2*. The commands are coded as given in *Table 7.5*.

Bits 0, 1, 2, 3, and 7 are supported to control the DS402 state machine (see *chapter 2.3.1 State Machine*). Bits 4, 5, 6, and 9 are supported according to the mode of operation. The descriptions of these bits are found in the modes of operation (see *chapter 7.2.1.1 Controlword in Profile Position Mode* to *chapter 7.2.1.9 Controlword in Cyclic Synchronous Velocity Mode*).

For details about the *Halt* bit, refer to the corresponding descriptions for the different modes of operation. *Halt* does not necessarily have to reach zero velocity if the *Halt* bit has been reset before reaching speed zero.

The manufacturer-specific bits 11 (dcs) and 12 (do) are processed according to the configuration of object *Digital Output Configuration (0x2FFF)*. If the configuration is set to *2: Control over Controlword*, then only bit 12 (do) is processed. If the configuration is set to *3: Control over Digital CAM Switch*, then only bit 11 (dcs) is processed. Otherwise, bits 11 and 12 are not processed. For more information, see *chapter 7.21.5 Parameter 52-05: Digital Output Configuration (0x2FFF)*.

Attribute	**Value**
Index	0x6040
LCP parameter number	16-00
Name	Controlword
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See tables in this chapter.
Default value	–

Table 7.1 0x6040: Controlword

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">ms</td><td rowspan="2">r</td><td rowspan="2">oms</td><td rowspan="2">h</td><td rowspan="2">fr</td><td colspan="3" rowspan="2">oms</td><td rowspan="2">eo</td><td rowspan="2">qs</td><td rowspan="2">ev</td><td rowspan="2">so</td></tr>
<tr><td>cs</td><td>–</td><td>–</td><td>do</td><td>dcs</td></tr>
<tr><td colspan="8">MSB</td><td colspan="8">LSB</td></tr>
</table>

Table 7.2 0x6040: Controlword

ms	Manufacturer-specific
r	Reserved
oms	Operation mode specific
h	Halt
fr	Fault reset
eo	Enable operation
qs	Quick stop
ev	Enable voltage
so	Switch on
cs	Control set selection
do	Set/reset the digital output
dcs	Digital CAM switching functionality enabled/disabled

Table 7.3 Definition of Controlword Bits

Bit	Value	Definition
11	0	Digital CAM switching functionality must be disabled (see *chapter 2.5.1 Digital CAM Switch*).
	1	Digital CAM switching functionality must be enabled (see *chapter 2.5.1 Digital CAM Switch*).
12	0	Digital output is cleared.
	1	Digital output is set.
15	0	Control parameter set 1 is used (objects 0x2012 and 0x2013; see *chapter 2.3.6 Control Loops*).
	1	Control parameter set 2 is used (objects 0x2014 and 0x2015; see *chapter 2.3.6 Control Loops*).

Table 7.4 Manufacturer-specific Bits in Controlword

Command	Controlword bits					Transitions
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X[1)]	1	1	0	2, 6, 8
Switch on	0	0	1	1	1	3
Switch on and enable operation	0	1	1	1	1	3, 4[2)]
Disable voltage	0	X[1)]	X[1)]	0	X[1)]	7, 9, 10, 12
Quick stop	0	X[1)]	0	1	X[1)]	7, 10, 11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4, 16
Fault reset	0→1	X[1)]	X[1)]	X[1)]	X[1)]	15

Table 7.5 Command Coding

1) X denotes Don't care.

2) Automatic transition to Enable operation state after executing Switched on state functionality.

NOTICE

See *chapter 2.3.1 State Machine* for the relevant transitions.

7.2.1.1 Controlword in Profile Position Mode

Table 7.6 shows the structure of the *Controlword*. *Table 7.7* defines the values for bit 4, 5 and 9 of the *Controlword*. *Table 7.8* defines the values for bit 6 and 8 of the *Controlword*. If no positioning is in progress, the rising edge of bit 4 starts the positioning of the axis. If positioning is in progress, the values given in *Table 7.7* define the behavior.

15 10	9	8	7	6	5	4	3 0
See *Table 7.2*	Change on setpoint	Halt	See *Table 7.2*	Abs/rel	Change set immediately	New setpoint	See *Table 7.2*
MSB							LSB

Table 7.6 Controlword for Profile Position Mode

Bit 9	Bit 5	Bit 4	Definition
0	0	0 → 1	Positioning is completed (target reached) before the next one is started.
X (= don't care)	1	0 → 1	Next positioning is started immediately.
1	0	0 → 1	Positioning with the current profile velocity is carried out up to the current setpoint and then the next positioning is applied.

Table 7.7 Definition of Bits 4, 5, and 9

Bit	Value	Definition
6	0	Target position is an absolute value.
	1	Target position is a relative value depending on object 0x60F2 (see *chapter 7.10.3 Parameter: Positioning Option Code (0x60F2)*).
8	0	The motion is executed or continued.
	1	Stop axis according to halt option code (see *chapter 7.20.7 Parameter 50-47: Halt Option Code (0x605D)*).

Table 7.8 Definition of Bits 6 and 8

7.2.1.2 Controlword in Profile Velocity Mode

Table 7.9 shows the structure of the *Controlword*. *Table 7.10* defines the values for bit 8 of the *Controlword*.

15 10	9	8	7	6 4	3 0
See *Table 7.2*	Reserved (0)	Halt	See *Table 7.2*	Reserved (0)	See *Table 7.2*
MSB					LSB

Table 7.9 Controlword for Profile Velocity Mode

Bit	Value	Definition
8	0	The motion is executed or continued.
	1	Stop axis according to halt option code (see *chapter 7.20.7 Parameter 50-47: Halt Option Code (0x605D)*).

Table 7.10 Definition of Bit 8

7.2.1.3 Controlword in Profile Torque Mode

Table 7.11 shows the structure of the *Controlword*. *Table 7.12* defines the values for bit 8 of the *Controlword*.

15 10	9	8	7	6 4	3 0
See *Table 7.2*	Reserved (0)	Halt	See *Table 7.2*	Reserved (0)	See *Table 7.2*
MSB					LSB

Table 7.11 Controlword for Profile Torque Mode

Bit	Value	Definition
8	0	The motion is executed or continued.
	1	Stop axis according to halt option code (see *chapter 7.20.7 Parameter 50-47: Halt Option Code (0x605D)*).

Table 7.12 Definition of Bit 8

7.2.1.4 Controlword in Homing Mode

Table 7.13 shows the structure of the *Controlword*. *Table 7.14* defines the values for bits 4 and 8 of the *Controlword*.

15 10	9	8	7	6	4	3 0
See *Table 7.2*	Reserved (0)	Halt	See *Table 7.2*	Reserved (0)	Homing operation start	See *Table 7.2*
MSB						LSB

Table 7.13 Controlword in Homing Mode

Bit	Value	Definition
4	0	Do not start homing procedure.
	1	Start or continue homing procedure.
8	0	Enable bit 4.
	1	Stop axis according to halt option code (see *chapter 7.20.7 Parameter 50-47: Halt Option Code (0x605D)*).

Table 7.14 Definition of Bits 4 and 8

7

7.2.1.5 Controlword in CAM Mode

Table 7.15 shows the structure of the *Controlword*. The rising edge of bit 4 starts the CAM profile activation request. *Table 7.16* defines the values for bits 5, 6, and 9 of the *Controlword*. It is assumed that the activation request of a CAM is edge-triggered 0→1, otherwise the axis has no clear point in time when to activate a new CAM profile, or to reactivate a CAM profile without any content changes.

15 10	9	8	7	6	5	4	3 0
See *Table 7.2*	Use blend distance	Reserved (0)	See *Table 7.2*	Control parameter source	Change CAM immediately	New CAM	See *Table 7.2*
MSB							LSB

Table 7.15 Controlword in CAM Mode

Bit	Value	Definition
5	0	Currently active CAM profile is finished first (target reached).
	1	Next CAM profile is started immediately.
6	0	Control parameter set selection uses the manufacturer-specific bit of the *Controlword* (bit 15). This is default behavior as in other modes of operation.
	1	Control parameter set selection is done automatically inside the CAM profile. The manufacturer-specific bit of the *Controlword* (bit 15) for selection is ignored by the axis.
9	0	Automatically blend to the beginning of the new CAM: • Basic CAM: 1st data point • Advanced CAM: starting node
	1	Uses value of object 0x380A (see *chapter 7.14.11 Parameter: Minimum Blending Distance (0x380A)*) as the minimum length for blending to a new CAM.

Table 7.16 Definition of Bits 5, 6, and 9

7.2.1.6 Controlword in Gear Mode

Table 7.17 shows the structure of the *Controlword*. *Table 7.18* defines the values for the available synchronization modes (bits 5 and 6). *Table 7.19* defines the values for bit 9 of the *Controlword*.

15 10	9	8	7	6 5	4	3 0
See *Table 7.2*	Master sync direction	Reserved (0)	See *Table 7.2*	Synchronization mode	Reserved (0)	See *Table 7.2*
MSB						LSB

Table 7.17 Controlword in Gear Mode

Bit 6	Bit 5	Definition
0	0	Synchronize shortest way.
0	1	Synchronize according to slow down mode.
1	0	Synchronize according to catch up mode.
1	1	Velocity-controlled synchronization (during gear in).

Table 7.18 Definition of Bits 5 and 6 (Synchronization Modes)

Bit	Value	Definition
9	0	Not relevant for synchronization mode 11; For all other modes: Master start distance is in the positive direction of the guide value.
	1	Not relevant for synchronization mode 11; For all other modes: Master start distance is in the negative direction of the guide value.

Table 7.19 Definition of Bit 9

7.2.1.7 Controlword in ISD Inertia Measurement Mode

Table 7.17 shows the structure of the *Controlword*. A value of 1 in bit 4 starts the inertia measurement procedure. It must remain high as long as the procedure is running, otherwise it is aborted. *Table 7.21* defines the values for bits 4 and 8 of the *Controlword*.

15 10	9	8	7	6 5	4	3 0
See *Table 7.2*	Reserved (0)	Halt	See *Table 7.2*	Reserved (0)	Measurement start	See *Table 7.2*
MSB						LSB

Table 7.20 Controlword in ISD Inertia Measurement Mode

Bit	Value	Definition
4	0	Do not start the measurement or stop an ongoing measurement.
	1	Start measurement.
8	0	Enable bit 4.
	1	Stop axis according to halt option code (see *chapter 7.20.7 Parameter 50-47: Halt Option Code (0x605D)*).

Table 7.21 Definition of Bits 4 and 8

7.2.1.8 Controlword in Cyclic Synchronous Position Mode

The operation mode-specific bits and the *Halt* bit in the *Controlword* are ignored by the servo drive. The *Halt* function is controlled by the control device.

7.2.1.9 Controlword in Cyclic Synchronous Velocity Mode

The operation mode-specific bits and the *Halt* bit in the *Controlword* are ignored by the servo drive. The *Halt* function is controlled by the control device.

7.3 Statusword Object

7.3.1 Parameter 16-03 Statusword (0x6041)

This object provides the status of the state machine. The structure of the object is defined in *Table 7.23*. The oms bits are supported if the mode of operation is supported. If the related functionality of the oms bits is not available, the corresponding bit is 0. All implemented bits of the *Statusword* are valid regardless of the state of the state machine.

Attribute	**Value**
Index	0x6041
LCP parameter number	16-03
Name	Statusword
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See tables in this chapter.
Default value	–

Table 7.22 0x6041: Statusword

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">ms</td><td colspan="2" rowspan="2">oms</td><td rowspan="2">ila</td><td rowspan="2">tr</td><td rowspan="2">rm</td><td>ms</td><td rowspan="2">w</td><td rowspan="2">sod</td><td rowspan="2">qs</td><td rowspan="2">ve</td><td rowspan="2">f</td><td rowspan="2">oe</td><td rowspan="2">so</td><td rowspan="2">rtso</td></tr>
<tr><td>cs</td><td>sto</td><td>h</td></tr>
<tr><td colspan="8">MSB</td><td colspan="8">LSB</td></tr>
</table>

Table 7.23 0x6041: Statusword

ms	Manufacturer-specific
oms	Operation mode specific
ila	Internal limit active
tr	Target reached
rm	Remote
w	Warning
sod	Switch on disabled
qs	Quick stop
ve	Voltage enabled
f	Fault
oe	Operation enabled
so	Switched on
rtso	Ready to switch on
h	Is homed
ce	Command error
sto	STO active

Table 7.24 Definition of Statusword Bits

Statusword	State
xxxx xxxx x0xx 0000b	Not ready to switch on
xxxx xxxx x1xx 0000b	Switch on disabled
xxxx xxxx x01x 0001b	Ready to switch on
xxxx xxxx x01x 0011b	Switched on
xxxx xxxx x01x 0111b	Operation enabled
xxxx xxxx x00x 0111b	Quick stop active
xxxx xxxx x0xx 1111b	Fault reaction active
xxxx xxxx x0xx 1000b	Fault

Table 7.25 State Coding

If bit 4 (voltage enabled) of the *Statusword* is 1, this indicates that high voltage is switched on.

If bit 5 (quick stop) of the *Statusword* is 0, this indicates that the servo drive is reacting on a quick stop request.

If bit 7 (warning) of the *Statusword* is 1, this indicates the presence of a warning condition. Warning is not an error or fault. The status of the state machine is not changed. The cause of the warning is given in a special object, described in *chapter 7.22.10 Parameter 16-92: Warning Code (0x5FFE)*.

7

If bit 8 (manufacturer-specific: *Is Homed*) of the *Statusword* is 1, this indicates that there has been a successful homing procedure. For details, refer to *chapter 2.4.4 Homing Mode*.

If bit 9 (remote) of the *Statusword* is 1, this indicates that the *Controlword* is processed. If it is 0 (local), this indicates that the *Controlword* is not processed and the axis is controlled using the LCP.

If bit 10 (target reached) of the *Statusword* is *1*, this indicates that the servo drive has reached the setpoint (see *chapter 7.20.2 Parameter 50-42: Target Reached Option Code (0x2054)* for details on the meaning of "reached"). The setpoint is operating mode-specific and is defined in detail in the corresponding clauses. The change of a target value by software alters this bit. If the *Quick stop option code* (see *chapter 7.20.6 Parameter 50-46: Quick Stop Option Code (0x605A)*) is *5*, *6*, or *7*, then bit 10 is set to *1* as soon as the quick stop operation is finished and the servo drive is halted. If the same internal value is commanded, then bit 10 does not alter.

If bit 11 (internal limit active) of the *Statusword* is *1*, this indicates that an internal limit is active. This bit is set as soon as an internal limit occurs (for example, current limit) or when a trajectory is calculated and this is affected by at least 1 limitation (for example, maximum acceleration versus profile acceleration). When a new trajectory is calculated, the bit is re-evaluated.

Bits 12 and 13 of the *Statusword* are operation mode-specific. Refer to the corresponding descriptions for the different modes of operation for details on these bits.

If bit 14 (manufacturer-specific: STO) of the *Statusword* is *1*, this indicates that the Safe Torque Off has been activated. No torque is applied to the servo drive. Check the safety voltage.

If bit 15 (manufacturer-specific: command error) of the *Statusword* is *1*, this indicates that there has been a problem in executing the command that has been sent over PDO.

7.3.1.1 Statusword in Profile Position Mode

The *Profile Position* mode uses some bits of the *Statusword* for operation mode-specific purposes. *Table 7.26* shows the structure of the *Statusword*. *Table 7.27* defines the values for bits 10, 12, and 13.

15 14	13	12	11	10	9 0
See *Table 7.23*	Following error	Setpoint acknowledge	See *Table 7.23*	Target reached	See *Table 7.23*
MSB					LSB

Table 7.26 Statusword in Profile Position Mode

Bit	Value	Definition
13	0	No following error.
	1	Following error.
12	0	Previous setpoint already processed, waiting for new setpoint.
	1	Previous setpoint still in process, setpoint overwriting is accepted.
10	0	Halt (bit 8 in *Controlword*) = 0: Target position not reached. Halt (bit 8 in *Controlword*) = 1: Axis decelerates.
	1	Halt (bit 8 in *Controlword*) = 0: Target position reached. Halt (bit 8 in *Controlword*) = 1: Velocity of axis is 0.

Table 7.27 Definition of Bits 10, 12, and 13

NOTICE

Bit 10: Refer to *Target reached option code* (see *chapter 7.20.2 Parameter 50-42: Target Reached Option Code (0x2054)*) to influence the behavior of this bit. Also refer to *Illustration 2.17* in *chapter 2.4.1 Profile Position Mode* for the functional description.
Bit 13: Refer to *chapter 2.7.3 Following Error Detection* for more information about the following error detection.

7.3.1.2 Statusword in Profile Velocity Mode

The *Profile Velocity* mode uses some bits of the *Statusword* for operation mode-specific purposes. *Table 7.28* shows the structure of the *Statusword*. *Table 7.29* defines the values for bits 10 and 12.

15 14	13	12	11	10	9 0
See *Table 7.23*.	Reserved (0)	Speed	See *Table 7.23*.	Target reached	See *Table 7.23*.
MSB					LSB

Table 7.28 Statusword in Profile Velocity Mode

Bit	Value	Definition
10	0	Halt (bit 8 in *Controlword*) = 0: Target velocity not reached. Halt (bit 8 in *Controlword*) = 1: Axis decelerates.
	1	Halt (bit 8 in *Controlword*) = 0: Target reached. Halt (bit 8 in *Controlword*) = 1: Velocity of axis is 0.
12	0	Speed is not equal to 0.
	1	Speed is equal to 0.

Table 7.29 Definition of Bits 10 and 12

NOTICE

Bit 10: Refer to object 0x2054 (see *chapter 7.20.2 Parameter 50-42: Target Reached Option Code (0x2054)*) to influence the behavior of this bit. Also refer to *chapter 2.4.2 Profile Velocity Mode* for the functional description.
Bit 12: As soon as the velocity value exceeds the *Velocity threshold* (see *chapter 7.22.2.1 Parameter: Velocity Threshold (0x606F)*) longer than the *Velocity threshold time* (see *chapter 7.22.2.2 Parameter: Velocity Threshold Time (0x6070)*), then bit 12 is set to *0* in the *Statusword*.
Below this threshold, the bit is set to *1* and indicates that the axis is stationary.

7.3.1.3 Statusword in Profile Torque Mode

The *Profile Torque* mode uses some bits of the *Statusword* for operation mode-specific purposes. *Table 7.30* shows the structure of the *Statusword*. *Table 7.31* defines the values for bit 10.

15 14	13 12	11	10	9 0
See *Table 7.23.*	Reserved (0)	See *Table 7.23.*	Target reached	See *Table 7.23.*
MSB				LSB

Table 7.30 Statusword in Profile Torque Mode

Bit	Value	Definition
10	0	Halt (bit 8 in *Controlword*) = 0: Target not reached. Halt (bit 8 in *Controlword*) = 1: Axis decelerates.
	1	Halt (bit 8 in *Controlword*) = 0: Target reached. Halt (bit 8 in *Controlword*) = 1: Velocity of axis is 0.

Table 7.31 Definition of Bit 10

NOTICE

Bit 10: Refer to *Target reached option code* (see *chapter 7.20.2 Parameter 50-42: Target Reached Option Code (0x2054)*) to influence the behavior of this bit. Also refer to *chapter 2.4.2 Profile Velocity Mode* for the functional description.

7.3.1.4 Statusword in Homing Mode

The *Homing* mode uses some bits of the *Statusword* for operating mode-specific purpose. *Table 7.32* shows the structure of the *Statusword*. *Table 7.33* defines the values for bits 10, 12, and 13.

15 14	13	12	11	10	9 0
See *Table 7.23.*	Homing error	Homing attained	See *Table 7.23.*	Target reached	See *Table 7.23.*
MSB					LSB

Table 7.32 Statusword in Homing Mode

Bit 13	Bit 12	Bit 10	Definition
0	0	0	Homing procedure is in progress.
0	0	1	Homing procedure is interrupted or not started.
0	1	0	Reserved.
0	1	1	Homing procedure is completed successfully.
1	0	0	Homing error occurred, velocity is not 0.
1	0	1	Homing error occurred, velocity is 0.
1	1	X	Reserved.

Table 7.33 Definition of Bits 10, 12, and 13

NOTICE

The manufacturer-specific bit 8 of the *Statusword* is also influenced by the homing mode. The bit is set if the homing procedure was successful. This bit remains set until the servo drive is power-cycled, reset, a position overflow occurred, or a new homing procedure is started. The bit is reset when the encoder offset mode has been used because this functionality can also influence the position.
The manufacturer-specific bit 15 of the *Statusword* is also used by the homing mode. The bit is set if a homing procedure is started but the configuration does not allow it (for example, limit switch is not configured when trying to do a homing on limit switch).

7.3.1.5 Statusword in CAM Mode

The *CAM* mode uses some bits of the *Statusword* for operation mode-specific purposes. *Table 7.34* shows the structure of the *Statusword*. *Table 7.35* defines the values for bits 10, 12, and 13. For more status information, see *chapter 7.14.8 Parameter: CAM Profile Status (0x3805)*.

15 … 14	13	12	11	10	9 … 0
See *Table 7.23*.	Following error	CAM ack	See *Table 7.23*.	Target reached (InSync)	See *Table 7.23*.
MSB					LSB

Table 7.34 Statusword in CAM Mode

Bit	Value	Definition
10	0	Axis is blending. Current position is automatically calculated by the CAM.
	1	Setpoints of the CAM are processed.
12	0	Previous CAM profile already processed, waiting for new CAM profile.
	1	Previous CAM profile still in progress, new CAM profile will be accepted.
13	0	No following error.
	1	Following error.

Table 7.35 Definition of Bits 10, 12, and 13

NOTICE

Bit 13: Refer to *chapter 2.7.3 Following Error Detection* for more information about the following error detection.

7.3.1.6 Statusword in Gear Mode

The *Gear* mode uses some bits of the *Statusword* for operation mode-specific purposes. *Table 7.36* shows the structure of the *Statusword*. *Table 7.37* defines the values for bits 10, 12, and 13.

15 … 14	13	12	11	10	9 … 0
See *Table 7.23*	Followin g error	Sync started	See *Table 7.23*	Target reached (InSync)	See *Table 7.23*
MSB					LSB

Table 7.36 Statusword in Gear Mode

Bit	Value	Definition
10	0	Axis is not (yet) in sync with the given guide value.
	1	Axis is in sync with the given guide value.
12	0	Not in synchronization movement (either not started or already in sync).
	1	In synchronization movement.

Bit	Value	Definition
13	0	No following error.
	1	Following error.

Table 7.37 Definition of Bits 10, 12, and 13

NOTICE

Bit 13: Refer to *chapter 2.7.3 Following Error Detection* for more information about the following error detection.

7.3.1.7 Statusword in ISD Inertia Measurement Mode

The *ISD Inertia Measurement Mode* uses some bits of the *Statusword* for operation mode-specific purpose. *Table 7.38* shows the structure of the *Statusword*. *Table 7.39* defines the values for bits 10, 12, and 13.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>0</td></tr>
<tr><td colspan="2">See Table 7.23.</td><td>Measurement error</td><td>Standstill</td><td>See Table 7.23.</td><td>Target reached</td><td colspan="2">See Table 7.23.</td></tr>
<tr><td colspan="4">MSB</td><td colspan="4">LSB</td></tr>
</table>

Table 7.38 Statusword in ISD Inertia Measurement Mode

Bit	Value	Definition
10	0	Result of measurement is not available.
	1	Measurement has finished and result can be read.
12	0	Velocity is not 0 (measurement is ongoing or the servo drive is coasting).
	1	Velocity is 0.
13	0	No error in measurement.
	1	Error occurred during measurement.

Table 7.39 Definition of Bits 10, 12, and 13

NOTICE

Bit 13: The error reason can be read from object 0x2009 (see *chapter 7.16.1 Parameter 52-60: Measured Inertia (0x2009)*).

7.3.1.8 Statusword in Cyclic Synchronous Position Mode

Table 7.40 shows the structure of the *Statusword*.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>0</td></tr>
<tr><td colspan="2">See Table 7.23</td><td>Following error</td><td>Servo drive follows the commanded value</td><td>See Table 7.23</td><td>Status toggle</td><td colspan="2">See Table 7.23</td></tr>
<tr><td colspan="4">MSB</td><td colspan="4">LSB</td></tr>
</table>

Table 7.40 Statusword in Cyclic Synchronous Position Mode

NOTICE

Bit 10: Used as status toggle information to indicate if the device provides updated input data. The bit is toggled with every update of the input process data.
Bit 12: Is 0 if the servo drive does not follow the target value because of local control. Bit 12 is set if the servo drive is in state *Operation enabled* and follows the target and setpoint values of the control device. In all other cases, it is 0.
Bit 13: The following error bit. The following error value is only evaluated in state *Operation enabled*. After a reset, the setpoint is set to the actual value so that the following error is 0.

7.3.1.9 Statusword in Cyclic Synchronous Velocity Mode

Table 7.41 shows the structure of the *Statusword*.

15 14	13	12	11	10	9 0
See *Table 7.23*	Reserved	Servo drive follows the commanded value	See *Table 7.23*	Status toggle	See *Table 7.23*
MSB					LSB

Table 7.41 Statusword in Cyclic Synchronous Velocity Mode

NOTICE

Bit 10: Used as status toggle information to indicate if the device provides updated input data. The bit is toggled with every update of the input process data.
Bit 12: Is 0 if the servo drive does not follow the target value because of local reasons. Bit 12 is set if the servo drive is in state *Operation enabled* and follows the target and setpoint values of the control device. In all other cases, it is 0.

7

7.4 Factor Group Objects

7.4.1 Parameters 55-00 and 55-01: Position Encoder Resolution (0x608F)

This object indicates the configured encoder increments and number of motor revolutions. The position encoder resolution is calculated by the following formula:

$$\textit{position encoder resolution} = \frac{\textit{encoder increments}}{\textit{motor revolutions}}$$

All values are dimensionless.

Attribute	Value
Index	0x608F
Name	Position encoder resolution
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	55-00
Description	Encoder increments
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	2^{20}
Sub-index	0x02
LCP parameter number	55-01
Description	Motor revolutions
Access	Read only
PDO mapping	Optional

Attribute	Value
Value range	UNSIGNED32 (0 = 2^{32})
Default value	1

Table 7.42 0x608F: Position Encoder Resolution

7.4.2 Parameters 55-10 and 55-11: Gear Ratio (0x6091)

This object indicates the configured number of motor shaft revolutions and the number of drive shaft revolutions. The gear ratio is calculated by the following formula:

$$\text{gear ratio} = \frac{\text{motor shaft revolutions}}{\text{drive shaft revolutions}}$$

All values are dimensionless.

Attribute	Value
Index	0x6091
Name	Gear ratio
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	55-10
Description	Motor shaft revolutions
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	1
Sub-index	0x02
LCP parameter number	55-11
Description	Drive shaft revolutions
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	1

Table 7.43 0x6091: Gear Ratio

7.4.3 Parameters 55-20 and 55-21: Feed Constant (0x6092)

This object indicates the configured feed constant. This is defined as the measurement distance per 1 revolution of the drive shaft of the gearbox. The feed constant is calculated by the following formula:

$$\textit{feed constant} = \frac{\textit{feed}}{\textit{drive shaft revolutions}}$$

The feed is given in user-defined position units. The drive shaft revolution is dimensionless.

Using the default values, the results in position values are given in 1/100 degree.

Attribute	Value
Index	0x6092
Name	Feed constant
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	55-20
Description	Feed
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	36000
Sub-index	0x02
LCP parameter number	55-21
Description	Shaft revolutions
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	1

Table 7.44 0x6092: Feed Constant

7.4.4 Parameters 55-30 and 55-31: Velocity Factor (0x6096)

The velocity factor is used to match the velocity units to the user-defined velocity units.

Attribute	Value
Index	0x6096
Name	Velocity factor
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	55-30
Description	Numerator
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	1
Sub-index	0x02
LCP parameter number	55-31
Description	Divisor
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	1

Table 7.45 0x6096: Velocity Factor

7.4.5 Parameters 55-40 and 55-41: Acceleration Factor (0x6097)

The acceleration factor can be used to match the acceleration units to the user-defined acceleration units. The acceleration factor is also valid for deceleration values.

Attribute	Value
Index	0x6097
Name	Acceleration factor
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	55-40
Description	Numerator

Attribute	Value
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	1
Sub-index	0x02
LCP parameter number	55-41
Description	Divisor
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	1 to 2^{32} –1
Default value	1

Table 7.46 0x6097: Acceleration Factor

7.5 Commonly Used Objects

7.5.1 Parameter 52-00: Modes of Operation (0x6060)

This object indicates the requested operation mode and only shows the value of the requested operation mode. The actual mode of operation of the servo drive is reflected in object 0x6061 (see *chapter 7.5.2 Parameter 52-01: Modes of Operation Display (0x6061)*). A value of *0* does not change the currently active mode of operation.

Attribute	Value
Index	0x6060
LCP parameter number	52-00
Name	Modes of operation
Object code	Var
Data type	INTEGER8
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.48.*
Default value	0

Table 7.47 0x6060: Modes of Operation

This object provides the output value of the trajectory generator. The value is given in user-defined velocity units.

Value	Definition	Abbreviation	Control
–7 = 0xF9	Gear mode	gr	Position controlled (can be velocity controlled during synchroni-zation phase).
–6 = 0xFA	CAM mode	cam	Position controlled
–5 = 0xFB	ISD Inertia measurement mode	im	Torque controlled
0	No mode change	–	–
+1	Profile position mode	pp	Position controlled
+3	Profile velocity mode	pv	Velocity controlled
+4	Torque profile mode	tq	Torque controlled
+6	Homing mode	hm	Velocity controlled
+8	Cyclic synchronous position mode	csp	Position controlled
+9	Cyclic synchronous velocity mode	csv	Velocity controlled

Table 7.48 Supported Modes of Operation

7.5.2 Parameter 52-01: Modes of Operation Display (0x6061)

This object provides the actual operation mode.

Attribute	Value
Index	0x6061
LCP parameter number	52-01
Name	Modes of operation display
Object code	Var
Data type	INTEGER8
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.48*.
Default value	1

Table 7.49 0x6061: Modes of Operation Display

7.5.3 Parameter: Supported Drive Modes (0x6502)

This object provides information on the supported drive modes.

Attribute	Value
Index	0x6502
LCP parameter number	–
Name	Supported drive modes
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.51*.
Default value	Dependent on firmware version.

Table 7.50 0x6502: Supported Drive Modes

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r	r	r	r	im	cam	gr	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r	r	r	r	r	cstca	cst	csv	csp	ip	hm	r	tq	pv	vl	pp

Table 7.51 Value Definition for 0x6502

Value	Definition
0	Mode is not supported
1	Mode is supported
r	Reserved bits

Table 7.52 Value Definition for 0x6502

7.5.4 Parameter 50-16: Maximum Profile Velocity (0x607F)

This object indicates the configured maximum allowed velocity in either direction during a profiled motion. The value is given in used-defined velocity units.

Attribute	Value
Index	0x607F
LCP parameter number	50-16
Name	Maximum profile velocity
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	Maximum drive limit

Table 7.53 0x607F: Maximum Profile Velocity

7.5.5 Parameter 52-37: Maximum Motor Speed (0x6080)

This object is used to limit the maximum speed of the servo drive in either direction and in every available mode of operation. The value is given in user-defined velocity units.

Attribute	Value
Index	0x6080
LCP parameter number	52-37
Name	Maximum motor speed
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	Maximum value (dependent on servo drive size).

Table 7.54 0x6080: Maximum Motor Speed

7

7.5.6 Parameter 52-12: Profile Velocity (0x6081)

This object indicates the configured velocity normally attained at the end of the acceleration ramp during a profiled motion and is valid for both directions of motion. The value is given in user-defined velocity units.

Attribute	Value
Index	0x6081
LCP parameter number	52-12
Name	Profile velocity
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	100

Table 7.55 0x6081: Profile Velocity

7.5.7 Parameter 50-11: Profile Acceleration (0x6083)

This object indicates the configured acceleration. The value is given in user-defined acceleration units.

Attribute	Value
Index	0x6083
LCP parameter number	50-11, 52-13, and 52-21
Name	Profile acceleration
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	1000

Table 7.56 0x6083: Profile Acceleration

7.5.8 Parameter 50-12: Profile Deceleration (0x6084)

This object indicates the configured deceleration. The value is given in user-defined deceleration units.

Attribute	Value
Index	0x6084
LCP parameter number	50-12, 52-14, and 52-22
Name	Profile deceleration
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	1000

Table 7.57 0x6084: Profile Deceleration

7.5.9 Parameter 50-13: Quick Stop Deceleration (0x6085)

This object indicates the configured deceleration used to stop the servo drive when the quick stop function is activated and the quick stop option code is set to *2* or *6*.

The quick stop deceleration is also used if the fault reaction option code object is *2* and the halt option code is *2*. The value is given in user-defined acceleration units.

Attribute	Value
Index	0x6085
LCP parameter number	50-13
Name	Quick stop deceleration
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0x7FFF FFFF

Table 7.58 0x6085: Quick Stop Deceleration

7.5.10 Parameter 50-14: Maximum Acceleration (0x60C5)

This object indicates the configured maximal acceleration. It is used to limit the acceleration to an acceptable value in order to prevent the motor and the moved mechanics from being damaged. The value is given in user-defined acceleration units.

Attribute	Value
Index	0x60C5
LCP parameter number	50-14
Name	Maximum acceleration
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0x7FFF FFFF

Table 7.59 0x60C5: Maximum Acceleration

7.5.11 Parameter 50-15: Maximum Deceleration (0x60C6)

This object indicates the configured maximal deceleration. It is used to limit the deceleration to an acceptable value in order to prevent the motor and the moved mechanics from being damaged. The value is given in user-defined acceleration units.

Attribute	Value
Index	0x60C6
LCP parameter number	50-15
Name	Maximum deceleration
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0x7FFF FFFF

Table 7.60 0x60C6: Maximum Deceleration

7.5.12 Parameter: Maximum Torque (0x6072)

This object indicates the configured maximum allowed torque in the motor. The value is given per thousand of rated torque.

This object is set to protect the application from damaging the machine. It is set at the start-up of the PLC project.

The servo drive always uses the minimum of objects 0x6072 and 0x2053.

Attribute	Value
Index	0x6072
LCP parameter number	–
Name	Maximum torque
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	Maximum possible torque

Table 7.61 0x6072: Maximum Torque

7.5.13 Parameters 52-15, 52-23, and 52-36: Application Torque Limit (0x2053)

This object indicates the configured maximum allowed torque in the motor. The value is given per thousand of rated torque.

This object is used to limit the current during a motion command and it can be mapped to the PDO.

The servo drive always uses the minimum of objects 0x6072 and 0x2053.

Attribute	Value
Index	0x2053
LCP parameter number	52-15, 52-23, and 52-36
Name	Application torque limit
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0xFFFF

Table 7.62 0x2053: Application Torque Limit

7.6 Control Parameters

7.6.1 Parameter 51-07 to 51-09: Used Task Cycle Times (0x201D)

This object provides information about the cycle times for the control loops. The values are given in microseconds.

Attribute	Value
Index	0x201D
Name	Used task cycle times
Object code	Array
Data type	FLOAT
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x03
Sub-index	0x01
LCP parameter number	51-07
Description	Real-time task
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	–
Default value	–
Sub-index	0x02
LCP parameter number	51-08
Description	Fast task
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	–
Default value	–
Sub-index	0x03
LCP parameter number	51-09
Description	Slow task
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	–
Default value	–

Table 7.63 0x201D: Used Task Cycle Times

7.6.2 Parameter 51-01: Control Parameter Blending Time (0x201B)

This object is used to configure the blending time between the 2 sets of control parameters. The value is given in milliseconds.

Attribute	Value
Index	0x201B
LCP parameter number	51-01
Name	Control parameter blending time
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	0–100
Default value	0

Table 7.64 0x201B: Control Parameter Blending Time

7.6.3 Parameter 51-00: Control Parameter Usage (0x201C)

This object contains the number of the control parameter set used.

Attribute	Value
Index	0x201C
LCP parameter number	51-00
Name	Control parameter usage
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	1 or 2
Default value	1

Table 7.65 0x201C: Control Parameter Usage

7.6.4 Position Controller

7.6.4.1 Parameters 51-16 and 51-17: Position Controller Parameters (0x2013)

This object contains the parameters of the position controller. The value for derivative component is time constant.

Attribute	Value
Index	0x2013
Name	Position controller parameters
Object code	Array
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
Data type	UNSIGNED8
PDO mapping	No

Attribute	Value
Default value	0x02
Sub-index	0x01
LCP parameter number	51-16
Description	Position controller proportional
Access	Read/write
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	15
Sub-index	0x02
LCP parameter number	51-17
Description	Position controller differential
Access	Read/write
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0

Table 7.66 0x2013: Position Controller Parameters

7.6.4.2 Parameters 51-26 and 51-27: Position Controller Parameters 2 (0x2015)

This object contains a 2nd set of position controller parameters. The value for derivative component is time constant.

Attribute	Value
Index	0x2015
Name	Position controller parameters 2
Object code	Array
Data type	UNSIGNED8
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	51-26
Description	Position controller proportional
Access	Read/write
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	15
Sub-index	0x02
LCP parameter number	51-27
Description	Position controller differential
Access	Read/write
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT

Attribute	Value
Default value	0

Table 7.67 0x2015: Position Controller Parameters 2

7.6.5 Speed Controller

7.6.5.1 Parameters 51-10 to 51-15: Speed Controller Parameters (0x2012)

This object contains the speed controller parameters. The values for integral and derivative components are time constants.

Attribute	Value
Index	0x2012
Name	Speed controller parameters
Object code	Array
Data type	UNSIGNED8
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x06
Sub-index	0x01
LCP parameter number	51-10
Description	Speed controller proportional
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0.1
Sub-index	0x02
LCP parameter number	51-11
Description	Speed controller integral
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0.01
Sub-index	0x03
LCP parameter number	51-12
Description	Speed controller inertia
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	Dependent on motor type and brake configuration.
Sub-index	0x04
LCP parameter number	51-13
Description	Speed controller differential
Access	Read only
Data type	FLOAT
PDO mapping	Optional

Attribute	Value
Value range	FLOAT
Default value	0
Sub-index	0x05
LCP parameter number	51-14
Description	Center frequency
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0
Sub-index	0x06
LCP parameter number	51-15
Description	Bandwidth
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	50

Table 7.68 0x2012: Speed Controller Parameters

7.6.5.2 Parameters 51-20 to 51-25: Speed Controller Parameters 2 (0x2014)

This object contains a 2nd set of speed controller parameters. The values for integral and derivative components are time constants.

Attribute	Value
Index	0x2014
Name	Speed controller parameters 2
Object code	Array
Data type	UNSIGNED8
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x06
Sub-index	0x01
LCP parameter number	51-20
Description	Speed controller proportional
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0.1
Sub-index	0x02
LCP parameter number	51-21
Description	Speed controller integral
Access	Read only
Data type	FLOAT
PDO mapping	Optional

Attribute	Value
Value range	FLOAT
Default value	0.01
Sub-index	0x03
LCP parameter number	51-22
Description	Speed controller inertia
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	Dependent on motor type and brake configuration.
Sub-index	0x04
LCP parameter number	51-23
Description	Speed controller differential
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0
Sub-index	0x05
LCP parameter number	51-24
Description	Center frequency
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0
Sub-index	0x06
LCP parameter number	51-25
Description	Bandwidth
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0

Table 7.69 0x2014: Speed Controller Parameters 2

7.7 Positions and Offset Objects

7.7.1 Parameter: Position Demand Value (0x6062)

This object provides the demanded position value. The value is given in user-defined position units.

Attribute	Value
Index	0x6062
LCP parameter number	–
Name	Position demand value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.70 0x6062: Position Demand Value

7.7.2 Parameter: Position Demand Internal Value (0x60FC)

This object provides the output of the trajectory generator in position controlled modes of operation. The value is given in increments of the position encoder.

Attribute	Value
Index	0x60FC
LCP parameter number	–
Name	Position demand internal value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.71 0x60FC: Position Demand Internal Value

7.7.3 Parameter: Drive Position (0x2022)

This object provides the actual value of the position measurement before applying the position range limits. The drive position is the reference figure for all the trajectory calculations. The value is given in user-defined position units.

Attribute	Value
Index	0x2022
LCP parameter number	16-20
Name	Drive position
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.72 0x2022: Drive Position

7.7.4 Parameter: Position Actual Internal Value (0x6063)

This object provides the actual value of the position measurement device, which is 1 of the 2 input values of the closed-loop control. The value is given in increments.

Attribute	Value
Index	0x6063
LCP parameter number	–
Name	Position actual internal value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.73 0x6063: Position Actual Internal Value

7.7.5 Parameter 50-03: Position Actual Value (0x6064)

This object provides the actual value of the position measurement device. The value is given in user-defined position units.

Attribute	Value
Index	0x6064
LCP parameter number	50-03
Name	Position actual value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.74 0x6064: Position Actual Value

7.7.6 Parameters 50-30 and 50-31: Position Range Limit (0x607B)

This object indicates the configured maximum and minimum position range limits. It limits the numerical range of the input value. On reaching or exceeding these limits, the input value automatically wraps to the other end of the range.

To disable the position range limits, set the minimum position range limit (sub-index 0xd01) and the maximum position range limit (sub-index 0x02) to *0* (default value). The values are given in user-defined position units.

Attribute	Value
Index	0x607B
Name	Position range limit
Object code	Array
Data type	UNSIGNED8
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	50-30
Description	Minimum position range limit
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32
Default value	0
Sub-index	0x02
LCP parameter number	51-31
Description	Maximum position range limit
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32

Attribute	Value
Default value	0

Table 7.75 0x607B: Position Range Limit

7.7.7 Parameters 50-32 and 50-33: Software Position Limit (0x607D)

This object is used to limit the maximum and minimum valid positions of the servo drive using software position limits. It is also used for monitoring the position limits in all available modes of operation.

The limit positions are given in user-defined position units. Supervision of software position limits requires a defined home position. The *Is homed* bit in the *Statusword* must be set by successfully completing the homing procedure. Setting *Min position limit* to MININT (default setting) disables the minimum position limit. Setting the *Max position limit* to MAXINT (default setting) disables the maximum position limit.

When the servo drive is in position-controlled mode, it behaves differently to all other modes of operation. In position-controlled mode of operation, the servo drive does not pass over the *Software position limit* (see *chapter 2.3.4.2 Software Position Limit*). The target position is limited to the software position limit. In all other modes of operation, the servo drive immediately ramps down using the quick-stop deceleration when passing the software position limit. This means that the servo drive always stops after the software position limit.

Attribute	Value
Index	0x607D
Name	Software position limit
Object code	Array
Data type	INTEGER32
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	50-32
Description	Minimum position limit
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32
Default value	0x7FFF FFFF
Sub-index	0x02
LCP parameter number	51-33
Description	Maximum position limit
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32
Default value	0x8000 0000

Table 7.76 0x607D: Software Position Limit

7.7.8 Parameters 51-02, 52-04, and 52-49: Application Settings (0x2016)

There are several general settings regarding the behavior of the servo drive.

Sub-index 01: Observer speed enable
Enables the speed observer. The purpose of the observer is to improve the quality of the calculated speed signal from position sensor. When enabled, the control bandwidth is higher.

Sub-index 02: Drive mirror mode
This object provides a way to set/clear the drive mirror mode. Setting mirror mode reverses the turning direction of the axis, useful when, for example, it is used with an inverting gear box.

Sub-index 03: Save position offset
Write *1* to save the position offset to non-volatile memory. After it is saved, the servo drive writes a *0* back to the object so that it can be rewritten with *1*. If a new homing process starts, the position offset is overwritten, but not stored automatically. If the servo drive is reset, the last saved position offset is reused. The *Is Homed* bit in the *Statusword* is not set during start-up. It is only set if a new homing procedure completed successfully.

For single-turn encoders, only the fractional part is preserved.

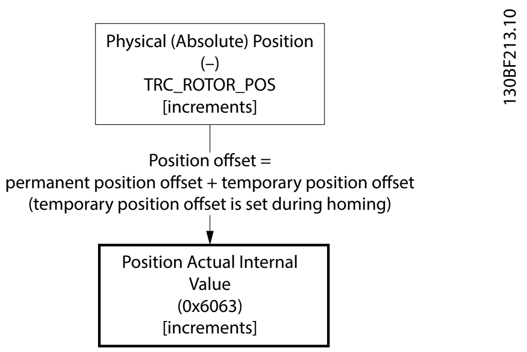


Illustration 7.1 Explanation of Position Offset

Attribute	Value
Index	0x2016
Name	Application settings
Object code	Array
Data type	UNSIGNED16
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x03
Sub-index	0x01
LCP parameter number	51-02
Description	Observer speed enable
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	0, 1
Default value	0
Sub-index	0x02
LCP parameter number	52-04
Description	Drive mirror mode

Attribute	Value
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	0, 1
Default value	–
Sub-index	0x03
LCP parameter number	52-49
Description	Save position offset
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	0, 1
Default value	–

Table 7.77 0x2016: Application Settings

7.8 Guide Value Objects

7.8.1 Parameter: Position Guide Value (0x2060)

The servo drive supports modes of operation that require synchronization with a position guide value. This position guide value can come from an external encoder, an existing master axis, or virtual master axis. The servo drive is provided with this guide value by the master controller (usually a PLC).

The position guide value is an object that represents a position and is given in the logical range of 0–1, and is encoded in 32 bits (logical revolution). For example, a value of 0x8000 0000 represents half a round. The value range is unlimited (explicitly not limited to 360°) and the direction of the value can run in both directions. However, the processing of the guide value can be parameterized using the *Guide value option code* object (see *chapter 7.8.3 Parameter: Guide Value Option Code (0x2061)*).

Wrap-around of the guide value is allowed and does not need special handling. The servo drive can calculate the velocity from the position differences internally.

Attribute	Value
Index	0x2060
LCP parameter number	–
Name	Position guide value
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0x03

Table 7.78 0x2060: Position Guide Value

7.8.2 Parameter: Velocity Guide Value (0x2064)

The servo drive supports modes of operation that require synchronization with a velocity guide value. This velocity guide value can come from an external encoder, an existing master axis, or virtual master axis. The servo drive is provided with this guide value by the master controller (usually a PLC). The velocity guide value is a floating point object that represents a velocity and is given in rounds per second.

The direction of the value can run in both directions.

If this value has to be used, or if the servo drive has to calculate the value itself (using the *Position guide value*), it must be parameterized using the *Guide value option code* (see *chapter 7.8.3 Parameter: Guide Value Option Code (0x2061)*).

Attribute	Value
Index	0x2064
LCP parameter number	–
Name	Velocity guide value
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	FLOAT
Default value	–

Table 7.79 0x2064: Velocity Guide Value

7.8.3 Parameter: Guide Value Option Code (0x2061)

In some applications, a backwards moving position guide value should not be processed by the servo drive. Therefore, this position guide value direction option is used to specify whether the servo drive should process the guide value.

Attribute	Value
Index	0x2061
LCP parameter number	–
Name	Guide value option code
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.81*.
Default value	2

Table 7.80 0x2061: Guide Value Option Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
–													ce	vel	fo
MSB															LSB

Table 7.81 Guide Value Option Code

fo	Forward only: 0: Forward and backward movement is processed. 1: Only forward movement is processed. If the movement of the reference input (master axis) is going backwards, the last forward directed position is kept. The further forward movement is started when the last seen forward position of the reference input is reached again. If the guide value goes back more than half of the range, the servo drive goes into fault state.
vel	Use velocity: 0: Servo drive calculates the velocity from position differences internally. 1: Servo drive uses *Velocity guide value object* (0x2064).
ce	Command error: 0: Plausibility check disabled. 1: Plausibility check enable.

7.8.4 Parameter: Guide Value Scaling Factor (0x3808)

The servo drive also supports scaling of the guide value. The scaling factor consists of a numerator and a denominator.

The value 0x0000 0000 must be reserved and not used for the numerator and the denominator objects. To be able to change both values simultaneously, a writing order is defined. The activation of the new factor is done after the writing of the numerator. This means to change both values at the same time, the denominator must be written first and the numerator afterwards. It is also possible to only change the numerator by writing a new value (the denominator stays the same). The numerator and denominator accept negative values. The division of 2 negative values results in a positive value.

The *Position guide value* is multiplied by the quotient of numerator and denominator.

Guide value scaled = (Guide value x Scaling value) + Guide value offset

Internally used *0x2060* *0x3808.01/02* *0x3806*

Attribute	Value
Index	0x3808
Name	Guide value scaling factor
Object code	Array
Data type	INTEGER32
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	UNSIGNED8
Default value	0x02
Sub-index	0x01
LCP parameter number	–
Description	Guide value scaling numerator
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	1
Sub-index	0x02
LCP parameter number	–
Description	Guide value scaling denominator
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32

Attribute	Value
Default value	1

Table 7.82 0x3808: Guide Value Scaling Factor

7.8.5 Parameter: Guide Value Offset (0x3806)

The *Guide value offset* modifies the guide value used by the servo drive. It internally adds an offset to the *Position guide value*.

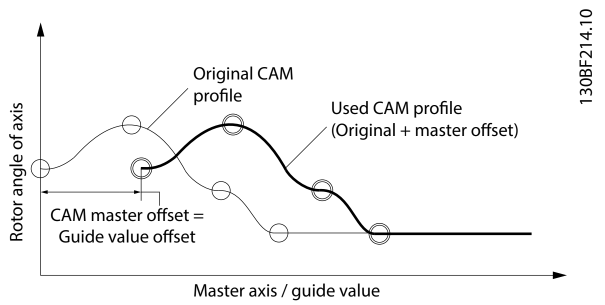


Illustration 7.2 Example of Position Guide Value Offset in CAM Mode

When using the guide value as master position as a relative value in CAM mode, only the change of the value of object *Guide value offset* during operation has an effect. At the beginning of a CAM, the offset value is compensated by the internal offset that is calculated for relative operation. Objects for *Guide value offset* and for *CAM slave offset* can be used in parallel, that means that it is possible to have a master and a slave offset at the same time.

The value of the *Guide value offset* must be given in position guide value unit. That means that 1 master cycle is represented by values from 0 to 0xFFFF FFFF. The offset is a signed value, so it is possible to advance or delay the position guide value for half a round.

Example:
To advance the *Position guide value* for 45°, means having a *Guide value offset* of 0x2000 0000. To delay the *Position guide value* for 90°, use 0xC0000 0000.
Changing a value has an immediate effect if the axis is currently processing the guide value.

Attribute	Value
Index	0x3806
LCP parameter number	–
Name	Guide value offset
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	0

Table 7.83 0x3806: Guide Value Offset

7.9 Guide Value Reference Objects

7.9.1 Parameter: Position Guide Value Reference (0x2062)

This object is used to provide the position of the servo drive in a way that it can be used as the *Position guide value* for synchronized motion of other servo drives. The source of this value is given by the *Guide value reference option code*.

The value must be left-aligned. To achieve this, shift the original value using the *Guide value reference option code* (see *chapter 7.9.3 Parameter: Guide Value Reference Option Code (0x2063)*).

Attribute	Value
Index	0x2062
LCP parameter number	–
Name	Position guide value reference
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.84 0x2062: Position Guide Value Reference

7.9.2 Parameter: Velocity Guide Value Reference (0x2065)

This object is used to provide the velocity of the servo drive in a way that it can be used as the*Velocity guide value* for synchronized motion of other servo drives. The source of this value is given by the *Guide value reference option code* (see *chapter 7.9.3 Parameter: Guide Value Reference Option Code (0x2063)*). The value is given in floating point.

Attribute	Value
Index	0x2065
LCP parameter number	–
Name	Velocity guide value reference
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	FLOAT
Default value	0

Table 7.85 0x2065: Velocity Guide Value Reference

7.9.3 Parameter: Guide Value Reference Option Code (0x2063)

Use this object to select the source that is used as provider for the guide value reference objects.

Attribute	Value
Index	0x2063
LCP parameter number	–
Name	Guide value reference option code
Object code	Var

Attribute	Value
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.87*.
Default value	0

Table 7.86 0x2063: Guide Value Reference Option Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
source		–									shift				
MSB															LSB

Table 7.87 Guide Value Reference Option Code

Bit 15	Bit 14	Definition
0	0	Actual value.
0	1	Set value.
1	0	External encoder.
1	1	Guide value reference simulation.

Table 7.88 Value Definition for Bits 14 and 15

Bits 0–4, shift:
Number of bits the value must be shifted to the left.

7.9.4 Parameter: Position Guide Value Reference Set (0x2068)

By writing this object, the position guide value reference object is set to the specified value. The internally calculated offset is stored permanently. It is recommended that the guide value is constant while this object is written. The value is given in guide value units.

Attribute	Value
Index	0x2068
LCP parameter number	–
Name	Position guide value reference set
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.89 0x2068: Position Guide Value Reference Set

7.9.5 Parameter: Guide Value Plausibility Distance (0x2067)

This object is used to provide the maximum allowed position increment per cycle for the guide value reference plausibility check. The value of object *0x2062 Position guide value reference* is used for monitoring.

Attribute	Value
Index	0x2067
LCP parameter number	–
Name	Guide value plausibility distance
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0

Table 7.90 0x2067: Guide Value Plausibility Distance

7

7.9.6 Guide Value Reference Simulation

7.9.6.1 Parameter: Guide Value Reference Simulation Control (0x2070)

This object is used to start and stop the guide value reference simulation.

Attribute	Value
Index	0x2070
LCP parameter number	–
Name	Guide value reference simulation control
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.92*.
Default value	0

Table 7.91 0x2070: Guide Value Reference Simulation Control

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
–															en
MSB															LSB

Table 7.92 Guide Value Reference Simulation

Bit 0, en:
0: Guide reference simulation off.
1: Guide reference simulation on.

7.9.6.2 Parameter: Guide Value Reference Speed Limit (0x2071)

This object defines the maximum speed for the simulation. The value is given in rps.

Attribute	Value
Index	0x2071
LCP parameter number	–
Name	Guide value reference speed limit
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	FLOAT
Default value	0 rps

Table 7.93 0x2071: Guide Value Reference Speed Limit

7.9.6.3 Parameter: Guide Value Reference Target Velocity (0x2072)

This object defines the target velocity. Ramp-up and ramp-down takes place with the values specified in objects 0x2073 (acceleration) and 0x2074 (deceleration). The value is given in rps.

Attribute	Value
Index	0x2072
LCP parameter number	–
Name	Guide value reference target velocity
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	FLOAT
Default value	0 rps

Table 7.94 0x2072: Guide Value Reference Target Velocity

7.9.6.4 Parameter: Guide Value Reference Acceleration (0x2073)

The guide value reference simulation speed is increased with this acceleration. *Illustration 2.132* in *chapter 2.5.3.2 Guide Value Reference Simulation* shows that acceleration means increasing the speed in an absolute way. The value is given in rps/s.

Attribute	Value
Index	0x2073
LCP parameter number	–
Name	Guide value reference acceleration
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	FLOAT

Attribute	Value
Default value	0 rps/s

Table 7.95 0x2073: Guide Value Reference Acceleration

7.9.6.5 Parameter: Guide Value Reference Deceleration (0x2074)

The guide value reference simulation speed is decreased with this deceleration. *Illustration 2.132* in *chapter 2.5.3.2 Guide Value Reference Simulation* shows that deceleration means decreasing the speed in an absolute way. The value is given in rps/s.

Attribute	Value
Index	0x2074
LCP parameter number	–
Name	Guide value reference deceleration
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	FLOAT
Default value	0 rps/s

Table 7.96 0x2074: Guide Value Reference Deceleration

7.10 Profile Position Mode Objects

7.10.1 Parameter 52-10: Target Position (0x607A)

This object indicates the commanded position that the servo drive should move to in profile position mode. It uses the current settings of the motion control parameters, for example velocity, acceleration, deceleration, and motion profile type. The value of this object is interpreted as absolute or relative depending on the *abs/rel* flag in the *Controlword*. The value is given in user-defined position units.

Attribute	Value
Index	0x607A
LCP parameter number	52-10
Name	Target position
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	0

Table 7.97 0x607A: Target Position

7.10.2 Parameter 52-16: End Velocity (0x6082)

This object indicates the configured velocity of the servo drive on reaching the target position. Normally, the servo drive stops at the target position, meaning that the end velocity is *0*. The value is given user-defined velocity units.

Attribute	**Value**
Index	0x6082
LCP parameter number	52-16
Name	End velocity
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0

Table 7.98 0x6082: End Velocity

7.10.3 Parameter: Positioning Option Code (0x60F2)

This object indicates the configured positioning behavior, as described by the *Profile position mode*. Bits 0, 1, 4, 5, and 15 are supported.

Attribute	**Value**
Index	0x60F2
LCP parameter number	–
Name	Positioning option code
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.101* and *Table 7.102*.
Default value	0

Table 7.99 0x60F2: Positioning Option Code

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>ms</td><td colspan="3">reserved (0)</td><td colspan="4">not supported</td><td colspan="2">rado</td><td colspan="2">reserved (0)</td><td colspan="2">not supported</td><td colspan="2">relative option</td></tr>
<tr><td colspan="8">MSB</td><td colspan="8">LSB</td></tr>
</table>

Table 7.100 0x60F2: Positioning Option Code

ms	Manufacturer-specific
rado	Rotary axis direction option

The relative option bits control the behavior of positioning tasks in detail when the abs/rel bit (bit 6) in the *Controlword* is set to *1* in profile position mode (= relative positioning).
Table 7.101 shows the bit value definitions.

Bit 1	Bit 0	Definition
0	0	Positioning moves are performed relative to the preceding (internal absolute) target position (relative to 0 if there is no preceding target position).
0	1	Positioning moves are performed relative to the position internal demand value (object 0x60FC) output of the trajectory generator.
1	0	Positioning moves are performed relative to the position actual value (object 0x6064).
1	1	Reserved.

Table 7.101 Definition of Bits 0 and 1

The *Rotary axis direction option* bits (bits 6, 7, and 15) control the behavior of positioning tasks when the *abs/rel* bit (bit 6) in the *Controlword* is set to *0* in profile position mode (= absolute positioning).

Controlled rotary axes are needed for rotary tables and conveyor belts. They are also needed for cases where the host controller position limits are wider than the position range limits available in the servo drive. In these cases, the software position limits (see *chapter 7.7.7 Parameters 50-32 and 50-33: Software Position Limit (0x607D)*) are out of the position range limits (see *chapter 7.7.6 Parameters 50-30 and 50-31: Position Range Limit (0x607B)*). To disable the position range limits, set the minimum position range limit (0x607B, sub-index 1) and maximum position range limit (0x607B, sub-index 2) to *0*. Depending on the application, different rotary axis movements are possible. These are coded by bit 6, 7, and 15 (see *Table 7.102*).

Bit 15	Bit 7	Bit 6	Definition
0	0	0	Normal positioning similar to linear axis; If reaching or exceeding the position range limits (0x607B), the input value automatically wraps to the other end of the range. Positioning can be relative or absolute. Movement greater than a modulo value is only possible with this bit combination.
0	0	1	Positioning only in negative direction; If the target position is higher than the actual position, then the axis moves over the minimum position limit (0x607B, sub-index 01) to the target position.
0	1	0	Positioning only in positive direction; If the target position is lower than the actual position, then the axis moves over the maximum position limit (0x607B, sub-index 02) to the target position.
0	1	1	Positioning with the shortest way to the target position. If the distance in both directions is the same, the axis moves in positive direction.
1	0	0	Positioning in last direction; Similar to positioning, however the direction (negative or positive) depends on the last known positioning direction. If no previous direction is available, then positive direction is used.
1	0	1	Reserved.
1	1	x	Reserved.

Table 7.102 Definition of Bits 6, 7, and 15

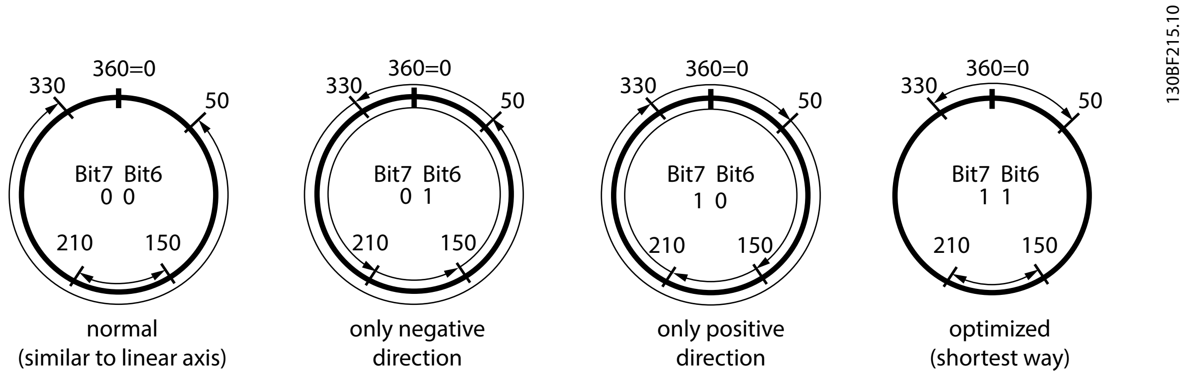


Illustration 7.3 Possible Rotary Axis Movements

If rado bits are set to *0* (normal), the servo drive is able to turn >1 revolution. This applies if object 0x607B is set to 0° and 360°, and the target positon is above 1 revolution.

For the options *Only negative direction*, *Only positive direction*, and *Last direction*, a movement bigger than the modulo value is not possible. If the value of the target position is bigger than the modulo value, then the modulo operation is used to determine the position within the range. This position is then used as the target position (for example, if the target position is set to 410°, then the actual target position is 50°). The direction to reach this position is then determined by the rado bits.

If rado bits are set to *1* (optimized), the shortest way to the new target position must be found. In this case, the servo drive must never turn more than 180°, even if the distance between target position is >1 revolution. However, it is allowed to cross the maximum/minimum limits of object 0x607B to reach the target position.

7.10.4 Parameter: Position Window (0x6067)

This object indicates the configured symmetrical range of accepted positions relative to the target position. If the actual value of the position encoder is within the position window, this target position is regarded as having been reached.

The value is given in user-defined position units.

If the value of the position window is *0xFFFF FFFF*, the position window control is switched off (*Target reached* bit is not set).

Attribute	Value
Index	0x6067
LCP parameter number	–
Name	Position window
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	5°

Table 7.103 0x6067: Position Window

7.10.5 Parameter: Position Window Time (0x6068)

This object indicates the configured time, during which the actual position within the position window is measured. The value is given in ms.

Attribute	Value
Index	0x6068
LCP parameter number	–
Name	Position window time
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	20 ms

Table 7.104 0x6068: Position Window Time

7.11 Profile Velocity Mode Objects

7.11.1 Parameter 52-20: Target Velocity (0x60FF)

This object indicates the configured target velocity and is used as input for the trajectory generator. The value is given in user-defined velocity units.

Attribute	Value
Index	0x60FF
LCP parameter number	52-20
Name	Target velocity

7

Attribute	Value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	0

Table 7.105 0x60FF: Target Velocity

7.11.2 Parameter: Velocity Demand Value (0x606B)

This object provides the output value of the trajectory generator. The value is given in user-defined velocity units.

Attribute	Value
Index	0x606B
LCP parameter number	–
Name	Velocity demand value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.106 0x606B: Velocity Demand Value

7.11.3 Parameter 50-04: Velocity Actual Value (0x606C)

This object provides the actual velocity value derived from the position sensor. The value is given in user-defined velocity units.

Attribute	Value
Index	0x606C
LCP parameter number	16-17, 50-04
Name	Velocity actual value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.107 0x606C: Velocity Actual Value

7.11.4 Parameter: Velocity Window (0x606D)

This object indicates the configured velocity window. The value is given in user-defined velocity units.

Attribute	Value
Index	0x606D
LCP parameter number	–
Name	Velocity window
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	6000 (10 RPM with default values for factor group)

Table 7.108 0x606D: Velocity Window

7.11.5 Parameter: Velocity Window Time (0x606E)

This object indicates the configured velocity window time. The value is given in milliseconds.

Attribute	Value
Index	0x606E
LCP parameter number	–
Name	Velocity window time
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	20 ms

Table 7.109 0x606E: Velocity Window Time

7.12 Profile Torque Mode Objects

7.12.1 Parameter 52-30: Target Torque (0x6071)

This object indicates the configured target velocity and is used as input for the trajectory generator. The value is given per thousand of rated torque.

Attribute	Value
Index	0x6071
LCP parameter number	52-30
Name	Target torque
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional

Attribute	Value
Value range	INTEGER16
Default value	0

Table 7.110 0x6071: Target Torque

7.12.2 Parameter: Torque Demand (0x6074)

This object provides the output value of the trajectory generator. The value is given per thousand of rated torque.

Attribute	Value
Index	0x6074
LCP parameter number	–
Name	Torque demand
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	INTEGER16
Default value	0

Table 7.111 0x6074: Torque Demand

7.12.3 Parameter 50-20: Motor Rated Current (0x6075)

This object indicates the configured motor rated current. All relative current data refers to this value. The value is given in mA.

Attribute	Value
Index	0x6075
LCP parameter number	50-20
Name	Motor rated current
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	Dependent on motor type.

Table 7.112 0x6075: Motor Rated Current

7.12.4 Parameter 50-21: Motor Rated Torque (0x6076)

This object indicates the configured motor rated torque. All relative torque data refers to this value. The value is given in mNm (milli Newton meter).

Attribute	Value
Index	0x6076
LCP parameter number	50-21
Name	Motor rated torque
Object code	Var

Attribute	Value
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	Dependent on motor type.

Table 7.113 0x6076: Motor Rated Torque

7.12.5 Parameter 52-31: Torque Actual Value (0x6077)

This object provides the actual value of the torque. It corresponds to the instant torque in the motor. The value is given per thousand of rated torque. The value in the LCP is given in Nm.

Attribute	Value
Index	0x6077
LCP parameter number	16-16, 52-31
Name	Torque actual value
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER16
Default value	–

Table 7.114 0x6077: Torque Actual Value

7.12.6 Parameter 16-14: Current Actual Value (0x6078)

This object provides the actual value of the current. It corresponds to the current in the motor. The value in the object is given per thousand of rated current. The value in the LCP is given in A.

Attribute	Value
Index	0x6078
LCP parameter number	16-14
Name	Current actual value
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER16
Default value	–

Table 7.115 0x6078: Current Actual Value

7.12.7 Parameter 52-32: Torque Slope (0x6087)

This object indicates the configured rate of change of torque. The value is given in units of per thousand of rated torque per second.

Attribute	Value
Index	0x6087
LCP parameter number	52-32
Name	Torque slope
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.116 0x6087: Torque Slope

7.12.8 Parameter: Torque Window (0x2050)

This object indicates the configured torque window for the *Target reached* bit in the *Statusword*. The value is given per thousand of rated torque.

Attribute	Value
Index	0x2050
LCP parameter number	–
Name	Torque window
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	20

Table 7.117 0x2050: Torque Window

7.12.9 Parameter: Torque Window Time (0x2051)

This object indicates the configured torque window time for target reached. The value is given in milliseconds.

Attribute	Value
Index	0x2051
LCP parameter number	–
Name	Torque window time
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16

Attribute	Value
Default value	50 ms

Table 7.118 0x2051: Torque Window Time

7.13 Homing Mode Objects

7.13.1 Parameter 52-40: Home Offset (0x607C)

This object indicates the configured difference between the zero position for the application and the machine home position (found during homing). During homing, the machine home position is found and once the homing is completed, the zero position is offset from the home position by adding the home offset to the home position.

The zero position is calculated as:
zero position = home position + home offset (see *Illustration 7.4*).

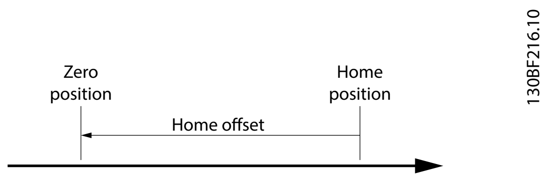


Illustration 7.4 Home Offset Definition

All subsequent absolute moves are taken relative to this new zero position. The value of this object is given in user-defined position units. Negative values indicate the opposite direction.

Attribute	Value
Index	0x607C
LCP parameter number	52-40
Name	Home offset
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	0

Table 7.119 0x607C: Home Offset

7.13.2 Parameter 52-41: Homing Method (0x6098)

This object indicates the configured homing method to be used. All supported homing methods can be written into this object.

Attribute	Value
Index	0x6098
LCP parameter number	52-41
Name	Homing method
Object code	Var
Data type	INTEGER8

Attribute	Value
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 2.4* in *chapter 2.4.4 Homing Mode.*
Default value	37

Table 7.120 0x6098: Homing Method

7.13.3 Parameters 52-42 and 52-43: Homing Speeds (0x6099)

This object indicates the configured speeds used during homing procedure. The values are given in user-defined velocity units. The speed at sub-index 1 is used to initially search for the limit or home switch. The speed at sub-index 2 is used for the subsequent reverse motion after the switch is triggered.

Attribute	Value
Index	0x6099
Name	Homing speeds
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	52-42
Description	Speed during search for switch
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	48000 (corresponds to 80 RPM)
Sub-index	0x02
LCP parameter number	52-43
Description	Speed during search for zero
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	12000 (corresponds to 20 RPM)

Table 7.121 0x6099: Homing Speeds

7.13.4 Parameter 52-44: Homing Acceleration (0x609A)

This object indicates the configured acceleration to be used during homing operation. The value is given in user-defined acceleration units.

Attribute	Value
Index	0x609A
LCP parameter number	52-44

Attribute	Value
Name	Homing acceleration
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	No
Value range	UNSIGNED32
Default value	60000 (corresponds to 100 RPM/s)

Table 7.122 0x609A: Homing Acceleration

7.13.5 Parameter 52-50 to 52-57: Supported Homing Methods (0x60E3)

This object provides the supported homing methods of the servo drive. The values of the homing methods are described in *Table 2.4* in *chapter 2.4.4 Homing Mode*.

Methods –2, –1, 17, 18, 19, 21, and 37 are represented by sub-indexes 1–7.

Methods –3, –2, –1, and 37 are always available. If a method is not available, the value of the corresponding sub-index is set to *0*.

For sub-index 3, 4, 5, and 6, the availability of the related method is dependent on the Input configuration object (see *chapter 7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)*), as shown in *Table 7.123*.

Methods 17 and 18 are available at the same time.

Sub-index	Value and definition	Availability
1	Homing on actual position -3	Always available.
2	Homing on positive block -2	Always available.
3	Homing on negative block -1	Always available.
4	Homing on negative limit switch 0 or 17	Only available if the bit field *Function input 1* or *Function input 2* of object 0x200F is configured as *Left Limit* (010b).
5	Homing on positive limit switch 0 or 18	Only available if the bit field *Function input 1* or *Function input 2* of object 0x200F is configured as *Right Limit* (011b).
6	Homing on positive home switch 0 or 19	Only available if the bit field *Function input 1* or *Function input 2* of object 0x200F is configured as *Home* (100b).
7	Homing on negative home switch 0 or 21	
8	Homing on current position 37	Always available.

Table 7.123 Definition of Sub-indexes for 0x60E3

Attribute	Value
Index	0x60E3
Name	Supported homing methods
Object code	Array
Data type	UNSIGNED8
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x08
Sub-index	0x01

Attribute	Value
LCP parameter number	52-50
Description	1st supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	-3
Sub-index	0x02
LCP parameter number	52-51
Description	2nd supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	-2
Sub-index	0x03
LCP parameter number	52-52
Description	3rd supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	-1
Sub-index	0x04
LCP parameter number	52-53
Description	4th supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	Dependent on the configuration of object 0x200F.
Sub-index	0x05
LCP parameter number	52-54
Description	5th supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	Dependent on the configuration of object 0x200F.
Sub-index	0x06
LCP parameter number	52-55
Description	6th supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	Dependent on the configuration of object 0x200F.
Sub-index	0x07
LCP parameter number	52-56

Attribute	Value
Description	7th supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	Dependent on the configuration of object 0x200F.
Sub-index	0x08
LCP parameter number	52-57
Description	8th supported homing method
Access	Read only
Data type	INTEGER8
PDO mapping	No
Value range	See *Table 7.123*.
Default value	37

Table 7.124 0x60E3: Supported Homing Methods

7.13.6 Parameter 52-45 to 52-48: Additional Homing objects (0x2040)

This object provides additional parameters for the configuration of the homing mode.

Sub-index 01: Homing blocking window velocity
The servo drive assumes that it is blocked when the actual speed falls below the limit that is given in this object, for at least the time that is given in the *Homing blocking window time* object. The value is given in user-defined velocity units.

Sub-index 02: Homing blocking window time
This object is used for block detection. The time is given in milliseconds. A value of *0* means immediately.

Sub-index 03: Homing limit distance
This object indicates the maximal distance in which the homing procedure has to be finished. Otherwise, the home procedure is aborted with an error. The value is given in user-defined position units. A value of *0* disables the limitation.

Sub-index 04: Homing deceleration
This object indicates the configured deceleration to be used during homing procedure. The value is given in user-defined acceleration units.

Attribute	Value
Index	0x2040
Name	Additional homing methods
Object code	Array
Data type	UNSIGNED8
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x04
Sub-index	0x01
LCP parameter number	52-45
Description	Homing blocking window velocity
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	6000 (corresponding to 10 RPM)

Attribute	Value
Sub-index	0x02
LCP parameter number	52-46
Description	Homing blocking window time
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	100
Sub-index	0x03
LCP parameter number	52-47
Description	Homing limit distance
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0
Sub-index	0x04
LCP parameter number	52-48
Description	Homing deceleration
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	600000 (corresponding to 1000 rpm/s)

Table 7.125 0x2040: Additional Homing Objects

7.14 CAM Mode Objects

7.14.1 Parameter: CAM Profile Memory Layout (0x380F)

There are 2 memory layouts available for CAM profiles:

- 8 CAM profiles (for example, basic CAM with 256 data points)
- 2 CAM profiles (for example, basic CAM with 1024 data points)

Write to sub-index 1 to select the memory layout type. The switch between the layouts takes place after a power cycle. The currently used layout can be read in sub-index 2.

Attribute	Value
Index	0x380F
Name	CAM profile memory layout
Object code	Array
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x02
Default value	0x02
Sub-index	0x01
LCP parameter number	–

Attribute	Value
Description	CAM profile memory layout set
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	2 or 8
Default value	8
Sub-index	0x02
LCP parameter number	–
Description	CAM profile memory layout act
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	2 or 8
Default value	8

Table 7.126 0x380F: CAM Profile Memory Layout

7.14.2 Parameter: CAM Status (0x3801)

See *chapter 2.4.5.7 Notifications from the Servo Drive* for the description of this object.

Attribute	Value
Index	0x3801
Name	CAM status
Object code	Array
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x04
Default value	0x04
Sub-index	0x01
LCP parameter number	–
Description	CAM status code
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.56* in *chapter 2.4.5.7 Notifications from the Servo Drive.*
Default value	–
Sub-index	0x02
LCP parameter number	–
Description	CAM status parameter 1
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.56* in *chapter 2.4.5.7 Notifications from the Servo Drive.*
Default value	–
Sub-index	0x03
LCP parameter number	–
Description	CAM status parameter 2

Attribute	Value
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.56* in *chapter 2.4.5.7 Notifications from the Servo Drive*.
Default value	–
Sub-index	0x04
LCP parameter number	–
Description	CAM status parameter 3
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.56* in *chapter 2.4.5.7 Notifications from the Servo Drive*.
Default value	–

Table 7.127 0x3801: CAM Status

7

7.14.3 Parameter: CAM Control (0x3800)

See *chapter 2.4.5.6 Commands During Operation* for the description of this object.

Attribute	Value
Index	0x3800
Name	CAM control
Object code	Array
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x04
Default value	0x04
Sub-index	0x01
LCP parameter number	–
Description	CAM control code
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.53* in *chapter 2.4.5.6 Commands During Operation*.
Default value	–
Sub-index	0x02
LCP parameter number	–
Description	CAM control parameter 1
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.53* in *chapter 2.4.5.6 Commands During Operation*.
Default value	–
Sub-index	0x03
LCP parameter number	–
Description	CAM control parameter 2
Access	Read/write

Attribute	Value
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.53* in *chapter 2.4.5.6 Commands During Operation*.
Default value	–
Sub-index	0x04
LCP parameter number	–
Description	CAM control parameter 3
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 2.53* in *chapter 2.4.5.6 Commands During Operation*.
Default value	–

Table 7.128 0x3800: CAM Control

7.14.4 Parameters: CAM Profile 1–8 (0x3810–0x3817)

These objects contain the configuration of the CAM profiles. Together with the CAM data in objects 0x3820 to 0x3827, and optionally the CAM pattern files in objects 0x3830-0x3837, they specify a CAM.

Use object 0x3804 to select the CAM profile for the movement. Object 0x3805 provides the current number of the selected CAM profile.

If a certain CAM profile is active, SDO write access to the object, which contains the currently running profile, is rejected by means of the SDO abort code 0x0800 0020 (write access for both: CAM and CAM configuration).

Attribute	Value
Index	0x3810
Name	CAM profile 1
Object code	Record
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x03
Default value	0x03
Sub-index	0x01
LCP parameter number	–
Description	CAM configuration 1
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	See *Table 7.130*.
Default value	0
Sub-index	0x02
LCP parameter number	–
Description	CAM parsing state
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional

Attribute	Value
Value range	See *Table 7.132*.
Default value	–
Sub-index	0x03
LCP parameter number	–
Description	CAM parsing error info
Access	Read only
Data type	UNSIGNED32
PDO mapping	Optional
Value range	See *Table 7.132*.
Default value	–

Table 7.129 0x3810: CAM Profile 1

NOTICE

Objects 0x3811 to 0x3817 are the same as CAM profile 1 (0x3810) as shown in *Table 7.129*. Only the CAM profile number increases by 1 for each object.

15	14 13 12 11 10 9 8 7 6 5 4 3	2	1	0
Parse CAM data	reserved	cyclic/non-cyclic	master abs/rel	slave abs/rel
MSB				LSB

Table 7.130 CAM Configuration

Bit	Value	Definition
0	0	Slave position is an absolute value.
	1	Slave position is a relative value.
1	0	Master position is an absolute value.
	1	Master position is a relative value.
2	0	Axis repeats the execution of the CAM in cyclic mode.
	1	Axis stops after 1 execution of the CAM in non-cyclic mode.
15	0	The servo drive resets this bit automatically to 0 if the parsing was started.
	1	Triggers a new CAM parsing (needed if new CAM data was transferred over SDO and if the CAM data was transferred as a file).

Table 7.131 Definition of Bits of the CAM Slave Table Selector

Both supported CAM types can be executed in cyclic and non-cyclic mode:

- Cyclic mode: Repeats the execution of the CAM profile continuously on the master cycle basis.
- Non-cyclic mode: The CAM profile is only run once. If the master position is already outside of the CAM profile, the slave axis stays in synchronized motion and keeps the last position.

CAM parsing state (sub-index 02)	Description	CAM parsing error information (sub-index 03)			
		Byte 0	Byte 1	Byte 2	Byte 3
0x0000	No error/Valid CAM profile	Reserved			
0x0001	File not found.	Reserved			
0x0002	Error in 1 of the elements *controlParam*.	Number of the set	Reserved		
0x0004	Profile currently active. No parsing possible.	Reserved			
0x0005	Out of CAM memory (CAM profile is too large).	Reserved			

CAM parsing state (sub-index 02)	Description	CAM parsing error information (sub-index 03)			
		Byte 0	Byte 1	Byte 2	Byte 3
0x0007	Advanced CAM only: Wrong node type.	Node ID		Segment ID (non-fitting)	
0x0009	Advanced CAM only: Unknown node type.	Number of Element in Node section		Normal: 0 EventSegmentContainer: segment ID	
0x000A	Advanced CAM only: Start node has wrong type.	Segment ID		Start node ID	
0x000B	CAM has <2 nodes.	Reserved			
0x000C	Basic CAM only: Parameter is out of range.	Type of parameter: 0: Reserved 1: Master position 2: Slave position 3: Velocity 4: Acceleration	Data point index	Reserved	
0x000D	Basic CAM only: A mandatory parameter is missing.				
0x000E	Advanced CAM only: End node has wrong type.	Segment ID		End node ID	
0x000F	Parameter in header has wrong range.	Parameter ID		Reserved	
0x0010	A mandatory parameter is missing in the header.	Parameter type (see *Table 7.133*).		Reserved	
0x0011	CAM file is empty	Reserved			
0x0012	CAM storage buffer not ready.	Reserved			
0x0013	CAM storage buffer too small.	Reserved			
0x0014	CAM corrupt file.	Reserved			
0x0015	Node allocation overflow.	Node ID		Reserved	
0x0016	Segment allocation overflow.	Segment ID		Start node ID	
0x0017	Multiple segments as default.	Segment ID		Start node ID	
0x0018	Advanced CAM only: Unknown segment type.	Number of elements in segment section.	Reserved		
0x0019	Starting node failure.	1: Not found 2: Wrong type 3: No following segment	Reserved		
0x001A	Advanced CAM only: node parameter is out of range.	Parameter type (see *Table 7.134* and *Table 7.135*).		Node type 0: Reserved 1: Guide node 2: Event node	Node ID
0x001B	Advanced CAM only: A mandatory node parameter is missing.				
0x001C	Advanced CAM only: segment parameter is out of range.	Parameter type (see *Table 7.136* to *Table 7.146*).		Segment type 0: Reserved 1: Guide poly 5: Move distance 6: Flying stop 7: Return 8: Event segment container 0x50: Time poly 0x51: Velocity 0x52: Torque 0x53: Sync	Segment ID
0x001D	Advanced CAM only: A mandatory segment parameter is missing.				
0x001E	Only 1 friction segment allowed.	Segment ID		Reserved	
0x0020	Undefined CAM file type.	Reserved			

7

CAM parsing state (sub-index 02)	Description	CAM parsing error information (sub-index 03)			
		Byte 0	**Byte 1**	**Byte 2**	**Byte 3**
0x0021	No valid CAM file (error in XML file definition).	Reserved			
0x1000	Action ID element invalid. Attribute *actionID* is missing or has invalid format	Action ID	Action node number	Reserved	
0x1001	Action ID element invalid. Attribute *actionID* is out of range	Action ID	Action node number	Reserved	
0x1002	Action command element invalid. Mandatory attribute is missing or has invalid format.	5: control-ParamN.speedP 6: control-ParamN.speedI 7: control-ParamN.speedD 8: control-ParamN.inertia 9: control-ParamN.positionP 10: control-ParamN.positionD 11: selControlParam.set 12: compensateR-ounding.partition 13: compensateR-ounding.revolutions 14: compensateR-ounding.offsetRev 16: resetCounter 17: startCounter 18: stopCounter 19: logValue 20: setFollowSegment 21: setDigOut	Action ID	Reserved	
0x1003	Action command element invalid. Mandatory attribute is out of range.				
0x1004	Node action list is invalid.	Reserved	Node ID	Reserved	
0x1005	Segment start action list is invalid.	Segment ID	Reserved		
0x1006	Segment end action list is invalid.	Segment ID	Reserved		
0x1007	Segment ID invalid.	Segment ID		Start node ID	
0x1008	Node ID invalid.	Reserved		Start node ID	
0x1009	EventSegment invalid starting event node.	Segment ID		Event node ID	
0x1010	Action list ID invalid.	Reserved			
0x1100	Exit ID element invalid. Attribute *exit ID* is missing or has invalid format.	Exit ID	Exit node number	Reserved	
0x1101	Exit ID element invalid. Attribute *exit ID* is out of range.				

CAM parsing state (sub-index 02)	Description	CAM parsing error information (sub-index 03)			
		Byte 0	Byte 1	Byte 2	Byte 3
0x1102	Exit condition element invalid. Mandatory attribute is missing or has invalid format.	3: rectMark.input 4: rectMark.mode 5: rectMark.threshold 6: rectMark.minLength 7: rectMark.maxLength 8: pattern.input 9: pattern.threshold 10: pattern.subsample 11: pattern.checklength 12: checkDigInput.input 13: checkDigInput.value 14: checkCounter.input 15: checkCounter.threshold 16: checkAnaInput.input 17: checkAnaInput.threshold 18: checkAnaInput.condition 19: checkVelocity.threshold 20: checkVelocity.condition 21: checkTorque.threshold 22: checkTorque.condition 23: checkDistance.threshold	Exit ID	Reserved	
0x1103	Exit condition element invalid. Mandatory attribute is out of range.				
0x1104	Exit ID list of a segment is invalid.	Segment ID	Reserved		
0x1106	Pattern file invalid.	Reserved	Line number	Reserved	
0x1107	Out of pattern memory. Pattern too large.	Reserved			
0x7FFE	An unknown error occurred.	Reserved			
0x7FFF	Parsing	Reserved			

Table 7.132 CAM Parsing State and Error Information

Number	Description
0	Reserved
2	windowRev of element followingError.
3	time of element followingError.
4	positionP of element controlParam1.
5	positionD of element controlParam1.
6	speedP of element controlParam1.
7	speedI of element controlParam1.
8	speedD of element controlParam1.
9	inertia of element controlParam1.
10	positionP of element controlParam2.
11	positionD of element controlParam2.
12	speedP of element controlParam2.
13	speedI of element controlParam2.
14	speedD of element controlParam2.
15	inertia of element controlParam2.
16	numerator of element masterScaling.
17	denominator of element masterScaling.
18	numerator of element slaveScaling.
19	denominator of element slaveScaling.
20	Control loop set 1 has wrong values.
21	Control loop set 2 has wrong values.

Table 7.133 Definition of Parameter Type for CAM Profile Header

Value	Definition
0	Reserved
1	Node ID
2	MasterPos
3	Signal
4	Action

Table 7.134 Parameter IDs for GuideNodes

Value	Definition
0	Reserved
1	Node ID
2	Signal
3	Action

Table 7.135 Parameter IDs for EventNodes

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Movement type
8	Start position
9	Start velocity
10	Start acceleration
11	Distance
12	End velocity
13	End acceleration
64–69	Coeffs a0–a5

Table 7.136 Parameter IDs for GuidePoly Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Start position
8	Start velocity
9	Start acceleration
10	End velocity
11	End acceleration

Table 7.137 Parameter IDs for Move Distance Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Partition
8	Revolutions
9	Offset

Table 7.138 Parameter IDs for Return Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Start Position
8	Maximal constant distance
9	Brake length
10	Start velocity
11	Brake distance
64	A0
65	A1
66	A2
67	A3

Table 7.139 Parameter IDs for Flying Stop Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Starting node

Table 7.140 Parameter IDs for Event Segment Container

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Duration
8	Exit condition
9	Movement type
10	Start position
11	Start velocity
12	Distance
13	End velocity
14	End acceleration
64–69	Coeffs a0–a5

Table 7.141 Parameter IDs for Time Poly Segment with Positions

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Duration
8	Exit condition
9	Movement type
10	Start position
11	Velocity
12	Acceleration
13	Deceleration
14	Torque limit

Table 7.142 Parameter IDs for Velocity Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Duration
8	Exit condition
9	Movement type
10	Start position
11	Torque
12	Ramp
13	Speed limit

Table 7.143 Parameter IDs for Torque Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Duration
8	Exit condition
9	Movement type
10	Start position
11	Velocity ration
12	Acceleration
13	Deceleration
14	Torque limit

Table 7.144 Parameter IDs for Sync Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Duration
8	Exit condition
9	Movement type
10	Start position

Table 7.145 Parameter IDs for PWM Off Segment

Value	Definition
0	Reserved
1	Segment ID
2	Starting node
3	Ending node
4	Start action
5	End action
6	Default segment
7	Duration
8	Exit condition
9	Movement type
10	Start position
11	Velocity low
12	Velocity high
13	Do compensation
14	Acceleration
15	Deceleration
16	Guide value offset
17	Timeout

Table 7.146 Parameter IDs for Friction Segment

Value	Definition
0	Reserved
1	MasterPos
2	SlavePos
3	vel
4	acc

Table 7.147 Definition of Parameter Type for Basic CAM Data Points

7.14.5 Parameters: CAM Data 1–8 (0x3820–3827)

These objects contain the CAM files. If the new CAM data is to be parsed, set bit 15 of the CAM configuration object (0x3810–0x3817).

Attribute	Value
Index	0x3820
Name	CAM data 1
Object code	Var
Data type	DOMAIN
Sub-index	0x00
Access	Read/write
PDO mapping	No
Value range	DOMAIN
Default value	–

Table 7.148 0x3820: CAM Data 1

NOTICE

Objects 0x3821 to 0x3827 are the same as CAM data 1 (0x3820) as shown in *Table 7.148*. Only the CAM data number is different.

7.14.6 Parameters: CAM Pattern 1–8 (0x3830–3837)

The pattern data for the CAMs can be transmitted by transferring it to these objects.

The format is a csv file with white space as the separator between the values. The 1st line contains the number of channels (= columns) and the number of samples per channel. The values start in the 2nd line of the file.

If >2 channels are included in the pattern file, only the 1st 2 columns are used. The values are interpreted alternating (1st value of 1st column, 1st value of 2nd column, 2nd value of 1st column, and so on). The remaining channels are ignored.

If the new CAM data is to be parsed (CAM data + CAM pattern), set bit 15 of the CAM configuration object (see *chapter 7.14.4 Parameters: CAM Profile 1–8 (0x3810–0x3817)*).

Attribute	Value
Index	0x3830
Name	CAM pattern 1
Object code	Var
Data type	DOMAIN
Sub-index	0x00
Access	Read/write
PDO mapping	No
Value range	DOMAIN
Default value	–

Table 7.149 0x3830: CAM Pattern 1

NOTICE

Objects 0x3831 to 0x3837 are the same as CAM pattern 1 (0x3830) as shown in *Table 7.149*. Only the CAM pattern number increases by 1 for each sub-index.

7.14.7 Parameter: CAM Profile Selector (0x3804)

This object indicates the selected CAM slave table. To activate a new CAM or reactivate an already processed one, write the number of the required table to this object and start it by using the handshaking procedure described in *chapter 2.4.5.1 Activating a CAM profile*.

If bits 13–15 are set, the real start of the CAM is delayed until the configured input is on. The behavior of the activation is then as if the handshake is taking place at the occurrence of the (input) event. If the start of that CAM is *Slave absolute*, the servo drive jumps to the correct position.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
delay code			reserved									number of CAM profile			
MSB															LSB

Table 7.150 0x3804: CAM Slave Table Selector

The value range of the *number of CAM profile* is limited to 1–8 according to *Table 7.149*.

Bit	Value	Definition
0–3	0	Not allowed.
	1–8	Selects the CAM profile to be activated.
	9–15	Reserved.
13–15	0	Normal CAM activation behavior, independent of digital input signals.
	1	Delayed activation of CAM: Processing of CAM is delayed until digital input 1 is on; input 2 is irrelevant.
	2	Delayed activation of CAM: Processing of CAM is delayed until digital input 2 is on; input 1 is irrelevant.
	3	Delayed activation of CAM: Processing of CAM is delayed until digital input 1 or digital input 2 is on.
	4	Delayed activation of CAM: Processing of CAM is delayed until digital input 1 and digital input 2 is on.
	5–7	Reserved.

Table 7.151 Definition of Bits of the CAM Slave Table Selector

Attribute	Value
Index	0x3804
LCP parameter number	–
Name	CAM profile selector
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.151*.
Default value	0

Table 7.152 0x3804: CAM Profile Selector

7.14.8 Parameter: CAM Profile Status (0x3805)

This object provides the current number of the selected CAM slave table (bits 0–3) and the used starting behavior. The value 0 is allowed for the field *number of active CAM profile* to indicate that no CAM is active. Bits 4–11 indicate if the CAM profile is valid, meaning that it can be activated.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAM error	buffered only	end of profile	reserved (0)	CAM profile validity								number of active CAM profile			
MSB															LSB

Table 7.153 0x3805: CAM Profile Status

Bit	Value	Definition
0–3	0	No CAM slave table selected.
	1–8	Activated CAM profile.
4	0	CAM profile 1 is invalid (and cannot be activated).
	1	CAM profile 1 is valid.
5	0	CAM profile 2 is invalid (and cannot be activated).
	1	CAM profile 2 is valid.
6	0	CAM profile 3 is invalid (and cannot be activated).
	1	CAM profile 3 is valid.
7	0	CAM profile 4 is invalid (and cannot be activated).
	1	CAM profile 4 is valid.
8	0	CAM profile 5 is invalid (and cannot be activated).
	1	CAM profile 5 is valid.
9	0	CAM profile 6 is invalid (and cannot be activated).
	1	CAM profile 6 is valid.
10	0	CAM profile 7 is invalid (and cannot be activated).
	1	CAM profile 7 is valid.
11	0	CAM profile 8 is invalid (and cannot be activated).
	1	CAM profile 8 is valid.
13	0	Axis is in the CAM or past the end of the profile.
	1	End of profile is reached (only active for 1 cycle).
14	0	Only available if bit 15 = 1: General CAM error related to the active and the buffered CAM profile.
	1	Only available if bit 15 = 1: CAM error only related to the buffered CAM, not the currently active CAM profile.
15	0	No error.
	1	Error occurred at CAM activation request; Remains at 1 until a new command is issued.

Table 7.154 Definition of Bits of the CAM Profile Status

Additional information for bit 13:
Bit 13 is a pulsed output signaling the cyclic end of the CAM profile. It is set every time that the end of the CAM profile is reached. In reverse direction, bit 13 is shown at the end of the CAM profile (in this case, the first point of the CAM profile). This bit is only set for 1 fieldbus cycle.

Additional information for bit 15:
Table 7.155 shows which situations lead to a CAM error (bit 15 of the CAM profile status object), and which lead to command errors (signaled in the *Statusword*).

CAM error bit		Command error	Description
15	14		
		X	• Tried to activate a CAM when there is already a buffered CAM (see *chapter 2.4.5.1 Activating a CAM profile*). or • An invalid CAM profile number is selected in the CAM profile selector. This can either be an out of range number or an invalid profile.
X			Guide value reversed while an advanced CAM is active or blending is active.
X	X		Error during CAM blending: The non-cyclic end of the blending would be after the end of the new CAM profile or would be outside of the cycle.

Table 7.155 Definition of CAM Error Versus Command Error

Attribute	Value
Index	0x3805
LCP parameter number	–
Name	CAM profile status
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.155*.
Default value	No

Table 7.156 0x3805: CAM Profile Status

7.14.9 Parameter: CAM Slave Offset (0x3807)

This object contains an offset value to the slave position of a CAM. When using the slave position as a relative value, only changing the value of object *CAM slave offset* during operation has an effect. At the beginning of a CAM, the offset value is compensated by the internal offset that is calculated for relative operation.

Objects for *Guide value offset* (master) and for slave offset can be used in parallel, meaning it is possible to have a master and a slave offset at the same time.

CAM slave offset accepts negative values. The values are always added to the current value.

The value of the *CAM slave offset* is given in revolutions of rotor position. Changing a value has an immediate effect if the axis is processing a CAM.

Attribute	Value
Index	0x3807
LCP parameter number	–
Name	CAM slave offset
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	FLOAT

Attribute	Value
Default value	No

Table 7.157 0x3807: CAM Slave Offset

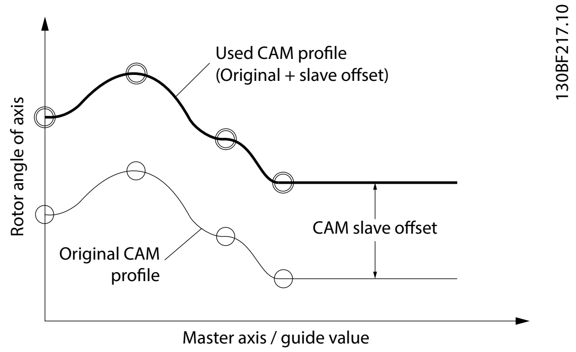


Illustration 7.5 CAM Slave Offset

7.14.10 Parameter: CAM Slave Scaling (0x3809)

The axis supports scaling in slave direction. The scaling factor consists of a numerator and a denominator. The value 0x0000 0000 must be reserved and cannot be used for the numerator and the denominator objects. A writing order is defined so that both values can be changed simultaneously. The activation of the new factor is done after the writing of the numerator. This means to change both values at the same time, the denominator must be written first and the numerator afterwards. It is also possible to only change the numerator by just writing a new value (the denominator remains the same).

Guide value scaling (master) and *CAM slave scaling* can be used in parallel. The numerator and the denominator accept negative values. The division of 2 negative values results in a positive value. The slave position is multiplied by the quotient of numerator and denominator.

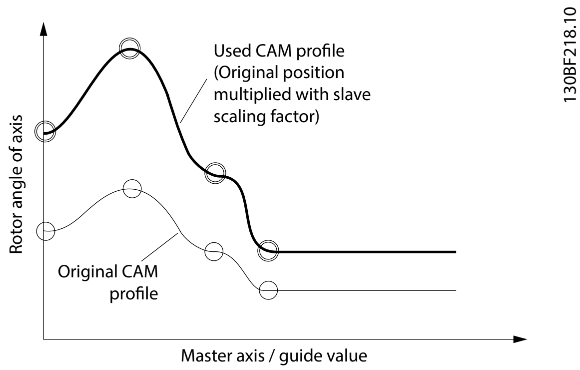


Illustration 7.6 CAM Slave Scaling

Attribute	Value
Index	0x3809
Name	CAM slave scaling
Object code	Array
Data type	INTEGER32
Sub-index	0x00
Description	Number of entries

Attribute	Value
Access	Read only
PDO mapping	No
Value range	UNSIGNED8
Default value	0x02
Sub-index	0x01
LCP parameter number	–
Description	CAM slave numerator
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	1
Sub-index	0x02
LCP parameter number	–
Description	CAM slave denominator
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	1

Table 7.158 0x3809: CAM Slave Scaling

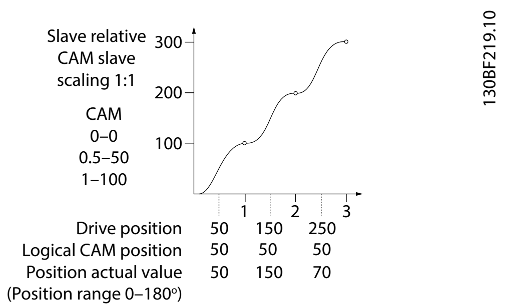


Illustration 7.7 Influence of CAM Scaling on the Different Position Objects (1)

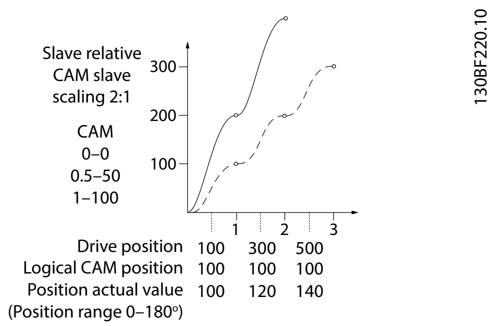


Illustration 7.8 Influence of CAM Scaling on the Different Position Objects (2)

7.14.11 Parameter: Minimum Blending Distance (0x380A)

This parameter defines the minimum distance for blending to a new CAM. The minimum blending distance is calculated in the direction of the guide value when blending is activated. If the guide value is at standstill at the moment of activation, the last known direction of the guide value is used.

Attribute	Value
Index	0x380A
Name	Minimum blending distance
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32

Attribute	Value
Default value	0

Table 7.159 0x380A: Minimum Blending Distance

7.14.12 Parameter: Logical CAM Position (0x2020)

This object provides the logical CAM position. It is only up to date if the servo drive is in CAM mode. Otherwise, the latest value is preserved. The value is given in revolutions of motor shaft.

Attribute	Value
Index	0x2020
Name	Logical CAM Position
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	FLOAT
Default value	–

Table 7.160 0x2020: Logical CAM Position

7.14.13 Parameter: Logical CAM Set Point (0x2021)

This object provides the logical CAM set point. It is only up to date if the servo drive is in CAM mode. Otherwise, the latest value is preserved. The value is given in revolutions of motor shaft.

Attribute	Value
Index	0x2021
Name	Logical CAM set point
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	FLOAT
Default value	–

Table 7.161 0x2021: Logical CAM Set Point

7.14.14 Parameter: Active Segment ID (0x2019)

This object contains the segment ID of the currently active segment.

The following values indicate special states during CAM processing:

- 65534: Segment ID during blending to starting node in profile
- 65535: Segment ID during blending between CAM profiles

Attribute	Value
Index	0x2019
Name	Active Segment ID

7

Attribute	Value
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–

Table 7.162 0x2019: Active Segment ID

7.14.15 Parameter: Last Node ID (0x201A)

This object contains the node ID of the last node passed.

Attribute	Value
Index	0x201A
Name	Last node ID
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–

Table 7.163 0x201A: Last Node ID

7.14.16 Parameter: Logged Values (0x3870)

The object contains the values that are logged by a CAM action.

Attribute	Value
Index	0x3870
Name	Logged values
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x10
Sub-index	0x01
LCP parameter number	–
Description	Memory cell 1
Access	Read only
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.164 0x3870: Logged Values

NOTICE

Sub-indexes 0x02 to 0x10 are the same as sub-index 1 as shown in *Table 7.164*. Only the memory cell number is increased by 1 for each sub-index.

7.14.17 Parameter: Digital Input Counters (0x3860)

The object contains the counter values accumulated by actions in CAM mode. The values are read/write so that the counters can be modified manually.

Attribute	Value
Index	0x3860
Name	Digital input counters
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	–
Description	Counter for digital input 1
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0
Sub-index	0x02
LCP parameter number	–
Description	Counter for digital input 2
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0

Table 7.165 0x3860: Digital Input Counters

7.15 Gear Mode Objects

7.15.1 Parameter: Gear Ratio (0x3900)

The gear ratio consists of a numerator and a denominator.

The value 0x0000 0000 must be reserved and cannot be used for the numerator or the denominator object. A writing order is defined so that both values can be changed simultaneously (using SDO). The activation of the new factor is done after the writing of the numerator. This means, to change both values at the same time, the denominator must be written first and the numerator afterwards. It is also possible to only change the numerator by just writing a new value (the denominator remains the same).

Both objects also accept negative values. The division of 2 negative values results in a positive value. When writing both objects over PDO (in 1 PDO), they are updated at the same time.

Attribute	Value
Index	0x3900
Name	Gear ratio
Object code	Array
Data type	INTEGER32
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	UNSIGNED8
Default value	0x02
Sub-index	0x01
LCP parameter number	–
Description	Gear ratio numerator
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	1
Sub-index	0x02
LCP parameter number	–
Description	Gear ratio denominator
Access	Read/write
PDO mapping	Optional
Value range	INTEGER32
Default value	1

Table 7.166 0x3900: Gear Ratio

7.15.2 Parameter: Gear Synchronization Option Code (0x3901)

Indicates what action is performed when the *Gear In Pos* functionality is used but it is not possible to sync to that position.

Attribute	Value
Index	0x3901
LCP parameter number	–
Name	Gear synchronization option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	0: Raise a command error 1: Sync to the next guide value cycle
Default value	0

Table 7.167 0x3901: Gear Synchronization Option Code

7.15.3 Parameter: Gear Master Start Distance (0x3902)

This object defines the master distance for gear in procedure (when the slave axis is started to get into synchronization). The value must be given in *Guide value position* units. The direction is taken from the *Master sync direction* bit in the *Controlword (0x6040)* (see *Illustration 7.9*).

NOTICE

The sign of *Master Start Distance* is represented by the *MasterSyncDirection* bit in the *Controlword* (0x6040).

Attribute	Value
Index	0x3902
LCP parameter number	–
Name	Gear master start distance
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0

Table 7.168 0x3902: Gear Master Start Distance

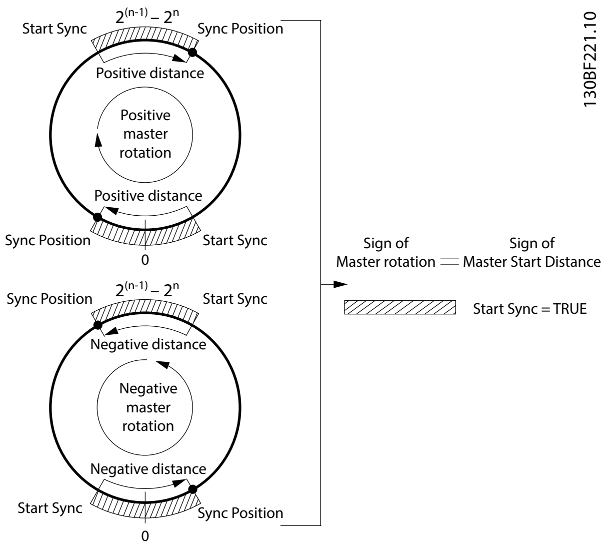


Illustration 7.9 Synchronization Executed at StartSync Position

NOTICE

***Master Rotation* direction and *MasterStartDistance* have the same signs, therefore synchronization is executed.**

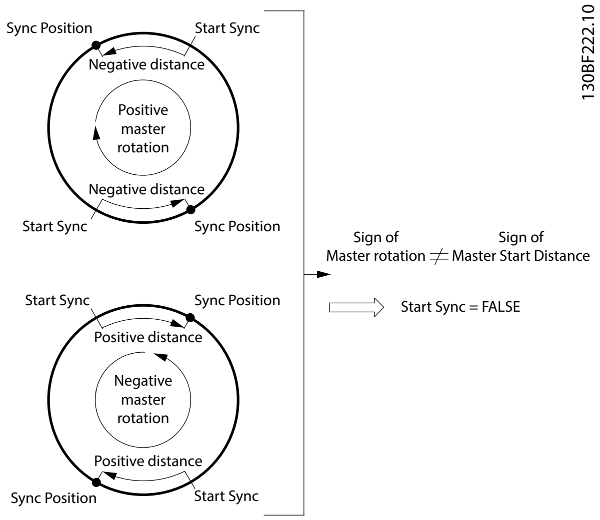


Illustration 7.10 Synchronization Not Executed

NOTICE

***Master Rotation* direction and *MasterStartDistance* have opposite signs, therefore synchronization is not executed.**

7.15.4 Parameter: Gear Master Sync Position (0x3903)

This object defines the position of the master where the slave is in sync with the master. The value is given in *Guide value position* units.

Attribute	Value
Index	0x3903
LCP parameter number	–
Name	Gear master sync position
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0

Table 7.169 0x3903: Gear Master Sync Position

7.15.5 Parameter: Gear Slave Sync Position (0x3904)

This object defines the position of the slave where the slave is in sync with the master. The value is given in user-defined position units.

Attribute	Value
Index	0x3904
LCP parameter number	–
Name	Gear slave sync position
Object code	Var
Data type	SIGNED32

Attribute	Value
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	SIGNED32
Default value	0

Table 7.170 0x3904: Gear Slave Sync Position

7.16 ISD Inertia Measurement Objects

7.16.1 Parameter 52-60: Measured Inertia (0x2009)

This object indicates the result after a measurement process. If the measurement completed successfully, the result of the measurement is given in kg m^2. The value is not stored in the servo drive and it is not automatically used by the control loop. If the measurement was not processed successfully, the error reason can be read from this object. See *Table 7.171* for the coding.

Value	Description
>0	Legal result of the measurement. Given in kg m^2.
0	No result available or measurement ongoing.
–1	Target velocity could not be reached.
–2	Torque could not be produced due to voltage limit.
–3	Measurement has been aborted (according to request of the *Controlword*).
–4	Measurement torque is limited by the application torque limit or maximum torque limit.

Table 7.171 Result Coding of ISD Inertia Measurement

Attribute	Value
Index	0x2009
LCP parameter number	52-60
Name	Measure inertia
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read only
PDO mapping	No
Value range	FLOAT
Default value	–

Table 7.172 0x2009: Measured Inertia

7.16.2 Parameters 52-61 and 52-62: Inertia Measurement Parameters (0x200A)

These parameters can be used to limit the measurement process. If the values used are too small, the measurement cannot be done and an error is issued (see *chapter 7.16.1 Parameter 52-60: Measured Inertia (0x2009)*).

Sub-index 01 contains the velocity used for the measurement in user-defined velocity units.

Sub-index 02 contains the acceleration torque for the inertia measurement given per thousand of rated torque.

Attribute	Value
Index	0x200A
Name	Inertia measurement parameters
Object code	Array
Data type	RECORD
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	52-61
Description	Inertia measurement velocity
Access	Read/write
Data type	UNSIGNED32
PDO mapping	No
Value range	UNSIGNED32
Default value	90% of maximum servo drive limit
Sub-index	0x02
LCP parameter number	52-62
Description	Inertia measurement torque
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	UNSIGNED16
Default value	30% of maximum servo drive limit

Table 7.173 0x200A: Inertia Measurement Parameters

7.17 Digital CAM Switch Objects

7.17.1 Parameter: On Compensation (0x3840)

This parameter shows the compensation time with which the switching on is advanced or delayed. A negative value means that the output changes before the switching position is reached.

The value can be changed while the digital CAM switching functionality is enabled. It has an immediate effect.

The value is given in milliseconds.

Attribute	Value
Index	0x3840
LCP parameter number	–

Attribute	Value
Name	On compensation
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	INTEGER16
Default value	0

Table 7.174 0x3840: On Compensation

7.17.2 Parameter: Off Compensation (0x3841)

This parameter shows the compensation time with which the switching off is advanced or delayed. A negative value means that the output changes before the switching position is reached.

The value can be changed while the digital CAM switching functionality is enabled. It has an immediate effect.

The value is given in milliseconds.

Attribute	Value
Index	0x3841
Name	Off compensation
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	INTEGER16
Default value	0

Table 7.175 0x3841: Off Compensation

7.17.3 Parameter: Hysteresis (0x3842)

This parameter shows the distance from the switching point (in positive and negative direction). The switch is not executed until the axis has left this area in order to avoid multiple switching around the switching point. Setting this parameter avoids the phenomenon where the output continually switches if the axis is near the switching point and the actual position is jittering around the switching point.

The value can be given in user-defined position units (using sub-index 0x01) or in revolutions (using sub-index 0x02). When writing to one of the indexes, the other one is updated automatically by the servo drive.

Attribute	Value
Index	0x3842
Name	Hysteresis
Object code	Array
Data type	RECORD
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only

7

Attribute	Value
PDO mapping	No
Default value	0x02
Sub-index	0x01
Description	Hysteresis in user-defined units
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0
Sub-index	0x02
Description	Hysteresis in revolutions
Access	Read/write
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0

Table 7.176 0x3842: Hysteresis

7.17.4 Parameters: Digital CAM Switch Parsing Control (0x3843)

If the digital CAM profile is currently active, SDO write access is rejected by SDO abort code 0x080 0020.

Attribute	Value
Index	0x3843
Name	Digital CAM switch parsing control
Object code	Array
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x03
Default value	0x03
Sub-index	0x01
Description	Digital CAM switch configuration
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	See *Table 7.178*.
Default value	0
Sub-index	0x02
Description	Digital CAM switch parsing state
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 7.180*.
Default value	–

Table 7.177 0x3843: Digital CAM Switch Parsing Control

15	14	0
Parse digital CAM switch data	Reserved	
MSB		LSB

Table 7.178 Digital CAM Switch Configuration

Bit	Value	Definition
15	0	The servo drive automatically resets this bit to 0 if the parsing is started.
	1	Triggers a new digital CAM switch parsing. This is needed if a new digital CAM switch data was transferred over SDO, and if the data was transferred as a file.

Table 7.179 Definition of Digital CAM Switch Configuration Bits

<table>
<tr><th rowspan="2">Digital CAM switch parsing state (sub-index 02)</th><th rowspan="2">Description</th><th colspan="4">Digital CAM switching parsing error information (sub-index 03)</th></tr>
<tr><th>Byte 0</th><th>Byte 1</th><th>Byte 2</th><th>Byte 3</th></tr>
<tr><td>0x0000</td><td>No error/valid digital CAM switch data</td><td colspan="4">Reserved</td></tr>
<tr><td>0x0001</td><td>File not found.</td><td colspan="4">Reserved</td></tr>
<tr><td>0x0004</td><td>Switches currently active. No parsing possible.</td><td colspan="4">Reserved</td></tr>
<tr><td>0x0005</td><td>Out of digital CAM switch memory (digital CAM switches data is too large).</td><td colspan="4">Reserved</td></tr>
<tr><td>0x000B</td><td>Digital CAM switch has no switch.</td><td colspan="4">Reserved</td></tr>
<tr><td>0x000C</td><td>Parameter is out of range.</td><td rowspan="4">Type of parameter: See Table 7.133 in chapter 7.14.4 Parameters: CAM Profile 1–8 (0x3810–0x3817).</td><td rowspan="2">Switch ID</td><td colspan="2" rowspan="2">Reserved</td></tr>
<tr><td>0x000D</td><td>A mandatory parameter is missing.</td></tr>
<tr><td>0x000F</td><td>A parameter in the header has wrong range.</td><td colspan="3">Reserved</td></tr>
<tr><td>0x0010</td><td>A mandatory parameter is missing in the header.</td><td colspan="3">Reserved</td></tr>
<tr><td>0x7FFE</td><td>An unknown error occurred.</td><td colspan="4">Reserved</td></tr>
<tr><td>0x7FFF</td><td>Parsing</td><td colspan="4">Reserved</td></tr>
</table>

Table 7.180 Digital CAM Switching State and Error Information

NOTICE

The counting of switches ID starts with 1.

Value	Definition
0	Reserved
1	Unit
32	Hysteresis
3	CamSwitchMode
4	FirstOnPosition
5	LastOnPosition
6	Duration
7	AxisDirection
8	OnCompensation
9	OffCompensation

Table 7.181 Definition of Parameter Type for Switch Element

7.17.5 Parameter: Digital CAM Switches Data (0x3844)

The digital CAM switches can be transmitted by transferring the byte stream of a digital CAM switch file over standard SDO. The information cannot be transferred while the digital CAM switching functionality is enabled. The content of the XML file is transferred when the object is read.

Attribute	Value
Index	0x3844
LCP parameter number	–
Name	Digital CAM switches data
Object code	Var
Data type	DOMAIN
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Default value	–

Table 7.182 0x3844: Digital CAM Switches Data

7.18 Touch Probe Objects

7.18.1 Parameter: Touch Probe Function (0x60B8)

This object configures the function of the touch probe.

Bit	Value	Definition
0	0	Switch off touch probe 1.
	1	Enable touch probe 1.
1	0	Trigger first event.
	1	Continuous.
3, 2	00	Trigger with touch probe 1 input.
	01	Trigger with zero impulse signal or position encoder.
	10	Touch probe source as defined in object 0x60D0, sub-index 01.
	11	Reserved.
4	0	Switch off sampling at positive edge of touch probe 1.
	1	Enable sampling at positive edge of touch probe 1.
5	0	Switch off sampling at negative edge of touch probe 1.
	1	Enable sampling at negative edge of touch probe 1.

Bit	Value	Definition
6	0	Always accept trigger event.
	1	Accept trigger event only within window defined by objects 0x3853, sub-index 01 and 0x3854, sub-index 01.
7	0	Accept bits 0–6 of this object.
	1	Ignore bits 0–6 of this object.
8	0	Switch off touch probe 2.
	1	Enable touch probe 2.
9	0	Trigger first event.
	1	Continuous.
11, 10	00	Trigger with touch probe 2 input .
	01	Trigger with zero impulse signal or position encoder.
	10	Touch probe source as defined in object 0x60D0, sub-index 02.
	11	Reserved.
12	0	Switch off sampling at positive edge of touch probe 2.
	1	Enable sampling at positive edge of touch probe 2.
13	0	Switch off sampling at negative edge of touch probe 2.
	1	Enable sampling at negative edge of touch probe 2.
14	0	Always accept trigger event.
	1	Accept trigger event only within window defined by objects 0x3853, sub-index 02 and 0x3854, sub-index 02.
15	0	Accept bits 8–14 of this object.
	1	Ignore bits 8–14 of this object.

Table 7.183 Definition of Touch Probe Function Bits

NOTICE

In order to only influence 1 of the touch probe functionalities, use bit 7 to disable the writing of bits 0–6, and bit 15 to disable the writing of bits 8–14. Those bits are then discarded and the old value is kept.

Attribute	Value
Index	0x60B8
LCP parameter number	–
Name	Touch probe function
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.183*.
Default value	0

Table 7.184 0x60B8: Touch Probe Function

7.18.2 Parameter: Touch Probe Status (0x60B9)

This object provides the status of the touch probe.

Bits 1 and 2 are set to *0* when touch probe 1 is switched off (object 0x60B8 bit 0 is 0). Bits 9 and 10 are set to *0* when touch probe 2 is switched off (object 0x60B8 bit 8 is 0).

Use bits tp1 and tp2 of object 0x2006 (see *chapter 7.22.12 Parameter 50-08: Motion and Input Status (0x2006)*) to determine if a value has been stored (positive or negative edge).

Bit	Value	Definition
0	0	Touch probe 1 is switched off.
	1	Touch probe 1 is enabled.
1	0	Touch probe 1 no positive edge value stored.
	1	Touch probe 1 positive edge position stored.
2	0	Touch probe 1 no negative edge value stored.
	1	Touch probe 1 negative edge position stored.
3–5	0	Reserved.
7, 6	–	User-defined (reserved).
8	0	Touch probe 2 is switched off.
	1	Touch probe 2 is enabled.
9	0	Touch probe 2 no positive edge position stored.
	1	Touch probe 2 positive edge position stored.
10	0	Touch probe 2 no negative edge value stored.
	1	Touch probe 2 negative edge position stored.
11–13	0	Reserved.
15, 14	–	User-defined (reserved).

Table 7.185 Definition of Touch Probe Status Bits

Attribute	Value
Index	0x60B9
LCP parameter number	–
Name	Touch probe status
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.185*.
Default value	–

Table 7.186 0x60B9: Touch Probe Status

7.18.3 Parameter 51-51: Touch Probe 1 Positive Edge (0x60BA)

This object provides the position value of the touch probe 1 at positive edge. The value is given in user-defined position units.

Attribute	Value
Index	0x60BA
LCP parameter number	51-51
Name	Touch probe 1 positive edge
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.187 0x60BA: Touch Probe 1 Positive Edge

7.18.4 Parameter 51-54: Touch Probe 1 Negative Edge (0x60BB)

This object provides the position value of touch probe 1 at negative edge. The value is given in user-defined position units.

Attribute	Value
Index	0x60BB
LCP parameter number	51-54
Name	Touch probe 1 negative edge
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.188 0x60BB: Touch Probe 1 Negative Edge

7.18.5 Parameter 51-61: Touch Probe 2 Positive Edge (0x60BC)

This object provides the position value of the touch probe 2 at positive edge. The value is given in user-defined position units.

Attribute	Value
Index	0x60BC
LCP parameter number	51-61
Name	Touch probe 2 positive edge
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.189 0x60BC: Touch Probe 2 Positive Edge

7.18.6 Parameter 51-64: Touch Probe 2 Negative Edge (0x60BD)

This object provides the position value of touch probe 2 at negative edge. The value is given in user-defined position units.

Attribute	Value
Index	0x60BD
LCP parameter number	51-64
Name	Touch probe 2 negative edge
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32

Attribute	Value
Default value	–

Table 7.190 0x60BD: Touch Probe 2 Negative Edge

7.18.7 Parameters 51-50 and 51-60: Touch Probe Source (0x60D0)

This object provides the source of the touch probe functions.

Value	Definition
0	Reserved.
+1	Touch probe 1 input.
+2	Touch probe 2 input.

Table 7.191 Value Definition for Touch Probe Source

Attribute	Value
Index	0x60D0
Name	Touch probe source
Object code	Array
Data type	INTEGER16
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	51-50
Description	Touch probe 1 source
Access	Read only
Data type	INTEGER16
PDO mapping	No
Value range	See *Table 7.52*.
Default value	+2
Sub-index	0x02
LCP parameter number	51-60
Description	Touch probe 2 source
Access	Read only
Data type	INTEGER16
PDO mapping	No
Value range	See *Table 7.52*.
Default value	+1

Table 7.192 0x60D0: Touch Probe Source

7.18.8 Parameter: First Position (0x3853)

This object defines the start position of the window. The value is given in user-defined position units.

Attribute	Value
Index	0x3853
Name	First position
Object code	Array

Attribute	Value
Data type	INTEGER32
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	–
Description	First position of touch probe 1
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32
Default value	–
Sub-index	0x02
LCP parameter number	–
Description	First position of touch probe 2
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.193 0x3853: First Position

7.18.9 Parameter: Last Position (0x3854)

This object defines the end position of the window. The value is given in user-defined position units.

Attribute	Value
Index	0x3854
Name	Last position
Object code	Array
Data type	INTEGER32
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	–
Description	Last position of touch probe 1
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32
Default value	–
Sub-index	0x02
LCP parameter number	–

Attribute	Value
Description	Last position of touch probe 2
Access	Read/write
Data type	INTEGER32
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.194 0x3854: Last Position

7.18.10 Parameter 51-53: Touch Probe Time Stamp 1 Positive Value (0x60D1)

This object provides the time stamp value of touch probe 1 at positive edge. The value is given in nanoseconds.

Attribute	Value
Index	0x60D1
LCP parameter number	51-53
Name	Touch probe time stamp 1 positive value
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.195 0x60D1: Touch Probe Time Stamp 1 Positive Value

7.18.11 Parameter 51-56: Touch Probe Time Stamp 1 Negative Value (0x60D2)

This object provides the time stamp value of touch probe 1 at negative edge. The value is given in nanoseconds.

Attribute	Value
Index	0x60D2
LCP parameter number	51-56
Name	Touch probe time stamp 1 negative value
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.196 0x60D2: Touch Probe Time Stamp 1 Negative Value

7.18.12 Parameter 51-63: Touch Probe Time Stamp 2 Positive Value (0x60D3)

This object provides the time stamp value of touch probe 2 at positive edge. The value is given in nanoseconds.

Attribute	Value
Index	0x60D3
LCP parameter number	51-63

Attribute	Value
Name	Touch probe time stamp 2 positive value
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.197 0x60D3: Touch Probe Time Stamp 2 Positive Value

7.18.13 Parameter 51-66: Touch Probe Time Stamp 2 Negative Value (0x60D4)

This object provides the time stamp value of touch probe 2 at negative edge. The value is given in nanoseconds.

Attribute	Value
Index	0x60D4
LCP parameter number	51-66
Name	Touch probe time stamp 2 negative value
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–

Table 7.198 0x60D4: Touch Probe Time Stamp 2 Negative Value

7.18.14 Parameter 51-52: Touch Probe 1 Positive Edge Counter (0x60D5)

This object provides a continuous counter that is incremented with each positive edge at touch probe 1. The counter is only valid if the touch probe input is enabled (bit 0 is set to *1* in object 0x60B8).

For single event measuring, only the value of bit 0 is evaluated. For continuous measuring, the value is an unsigned 16-bit value with overflow.

Attribute	Value
Index	0x60D5
LCP parameter number	51-52
Name	Touch probe 1 positive edge counter
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–

Table 7.199 0x60D5: Touch Probe 1 Positive Edge Counter

7.18.15 Parameter 51-55: Touch Probe 1 Negative Edge Counter (0x60D6)

This object provides a continuous counter that is incremented with each negative edge at touch probe 1. The counter is only valid if the touch probe input is enabled (bit 0 is set to *1* in object 0x60B8).

For single event measuring, only the value of bit 0 is evaluated. For continuous measuring, the value is an unsigned 16-bit value with overflow.

Attribute	Value
Index	0x60D6
LCP parameter number	51-55
Name	Touch probe 1 negative edge counter
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–

Table 7.200 0x60D6: Touch Probe 1 Negative Edge Counter

7.18.16 Parameter 51-62: Touch Probe 2 Positive Edge Counter (0x60D7)

This object provides a continuous counter that is incremented with each positive edge at touch probe 2. The counter is only valid if the touch probe input is enabled (bit 8 is set to *1* in object 0x60B8).

For single event measuring, only the value of bit 0 is evaluated. For continuous measuring, the value is an unsigned 16-bit value with overflow.

Attribute	Value
Index	0x60D7
LCP parameter number	51-52
Name	Touch probe 2 positive edge counter
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–

Table 7.201 0x60D7: Touch Probe 2 Positive Edge Counter

7.18.17 Parameter 51-65: Touch Probe 2 Negative Edge Counter (0x60D8)

This object provides a continuous counter that is incremented with each negative edge at touch probe 2. The counter is only valid if the touch probe input is enabled (bit 8 is set to *1* in object 0x60B8).

For single event measuring, only the value of bit 0 is evaluated. For continuous measuring, the value is an unsigned 16-bit value with overflow.

Attribute	Value
Index	0x60D8
LCP parameter number	51-65
Name	Touch probe 2 negative edge counter
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–

Table 7.202 0x60D8: Touch Probe 2 Negative Edge Counter

7.19 Tracing Objects

7.19.1 Parameter: Signal Tracer Control (0x5000)

This object is used to configure and control the tracing of the servo drive internal signals. The object is not visible in the LCP, so the sub-indexes do not have LCP parameter numbers.

Attribute	Value
Index	0x5000
Name	Signal tracer control
Object code	RECORD
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x09
Sub-index	0x01
Description	Trace control flags
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	See *Table 7.204*.
Default value	–
Sub-index	0x02
Description	Trace control
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	0, 1
Default value	–

Attribute	Value
Sub-index	0x03
Description	Trace sample count
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	UNSIGNED16
Default value	–
Sub-index	0x04
Description	Trace subsampling
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	UNSIGNED16
Default value	–
Sub-index	0x05
Description	Trace pre-trigger count
Access	Read/write
Data type	UNSIGNED16
PDO mapping	No
Value range	UNSIGNED16
Default value	–
Sub-index	0x06
Description	Trace trigger variable ID
Access	Read/write
Data type	UNSIGNED32
PDO mapping	No
Value range	See *Table 7.204*.
Default value	–
Sub-index	0x07
Description	Trigger level
Access	Read/write
Data type	FLOAT
PDO mapping	No
Value range	FLOAT
Default value	–
Sub-index	0x08
Description	Trace buffer size
Access	Read only
Data type	UNSIGNED16
PDO mapping	No
Value range	UNSIGNED16
Default value	–
Sub-index	0x09
Description	Trace info flags
Access	Read only
Data type	UNSIGNED32
PDO mapping	No
Value range	UNSIGNED32

Attribute	Value
Default value	–

Table 7.203 0x5000: Signal Tracer Control

Writing *1* to *Trace control* (sub-index 2) starts the trace. Writing *0* aborts a running trace.

Trace sample count (sub-index 3) defines the amount of data to trace; each channel collects the indicated number of samples.

If the process data to be traced is changing slowly, it can be set to sample every Nth time by writing *N* to *Trace subsampling* (sub-index 4). The value 0 is treated like a 1.

Trace pre-trigger count (sub-index 5) specifies how much pre-trigger history the trace will contain. Trigger will not occur until at least the specified number of samples has been recorded.

Trace trigger variable ID (sub-index 6) selects the signal to be used as the trigger source.

Trigger level (sub-index 7) gives the level the signal selected for triggering has to cross to generate the trigger event.

Trace buffer size (sub-index 8) returns the size of the memory set aside by the servo drive for tracing. It may be different on different servo drive sizes. The trace sample count, multiplied by the number of channels must not exceed this value.

Trace info flags (sub-index 9) describes some features relevant for the trace. Only bit 0 is defined and is set to tell the PLC that the trace data per channel is 32 bit wide. See *chapter 9.2.3 Trace Signals* for the list of defined trace signals.

Bit	Name	Definition
3–0	ChannelCount	Specifies how many signals the servo drive should trace (1–8). Cannot be changed while a trace is active.
5, 4	TaskLevel	Specifies the sampling frequency for the trace data. 01: Realtime task 10: Fast task 11: Slow task
6	TriggerSlope	1: rising slope; 0: falling slope
7	Mode	1: data acquisition depends on trigger input; 0: acquisition starts with start command
8	HasTriggered	1 if trigger event has occurred
13–9	Reserved	–
14	DataReady	1 if all requested data has been traced and can be downloaded.
15	Acquiring	1 if data acquisition is in progress. No trace set-up changes are possible during this time.

Table 7.204 Definition of Bits for the Trace Control Flags

7.19.2 Parameter: Signal Trace Channel IDs (0x5001)

This object is used to define the process data signals that will be traced. The object is not visible in the LCP, so the sub-indexes do not have LCP parameter numbers.

Attribute	Value
Index	0x5001
Name	Signal trace channel IDs
Object code	Array
Data type	UNSIGNED32

7

Attribute	Value
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x08
Sub-index	0x01
Description	Trace variable ID 1
Access	Read/write
Data type	UNSIGNED32
PDO mapping	No
Value range	See *chapter 9.2.3 Trace Signals*.
Default value	–

Table 7.205 0x5001: Signal Trace Channel IDs

NOTICE

Sub-indexes 02 to 08 are the same as sub-index 1 as shown in *Table 7.205*. Only the sub-index number increases by 1.

7.19.3 Parameter: Trace Data (0x5002)

This object provides the data after the trace is finished. The samples are 32-bit wide floats in an interleaved way (for example, if the signals A, B, and C are traced, the data looks like this: A0, B0, C0, A1, B1, and so on).

Attribute	Value
Index	0x5002
LCP parameter number	–
Name	Trace Data
Object code	Var
Data type	DOMAIN
Sub-index	0x00
Access	Read only
PDO mapping	No
Value range	DOMAIN
Default value	–

Table 7.206 0x5002: Trace Data

7.19.4 Parameter: Trace Signal Info (0x5004)

This object contains a list of available trace signals. It can be used to verify whether the trace signal exists in the current firmware version of the device.

Attribute	Value
Index	0x5004
LCP parameter number	–
Name	Trace signal info
Object code	Var
Data type	DOMAIN
Sub-index	0x00
Access	Read only
PDO mapping	No

Attribute	Value
Value range	DOMAIN
Default value	–

Table 7.207 0x5004: Trace Signal Info

7.20 Option Code Objects

7.20.1 Parameter 50-41: Fault Reaction Option Code (0x605E)

This object indicates what action is performed when a fault is detected in the servo drive. The slow down ramp is the deceleration value of the used mode of operation.

Attribute	Value
Index	0x605E
LCP parameter number	50-41
Name	Fault reaction option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	0
Default value	0 (Disabled servo drive function, motor is free to rotate.)

Table 7.208 0x605E: Fault Reaction Option Code

7.20.2 Parameter 50-42: Target Reached Option Code (0x2054)

This object indicates how the target reached bit in the *Statusword* (0x6041) is evaluated. Different settings can be used for the different modes of operations: position, velocity, or torque controlled.

Attribute	Value
Index	0x2054
LCP parameter number	50-42
Name	Target reached option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.210*.
Default value	0

Table 7.209 0x2054: Target Reached Option Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved			ss	reserved			tor	reserved			vel	reserved			pos
MSB															LSB

Table 7.210 0x2054: Target Reached Option Code

Legend:
pos = position controlled
vel = velocity controlled
tor = torque controlled
ss = standstill detection

Value	Definition for bits 0, 4, and 8	Definition for bit 12
0	Target reached bit is evaluated based on the actual value using the window parameters.	Standstill bit is evaluated based on the actual value using the window parameters.
1	Target reached bit is evaluated based on the demand value.	Standstill bit is evaluated based on the demanded value.

Table 7.211 Value Definition for Target Reached Option Code

7

7.20.3 Parameter 50-43: Following Error Option Code (0x2055)

This object defines the reaction of the servo drive when the lag window is exceeded. This option code is not supported in csp and csv.

If a following error (lag error) occurs, the used profile velocity may be exceeded. Limit the speed used to try to catch up the lag error in object 0x6080 (see *chapter 7.5.5 Parameter 52-37: Maximum Motor Speed (0x6080)*). Different reactions to a lag error can be set, according to the needs of the application.

Attribute	Value
Index	0x2054
LCP parameter number	50-43
Name	Following error option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.213*.
Default value	0

Table 7.212 0x2055: Following Error Option Code

Value	Definition
0	No reaction, warning is raised.
+1	Ramp down with quick stop ramp and remain in state *Operation enabled*.
+2	Ramp down with quick stop ramp and transition to state *Fault*.
+3	Ramp down with current limit and remain in state *Operation enabled*.
+4	Ramp down with current limit and transition to state *Fault*.
+5	Transition to state *Switch on disabled* (disable servo drive function; motor is free to rotate).

Table 7.213 Definition of Values for Following Error Option Code

7.20.4 Parameter 50-44: Enable in Positioning Option Code (0x2052)

This object indicates what action is performed during a transition from state *Switched on* to state *Operation enabled* occurs. It is only applicable if the current mode of operation is position controlled and the servo drive is not in standstill during the transition.

Attribute	Value
Index	0x2052
LCP parameter number	50-44
Name	Enable in positioning option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.215*.
Default value	0

Table 7.214 0x2052: Enable in Positioning Option Code

Value	Definition
0	Will not switch to state *Operation enabled* unless the servo drive is standstill.
+1	Ramp down with quick stop ramp and transition to state *Operation enabled*.
+2	Continue movement with current velocity. Movement will be infinite or until the limit switch appears.

Table 7.215 Definition of Values for Enable in Positioning Option Code

7.20.5 Parameter 50-45: Abort Connection Option Code (0x6007)

This object indicates what action is performed when the connection to the PLC is interrupted.

Attribute	Value
Index	0x6007
LCP parameter number	50-45
Name	Abort connection option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.217*.
Default value	3

Table 7.216 0x6007: Abort Connection Option Code

Value	Definition
0	No action.
+2	Disable voltage command.
+3	Quick stop command.

Table 7.217 Definition of Values for Enable in Positioning Option Code

7

7.20.6 Parameter 50-46: Quick Stop Option Code (0x605A)

This object indicates what action is performed when the quick stop function is executed. The slow down ramp is the deceleration value of the mode of operation used.

Attribute	Value
Index	0x605A
LCP parameter number	50-46
Name	Quick stop option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.219*.
Default value	2

Table 7.218 0x605A: Quick Stop Option Code

Value	Definition
0	Disable servo drive function.
+1	Slow down with slow down ramp and transition to state *Switch On Disabled*.
+2	Slow down with quick stop ramp and transition to state *Switch On Disabled*.
+3	Slow down with current limit and transition to state *Switch On Disabled*.
+5	Slow down with slow down ramp and remain in state *Quick Stop Active*.
+6	Slow down with quick stop ramp and remain in state *Quick Stop Active*.
+7	Slow down with current limit and remain in state *Quick Stop Active*.

Table 7.219 Definition of Values for Quick Stop Option Code

7.20.7 Parameter 50-47: Halt Option Code (0x605D)

This object indicates what action is performed when the halt function is executed. The slow down ramp is the deceleration value of the mode of operation used.

Attribute	Value
Index	0x605D
LCP parameter number	50-47
Name	Halt option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.221*.
Default value	1

Table 7.220 0x605D: Halt Option Code

Value	Definition
+1	Slow down with slow down ramp and remain in state *Operation enabled*.
+2	Slow down with quick stop ramp and remain in state *Operation enabled*.
+3	Slow down with current limit and remain in state *Operation enabled*.

Table 7.221 Definition of Values for Halt Option Code

7.20.8 Parameter 50-48: Shutdown Option Code (0x605B)

This object indicates what action is performed if a transition from state *Operation enabled* to state *Ready to switch on* occurs.

Attribute	Value
Index	0x605B
LCP parameter number	50-48
Name	Shutdown option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	See *Table 7.223*.
Default value	0

Table 7.222 0x605B: Shutdown Option Code

Value	Definition
–1	Slow down with quick stop ramp; disable of the drive function.
0	Disable drive function (switch-off the drive power stage).
+1	Slow down with slow down ramp; disable of the drive function.

Table 7.223 Definition of Values for Shutdown Option Code

⚠CAUTION

UNAUTHORIZED COMMAND

Using the value *0* for a servo drive with brake stops the servo drive immediately and may result in damage to the holding brake.

- **Do not use the holding brake as active braking.**
- **Change the shutdown option to *-1*.**

7.20.9 Parameter 50-49: Disable Operation Option Code (0x605C)

This object indicates what action is performed if a transition from state *Operation enabled* to state *Switched on* occurs.

Attribute	Value
Index	0x605C
LCP parameter number	50-49
Name	Disable operation option code
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional

Attribute	Value
Value range	See *Table 7.225*.
Default value	1

Table 7.224 0x605C: Disable Operation Option Code

Value	Definition
–1	Slow down with quick stop ramp and disable the servo drive function.
0	Disable servo drive function by switching off the servo drive power stage.
+1	Slow down with slow down ramp and disable the servo drive function.

Table 7.225 Definition of Values for Disable Operation Option Code

7.21 Peripherals

7.21.1 Parameter 16-60: Digital Inputs (0x60FD)

This object provides information about the state of the digital inputs. It represents the physical input levels.

Attribute	Value
Index	0x60FD
LCP parameter number	16-60
Name	Digital inputs
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.227*.
Default value	–

Table 7.226 0x60FD: Digital Inputs

<table>
<tr><td>31</td><td>17</td><td>16</td><td></td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">ms</td><td rowspan="2">–</td><td rowspan="2">hom</td><td rowspan="2">rl</td><td rowspan="2">ll</td></tr>
<tr><td>–</td><td>di2</td><td>di1</td></tr>
<tr><td colspan="4">MSB</td><td colspan="3">LSB</td></tr>
</table>

Table 7.227 0x60FD: Digital Inputs

Legend:
di1: State of input *Digital 1* input.
di2: State of input *Digital 2* input.
ll: State of *Left Limit* input.
rl: State of *Right Limit* input.
hom: State of *Home* input.

Value	Definition
0	Switched off (0 V)
1	Switched on (supply voltage V DC)

Table 7.228 Definition of Values for Digital Inputs

7.21.2 Parameters 16-62 and 16-64: Analog Inputs (0x200D)

This object provides the information on the analog inputs. The values are given in volt.

Attribute	Value
Index	0x200D
Name	Analog inputs
Object code	Array
Data type	UNSIGNED8
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	16-62
Description	Analog input 1
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	–
Sub-index	0x02
LCP parameter number	16-64
Description	Analog input 2
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	–

Table 7.229 0x60FD: Digital Inputs

7.21.3 Parameter: Dual Analog User Inputs Configuration (0x200F)

This object configures the dual analog user inputs.

Each input can be defined as one of the following:

- Analog input (for example it can be used in CAM mode as an analog sensor for alignment).
- Digital input (for example, it can be used in CAM mode as a trigger).
- Left/right limit switch (for example, it can be used in *Homing* mode).
- Homing switch (for example, it can be used in *Homing* mode).
- Touch probe input.

The polarity of the input can be inverted.

15	14	13 12 11	10 9 8	7	6	5 4 3	2 1 0
Polarity input 2	Reserved	Debounce count	Function input 2	Polarity input 1	Reserved	Debounce count	Function input 1
1: Invert	–	Default: 0 (no debounce)	0: Analog 2 1: Digital 2 2: Left limit 3: Right limit 4: Home	1: invert	–	Default: 0 (no debounce)	0: Analog 1 1: Digital 1 2: Left limit 3: Right limit 4: Home
MSB							LSB

Table 7.230 0x200F: Input Configuration Definition

If both inputs are configured as *Left limit* or both inputs are configured as *Right limit*, the logical disjunction (OR) of the input values is evaluated for the limit switches.

If both inputs are configured as *Home*, the logical disjunction (OR) of the input values is evaluated for the home switch.

Other input conflicts are rejected.

The term *Left limit* denotes the negative limit switch, and the term *Right limit* denotes the positive limit switch.

The limit and home switch input must be configured using the polarity bit so that the logical state is high when the limit switch is active (limit is reached).

If an input is configured as digital, it can be used as a touch probe input. It is debounced by using a defined number of samples taken at the cycle time of the PWM frequency. The value of the analog input is given as a physical value.

Attribute	**Value**
Index	0x200F
LCP parameter number	–
Name	Dual analog user inputs config
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	No
Value range	See *Table 7.230*.
Default value	0

Table 7.231 0x200F: Dual Analog User Inputs Config

7.21.4 Parameter 16-66: Digital Outputs (0x60FE)

This object commands simple digital outputs. It represents the physical output levels.

Attribute	**Value**
Index	0x60FE
Name	Digital outputs
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Const

Attribute	Value
PDO mapping	No
Default value	0x01
Sub-index	0x01
LCP parameter number	16-66
Description	Physical outputs
Access	Read/write
Data type	UNSIGNED32
PDO mapping	Optional
Value range	See *Table 7.233*.
Default value	0

Table 7.232 0x60FE: Digital Outputs

31 1	0
reserved	set brake
MSB	LSB

Table 7.233 0x60FE: Digital Outputs

Value	Definition
0	Switch off (do not set brake).
1	Switch on (set brake).

Table 7.234 Definition of Values for Bit 0

NOTICE

The digital output can be read using object 0x2006 (see *chapter 7.22.12 Parameter 50-08: Motion and Input Status (0x2006)*). It can be set in various ways, depending on the configuration of the digital output (see *chapter 7.21.5 Parameter 52-05: Digital Output Configuration (0x2FFF)*).

7.21.5 Parameter 52-05: Digital Output Configuration (0x2FFF)

The digital output can be used to power external sensors, for example an external encoder. It can be switched on or off as follows:

- Manually.
- Automatically by the Digital CAM Switch functionality.
- Automatically by the ISD CAM Mode.

This object is used to set the digital output configuration - it specifies which source should be used for switching the digital output on or off.

Table 7.236 defines the possible digital output configurations.

Attribute	Value
Index	0x2FFF
LCP parameter number	52-05
Name	Digital output configuration
Object code	Var
Data type	UNSIGNED8
Sub-index	0x00
Access	Read/write
PDO mapping	Yes

Attribute	Value
Value range	UNSIGNED16
Default value	0

Table 7.235 0x2FFF: Digital Output Configuration

Value	Definition
0	Always off.
1	Always on.
2	Control over *Controlword* (manufacturer-specific, bit 12).
3	Control over Digital CAM Switch.
4	Control over CAM mode.

Table 7.236 Definition of Values for Digital Output Configuration

7.21.6 External Encoder Objects

7.21.6.1 Parameters 51-30 and 51-34 to 51-40: External Encoder Configuration (0x3000)

These objects configure the external encoder hardware.

Attribute	Value
Index	0x3000
Name	External encoder config
Object code	Array
Data type	Unsigned8
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x08
Sub-index	0x01
LCP parameter number	51-34
Description	External encoder multi-turn bits
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0
Sub-index	0x02
LCP parameter number	51-35
Description	External encoder single-turn bits
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0
Sub-index	0x03
LCP parameter number	51-36
Description	External encoder align bits
Access	Read/write
Data type	UNSIGNED16

Attribute	Value
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0
Sub-index	0x04
LCP parameter number	51-37
Description	External encoder flag bits
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0
Sub-index	0x05
LCP parameter number	51-38
Description	External encoder clock freq mHz
Access	Read/write
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	0
Sub-index	0x06
LCP parameter number	51-40
Description	External encoder crc polynomial
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0
Sub-index	0x07
LCP parameter number	51-39
Description	External encoder gray encoding
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0
Sub-index	0x08
LCP parameter number	51-30
Description	External encoder type
Access	Read/write
Data type	UNSIGNED16
PDO mapping	Optional
Value range	0, 2, 4
Default value	0

Table 7.237 0x3000: External Encoder Configuration

Sub-index 01: External encoder multi-turn bits
Set this object to a value representing the multi-turn resolution of the encoder in bits. Set the value to 0 for a single-turn encoder.

Sub-index 02: External encoder single-turn bits
Set this object to a value representing the single-turn resolution of the encoder in bits.

Sub-index 03: External encoder align bits
Some encoders have fill bits in the data that is transmitted to the servo drive. These fill bits are often used to have the same length of transmit data among encoders with different resolutions. Use the encoder manual to find out if there are any fill bits.

Sub-index 04: External encoder flag bits
Most encoders provide additional error signaling bits in the data that is transmitted to the servo drive. The servo transitions to error state when at least 1 of these bits is not 0. Use the encoder manual to find out if there are any error bits.

Sub-index 05: External encoder clock freq mHz
Some encoder types need a clock from the master, for example SSI from 0.1 MHz up to 2 MHz or BiSS up to 10 MHz. Consult the encoder manual to find out which frequencies can be used. The length and parameters of the cable also have an effect on the maximum usable cable length.

Sub-index 06: External encoder crc polynomial
Currently only used for BiSS encoders. Consult the encoder manual about the polynomial used to calculate the CRC in the encoder. For example, the Acuro BiSS encoder series from Hengstler uses a value of 67.

Sub-index 07: External encoder gray-encoding
Set this value to 1 when the data from the encoder is gray-coded, otherwise leave it as 0.

Sub-index 08: External encoder type
The following values are supported:

0: No encoder

2: BiSS B

4: SSI

7.21.6.2 Parameter 51-32 and 51-33: External Encoder (0x2011)

This object contains information regarding the external encoder.
Sub-index 1 contains the position of the external encoder. The value of this object represents the guide value from 0–1.
Sub-index 2 contains the speed of the external encoder. The value is given in revolutions per second.

Attribute	**Value**
Index	0x2011
Name	External encoder
Object code	Array
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
Data type	UNSIGNED8
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP parameter number	51-32
Description	External encoder position
Access	Read only
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32
Default value	–
Sub-index	0x02
LCP parameter number	51-33
Description	External encoder speed
Access	Read only
Data type	FLOAT

Attribute	Value
PDO mapping	Optional
Value range	FLOAT
Default value	–

Table 7.238 0x2011: External Encoder

7.21.6.3 Parameter 51-31: External Encoder Enable (0x3001)

Enables or disables the external encoder. The value 1 must be written if the external encoder parameters in object 0x3000 (see *chapter 7.21.6.1 Parameters 51-30 and 51-34 to 51-40: External Encoder Configuration (0x3000)*) were changed.

Attribute	Value
Index	0x3001
LCP parameter number	51-31
Name	External encoder enable
Object code	Var
Data type	UNSIGNED8
Sub-index	0x00
Access	Read/write
PDO mapping	Yes
Value range	0, 1
Default value	0

Table 7.239 0x3001: External Encoder Enable

7.22 Monitoring Objects

7.22.1 Following Error Detection Objects

7.22.1.1 Parameter: Following Error Window (0x6065)

This object indicates the configured range of tolerated position values symmetrically to the position demand value. If the position actual value is out of the following error window, a following error occurs. A following error can also occur if a servo drive is blocked, unreachable profile velocity occurs, or if closed-loop coefficients are incorrect.
The value is given in user-defined position units. If the value of the following error window is *0xFFFF FFFF*, the following control is switched off.

Attribute	Value
Index	0x6065
LCP parameter number	–
Name	Following error window
Object code	Var
Data type	UNSIGNED32
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED32
Default value	Corresponding to 5°

Table 7.240 0x6065: Following Error Window

7

7.22.1.2 Parameter: Following Error Time Out (0x6066)

This object indicates the configured time for a following error condition after which bit 13 of the *Statusword* is set to *1*. The reaction of the servo drive when a following error occurs depends on the *Following error option code* (see *chapter 7.20.3 Parameter 50-43: Following Error Option Code (0x2055)*).

The value is given in ms.

Attribute	Value
Index	0x6066
LCP parameter number	–
Name	Following error time out
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	50

Table 7.241 0x6066: Following Error Time Out

7.22.1.3 Parameter: Following Error Actual Value (0x60F4)

This object provides the actual value of the following error.

The value is given in user-defined position units.

Attribute	Value
Index	0x60F4
LCP parameter number	50-05
Name	Following error actual value
Object code	Var
Data type	INTEGER32
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	INTEGER32
Default value	–

Table 7.242 0x60F4: Following Error Actual Value

7.22.2 Standstill Detection Objects

7.22.2.1 Parameter: Velocity Threshold (0x606F)

This object indicates the configured velocity threshold.
The value is given in user-defined velocity units.

Attribute	Value
Index	0x606F
LCP parameter number	–
Name	Velocity threshold
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	6000 (corresponds to 10 RPM)

Table 7.243 0x606F: Velocity Threshold

7.22.2.2 Parameter: Velocity Threshold Time (0x6070)

This object indicates the configured velocity threshold time. The value is given in milliseconds.

Attribute	Value
Index	0x6070
LCP parameter number	–
Name	Velocity threshold time
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	5

Table 7.244 0x6070: Velocity Threshold Time

7.22.3 Constant Velocity Detection Objects

7.22.3.1 Parameter 51-70: Constant Velocity Window (0x2030)

This object indicates the configured symmetrical range of accepted velocity changes relative to the last velocity measured. The value is given in user-defined velocity units.

Attribute	Value
Index	0x2030
LCP parameter number	51-70
Name	Constant velocity window
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	6000 (corresponds to 10 RPM)

Table 7.245 0x2030: Constant Velocity Window

7.22.3.2 Parameter 51-71: Constant Velocity Window Time (0x2031)

This object indicates the configured time during which the velocity changes within the constant velocity window are measured. The value is given in milliseconds.

Attribute	Value
Index	0x2031
LCP parameter number	51-71
Name	Constant velocity window time
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	UNSIGNED16
Default value	20

Table 7.246 0x2031: Constant Velocity Window Time

7.22.4 Parameters 15-40, 15-41, and 15-43: Version log (0x4000)

This object contains all the information regarding the versioning of the firmware.

Attribute	Value
Index	0x4000
Name	Version log
Object code	Array
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
Data type	UNSIGNED8
PDO mapping	No

Attribute	Value
Default value	0x06
Sub-index	0x01
LCP Param number	15-40
Description	Application type
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 7.248*.
Default value	–
Sub-index	0x02
LCP Param number	15-41
Description	Software type
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	See *Table 7.249*.
Default value	–
Sub-index	0x03
LCP Param number	15-43 (major, minor, beta, build)
Description	SW Version major
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–
Sub-index	0x04
LCP Param number	15-43 (major, minor, beta, build)
Description	SW Version minor
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–
Sub-index	0x05
LCP Param number	15-43 (major, minor, beta, build)
Description	SW Version beta
Access	Read only
Data type	UNSIGNED16
PDO mapping	Optional
Value range	UNSIGNED16
Default value	–
Sub-index	0x06
LCP Param number	15-43 (major, minor, beta, build)
Description	Build number
Access	Read only
Data type	UNSIGNED32
PDO mapping	Optional
Value range	UNSIGNED32

7

Attribute	Value
Default value	–

Table 7.247 0x4000: Version Log

Sub-index 01: Application type
This object shows the application type for this device.

Application type	Value
784	ISD 510 servo drive POWERLINK®
800	ISD 510 servo drive EtherCAT®
1040	SAB POWERLINK®
1056	SAB EtherCAT®

Table 7.248 Application Type

Sub-index 02: Software type
This object shows the software type for firmware package for this device.

Software type	Value
768	ISD 510 servo drive
1024	SAB

Table 7.249 Software Type

Sub-index 03: SW Version major
This object shows the software major version for the firmware package for this device.

Sub-index 04: SW Version minor
This object shows the software minor version for the firmware package for this device.

Sub-index 05: SW Version beta
This object shows the software beta version for the firmware package for this device. Any value other than 0 means that it is not a publicly released firmware.

Sub-index 06: Build number
This object shows the software build number for the firmware package for this device.

7.22.5 Parameter 15-51: Serial String (0x4004)

This object contains the serial number for the device as a string.
The serial string uniquely identifies the device.

Attribute	Value
Index	0x4004
Name	Serial string
Object code	Var
Data type	VISIBLE_STRING
Sub-index	0x00
LCP parameter number	15-51
Access	Read only
PDO mapping	Yes
Value range	VISIBLE_STRING
Default value	–

Table 7.250 0x4004: Serial String

7.22.6 Parameters 12-00 to 12-05: Communication Settings (0x400A)

This object can be used to control the IP settings.
In most cases the PLC/master controls and writes these objects.

Sub-index 0x01:
Not applicable for Ethernet POWERLINK®.
0: IP communication (EoE) disabled.
1: Manually IP addresses configured.

Sub-index 0x02:
IP address for the device.
For Ethernet POWERLINK®: 192.168.100.NODE_ID (where NODE_ID is the node ID for the Ethernet POWERLINK® slave).

Sub-index 0x03:
Not applicable for Ethernet POWERLINK®.
IP mask for the device.

Sub-index 0x04:
Not applicable for Ethernet POWERLINK®.
IP gateway for the device.

Sub-index 0x08:
MAC address of the device.

Attribute	Value
Index	0x400A
Name	Communication settings
Object code	Record
Data type	COMM_SETTINGS
Sub-index	0x00
Access	Read only
PDO mapping	Yes
Value range	0x00–0xFF
Default value	13
Sub-index	0x01
LCP Parameter number	12-00 EtherCAT® only: 0: Disabled 1: Manually configured IP
Name	IP configuration
Data type	UNSIGNED8
Access	Read only
PDO mapping	Yes
Value range	0x00–0xFF
Default value	0
Sub-index	0x02
LCP Parameter number	12-01 Additionally for Ethernet POWERLINK® 12-60 (last number of the IP address)
Name	IP address
Data type	UNSIGNED32
Access	Read only
PDO mapping	Yes
Value range	– [maximum 200 characters]
Default value	0xC0A864EF

Attribute	Value
Sub-index	0x03
LCP Parameter number	12-02
Name	IP mask
Data type	UNSIGNED32
Access	Read only
PDO mapping	Yes
Value range	0x0000 0000–0xFFFF FFFF
Default value	0
Sub-index	0x04
LCP Parameter number	12-03
Name	IP gateway
Data type	UNSIGNED32
Access	Read only
PDO mapping	Yes
Value range	0x0000 0000–0xFFFF FFFF
Default value	–
Sub-index	0x08
LCP Parameter number	12-04
Name	Mac address
Data type	OCTET_STRING
Access	Read only
PDO mapping	Yes
Value range	–
Default value	–

Table 7.251 0x400A: Communication Settings

7.22.7 Parameters 15-01 and 15-02: Total Running Time (0x5807)

The servo drive has 2 counters for the running time.
The 1st object is a non-resettable counter (32 bits). It contains the time in seconds that the servo drive has been running (enabled). This means that the counter is running as long as the servo drive is in state *Switched on*, *Operation enabled*, or *Quick stop active*.
The 2nd object contains the time in seconds that the servo drive has been running (enabled). This means that the counter is running as long as the servo drive is in state *Switched on*, *Operation enabled*, or *Quick stop active*.
This 2nd counter is resettable by writing *0* to it, so it can be used to track service intervals.

Attribute	Value
Index	0x5807
Name	Total running time
Object code	Array
Data type	UNSIGNED32
Sub-index	0x00
Description	Value of highest sub-index
Access	Read only
PDO mapping	No
Default value	0x02
Sub-index	0x01
LCP Parameter number	15-01
Description	Total running time drive

Attribute	Value
Access	Read only
Data type	UNSIGNED32
PDO mapping	No
Value range	UNSIGNED32
Default value	–
Sub-index	0x02
LCP Parameter number	15-02
Description	Total running time user
Access	Read/write
Data type	UNSIGNED32
PDO mapping	No
Value range	UNSIGNED32
Default value	–

Table 7.252 0x5807: Total Running Time

7.22.8 Parameter 50-09: STO Voltage and Brake Status (0x2007)

Information about the current state of the STO or the brake can be read from this object.

Attribute	Value
Index	0x2007
LCP parameter number	50-09
Name	STO voltage and brake status
Object code	var
Data type	UNSIGNED8
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.254*.
Default value	–

Table 7.253 0x2007: STO Voltage and Brake Status

7	6	5	4	3	2	1	0
Brake status				STO activated			
MSB							LSB

Table 7.254 0x2007: STO Voltage and Brake Status

Bit	Value	Definition
7–4	0	No brake connected.
	1	Released, shaft is blocked.
	2	Boosting, shaft is being released.
	3	Lifted, shaft is free.
	4	WaitRelease, shaft is becoming locked.
	Otherwise	Reserved.
3–0	0	STO activated.
	1	STO voltage is present; normal operation is possible.
	Otherwise	Reserved.

Table 7.255 Definition of Bits of the STO Voltage and Brake Status

7.22.9 Parameter 15-30: Error Code (0x603F)

This object provides the error code of the last error that occurred in the servo drive.

Attribute	Value
Index	0x603F
LCP parameter number	15-30
Name	Error code
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *chapter 9.2.1 Troubleshooting*
Default value	–

Table 7.256 0x603F: Error Code

7.22.10 Parameter 16-92: Warning Code (0x5FFE)

This object provides the warning code of the last warning that occurred in the servo drive.

Attribute	Value
Index	0x5FFE
LCP parameter number	16-92
Name	Warning code
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *chapter 9.2.2 Error Codes*
Default value	–

Table 7.257 0x5FFE: Warning Code

7.22.11 Parameter: Control Source (0x5020)

This object defines the current control source.
0 means that the device is controlled via fieldbus (remote/PLC).
1 means that the device is controlled via LCP.

Attribute	Value
Index	0x5020
LCP parameter number	–
Name	Control source
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read/write
PDO mapping	Optional
Value range	0, 1

Attribute	Value
Default value	0

Table 7.258 0x5020: Control Source

7.22.12 Parameter 50-08: Motion and Input Status (0x2006)

This object contains the motion status (independent on the current mode of operation), and the state of the input switches (independent of the configuration of the digital inputs).

Attribute	Value
Index	0x2006
LCP parameter number	50-08
Name	Motion and input status
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.260*.
Default value	–

Table 7.259 0x2006: Motion and Input Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tp2	tp1	hom	rl	ll	di2	di1	do	nLim	pLim	neg	pos	st	con	dec	acc
MSB															LSB

Table 7.260 0x2006: Motion and Input Status

Digital input:
di1: State of input *Digital 1* input.
di2: State of input *Digital 2* input.
ll: State of *Left Limit* input.
rl: State of *Right Limit* input.
hom: State of *Home* input.
tp1: State of *Touch probe 1* input (object 0x60B9, bit 1 or 2).
tp2: State of *Touch probe 2* input (object 0x60B9, bit 9 or 10).
The state is coded as 0 = inactive; 1 = active for all different kinds of inputs. If no such input is configured, the value is set to *0*.
Digital output:
do: State of the digital output
The state is coded as 0 = inactive; 1 = active. It is not possible to change the state of the digital output using this object.

Software position limits:
pLim: Positive software limit reached (see *chapter 2.3.4.2 Software Position Limit*)
nLim: Negative software limit reached (see *chapter 2.3.4.2 Software Position Limit*)

Motion status:
acc: The servo drive is accelerating (increasing the absolute value of the velocity); Is determined based on the actual value.
dec: The servo drive is decelerating (decreasing the absolute value of the velocity); Is determined based on the actual value.
con: Velocity is constant. The velocity can be 0.
st: Velocity is constant. The velocity is 0.
pos: Positive direction: The position is increasing.
neg: Negative direction: The position is decreasing.

7.22.13 Parameter 50-07: Overlaying Motion Status (0x2005)

This object contains the status of all overlaying motions or functionalities (regardless of the current mode of operation).

Attribute	Value
Index	0x2005
LCP parameter number	50-07
Name	Overlaying motion status
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	See *Table 7.262*.
Default value	–

Table 7.261 0x2005: Overlaying Motion Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												Digital CAM switching			
r												dcsE	r	dcsV	dcsS
MSB															LSB

Table 7.262 Definition of Bitfield

Digital CAM switching functionality (see *chapter 2.5.1 Digital CAM Switch*):
dcsS: State of Digital CAM switching functionality (0 = inactive; 1 = active).
dcsE: Error in Digital CAM switching functionality (for example, an attempt was made to enable the Digital CAM switching functionality with invalid switches definition).
dcsV: Validity of the digital CAM switching data (0 = invalid: Digital CAM switching functionality cannot be activated; 1 = valid).
r: Reserved.

7.22.14 Parameter: Physical Limits (0x5100)

This object contains information needed for scaling of data that is given relative to these values.

Attribute	Value
Index	0x5100
Name	Physical limits
Object code	Array
Sub-index	0x00
Description	Value of highest sub-index
Access	Const
Data type	UNSIGNED8
PDO mapping	No
Default value	0x05
Sub-index	0x01
LCP Parameter number	–
Description	Measurement limit current
Access	Read only
Data type	FLOAT
PDO mapping	Optional

Attribute	Value
Value range	FLOAT
Default value	–
Sub-index	0x02
LCP Parameter number	–
Description	Measurement limit current
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	–
Sub-index	0x03
LCP Parameter number	50-24
Description	Maximum configured speed
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	Depends on motor size
Sub-index	0x04
LCP Parameter number	50-26
Description	Number of pole pairs
Access	Read only
Data type	UNSIGNED8
PDO mapping	Optional
Value range	FLOAT
Default value	–
Sub-index	0x05
LCP Parameter number	–
Description	Measurement limit brake current
Access	Read only
Data type	FLOAT
PDO mapping	Optional
Value range	FLOAT
Default value	–

Table 7.263 0x5100: Physical Limits

Sub-index 01: Measurement Limit Current
This object contains the maximum measurable current. The value is given in Ampere.

Sub-index 02: Measurement Limit Voltage
This object contains the maximum measurable voltage. The value is given in Volt.

Sub-index 03: Maximum Configured Speed
This object contains the maximum configured speed. The value is given in revolutions per minute.

Sub-index 04: Number of Pole Pairs
This object contains the number of pole pairs.

Sub-index 05: Measurement Limit Brake Current
This object contains the maximum measurable brake current. The value is given in Ampere.

7.22.15 Voltage Objects

7.22.15.1 Parameter 16-30: DC Link Voltage (0x2003)

This value provides the measured DC-link voltage. The value is given in Volt.

Attribute	Value
Index	0x2003
LCP parameter number	16-30
Name	DC link voltage
Object code	Var
Data type	INTEGER16
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	–
Default value	–

Table 7.264 0x2003: DC Link Voltage

7.22.15.2 Parameter 50-06: Auxiliary Voltage (0x200E)

This object provides the voltage of U_{AUX}. The value is given in Volt.

Attribute	Value
Index	0x200E
LCP parameter number	50-06
Name	Auxiliary voltage
Object code	Var
Data type	FLOAT
Sub-index	0x00
Access	Read only
PDO mapping	Optional
Value range	–
Default value	–

Table 7.265 0x200E: Auxiliary Voltage

7.22.16 Parameter 16-19, 16-31, 16-34, 16-39: Temperature (0x2000)

This object contains several internal temperatures of the device. All the values are given in °C.

Sub-index 1 contains the temperature of the module.
Sub-index 2 contains the temperature from the internal sensor 1.
Sub-index 3 contains the temperature from the internal sensor 2.
Sub-index 4 contains the temperature of the motor winding.

Attribute	Value
Index	0x2000
Name	Temperature
Object code	Array
Sub-index	0x00
Description	Value of highest sub-index

Attribute	Value
Access	Const
Data type	UNSIGNED8
PDO mapping	No
Default value	0x04
Sub-index	0x01
LCP Parameter number	16-34
Description	Temperature module
Access	Read only
Data type	INTEGER16
PDO mapping	Optional
Value range	INTEGER16
Default value	–
Sub-index	0x02
LCP Parameter number	16-39
Description	Temperature PCB 1
Access	Read only
Data type	INTEGER16
PDO mapping	Optional
Value range	INTEGER16
Default value	–
Sub-index	0x03
LCP Parameter number	16-31
Description	Temperature PCB 2
Access	Read only
Data type	INTEGER16
PDO mapping	Optional
Value range	INTEGER16
Default value	–
Sub-index	0x04
LCP Parameter number	16-19
Description	Temperature wire motor
Access	Read only
Data type	INTEGER16
PDO mapping	Optional
Value range	INTEGER16
Default value	–

Table 7.266 0x2000: Temperature

8 SAB Parameter Description

This chapter provides information on all user parameters that are accessible via the object dictionary.

8.1 Object 0x4040: Controlword

This object indicates the received command controlling the SAB state machine. Possible state transitions are described in *chapter 3.2 Control*.

Attribute	Value
Index	0x4040
Description	Controlword
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	50-10
Access	Read/write
PDO mapping	Yes
Default value	0

Table 8.1 Object 0x4040: Controlword

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	10
Function	–	–	–	r2	r1	–	–	–	fr	–	–	–	–	ev	–	eva

Table 8.2 Structure of Object 0x4040

Table 8.3 provides a description of the bits.

Bit	Description
eva	U_{AUX} 1 and 2 enable 0 = On 1 = Off
ev	UDC 1 and 2 enable 0 = Off 1 = On
fr	Fault reset (carried out when changed from 0 to 1)
r1	Relay 1 enable 0 = Open 1 = Closed
r2	Relay 2 enable 0 = Open 1 = Closed

Table 8.3 Controlword Bits

8.2 Object 0x4041: Statusword

This object provides the status of the state machine and some information bits.

Attribute	Value
Index	0x4041
Description	Statusword
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	50-11
Access	Read only
PDO mapping	Yes
Default value	0

Table 8.4 Object 0x4041: Statusword

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	10
Function	mains	–	–	–	–	–	rm	–	w	–	–	pu	f	oe	so	rtso

Table 8.5 Structure of Object 0x4041

Table 8.6 provides a description of the bits.

Bit	Description
rtso	Ready to switch on (U_{AUX} is disabled).
so	Switched on (U_{AUX} is enabled and UDC can be switched on).
oe	Operation enabled (UDC lines 1 and 2 are enabled).
f	Fault (Alarm has tripped).
pu	Power-up.
w	Warning.
rm	Remote controlled.
mains	Mains is applied.

Table 8.6 Statusword Bits

The encoding of the SAB states is defined as:

States	Statusword bits				
	4 (pu)	3 (f)	2 (oe)	1 (so)	0 (rtso)
Init	0	0	0	0	0
U_{AUX} disabled	0	0	0	0	1
Standby	0	0	0	1	1
Power-up	1	0	0	1	1
Operation enabled	0	0	1	1	1
Fault	0	1	0	0	0

Table 8.7 SAB States

8.3 Object 0x2000: SAB Temperatures

This object shows the temperature measured on the power card, control card, and SAB card in °C.

Attribute	Value
Index	0x2000
Description	Temperature
Object code	Array
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x03
Default value	0x03
Data type	UNSIGNED16
Sub-index	0x01
LCP parameter number	16-34
Description	Temperature power card
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	SIGNED16
Unit	°C
Sub-index	0x02
LCP parameter number	16-39
Description	Temperature control card
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	SIGNED16
Unit	°C
Sub-index	0x03
LCP parameter number	16-31
Description	Temperature SAB card
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	SIGNED16
Unit	°C

Table 8.8 Object 0x2000: SAB Temperatures

8.4 Object 0x2001: DC-link Related Values

This object shows several DC-link voltages and currents.

Attribute	Value
Index	0x2001
Description	DC-link related values
Object code	Record

Attribute	Value
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x07
Default value	0x07
Sub-index	0x01
LCP parameter number	51-73
Description	DC-link total current
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere
Sub-index	0x02
LCP parameter number	–
Description	UDC 1 current
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere
Sub-index	0x03
LCP parameter number	–
Description	UDC 2 current
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere
Sub-index	0x04
LCP parameter number	16-30
Description	DC-link voltage
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	SIGNED16
Unit	Volt
Sub-index	0x05
LCP parameter number	51-74
Description	DC leakage current
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere
Sub-index	0x06
LCP parameter number	–
Description	DC-link voltage readout

8

Attribute	Value
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	SIGNED16
Unit	Volt
Sub-index	0x07
LCP parameter number	51-71
Description	UDC 1 current readout
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere
Sub-index	0x08
LCP parameter number	51-72
Description	UDC 2 current readout
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere

Table 8.9 Object 0x2001: DC-link Related Values

8.5 Object 0x2003: U_{AUX} Related Values

This object shows values related to the AUX voltage. Use sub-indexes 04 and 05 to limit the current or set them to *0* to disable the limiting. A warning is issued when 90% of the set current limit is exceeded. Once 100% of the set current limit is exceeded, the line is switched off and the SAB transitions to state *Error*.

Attribute	Value
Index	0x2003
Description	U_{AUX} related values
Object code	Record
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x05
Default value	0x05
Sub-index	0x01
LCP parameter number	51-81
Description	AUX line voltage
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Volt

Attribute	Value
Sub-index	0x02
LCP parameter number	51-82
Description	AUX line 1 current
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere
Sub-index	0x03
LCP parameter number	51-83
Description	AUX line 2 current
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	REAL
Unit	Ampere
Sub-index	0x04
LCP parameter number	–
Description	AUX line 1 user current limit
Access	Read/write
PDO mapping	Yes
Object code	Var
Data type	UNSIGNED16
Unit	0.1 Ampere
Value range	0–150
Default value	0
Sub-index	0x05
LCP parameter number	–
Description	AUX line 2 user current limit
Access	Read/write
PDO mapping	Yes
Object code	Var
Data type	UNSIGNED16
Unit	0.1 Ampere
Value range	0–150
Default value	0

Table 8.10 Object 0x2003: U_{AUX} Related Values

8.6 Object 0x2008: ISD Power Consumption

This object shows the power consumed by the servo drives connected to the SAB, averaged over the last 60 s.

Attribute	Value
Index	0x2008
Description	ISD power consumption
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00

Attribute	Value
LCP parameter number	16-10
Access	Read only
PDO mapping	Yes
Unit	Watt

Table 8.11 Object 0x2008: ISD Power Consumption

8.7 Object 0x2009: Fan Speed Power Card

This object shows the speed of the fan on the power card as a percentage value (0–100%).

Attribute	Value
Index	0x2009
Description	Fan speed power card
Object code	Var
Data type	UNSIGNED16
Unit	%
Sub-index	0x00
LCP parameter number	51-61
Access	Read only
Value range	0–100
PDO mapping	Yes

Table 8.12 Object 0x2009: Fan Speed Power Card

8.8 Object 0x200D: Relay 1 Control

This object configures the function of relay 1 with the on and off delay times. The valid configuration values are detailed in *Table 3.2* in *chapter 3.2.1 Relay Outputs*.

Attribute	Value
Index	0x200D
Description	Relay 1 Config
Object code	Record
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x03
Default value	0x03
Sub-index	0x01
LCP parameter number	05-40
Description	Relay 1 Config
Access	Read/write
PDO mapping	No
Object code	Var
Data type	UNSIGNED16
Value range	See *Table 3.2* in *chapter 3.2.1 Relay Outputs*.
Default value	0
Sub-index	0x02
LCP parameter number	05-41
Description	Relay 1 on delay
Access	Read/write
PDO mapping	No
Object code	Var
Data type	UNSIGNED16
Unit	Seconds
Value range	0–600
Default value	0
Sub-index	0x03
LCP parameter number	05-42
Description	Relay 1 off delay
Access	Read/write
PDO mapping	No
Object code	Var
Data type	UNSIGNED16
Unit	Seconds
Value range	0–600
Default value	0

Table 8.13 Object 0x200D: Relay 1 Control

8.9 Object 0x200E: Relay 2 Control

This object configures the function of relay 2 with the on and off delay times. The valid configuration values are detailed in *Table 3.2* in *chapter 3.2.1 Relay Outputs*.

Attribute	Value
Index	0x200E
Description	Relay 2 Config
Object code	Record
Sub-index	0x00
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x03
Default value	0x03
Sub-index	0x01
LCP parameter number	05-40
Description	Relay 2 Config
Access	Read/write
PDO mapping	No
Object code	Var
Data type	UNSIGNED16
Value range	See *Table 3.2* in *chapter 3.2.1 Relay Outputs*.
Default value	0
Sub-index	0x02
LCP parameter number	05-41

Attribute	Value
Description	Relay 2 on delay
Access	Read/write
PDO mapping	No
Object code	Var
Data type	UNSIGNED16
Unit	Seconds
Value range	0–600
Default value	0
Sub-index	0x03
LCP parameter number	05-42
Description	Relay 2 off delay
Access	Read/write
PDO mapping	No
Object code	Var
Data type	UNSIGNED16
Unit	Seconds
Value range	0–600
Default value	0

Table 8.14 Object 0x200E: Relay 2 Control

8.10 Object 0x2030: Brake Control

8.10.1 Object 0x2030: Brake Control

This object enables the brake resistor on the SAB when the value is set to *1*.

Attribute	Value
Index	0x2030
Description	Brake control
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	02-10
Access	Read/write
PDO mapping	Yes
Value range	0–1
Default value	0

Table 8.15 Object 0x2030: Brake Control

8.11 Object 0x2031: Brake Resistor

This object sets the brake resistor value that is used for power limit monitoring. Only SAB device-specific range limits are accepted, all others are rejected with an *SDO_OUT_OF_RANGE* message.

Attribute	Value
Index	0x2031
Description	Brake resistor
Object code	Var
Data type	REAL
Sub-index	0x00
LCP parameter number	02-11
Access	Read/write
PDO mapping	No
Value range	REAL
Default value	0
Unit	Ohm

Table 8.16 Object 0x2031: Brake Resistor

8.12 Object 0x2032: Brake Resistor Power Limit

This object sets the power limit for the resistor. This value is used for the power limit monitoring function (see *chapter 8.13 Object 0x2033: Brake Resistor Power Monitoring*.

Attribute	Value
Index	0x2032
Description	Brake resistor power limit
Object code	Var
Data type	REAL
Sub-index	0x00
LCP parameter number	02-12
Access	Read/write
PDO mapping	No
Value range	REAL
Default value	0
Unit	Watt

Table 8.17 Object 0x2032: Brake Resistor Power Limit

8.13 Object 0x2033: Brake Resistor Power Monitoring

This object enables the power limit monitoring for the SAB.

Attribute	Value
Index	0x2033
Description	Brake resistor power monitoring
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	02-13
Access	Read/write
PDO mapping	No
Value range	0–3 (see *Table 8.19*)
Default value	0

Table 8.18 Object 0x2033: Brake Resistor Power Monitoring

0	No brake power monitoring.
1	Warning at 90%, no alarm.
2	Alarm at 100%, no warning.
3	Warning at 90% and alarm at 100%.

Table 8.19 Valid values for Object 0x2033

8.14 Object 0x2034: Brake Check

This object enables the brake check.

Attribute	Value
Index	0x2034
Description	Brake check
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	02-15
Access	Read/write
PDO mapping	No
Value range	0–2 (see *Table 8.21*)
Default value	0

Table 8.20 Object 0x2034: Brake Check

0	No brake check.
1	Warning on failure.
2	Alarm on failure.

Table 8.21 Valid values for Object 0x2034

8.15 Object 0x2035: Brake Duty Cycle Monitoring

This object shows the current duty cycle of the brake in %.

Attribute	Value
Index	0x2035
Description	Brake check
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	02-14
Access	Read only
PDO mapping	No
Value range	0–100
Default value	0
Unit	%

Table 8.22 Object 0x2035: Brake Duty Cycle Monitoring

8.16 Object 0x2036: Brake Resistor Power 120 s

This object holds the calculated brake power of the last 120 s.

Attribute	Value
Index	0x2036
Description	Brake resistor power 120 s
Object code	Var
Data type	REAL
Sub-index	0x00
LCP parameter number	02-16
Access	Read only
PDO mapping	No
Value range	REAL
Default value	0
Unit	Ws (Watt seconds)

Table 8.23 Object 0x2036: Brake Resistor Power 120 s

8

8.17 Object 0x2062: Position Guide Value Reference

This SAB object is identical to the drive object (see *chapter 7.9.1 Parameter: Position Guide Value Reference (0x2062)*).

8.18 Object 0x2063: Guide Value Reference Option Code

This SAB object is identical to the drive object (see *chapter 7.9.3 Parameter: Guide Value Reference Option Code (0x2063)*). However, in the SAB object, only the sources *External encoder* and *Encoder simulation* are available.

8.19 Object 0x2065: Velocity Guide Value Reference

This SAB object is identical to the drive object (see *chapter 7.9.2 Parameter: Velocity Guide Value Reference (0x2065)*).

8.20 Object 0x2068: Position Guide Value Reference Set

This SAB object is identical to the drive object (see *chapter 7.9.4 Parameter: Position Guide Value Reference Set (0x2068)*).

8.21 Object 0x2070: Guide Value Reference Simulation Control

This SAB object is identical to the drive object (see *chapter 7.9.6.1 Parameter: Guide Value Reference Simulation Control (0x2070)*).

8

8.22 Object 0x2071: Guide Value Reference Speed Limit

This SAB object is identical to the drive object (see *chapter 7.9.6.2 Parameter: Guide Value Reference Speed Limit (0x2071)*).

8.23 Object 0x2072: Guide Value Reference Target Velocity

This SAB object is identical to the drive object (see *chapter 7.9.6.3 Parameter: Guide Value Reference Target Velocity (0x2072)*).

8.24 Object 0x2073: Guide Value Reference Acceleration

This SAB object is identical to the drive object (see *chapter 7.9.6.4 Parameter: Guide Value Reference Acceleration (0x2073)*).

8.25 Object 0x2074: Guide Value Reference Deceleration

This SAB object is identical to the drive object (see *chapter 7.9.6.5 Parameter: Guide Value Reference Deceleration (0x2074)*).

8.26 Object 0x3000: External Encoder Configuration

This SAB object (and all of its sub-objects) is identical to the drive object (see *chapter 7.21.6.1 Parameters 51-30 and 51-34 to 51-40: External Encoder Configuration (0x3000)*).

8.27 Object 0x3001: External Encoder Enable

This SAB object is identical to the drive object (see *chapter 7.21.6.3 Parameter 51-31: External Encoder Enable (0x3001)*).

8.28 Object 0x4004: Serial String

This SAB object is identical to the drive object (see *chapter 7.22.5 Parameter 15-51: Serial String (0x4004)*).

8.29 Object 0x400A: Communication Settings

This object can be used to control the IP settings.
Usually the PLC/master controls and writes these objects.

Sub-index 0x01:
Not applicable for Ethernet POWERLINK®.
0: IP communication (EoE) disabled.
1: IP addresses configured manually.

Sub-index 0x02:
IP address for the device.
For Ethernet POWERLINK®: 192.168.100.NODE_ID (where NODE_ID is the node ID for the Ethernet POWERLINK® slave).

Sub-index 0x03:
Not applicable for Ethernet POWERLINK®.
IP mask for the device.

Sub-index 0x04:
Not applicable for Ethernet POWERLINK®.
IP gateway for the device.

Sub-index 0x08:
MAC address of the device.

Sub-index 0x0D:
Topology configuration of the device.
0: Line topology mode.
1: Ring topology mode (needed for ring redundancy network configurations).
For details on the wiring, see the *VLT® Integrated Servo Drive ISD 510 System Operating Instructions*.

Attribute	Value
Index	0x400A
Name	Communication settings
Object code	Record
Data type	COMM_SETTINGS
Sub-index	0x00
Access	Read only
PDO mapping	Yes
Value range	0x00–0xFF
Default value	13
Sub-index	0x01
LCP Parameter number	12-00 EtherCAT® only: 0: Disabled 1: Manually configured IP
Name	IP configuration
Data type	UNSIGNED8
Access	Read only
PDO mapping	Yes
Value range	0x00–0xFF
Default value	0
Sub-index	0x02
LCP Parameter number	12-01 Additionally for Ethernet POWERLINK® 12-60 (last number of the IP address)
Name	IP address
Data type	UNSIGNED32
Access	Read only
PDO mapping	Yes
Value range	– [maximum 200 characters]
Default value	0xC0A864EF

Attribute	Value
Sub-index	0x03
LCP Parameter number	12-02
Name	IP mask
Data type	UNSIGNED32
Access	Read only
PDO mapping	Yes
Value range	0x0000 0000–0xFFFF FFFF
Default value	0
Sub-index	0x04
LCP Parameter number	12-03
Name	IP gateway
Data type	UNSIGNED32
Access	Read only
PDO mapping	Yes
Value range	0x0000 0000–0xFFFF FFFF
Default value	–
Sub-index	0x08
LCP Parameter number	12-04
Name	Mac address
Data type	OCTET_STRING
Access	Read only
PDO mapping	Yes
Value range	–
Default value	–
Sub-index	0x0D
LCP Parameter number	–
Name	Network topology
Data type	UNSIGNED8
Access	Read/write
PDO mapping	No
Value range	0, 1
Default value	0

Table 8.24 0x400A: Communication Settings

8.30 Object 0x5020: Control Source

This object shows the control source for the SAB.

- Value 0: Fieldbus is in control (writes are only accepted from the fieldbus).
- Value 1: The LCP is in control (writes are only accepted from the LCP).

Attribute	Value
Index	0x5020
Description	Control source
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	–
Access	Read/write
PDO mapping	Yes
Value range	0 or 1
Default value	0

Table 8.25 Object 0x5020: Control Source

8.31 Object 0x5807: Total Running Time

The SAB has 2 counters for the running time:

- Non-resettable counter (32 bits)
    - Counts the time that the SAB has been running in state *Normal operation* in seconds.
- Resettable counter
    - Counts the time that the SAB has been running in state *Normal operation* in seconds.
    - Can be reset by writing 0 to it and can therefore be used to track service intervals.

Attribute	Value
Index	0x5807
Description	Total running time
Object code	Array
Sub-index	0x00
LCP parameter number	–
Description	Number of entries
Access	Read only
PDO mapping	No
Value range	0x02
Default value	0x02
Sub-index	0x01
LCP parameter number	15-01
Description	Total running time SAB
Access	Read only
PDO mapping	Yes
Object code	Var
Data type	UNSIGNED32
Sub-index	0x02
LCP parameter number	15-02
Description	Total running time user
Access	Read/write
PDO mapping	Yes
Object code	Var
Data type	UNSIGNED32

Table 8.26 Object 0x5807: Total Running Time

8

8.32 Object 0x503F: Error Code

This object shows the latest reported/detected error code.
If no errors are detected, it shows *0*.

Attribute	Value
Index	0x503F
Description	Error code
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	16-90
Access	Read/write
PDO mapping	Yes
Value range	See error codes in *chapter 9.3.2 Warnings and Alarms*.
Default value	0

Table 8.27 Object 0x503F: Error Code

8.33 Object 0x5FFE: Warning Code

This object provides the warning code of the last warning that occurred in the SAB.

Attribute	Value
Index	0x5FFE
Description	Warning code
Object code	Var
Data type	UNSIGNED16
Sub-index	0x00
LCP parameter number	19-92
Access	Read only
PDO mapping	Yes
Value range	See warning codes in *chapter 9.3.2 Warnings and Alarms*.
Default value	0

Table 8.28 Object 0x5FFE: Warning Code

9 Diagnostics

9.1 System Monitoring

A warning or an alarm is signaled by the relevant indicator light on the device and indicated by an error code. If faults occur during servo system operation, check:

- The LEDs on the servo drive for general problems relating to communication or device status.
- The LEDs on the SAB for general problems with communication, auxiliary supply, or STO voltage.

The error codes can be read using the ISD Toolbox software, the LCP, or the PLC. The LCP only shows faults relating to the device it is connected to.

A warning remains active until its cause is no longer present, however the operation of the device may still be continued. Warning messages may be critical, but are not necessarily so. In the event of an alarm, the device goes to *Fault* state.

Reset the alarm to resume operation. Reset alarms that are not trip-locked using 1 of these 3 methods:

- Using the [Reset] key on the LCP.
- Using the PLC function block MC_Reset_ISD51x or DD_Reset_SAB.
- Using the ISD Toolbox.

If an alarm cannot be reset, the reason may be that the cause has not been rectified, or the alarm is trip-locked (see *Table 9.2*) and (*Table 9.5*).

Alarms that are trip-locked offer extra protection, meaning that the supply voltage must be switched off before the alarm can be reset. After being switched back on, the device is no longer blocked and can be operated normally.

Use Wireshark® if there are network problems. Wireshark® is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, and software and communications protocol development. It can be downloaded via the Wireshark® webpage (*wireshark.org*).

NOTICE

If the fault cannot be eliminated by 1 of the measures listed in *Table 9.1* or *Table 9.4*, notify Danfoss Service.

Have the following information available to enable Danfoss to provide help quickly and effectively:

- Type number
- Error code
- Firmware version
- System set-up (for example, number of servo drives and lines).

9.2 Drive

9.2.1 Troubleshooting

First use *Table 9.1* to check the possible causes of the fault and possible solutions. The error codes are listed in *chapter 9.2.2 Error Codes*.

Fault	Possible cause	Possible solution
LCP display dark or has no function.	Missing input power.	Check the input power source.
	Missing or open fuses or circuit breaker tripped.	Check the fuses and circuit breaker.
	No power to the LCP.	• Check the LCP cable for proper connection or damage. • Replace any faulty LCP or connection cables.
	Incorrect contrast setting.	Press [Status] + [▲]/[▼] to adjust the contrast.
	Display is defective.	Replace the faulty LCP or connection cable.
Servo drive overheats (high surface temperature).	Excessive load.	Check the torques.
Servo drive not running.	No drive communication or drive in error mode.	Check the fieldbus connection and the status indicators (LEDs) on the servo drive.

Fault	Possible cause	Possible solution
Servo drive does not run or only starts up slowly or with difficulty.	• Bearing wear. • Incorrect parameter settings. • Incorrect control loop parameters. • Incorrect torque settings.	• Check the bearings and shaft. • Check the parameter settings.
Drive hums and draws high current.	Drive defective.	Contact Danfoss.
Drive stops suddenly and does not restart.	• No drive communication. • Servo drive in error mode.	Check the fieldbus connection and the status indicators (LEDs) on the servo drive.
Wrong motor rotation direction.	Parameter error.	• Check the parameter settings. • Change the rotation direction if appropriate.
Drive runs normally, but does not generate the expected torque.	• Drive defective. • Parameter error.	• Check the parameter settings. • Contact Danfoss.
Drive screaming.	• Incorrect calibration. • Faulty current measurement. • Incorrect control loop parameters.	• Check the parameter settings. • Contact Danfoss.
Uneven running.	Defective bearing.	Check the shaft.
Vibration.	• Defective bearing. • Incorrect control loop parameters.	• Check the shaft. • Check the parameter settings.
(Unusual) running noises	• Defective bearing. • Defects on connected mechanics. • Incorrect control loop parameters.	• Check the shaft. • Check for loose mechanical components on the attached mechanics. • Check the parameter settings.
System fuse blows, circuit breaker trips, or drive protection trips immediately.	• Short circuit. • Incorrect control loop parameters.	• Check the wiring. • Contact Danfoss.
Drive speed drops sharply under load.	• Drive is running at current limit. • Drive is running with incorrect parameters.	• Check the application. • Check the parameter settings.
Brake does not release.	Brake control defective.	Contact Danfoss.
Holding brake does not hold the servo drive.	• Mechanical brake defective. • Shaft load exceeds the holding torque of the brake.	Contact Danfoss.
Brake engagement delayed.	Software error.	Contact Danfoss.
Noises when power-off brake engaged.	Mechanical brake damaged.	Contact Danfoss.
LEDs do not light up.	No power supply.	Check the power supply.
Error 0xFF91 occurs.	Increments between succeeding values too big.	Check for velocity or guide value plausi-bility distance.

Table 9.1 Troubleshooting Servo Drive

9.2.2 Error Codes

Code	Name	Severity (Warning/ Error/Trip lock)	Description	LCP name
0x0000	No error	Error	No error.	–
0x1000	Generic application error	Error	Generic application error.	generic err
0x2310	Overcurrent on output	Error	Overcurrent on output.	overcurr out
0x239B	Overload on output (I2T)	Warning, error	I^2t thermal state.	overload
0x3210	DC link overvoltage	Error	Overvoltage on DC-link voltage	UDC overvolt
0x3220	DC link undervoltage	Error	Undervoltage on DC-link voltage.	UDC undervolt

Code	Name	Severity (Warning/ Error/Trip lock)	Description	LCP name
0x4290	Overtemperature: Power module	Error	Overtemperature on power module.	overtemp PM
0x4291	Overtemperature: Control card	Error	Overtemperature on control PCB.	overtemp CC
0x4295	Overtemperature: Power card	Error	Overtemperature on power PCB.	overtemp PC
0x4310	Overtemperature: Motor	Error	Overtemperature on motor.	overtemp motor
0x5112	UAUX undervoltage	Error, trip lock	Undervoltage on auxiliary voltage.	undervolt UAUX
0x5530	EE Checksum Error (parameter missing)	Trip lock	Missing parameter in internal drive configuration.	config err
0x6320	Parameter error	Trip lock	An internal parameter has an invalid value.	param err
0x7320	Internal position sensor error	Trip lock	Absolute position sensor error.	int sensor err
0x7380	External position sensor error	Error	External encoder data could not be read.	ext sensor err
0x8611	Following error	Warning, error	A following error has occurred.	following err
0x8693	Homing error on entering homing mode	Warning	Could not enter homing mode (for example velocity not 0).	Homing mode fail
0x8694	Homing error on start homing method	Warning	Could not start homing method (for example drive not in standstill).	Homing method fail
0x8695	Homing error distance	Warning	Homing distance reached.	Homing distance

Code	Name	Severity (Warning/ Error/Trip lock)	Description	LCP name
0xFF01	Mechanical brake failure	Trip lock	No brake or wire failure.	brake mech fail
0xFF02	Short circuit in mechanical brake control	Trip lock	Short circuit in brake control.	brake mech short
0xFF0A	External interface power failure	Error	External interface power supply failure.	ext IF pwr fail
0xFF60	Timing violation 1	Trip lock	Contact Danfoss.	timing err 1
0xFF61	Timing violation 2	Trip lock	Contact Danfoss.	timing err 2
0xFF62	Timing violation 3	Trip lock	Contact Danfoss.	timing err 3
0xFF63	Timing violation 4	Trip lock	Contact Danfoss.	timing err 4
0xFF64	Timing violation 5	Trip lock	Contact Danfoss.	timing err 5
0xFF65	Timing violation 6	Trip lock	Contact Danfoss.	timing err 6
0xFF70	Firmware: Package description mismatch	Trip lock	Firmware found does not match the package description.	FW pack err
0xFF71	Firmware: Power cycle needed	Warning, error	Firmware update transfer is completed but a power cycle is required before the new firmware is active.	need powercycle
0xFF72	Firmware: Update started	Warning, error	Firmware update in progress. The warning becomes an error when an attempt is made to enable the drive in this state.	FW update

9

Code	Name	Severity (Warning/ Error/Trip lock)	Description	LCP name
0xFF80	STO active while drive enabled	Error	STO activated while servo drive was enabled or tried to enable while STO active.	STO active
0xFF81	STO mismatch	Trip lock	Dual diagnosis of STO voltage not plausible.	STO mismatch
0xFF85	P_STO error	Trip lock	P_STO voltage on power card not within limits.	P_STO error
0xFF90	Guide value reversed	Error	Position guide value went backwards while servo drive in CAM mode.	guide val rev
0xFF91	Guide value implausible	Error	Increments between succeeding values too big.	guide val impl

Table 9.2 Error Codes for Servo Drive

9

9.2.3 Trace Signals

Signal ID (hex)	Signal name	PLC library - trace constant name	Description	OD object
00000001	Phase current U	ISD51x_TRC_ICTRL_IU	Phase current U	–
00000002	Phase current V	ISD51x_TRC_ICTRL_IV	Phase current V	–
00000003	Phase current W	ISD51x_TRC_ICTRL_IW	Phase current W	–
00000006	Current Id act	ISD51x_TRC_ICTRL_ID_ACT	Actual current - field component	–
00000007	Current Iq act	ISD51x_TRC_ICTRL_IQ_ACT	Actual current - torque component	–
00000015	DC-link voltage	ISD51x_TRC_VOLTAGE_DC	DC-link voltage	0x2003
00000021	Speed loop integral	ISD51x_TRC_NCTRL_INTEGRAL	I-gain of speed loop	–
00000022	Modes of operation	ISD51x_TRC_MODES_OF_OP	Modes of operation	0x6060
00000023	Modes of operation display	ISD51x_TRC_MODES_OF_OP_DISPLAY	Modes of operation display	ox6061
00000024	Rotor mech angle	ISD51x_TRC_ROTOR_POS	Rotor mechanical angle, compensated for resolver mounting to pole	–
00000027	Position guide value	ISD51x_TRC_GUIDE_VALUE_POSITION	Position guide value	0x2060
00000028	Guide value extrapolated	ISD51x_TRC_GUIDE_VALUE_EXTRAPOL	Guide value extrapolated	–
00000029	Guide value raw	ISD51x_TRC_GUIDE_VALUE_RAW	Guide value raw	–
0000002A	Guide value MT	–	Guide value multiple table revolutions	–
0000002B	Torque FF	ISD51x_TRC_NCTRL_TORQUE_FF	Torque feedforward	–
0000002C	Guide value scaled	ISD51x_TRC_GUIDE_VALUE_SCALED	Guide value scaled	–
0000002D	Guide value MT counter	–	Guide value MT counter	–
0000002E	Guide value scaled 2	–	Guide value scaled 2	–
0000002F	Velocity guide value	ISD51x_TRC_GUIDE_VALUE_SPEED	Velocity guide value	0x2064
00000030	Active segment ID	ISD51x_TRC_CAM_SEGMENT_ID	Active segment ID	0x2019
00000031	Logical cam position	ISD51x_TRC_CAM_POS	Logical cam position	0x2020
00000033	Pattern sensor act even	ISD51x_TRC_PAT_SENS_ACT_EVEN	Actual pattern sensor value – even	–
00000034	Pattern sensor act odd	ISD51x_TRC_PAT_SENS_ACT_ODD	Actual pattern sensor value – odd	–
00000035	Pattern sensor corr even	ISD51x_TRC_PAT_SENS_CORR_EVEN	Value of pattern sensor correlation – even	–
00000036	Pattern sensor corr odd	ISD51x_TRC_PAT_SENS_CORR_ODD	Value of pattern sensor correlation – odd	–

Signal ID (hex)	Signal name	PLC library - trace constant name	Description	OD object
00000037	Pattern sensor status	ISD51x_TRC_PAT_SENS_STATUS	Status of pattern sensor mark search	–
00000038	Temperature wire motor	–	Temperature wire motor	0x2000.4
00000039	I^2t energy	–	I^2t energy	–
0000003D	Warning code	ISD51x_TRC_WARNING	Warning code	0x5FFE
0000003E	Error code	ISD51x_TRC_ERROR	Error code	0x603F
00000040	Torque set ICtrl	ISD51x_TRC_ICTRL_TORQUE_SET	Torque setpoint to the current controller	–
00000051	Temperature PCB 1	–	Temperature from internal sensor 1	0x2000.2
00000052	Temperature PCB 2	–	Temperature from internal sensor 2	0x2000.3
00000053	Auxiliary voltage	ISD51x_TRC_VOLTAGE_UAUX	Auxiliary voltage (U_{AUX})	0x200E
00000054	Last node ID	ISD51x_TRC_CAM_CURRENT_NODE	Last node ID	0x201A
00000056	Active cam profile	ISD51x_TRC_CAM_ACTIVE_PROFILE	Active profile of cam	–
00000057	Cam target reached	ISD51x_TRC_CAM_TARGET_REACHED	Target reached in cam profile	–
00000063	Guide value relative	ISD51x_TRC_GUIDE_VALUE_REL_MASTR	Guide value relative	–
00000064	Cam control code	–	Cam control code	0x3800.1
00000065	Cam control param 1	–	Cam control parameter 1	0x3800.2
00000066	Cam control param 2	–	Cam control parameter 2	0x3800.3
00000067	Cam control param 3	–	Cam control parameter 3	0x3800.4
00000068	Cam status code	ISD51x_TRC_CAM_STAT_CODE	Cam Status Code	0x3801.1
00000069	Cam status param 1	ISD51x_TRC_CAM_STAT_PARAM_1	Cam status parameter 1	0x3801.2
0000006A	Cam status param 2	ISD51x_TRC_CAM_STAT_PARAM_2	Cam status parameter 2	0x3801.3
0000006B	Cam status param 3	ISD51x_TRC_CAM_STAT_PARAM_3	Cam status parameter 3	0x3801.4
00004001	Current Id set	ISD51x_TRC_ICTRL_ID_SET	Setpoint current of field component	–
00004002	Current Iq set	ISD51x_TRC_ICTRL_IQ_SET	Setpoint current of torque component	–
0000400D	Ud set	ISD51x_TRC_ICTRL_IU	Ud set	–
0000400E	Uq set	–	Uq set	–
00004015	Analog input 1	ISD51x_TRC_ANALOG_INPUT_1	Analog input 1	0x200D.1
00004016	Analog input 2	ISD51x_TRC_ANALOG_INPUT_2	Analog input 2	0x200D.2
00004017	Digital output	ISD51x_TRC_DIGITAL_OUTPUT	Digital output	–
00004018	STO state	ISD51x_TRC_STO_STATE	STO state	–
00004019	External encoder position	ISD51x_TRC_EXT_ENCODER_POSITION	External encoder position	0x2011.1
0000401A	External encoder speed	ISD51x_TRC_EXT_ENCODER_SPEED	External encoder speed	0x2011.2
00004027	V stator duty cycle	ISD51x_TRC_ICTRL_USTAT_DUTY_CYCL	Stator voltage duty cycle	–
00004028	Resolver sin	ISD51x_TRC_RESOLVER_SIN	Resolver sine signal	–
00004029	Resolver cos	ISD51x_TRC_RESOLVER_COS	Resolver cosine signal	–
00004043	Mech brake I act	–	Actual mechanical brake current	–
00004044	Cam I set	–	Current pre-control cam mode	–
00004045	Mech brake I set	–	Setpoint of mechanical brake current	–
0000404C	Observed velocity	ISD51x_TRC_VELOCITY_OBSERVER	Actual observed velocity	–
0000404D	PD position ctrl P	ISD51x_TRC_PCTRL_KP	PD position controller P-part	–
0000404E	PD position ctrl D	ISD51x_TRC_PCTRL_KD	PD position controller D-part	–
0000404F	PID speed ctrl P	ISD51x_TRC_NCTRL_KP	PID speed controller P-part	–
00004050	PID speed ctrl I	ISD51x_TRC_NCTRL_KI	PID speed controller I-part	–
00004051	PID speed ctrl D	ISD51x_TRC_NCTRL_KD	PID speed controller D-part	–
00004052	PID speed ctrl inertia	ISD51x_TRC_NCTRL_INERTIA	PID speed controller inertia	–
80000001	Rotor velocity	ISD51x_TRC_ROTOR_N_ACT	Rotor actual velocity	–
80000002	Rotor angle act	ISD51x_TRC_ROTOR_ANGLE_ACT	Rotation angle of motor shaft	–
80000003	Speed set	ISD51x_TRC_NCTRL_N_SET	Speed setpoint - input for speed controller	–
80000005	Target position	ISD51x_TRC_DS402_TARGET_POSITION	Position setpoint in user-defined units	0x607A
80000006	Target velocity	ISD51x_TRC_DS402_TARGET_VELOCITY	Velocity setpoint in user-defined units	0x60FF
80000007	Target torque	ISD51x_TRC_DS402_TARGET_TORQUE	Torque setpoint in ‰ of rated torque	0x6071

9

Signal ID (hex)	Signal name	PLC library - trace constant name	Description	OD object
80000008	Rotor position logical	ISD51x_TRC_ROTOR_POS_ACT_VIRT	Logical rotation position of motor shaft	–
80000009	Following error	ISD51x_TRC_PCTRL_LAG_ERROR	Following error (difference between act value and setpoint)	–
8000000A	Rotor mech angle raw	ISD51x_TRC_ROTOR_POS_RAW	Raw mechanical angle, not compensated for resolver mounting to pole, not mirrored	–
8000000B	Rotor speed relative	–	Rotor speed relative	–
8000000E	Position ctrl set	ISD51x_TRC_PCTRL_POS_SET	Setpoint of position control loop	–
8000000F	Logical CAM setpoint	ISD51x_TRC_CAM_POS_SET	Position setpoint in cam mode	0x2021
80000010	Cam velocity set	ISD51x_TRC_CAM_N_SET	Speed setpoint in cam mode	–
80000012	Controlword	ISD51x_TRC_CONTROLWORD	Controlword	0x6040
80000013	Statusword	ISD51x_TRC_STATUSWORD	Statusword	0x6041
80000014	Motion and input status	ISD51x_TRC_MOTION_AND_INPUT	Motion and input status	0x2006
80000015	Cam profile status	ISD51x_TRC_CAM_PROFILE_STATUS	Cam profile status	0x3085
80000016	Logical CAM setpoint scaled	–	Position setpoint in cam mode after slave scaling	–
80000017	CAM friction torque	ISD51x_TRC_CAM_FRICTION_COMP	Cam friction compensation torque	–
80000020	Position actual value	ISD51x_TRC_POSITION_ACTUAL_VALUE	Actual position in user-defined units	0x6064
80000021	Velocity actual value	ISD51x_TRC_DS402_VELOCITY_ACT_VAL	Actual velocity in user-defined units	0x606C
80000022	Torque actual value	ISD51x_TRC_TORQUE_ACTUAL_VALUE	Actual torque ‰ of rated torque	0x6077
80000023	Following error actual value	ISD51x_TRC_DS402_FOLLOW_ERR	Following error in user-defined units	0x60F4
80000025	Position guide value reference	ISD51x_TRC_GUIDE_REF_VAL_POS	Position guide value reference	0x2062
80000026	Velocity guide value reference	ISD51x_TRC_GUIDE_REF_VAL_SPEED	Velocity guide value reference	0x2065

Table 9.3 Trace Signals for Servo Drive

9.3 SAB

9.3.1 Troubleshooting

Table 9.4 lists potential faults on the SAB, their possible causes, and actions for correcting the faults.

Fault	Possible cause	Possible solution
LCP display dark or has no function.	Missing input power.	Check the input power source.
	Missing or open fuses or circuit breaker tripped.	Check the fuses and circuit breaker.
	No power to the LCP.	• Check the LCP cable for proper connection or damage. • Replace any faulty LCP or connection cables.
	Incorrect contrast setting.	Press [Status] + [▲]/[▼] to adjust the contrast.
	Display is defective.	Replace the faulty LCP or connection cable.
Open power fuses or circuit breaker trip.	Phase-to-phase short.	• Check the cabling. • Check for loose connections.
DC-link voltage too high.	Brake resistor not connected.	Check the brake resistor cabling.
	Brake resistor too high resistance.	Check if the lowest resistance value has been entered.
	Several servo drives are decelerating with insufficient ramp time.	• Avoid simultaneous deceleration of several servo drives. • Change the deceleration speed of the servo drives.
	Brake resistor functionality not activated.	Activate the brake function.

Fault	Possible cause	Possible solution
DC-link voltage too low.	Incorrect mains supply.	Check supply voltage matches the allowed specification detailed in chapter *ISD Safety Concept* in the *VLT® Integrated Servo Drive ISD® 510 System Operating Instructions*.
DC overcurrent.	The sum of the servo drive current exceeds the maximum rating of the SAB.	• Check the servo drive current consumption. • Avoid simultaneous acceleration of all servo drives.
U_{AUX} overcurrent.	The servo drives are consuming more power on the U_{AUX} line than allowed.	• Check the number of attached servo drives with the shell diagrams in the *VLT® Integrated Servo Drive ISD® 510 System Design Guide.* • Avoid simultaneous lifting of the servo drive brakes.
U_{AUX} overvoltage.	Incorrect U_{AUX} supply.	Check that the supply matches the allowed specification detailed in chapter *Electrical Installation* in the *VLT® Integrated Servo Drive ISD® 510 System Operating Instructions*.
U_{AUX} undervoltage.	Incorrect U_{AUX} supply.	• Check that the supply matches the allowed specification detailed in chapter *Electrical Installation* in the *VLT® Integrated Servo Drive ISD® 510 System Operating Instructions*. • Check that the output power of the supply is sufficient.
Mains phase loss.	A phase is missing on the supply side, or the voltage imbalance is too high.	Check the supply voltages and supply currents to the SAB.
Grounding fault.	Grounding fault.	• Check for proper grounding and loose connections. • Check the hybrid cables for short circuits or leakage currents.
Brake resistor error.	Faulty brake resistor.	Remove the power to the SAB, wait for the discharge time to elapse, then replace the brake resistor.
Brake chopper error.	Faulty brake chopper.	Check the setting in parameter *2-15 Brake Check*.

Table 9.4 Troubleshooting SAB

9

9.3.2 Warnings and Alarms

Code	Name	Severity (Warning/ error/trip lock)	Description	LCP name
0x0000	No error	Error	No error.	–
0x1000	Generic application error	Error	Generic application error.	generic err
0x2120	Ground fault	Error	There is current from the output phases to ground.	ground fault
0x2340	Short circuit	Error	There is a short circuit in UDC output from SAB (DC Line1 and/or DC Line2). Remove power to the SAB and repair the short circuit.	short circuit
0x2391	AUX 1 overcurrent	Error	Current on AUX Line 1 reached overcurrent limit.	AUX1 overcurr
0x2392	AUX 2 overcurrent	Error	Current on AUX Line 2 reached overcurrent limit.	AUX2 overcurr
0x2393	AUX 1 user limit current	Warning, error	Current on AUX Line 1 reached user-defined limit.	AUX1 curr limit
0x2394	AUX 2 user limit current	Warning, error	Current on AUX Line 2 reached user-defined limit.	AUX2 curr limit
0x2395	AUX 1 fuse failure	Error	HW fuse failure. Current or voltage above limit on AUX Line 1.	AUX1 fuse fail
0x2396	AUX 2 fuse failure	Error	HW fuse failure. Current or voltage above limit on AUX Line 2.	AUX2 fuse fail
0x2397	DC 1 overcurrent	Error	Overcurrent on DC Line 1. The SAB peak current limit (approximately 200% of the rated current) is exceeded.	DC1 overcurr
0x2398	DC 2 overcurrent	Error	Overcurrent on DC Line 2. The SAB peak current limit (approximately 200% of the rated current) is exceeded.	DC2 overcurr
0x2399	DC overcurrent	Error	Overcurrent. The SAB has reached the current limit and shuts down to prevent any damage to the hardware.	DC overcurr
0x239B	Overload on output (I2T)	Warning, error	The SAB is about to cut out due to an overload (more than 100% for too long). The counter for electronic, thermal SAB protection triggers a warning at 90% and trips with an error at 100%.	overload
0x239D	DC overcurrent	Warning, error	Overcurrent. The SAB has reached the current limit and shuts down to prevent any damage to the hardware.	DC overcurr
0x3130	Mains phase loss	Warning, error	Mains phase loss detected. This occurs when a phase on mains is missing, or when the mains is imbalanced.	phase loss
0x3210	DC link overvoltage	Error	The DC-link voltage exceeds the limit and the SAB trips.	UDC overvolt
0x3220	DC link undervoltage	Error	The DC-link voltage is below the limit and the SAB trips.	UDC undervolt
0x3291	U_{AUX} high voltage	Warning	U_{AUX} above warning limit.	UAUX high volt
0x3292	U_{AUX} overvoltage	Error	U_{AUX} above overvoltage limit.	UAUX overvolt

Code	Name	Severity (Warning/ error/trip lock)	Description	LCP name
0x3293	U_{AUX} low voltage	Warning	U_{AUX} below warning limit.	UAUX low volt
0x3294	U_{AUX} undervoltage	Error	U_{AUX} below undervoltage limit.	UAUX undervolt
0x3295	UDC high voltage	Warning	The DC-link voltage (DC) is higher than the high-voltage warning limit.	UDC high volt
0x3296	UDC low voltage	Warning	The DC-link voltage (DC) is lower than the low-voltage warning limit.	UDC low volt
0x4220	Too low temperature: Heat sink	Warning	Heat sink temperature low. The SAB is too cold to operate. This warning is based on the temperature sensor in the IGBT module. This warning only occurs when DC-link voltage is >250 V.	low temp PM
0x4290	Overtemperature: Heat sink	Warning, Error	The maximum temperature of the heat sink has been exceeded. The temperature fault does not reset until the temperature drops below a defined heat sink temperature (115 °C).	overtemp PM
0x4291	Overtemperature: Control card	Warning, Error	Control card overtemperature. The cutout temperature of the control card is 80 °C.	overtemp CC
0x4292	Overtemperature: SAB card	Warning, Error	SAB card overtemperature. The cutout temperature of the SAB card is 80 °C.	overtemp SC
0x4293	Inrush overtemperature: SAB card	Error	Inrush fault. Too many transitions into state *Normal operation* have occurred within a short time period.	inrush SC
0x4294	Inrush overtemperature: power module	Error	Inrush fault. Too many power-ups have occurred within a short time period.	inrush PM
0x4410	Overtemperature: SAB	Error	Logic OR of control card temperature (see 0x4291) and/or heat sink temperature (see 0x4290) and/or SAB card temperature (see 0x4292).	overtemp SAB
0x6320	Parameter error	Trip lock	A parameter has an invalid value.	param err
0x6380	Configuration error (parameter missing)	Trip lock	A parameter is missing.	config err
0x6381	Reinitialization of parameters from power card	Trip lock	Configuration reinitialization. Configuration parameter for power unit has been reinitialized.	config reinit
0x7111	Brake chopper short circuit	Error	The brake chopper is monitored during operation. This error appears if a short circuit occurs.	brake ch short
0x7181	Brake resistor failure	Error	The brake resistor is monitored during operation. This error appears if a short circuit occurs.	brake r short

Code	Name	Severity (Warning/ error/trip lock)	Description	LCP name
0x7182	Brake resistor power limit	Error	Brake resistor power limit exceeded. The power transmitted to the brake resistor is calculated as an average value over the last 120 s of run time. The calculation is based on the DC-link voltage and the brake resistor value set in parameter *2-16* (Brake resistor power 120 s). The error is reported when the value is exceeded within 120 s.	brake r pwr lim
0x7183	Brake chopper check failed	Error	Brake check failed. The brake resistor is not connected or not working.	brake ch check
0x7380	External position sensor error	Error	External encoder data could not be read.	ext sensor err
0xFF21	Internal fan fault	Warning	Internal fan fault. The fan warning function checks if the fan is running/ mounted.	fan fault
0xFF31	AUX Line 1 min off time	Warning	The minimum off time required to protect the internal hardware has not been met.	AUX1 min off
0xFF32	AUX Line 2 min off time	Warning	The minimum off time required to protect the internal hardware has not been met.	AUX2 min off
0xFF51	Internal error 1	Trip lock	Internal error 1, contact Danfoss.	PM int err 1
0xFF52	Internal error 2	Trip lock	Internal error 2, contact Danfoss.	PM int err 2
0xFF53	Internal error 3	Trip lock	Internal error 3, contact Danfoss.	PM int err 3
0xFF54	Internal error 4	Trip lock	Internal error 4, contact Danfoss.	PM int err 4
0xFF55	Internal error 5	Trip lock	Internal error 5, contact Danfoss.	PM int err 5
0xFF56	Internal error 6	Trip lock	Internal error 6, contact Danfoss.	PM int err 6
0xFF70	Firmware: Package description mismatch	Trip lock	Firmware found does not match package description.	FW pack err
0xFF71	Firmware: Power cycle needed	Warning, error	Firmware update transfer is completed but a power cycle is required before the new firmware is active.	need powercycle
0xFF72	Firmware: Update started	Warning, error	Firmware update in progress. The warning becomes an error when an attempt is made to enable the drive in this state.	FW update

Table 9.5 Error Codes for SAB

9.3.3 Trace Signals

Signal ID (hex)	Signal name	PLC library - trace constant name	Description	OD object
00000015	DC-link voltage	SAB_TRC_V_DC	DC-link voltage	0x 2003
0000003D	Warning code	SAB_TRC_WARNING	Warning code	0x 5FFE
0000003E	Error code	SAB_TRC_ERROR	Error code	0x 603F
0000005A	Temperature control card	–	Temperature control card	0x 2000.2
0000005B	Temperature power card	–	Temperature power card	0x 2000.1
0000005C	Temperature SAB card	–	Temperature SAB card	0x 2000.3
00004019	External encoder position	SAB_TRC_EXT_ENC_POSITION	External encoder position	0x 2011.1
0000401A	External encoder speed	SAB_TRC_EXT_ENC_SPEED	External encoder speed	0x 2011.2
0000404D	UDC 1 current	SAB_TRC_I_DC_LINE_1	Current flow on DC link line 1	0x 2001.2
0000404E	UDC 2 current	SAB_TRC_I_DC_LINE_2	Current flow on DC link line 2	0x 2001.3
0000404F	AUX line 1 current	SAB_TRC_I_AUX_LINE_1	Current on AUX Line 1	0x 2003.2
00004050	AUX line 2 current	SAB_TRC_I_AUX_LINE_2	Current on AUX Line 2	0x 2003.3
00004051	AUX line voltage	SAB_TRC_UAUX	AUX line voltage	0x 2003.1
00004052	Brake Chopper Gate	SAB_TRC_BRAKE_CH_GATE	Brake chopper gate	–
00004053	Brake Chopper Feedback	SAB_TRC_BRAKE_CH_FEEDB	Brake chopper feedback	–
00007D00	UDC Over-Inrush	SAB_TRC_I_INRUSH	UDC current over-inrush	0x 2002.5
00007D01	UDC Bypass-Inrush	SAB_TRC_I_NORMAL	UDC current bypass-inrush	0x 2002.6
00007D02	UDC Back Current	SAB_TRC_I_BACK	Current (link voltage) back from drives	0x 2002.7
00007D03	Inrush relay power card	SAB_TRC_INRUSH_RELAY_PC	Inrush relay power card	–
00007D04	Inrush relay SAB card	SAB_TRC_INRUSH_RELAY_SC	Inrush relay SAB card	–
00007D05	DC leakage current	SAB_TRC_I_DC_LEAKAGE	DC leakage current	0x 2001.5
00007D06	Brake resistor power monitoring	SAB_TRC_BRAKE_PWR_MON	Brake resistor power monitoring	0x 2033
00007D07	DC link total current	SAB_TRC_I_DC_SUM	DC link total current	0x 2001.1
00007D08	DC link total current raw	SAB_TRC_I_DC_SUM_RAW	DC link total current (unfiltered)	
00007D09	UDC 1 flow (filtered)	SAB_TRC_I_DC_LINE_1_FILT	Current flow on DC link line 1 (filtered)	0x 2001.7
00007D0A	UDC 2 flow (filtered)	SAB_TRC_I_DC_LINE_2_FILT	Current flow on DC link line 2 (filtered)	0x 2001.8
80000012	Controlword	SAB_TRC_CONTROLWORD	Controlword	0x 6040
80000013	Statusword	SAB_TRC_STATUSWORD	Statusword	0x 6041
80010025	Position guide value reference	SAB_TRC_POS_GUIDE_REF	Position guide value reference	0x 2062
80010026	Velocity guide value reference	SAB_TRC_VEL_GUIDE_REF	Velocity guide value reference	0x 2065

Table 9.6 Trace Signals for SAB

9.4 Operating Status Indicators

The operating status of the servo drive and SAB is indicated via the LEDs on each device.

9.4.1 Operating LEDs on the Servo Drive

Illustration 9.1 shows the operating LEDs on the servo drive.

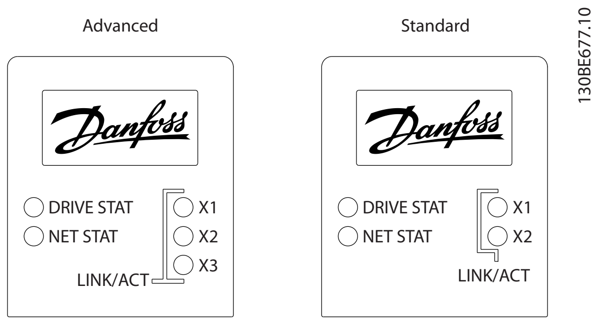


Illustration 9.1 Operating LEDs on the Servo Drive

LED	Color	Flash status	Description
DRIVE STAT	Green	On	Servo drive is in state *Operation enabled.*
		Flashing	Auxiliary voltage is applied.
	Red	On	Servo drive is in *Fault* or *Fault reaction active* state.
		Flashing	DC-link voltage is not applied.
NET STAT	Green/ red	Fieldbus dependent	Network status of the device (see corresponding fieldbus standard).
Link/A CT X1	Green	–	Link/activity status of *Hybrid In* (X1)
		On	Ethernet link established.
		Flashing	Ethernet link established and active.
		Off	No link.
Link/A CT X2	Green	–	Link/activity status of *Hybrid Out* (X2)
		On	Ethernet link established.
		Flashing	Ethernet link established and active.
		Off	No link.
Link/A CT X3[1]	Green	–	Link/activity status of the Ethernet port (X3).
		On	Ethernet link established.
		Flashing	Ethernet link established and active.
		Off	No link.

Table 9.7 Legend to *Illustration 9.1*

1) Advanced version only

9.4.2 Operating LEDs on the Servo Access Box

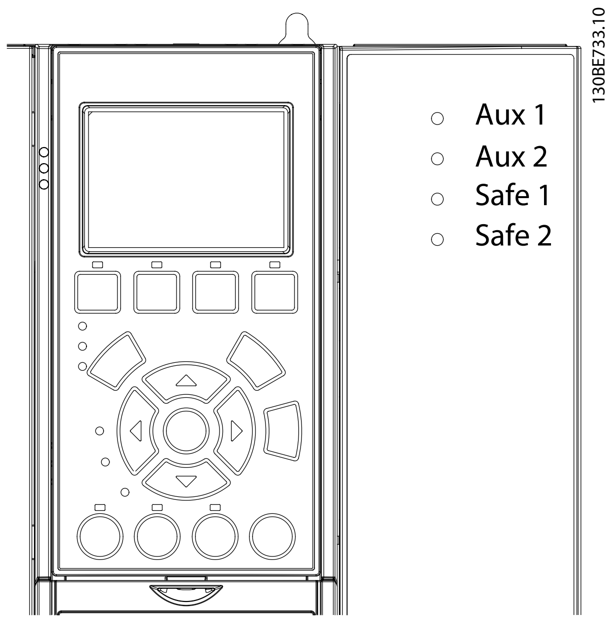


Illustration 9.2 Operating LEDs on the SAB

LED	Color	Flash status	Description
Aux 1	Green	–	State of the auxiliary voltage on line 1.
		On	State machine is in state *Standby*, *Power up*, or *Operation enabled*. Auxiliary voltage is applied to the output connectors on line 1.
		Off	State machine is in state U_{AUX} *disabled* or *Fault*. Auxiliary voltage is not applied to line 1.

LED	Color	Flash status	Description
Aux 2	Green	–	State of the auxiliary voltage on line 2.
		On	State machine is in state *Standby*, *Power up*, or *Operation enabled*. Auxiliary voltage is applied to the output connectors on line 2.
		Off	State machine is in state *U_{AUX} disabled* or *Fault*. Auxiliary voltage is not applied to line 2.
Safe 1	Green	On	24 V for STO is present on line 1.
		Off	24 V for STO is not present on line 1.
Safe 2	Green	On	24 V for STO is present on line 2.
		Off	24 V for STO is not present on line 2.
SAB STAT	Green	On	SAB is in state *Operation enabled*.
		Flashing	Auxiliary voltage is applied at the input.
		Off	No auxiliary voltage is applied at the input.
	Red	On	The SAB is in state *Fault*.
		Flashing	Mains is not applied at the input.
NET STAT	Green/ red	Fieldbus dependent.	Network status of the device (see corresponding fieldbus standard).
Link/A CT X1	Green	–	Link/activity status of *In*.
		On	Ethernet link established.
		Flashing	Ethernet link established and active.
		Off	No link.
Link/A CT X2	Green	–	Link/activity status of *Out*.
		On	Ethernet link established.
		Flashing	Ethernet link established and active.
		Off	No link.
Link/A CT X3	Green	–	Link/activity status of line 1.
		On	Ethernet link established.
		Flashing	Ethernet link established and active.
		Off	No link.
Link/A CT X4	Green	–	Link/activity status of line 2.
		On	Ethernet link established.
		Flashing	Ethernet link established and active.
		Off	No link.

Table 9.8 Legend to *Illustration 9.2*

9

10 Appendix

10.1 Glossary

A flange
The A side is the shaft side of the servomotor.

Ambient temperature
The temperature in the immediate vicinity of the servo system or component.

Automation Studio™
Automation Studio™ is a registered trademark of B&R. It is the integrated software development environment for B&R controllers.

Axial force
The force in newton-metres acting on the rotor axis in the axial direction.

Bearings
The ball bearings of the servomotor.

Beckhoff®
Beckhoff® is a registered trademark of and licensed by Beckhoff Automation GmbH, Germany.

B&R
Multi-national company, specializing in factory and process automation software and systems for a wide range of industrial applications.

B side
The rear side of the servo drive with the plug-and-socket connectors.

Brake
Mechanical holding brake on the servo drive.

CANopen®
CANopen® is a registered community trademark of CAN in Automation e.V.

CE
European test and certification mark.

CiA DS 402
Device profile for drives and motion control.
CiA® is a registered community trademark of CAN in Automation e.V.

Clamping set
A mechanical device, which, for example, can be used to secure gears to a motor shaft.

Connector (M23)
Servo drive hybrid connector.

Cooling
ISD servo drives are cooled by convection (without fans).

DC-link
Each servo drive has its own DC-link, consisting of capacitors.

DC-link voltage
A DC voltage shared by several servo drives connected in parallel.

DC voltage
A direct constant voltage.

EPSG
Ethernet POWERLINK® Standardization Group.

ETG
EtherCAT® Technology Group

EtherCAT®
EtherCAT® (Ethernet for Control Automation Technology) is an open high performance Ethernet-based fieldbus system. EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Illustration 10.1 EtherCAT® Logo

Ethernet POWERLINK®
Ethernet POWERLINK® is a deterministic real-time protocol for standard Ethernet. It is an open protocol managed by the Ethernet POWERLINK® Standardization Group (EPSG). It was introduced by Austrian automation company B&R in 2001.

Feed-in cable
Hybrid connection cable between the SAB and servo drive.

Feedback system
Feedback systems for servo drives in general.

Fieldbus
Communication bus between controller and servo axis and SAB; in general between controller and field nodes.

Firmware
Software in the unit; runs on the control board.

Function block
Device functionalities are accessible via the engineering environment software.

IGBT
The insulated-gate bipolar transistor is a 3-terminal semiconductor device, primarily used as an electronic switch to combine high efficiency and fast switching.

Installation elevation
Installation elevation above normal sea level, typically associated with a derating factor.

ISD
Integrated servo drive.

ISD devices
Refers to both the ISD 510 servo drives and the SAB.

ISD servomotor
Designates the ISD servomotor (without the drive electronics).

ISD Toolbox
A Danfoss pc software tool used for parameter setting and diagnostics of ISD servo drives and the SAB.

LCP
Local control panel.

Loop cable
Hybrid connection cable between 2 servo drives, with 2 M23 connectors.

M8 connectors
Fully functional real-time Ethernet port (X3) on the B side of the advanced servo drive.
Connector (X5) for connection of the LCP to the B side of the advanced servo drive.

M12 connector
Connector (X4) for connecting I/O and/or encoder on the B side of the advanced servo drive.

M23 connectors
Connectors (X1 & X2) for connecting the hybrid feed-in and loop cables on the B side of the standard and advanced servo drive.

Motor shaft
Rotating shaft on the A side of the servo motor, typically without a key groove.

Multi-turn encoder
Describes a digital absolute encoder, in which the absolute position remains known after several revolutions.

PLC
A programmable logic controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factor assembly lines.

PELV
Protected extra low voltage. Low Voltage Directive regarding voltage levels and distances between lines.

PLCopen®
The name PLCopen® is a registered trademark and, together with the PLCopen® logos, is owned by the association PLCopen®. PLCopen® is a vendor and product-independent worldwide association, that defines a standard for industrial control programming.

POU
Program organization unit. This can be a program, function block, or function.

PWM
Pulse width modulation.

Radial force
The force in newton-metres acting at 90° to the longitudinal direction of the rotor axis.

RCCB
Residual current circuit breaker.

Resolver
A feedback device for servomotors, typically with 2 analog tracks (sine and cosine).

Safety (STO)
A servo drive safety circuit that switches off the voltages of the driver components for the IGBTs.

Scope
Is part of the ISD Toolbox software and is used for diagnosis. It enables internal signals to be depicted.

Servo Access Box (SAB)
Generates the DC-link supply for the ISD 510 servo system and can host up to 64 servo drives.

SIL 2
Safety Integrated Level II.

Single-turn encoder
Describes a digital absolute encoder, in which the absolute position for 1 revolution remains known.

SSI
Synchronous serial interface.

Standstill (servo drive)
Power is on, there is no error in the axis, and there are no motion commands active on the axis.

STO
Safe Torque Off function. On activation of STO, the servo drive is no longer able to produce torque in the motor.

TwinCAT®
TwinCAT® is a registered trademark of and licensed by Beckhoff Automation GmbH, Germany. It is the integrated software development environment for controllers from Beckhoff.

U_{AUX}
Auxiliary supply, provides power to the control electronics of the drives and SAB.

Wireshark®
Wireshark® is a network protocol analyzer released under the GNU General Public License version 2.

10.2 General XML Conventions

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 specification.

XML attributes can be optional or mandatory. Mandatory attributes must be available in the element. If a mandatory attribute is missing, the parsing of the file fails. Optional attributes can be omitted. If they are not there, a default value is assumed automatically.

All numerical values accept a point as decimal separator. The scientific representation of numbers is also allowed, where a small “e” or a capital “E” may be used. Any other characters inside the value cannot be used and lead to an error.

Examples of allowed values:

- 532.4
- +532.4
- –532
- –9.398846E-6
- –9.398846e-6
- 9.398846e+3
- –0.000305

Index

A

B

C

D

E

L

M

S

T

U

V

W

X

Danfoss A/S
Ulsnaes 1
DK-6300 Graasten
vlt-drives.danfoss.com

130R0695 MG36D102 *MG36D102* 01/2017