

PC-based Automation

IEC 61131-3

PLC

PLC Designer - SoftMotion

Contents

1	About this documentation	10
1.1	Document history	11
1.2	Liability	11
1.3	Trademarks	11
1.4	Conventions used	12
1.5	References used	13
2	SoftMotion concept and overview of the components	14
3	The SoftMotion drive interface	17
3.1	Control configuration for SoftMotion	18
3.1.1	BusInterface	19
3.1.2	AxisGroup	20
3.1.3	CANDrive	22
3.2	SM DriveBasic.lib and automatic code generation	26
3.2.1	Mathematical auxiliary blocks	26
3.2.2	Axis group blocks	27
3.2.3	Configuration blocks	31
3.2.4	Control mode blocks	32
3.2.5	Direct setpoint selection	33
3.2.6	Virtual time axis (SMC_TimeAxisFB)	40
3.2.7	Homing via digital hardware inputs (SMC_Homing)	41
3.2.8	Diagnostic blocks	43
3.2.9	ENCODER (SMC_Encoder)	45
3.2.10	Visualisation templates	47
3.3	Drive element <BusInterfaceName>Drive.lib	48
3.3.1	LenzeCANdrive.lib	48
3.3.2	SM_CAN.lib	54
3.4	Variables of the AXIS_REF structure	56
3.5	Parameterisation of the drive	60
3.5.1	SMC_WriteDriveParamsToFile	61

4	<u>CNC editor</u>	62
4.1	<u>Supported and extended elements of the CNC language of DIN66025</u>	63
4.1.1	<u>Switching points, H-function</u>	67
4.1.2	<u>Additional function, M-function</u>	68
4.1.3	<u>Use of variables</u>	68
4.1.4	<u>Circular interpolation</u>	69
4.1.5	<u>Ellipse interpolation</u>	70
4.1.6	<u>Spline interpolation</u>	70
4.1.7	<u>Conditional jumps</u>	70
4.1.8	<u>Changing variable values</u>	71
4.2	<u>Starting the CNC editor, inserting and managing CNC programs</u>	72
4.2.1	<u>Creating a new CNC program</u>	72
4.2.2	<u>Rename CNC Program</u>	72
4.2.3	<u>Deleting a CNC program</u>	73
4.2.4	<u>Information</u>	73
4.2.5	<u>Define Queue Size</u>	73
4.2.6	<u>Defining the start position of the CNC program</u>	73
4.2.7	<u>Defining the angular tolerance for stopping in the CNC program</u>	74
4.2.8	<u>Shifting a CNC program</u>	74
4.2.9	<u>Rotating the CNC program</u>	74
4.2.10	<u>Stretch the CNC program</u>	74
4.2.11	<u>Invert Direction</u>	74
4.2.12	<u>Split Object</u>	75
4.2.13	<u>Read CNC Program from File</u>	75
4.2.14	<u>Write CNC Program to File</u>	75
4.2.15	<u>Import DXF File</u>	75
4.2.16	<u>Write OutQueue to File</u>	76
4.3	<u>CNC text editor</u>	77
4.4	<u>CNC graphic editor</u>	78

4.5	Commands in the "Extras" menu	79
4.5.1	Select Mode	79
4.5.2	Line Insert Mode	79
4.5.3	Circle CW Insert Mode	79
4.5.4	Circle CCW Insert Mode	80
4.5.5	Spline Insert Mode	80
4.5.6	Fit to Screen	80
4.5.7	Renumber Program	80
4.5.8	Convert Splines/Ellipses to Lines	80
4.5.9	Tool Radius Correction	81
4.5.10	Round Path	81
4.5.11	Smooth Path	81
4.5.12	Avoid Loop	81
4.5.13	Suppress Record	81
4.5.14	Show Interpolation Points	82
4.5.15	Set epsilon values for zero	82
4.6	Automatic structure filling in the CNC editor	83
5	CAM editor	84
5.1	Definition of a cam for SoftMotion	84
5.2	Starting the CAM editor	85
5.3	Creating a cam	85
5.4	Editing a cam	88
5.4.1	General editor settings	88
5.4.2	Editing properties of individual cam elements	89
5.4.3	Commands in the "Extras" menu	92
5.4.4	Commands in the "Insert" menu	95
5.5	Online mode of the cam editor	96
5.6	Using cams	97
5.6.1	How POU parameters work	97
5.6.2	Switching between cams	100
5.7	Cam data structures	103
5.7.1	MC CAM REF	103
5.7.2	SMC CAMXYVA	105
5.7.3	SMC CAMTable <Variable type> <Number of elements> 1	105
5.7.4	SMC CAMTable <Variable type> <Number of elements> 2	105
5.8	Example: manually generated cam	106

6	<u>The SM_PLCopen.lib library</u>	107
6.1	<u>PLCopen specification</u>	108
6.2	<u>POUs for the motion control of individual axes</u>	110
6.2.1	<u>MC_ReadStatus</u>	110
6.2.2	<u>MC_ReadAxisError</u>	110
6.2.3	<u>MC_Reset</u>	110
6.2.4	<u>MC_ReadParameter / MC_ReadBoolParameter</u>	111
6.2.5	<u>MC_WriteParameter / MC_WriteBoolParameter</u>	111
6.2.6	<u>MC_ReadActualPosition</u>	112
6.2.7	<u>MC_ReadActualVelocity</u>	112
6.2.8	<u>MC_ReadActualTorque</u>	112
6.2.9	<u>MC_Power</u>	113
6.2.10	<u>MC_Home</u>	114
6.2.11	<u>MC_Stop</u>	115
6.2.12	<u>MC_Halt</u>	116
6.2.13	<u>MC_MoveAbsolute</u>	118
6.2.14	<u>SMC_MoveContinuousAbsolute</u>	121
6.2.15	<u>MC_MoveAdditive</u>	123
6.2.16	<u>MC_MoveRelative</u>	126
6.2.17	<u>SMC_MoveContinuousRelative</u>	128
6.2.18	<u>MC_MoveSuperImposed</u>	130
6.2.19	<u>MC_MoveVelocity</u>	134
6.2.20	<u>MC_PositionProfile</u>	136
6.2.21	<u>MC_VelocityProfile</u>	139
6.2.22	<u>MC_AccelerationProfile</u>	140
6.2.23	<u>MC_SetPosition</u>	141
6.2.24	<u>MC_TouchProbe</u>	142
6.2.25	<u>MC_AbortTrigger</u>	145
6.3	<u>POUs for the synchronised motion control (multi-axis)</u>	146
6.3.1	<u>MC_CamTableSelect</u>	146
6.3.2	<u>MC_CamIn</u>	148
6.3.3	<u>MC_CamOut</u>	150
6.3.4	<u>MC_GearIn</u>	150
6.3.5	<u>MC_GearOut</u>	151
6.3.6	<u>MC_GearInPos</u>	152
6.3.7	<u>MC_Phasing</u>	156

6.4	<u>Additional program organisation units</u>	157
6.4.1	<u>SMC_GetCamSlaveSetPosition</u>	157
6.4.2	<u>SMC_CAMBounds</u>	158
6.4.3	<u>SMC_CAMEditor / SMC_CAMVisu</u>	159
6.4.4	<u>SMC_CAMRegister</u>	161
6.4.5	<u>SMC_GetTappetValue</u>	163
6.4.6	<u>MC_DigitalCamSwitch</u>	164
6.4.7	<u>SMC_ReadSetPosition</u>	167
6.4.8	<u>SMC_SetTorque</u>	167
7	<u>The SM CNC.lib library</u>	168
7.1	<u>SMC_NCDecoder</u>	169
7.2	<u>SMC_GCodeViewer</u>	171
7.3	<u>SMC_ToolCorr</u>	173
7.4	<u>SMC_AvoidLoop</u>	176
7.5	<u>SMC_SmoothPath</u>	178
7.6	<u>SMC_RoundPath</u>	180
7.7	<u>SMC_CheckVelocities</u>	182
7.8	<u>SMC_LimitCircularVelocities</u>	183
7.9	<u>SMC_Interpolator</u>	185
7.10	<u>SMC_GetMParameters</u>	191
7.11	<u>SMC_Interpolator2Dir</u>	192
7.12	<u>Function blocks for spatial transformation processes</u>	194
7.12.1	<u>SMC_CoordinateTransformation3D</u>	194
7.12.2	<u>SMC_TeachCoordinateTransformation</u>	195
7.12.3	<u>SMC_UnitVectorToRPY</u>	196
7.12.4	<u>SMC_DetermineCuboidBearing</u>	197
7.13	<u>Extension of the interpolation process from 2.5-D to real 3-D</u>	199
7.14	<u>Auxiliary functions and function blocks for path rotations, translations and scalings</u>	200
7.14.1	<u>SMC_ROTATEQUEUE2D</u>	200
7.14.2	<u>SMC_TRANSLATEQUEUE3D</u>	200
7.14.3	<u>SMC_SCALEQUEUE3D</u>	200
7.15	<u>Settings via global variables</u>	201

7.16	Structures of SM CNC.lib	202
7.16.1	SMC_POSINFO	202
7.16.2	SMC_GEOINFO	203
7.16.3	SMC_VECTOR3D	206
7.16.4	SMC_VECTOR6D	206
7.16.5	SMC_OUTQUEUE	207
7.16.6	SMC_CNC_REF	210
7.16.7	SMC_GCODE_WORD	211
7.17	Path cams with SMC XInterpolator	212
8	The SM CNCDiagnostic.lib library	215
8.1	Function blocks for analysing SMC CNC_REF data	215
8.1.1	SMC_ShowCNCREF	215
8.1.2	Function blocks for analysing SMC OutQueue data	215
8.1.3	SMC_ShowQueue	215
9	The SM Trafo.lib library	216
9.1	Transformation function blocks	216
9.1.1	Gantry systems	216
9.1.2	Gantry systems with tool offset	222
9.1.3	H gantry system with stationary drives	228
9.1.4	Two-jointed Scara systems	231
9.1.5	Three-jointed Scara systems	233
9.1.6	Parallel kinematics	235
9.1.7	Spatial transformations	238
10	The SM Error.lib library	244
10.1	SMC_ErrorString	244
10.2	The enumeration SMC_Error	245
11	The SM FileFBs.lib library	250
11.1	CNC function blocks	250
11.1.1	SMC_ReadNCQueue	250
11.1.2	SMC_ReadNCFile	251
11.1.3	SMC_VARLIST structure	253
11.2	CAM function blocks	254
11.2.1	SMC_ReadCAM	254
11.2.2	SMC_WriteCAM	254
11.3	Diagnostic function blocks	255
11.3.1	SMC_AxisDiagnosticLog	255

12	<u>The L_SMC_DriveUtil.lib library (Lenze specifications)</u>	257
12.1	<u>L_SMC_TouchProbe</u>	257
12.2	<u>Function blocks for EtherCAT</u>	262
12.2.1	<u>L_SMC_SetTorqueLimit</u>	262
12.2.2	<u>L_SMC_AbortTrigger</u>	264
12.2.3	<u>L_SMC_ReadParameter</u>	265
12.2.4	<u>L_SMC_WriteParameter</u>	266
13	<u>SoftMotion programming examples</u>	267
13.1	<u>Single-axis motion control in ST</u>	268
13.2	<u>Single-axis motion control in CFC with visualisation template</u>	270
13.3	<u>Cam drive control with the aid of a virtual time axis</u>	272
13.4	<u>Alternating cams</u>	274
13.5	<u>Drive control with the aid of the CNC editor</u>	275
13.5.1	<u>Direct creation of the OutQueue</u>	275
13.5.2	<u>Online decoding with the use of variables</u>	278
13.5.3	<u>Online path preprocessing</u>	281
13.6	<u>Dynamic SoftMotion programming</u>	283
13.6.1	<u>Preliminary steps</u>	283
13.6.2	<u>Dynamic Path-Programming</u>	283
14	<u>Index</u>	285

1 About this documentation

This documentation contains information on the SoftMotion concept and its components for the implementation of complex applications with the »PLC Designer«, where the sequence and process control and auxiliary functions play a major role.



Note!

This documentation supplements the software manual for the »PLC Designer«.



Tip!

Documentation and software updates for Lenze products can be found in the Download area at:

<http://www.Lenze.com>

Target group

This documentation addresses to persons who carry out the PLC programming with the »PLC Designer«.

Information on the validity

The information in this documentation is valid for the following Lenze software:

Software	From software version
L-force »PLC Designer«	2.3.0.33

1.1 Document history

Version			Description
1.0	06/2008	TD17	First edition for the »PLC Designer« - SoftMotion
2.0	08/2008	TD17	Revision
3.0	05/2009	TD29	Revision
4.0	10/2009	TD29	Extended by descriptions of the function blocks L_SMC_ReadParameter, L_SMC_WriteParameter and SMC_Encoder, as well as the 3D interpolation process. Revised description of the L_SMC_TouchProbe function block.
5.0	11/2010	TD29	Updated to software version 2.3
5.1	01/2011	TD17	- Note on the cyclic communication data for Lenze ECSxM (Motion) controllers added (see the chapter CANDrive (□ 22)) - Index entries added: – Word identifiers (□ 65) – Travel commands (□ 66) - G90 and G91 travel commands updated.
5.2	03/2011	TD17	Description of the L_SMC_TouchProbe (□ 257) function block adapted.

Your opinion is important to us!

These Instructions were created to the best of our knowledge and belief to give you the best possible support for handling our product. If you have suggestions for improvement, please e-mail us to:

feedback-docu@lenze.de

Thank you for your support.

Your Lenze documentation team

1.2 Liability

All information provided in this documentation has been carefully selected and checked to ensure that it accurately reflects the hardware and software described. Nevertheless, discrepancies cannot be ruled out. We do not accept any responsibility or liability for any damage that may occur due to such discrepancies. Any necessary corrections will be included in future updates of the documentation.

This documentation is based on the original »CoDeSys« documentation of 3S - Smart Software Solutions GmbH (Kempten). The depicted screenshots can vary from the »PLC Designer« images.

1.3 Trademarks

Microsoft, Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the USA and/or other countries.

Adobe and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the USA and/or other countries.

All other brand names mentioned in this documentation are trademarks of the relevant owners.

1.4 Conventions used

This documentation uses the following conventions to distinguish between different types of information:

Type of information	Writing	Examples/Notes
Spelling of numbers		
Decimal separator	Point	Generally the decimal point is used. For example: 1234.56
Text		
Version information	Text colour blue	All pieces of information that only apply to or from a specific software version of the controller are highlighted accordingly in this documentation. Example: This function extension is available from software version V3.0 onwards!
Program name	» «	The Lenze PC software »PLC Designer«
Window	<i>Italics</i>	The <i>Message window...</i> / The <i>Options</i> dialog box...
Variable identifiers		Set <i>bEnable</i> to TRUE to...
Control element	Bold	The OK button... / The Copy command... / The Properties tab... / The Name input field...
Sequence of menu commands		If several commands must be used in sequence to carry out a function, the individual commands are separated by an arrow: Select File→Open to...
Shortcut	< Bold >	Press < F1 > to open the online help. If a command requires a combination of keys, a "+" is placed between the key symbols: With < Shift >+< ESC >...
Program code	Courier	IF var1 < var2 THEN a = a + 1 END IF
Keyword	Courier bold	
Hyperlink	<u>Underlined</u>	Optically highlighted reference to another topic. Is activated via mouse-click in this documentation.
Symbols		
Page reference	(12)	Optically highlighted reference to another page. Is activated via mouse-click in this documentation.
Step-by-step instructions		Step-by-step instructions are indicated by a pictograph.

1.5 References used

This documentation uses the following signal words and symbols to indicate dangers and important information:

Safety instructions

Structure of the safety instructions:



Pictograph and signal word!

(characterise the type and severity of the danger)

Note

(describes the danger and gives information about how to prevent dangerous situations)

Pictograph	Signal word	Meaning
	Danger!	Danger of personal injury through dangerous electrical voltage Indicates an impending danger that may lead to death or severe personal injury if the corresponding measures are not taken.
	Danger!	Danger of personal injury through a general source of danger Indicates an impending danger that may lead to death or severe personal injury if the corresponding measures are not taken.
	Stop!	Danger of damage to material assets Indicates a potential danger that may lead to damage to material assets if the corresponding measures are not taken.

Application notes

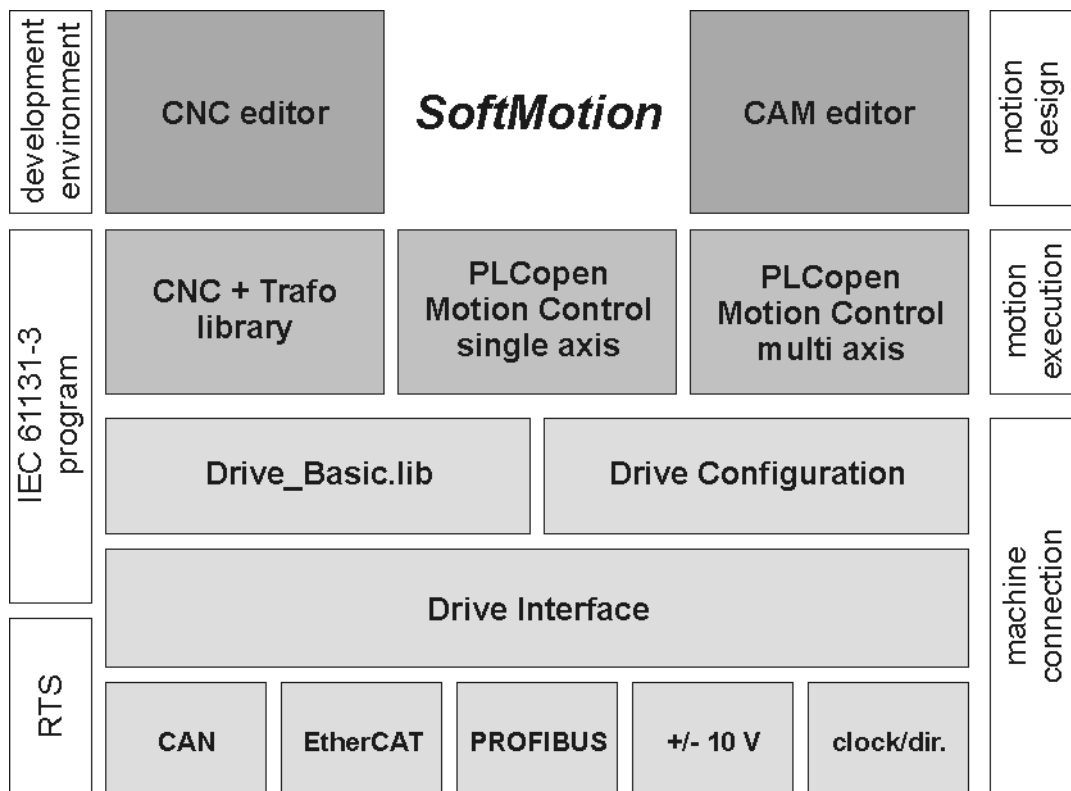
Pictograph	Signal word	Meaning
	Note!	Important note to ensure trouble-free operation
	Tip!	Useful tip for easy handling
		Reference to another documentation

2 SoftMotion concept and overview of the components

SoftMotion allows you to realise motions - from simple single-axis motions to cam motions and complex motions in several dimensions - in the »PLC Designer« integrated development environment. Those applications where sequence and process control or auxiliary functions play a key role alongside the motion functions are ideal for use with SoftMotion (as opposed to those that are exclusively concerned with motion functions).

SoftMotion should be viewed as a kind of basic toolkit, which you can use to influence procedures in various ways (even during runtime, should the process allow it) without having to go to a great deal of effort or having to have in-depth knowledge in order to realise the desired motions.

The entire program logic is handled in the PLC program and only pure motion information is processed in the library POU's.



SoftMotion is divided into the following components:

Drive Interface

This SoftMotion component is responsible for communication between the PLC program and the drives. It consists of the SM_DriveBasic.lib library as well as drive and bus system-specific libraries and drivers.

The "LenzeCAN (CAN)" drive driver and the LenzeCANdrive.lib library support the motion control of the Lenze 9400 Servo Drives StateLine and the ECSxM axis modules (Motion) via the CANopen bus system.

The programmer uses the configuration editor in the »PLC Designer« to map the structure and parameterisation of the drive hardware. With the help of the Drive Interface libraries, the "PLC Designer" then uses this information to create IEC data structures that represent the drives in an abstract manner. The Drive Interface communicates with the drives automatically, i.e. without the IEC programmer having to do anything else, thus ensuring that the drive data structures are updated continuously and that any data modified within them is transferred. To control the drive hardware, the IEC program can either access these data structures via standard POU's in the SoftMotion libraries (SM_CNC.lib, SM_PLCOpen.lib) or by means of programs actually created by the IEC programmer.

The setpoints are always set cyclically, i.e. setpoints (positions, velocities, accelerations, etc.) are calculated and transferred from the Drive Interface to the drives with each IEC task cycle. It is not possible to "instruct" the drives, for example, to set a target position so that a drive moves independently and, once the position has been reached, sends a message to the control that the instruction has been completed successfully. Amongst other reasons, this is because in such a case it would not be possible to coordinate the motions of several axes and the central control would not have any influence on the drives during execution of an instruction.

CNC editor

The CNC editor in the »PLC Designer« is used for the graphical and textual programming of multidimensional drive motions, based on the DIN 66205 CNC language. Basically, motions of up to 9 dimensions can be realised, whereby only two dimensions will not be interpolated linearly. In other words, lines, circles, circular arcs, parabolas, ellipses and splines can be programmed in two dimensions; the other directions will merely be interpolated linearly. For each path generated, the "PLC Designer" will automatically create a global data structure (CNC data) that will be available in the IEC program.

CAM editor

The graphically operated CAM editor integrated in the »PLC Designer« is used to program cams for the purpose of controlling multi-axis drives. On the basis of each programmed cam, the »PLC Designer« will automatically generate a global data structure (CAM data), which will be made available in the IEC program.

CNC libraries

The "SM_CNC.lib", "SM_CNCDiagnostic.lib" and "SM_Trafo.lib" libraries provide the IEC programmer with program organisation units that can be used to edit, display and execute motions generated in the CNC editor or planned during runtime.

PLCopen library

The "SM_PLCopen.lib" PLCopen motion control library consists of, inter alia, program organisation units that can be used not only for easy programming and control of single axes, but also the synchronised motion of two axes. Besides program organisation units that can be used for status queries, parameterizing and operation in general, it also contains function blocks that can move an axis in different ways according to the defined velocity and acceleration parameters. This specification can be, for example, a cam generated in the CAM editor that uses available program organisation units to link the slave axis to the master axis. There are also function blocks for programming electronic gears or phase shifts.

File service library

The "SM_FileFBs.lib" library is based on the "SysLibFile.lib" system library and can, therefore, only be used on controls which support that library.

Error library

The "SM_Error.lib" library contains all error outputs that can be produced by the program organisation units of the other libraries. It can also be used to generate English and German error messages from numeric error variables.

Portability

With the exception of some Drive Interface drivers that serve hardware components directly, all SoftMotion runtime components are programmed in IEC 1131-3, allowing maximum platform independence to be achieved.

In order to get to grips with the SoftMotion components quickly, we recommend that the reader studies the relevant examples.

Information on exemplary projects

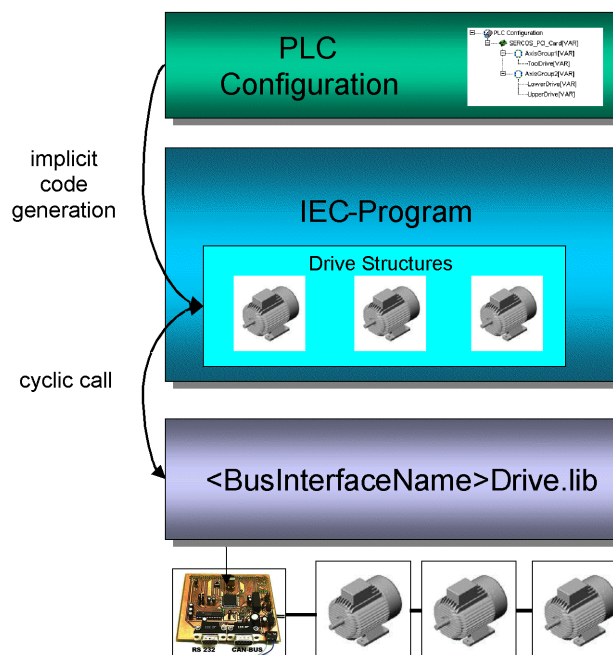
For a quick understanding of the SoftMotion components it is recommended to study the corresponding "[SoftMotion programming examples](#)" (□ 267). The projects referenced therein are included in the standard installation of »PLC Designer« (within the directory "../PLC Designer V2.3/Projects/Motion/Tutorial") and may be opened and reworked by the reader within the »PLC Designer« Softmotion environment. All projects can be translated and downloaded to a PLC of type RTE. Furthermore many of these projects contain a visualization assisting a deeper understanding of the SoftMotion modules.

3 The SoftMotion drive interface

The Softmotion Drive Interface is a standardised interface which offers the programmer the possibility of integrating the abstract image of a drive into the IEC program, configuring it there, and addressing it. It automatically provides for the updating and transmission of the data required for the control of the drive hardware. Like this drives cannot only be easily replaced and IEC program reused, also the user is saved the difficulties and trouble of the drive connection.

The Drive Interface works with the following components:

- The »PLC Designer« control configurator: Here the structure of the drives to be operated is to be mapped and parameterised on the basis of a corresponding configuration file. This structure is then made accessible to the application (IEC program) by means of the Drive-Interface libraries via implicitly generated and assigned (system) variables.
- The internal SM_DriveBasic.lib library provides IEC data structures and global variables which represent the drives, axis groups, and BusInterfaces shown in the control configurator.
- The drive element, i.e. the specific library <BusInterfaceName>Drive.lib (e.g. LenzeCANDrive.lib) that is to be supplied by the manufacturer of the control system for the drives and the bus system used offers special functions which are required for the data exchange between the structures and the corresponding hardware (see chapter [\[3.3\]](#) ([48](#))).





Note!

All SoftMotion function blocks, SoftMotion functions and the write and read function parameters which access the SoftMotion drives (e.g. MC_WriteParameter, MC_WriteBoolParameter, LenzeECSReadParameter, LenzeECSWriteParameter) may only be called in the Motion Task.

If the call is carried out in a different task, the execution may be incorrect.

3.1 Control configuration for SoftMotion

To map the structure of the drive, usually the following elements are provided in the »PLC Designer« control configuration (the corresponding configuration file has to be available in the directory for configuration files!):

	BusInterface_CAN BusInterface_EtherCAT BusInterface_Dummy	Fieldbus interface via which communication with the drives is carried out.
	Axis_IO_Group Axis_Group_CAN Axis_Group_Dummy	Physically connected group (axis group) of drives
	Drive	Drive

The Businterfaces, axis groups, and drives can be designated with optional, but definite IEC 61131-3 identifiers:

Each of these configuration objects can be configured via dialogues if this is supported by the configuration description of the target. In addition to the comfortable configuration via dialogues, the parameters can also be set in a configuration list ("module parameters"). There you can also find target-specific parameters starting with "MS:".

**Note!**

The configuration of the axis group is cancelled if

- the control system has detected an error on a drive;
- a drive is not available physically;
- a module parameter value beyond the value range specified for this purpose has been set for a drive in the **Control configuration** under the **Service Data Objects** tab.

This can be checked with the AXIS_REF-Strukturvariable *bCommunication* structure variable:

- *bCommunication* = *TRUE*: No error, drive responds
- *bCommunication* = *FALSE*: An error has occurred

Do not rename the axis groups and drives added to the control configuration after the import of a configuration file (XML). If you carry out another import, the names will be overwritten.

3.1.1 BusInterface

Here usually only the communication driver is selected.

► BusInterface_CAN:

► BusInterface_EtherCAT:

► BusInterface_Dummy:

Parameters of the PLC configuration

Parameter	Description
wParam1	Not used
wParam2	Not used
dwParam1	Not used
dwParam2	Not used

3.1.2 AxisGroup

Here the cyclic task where communication to the drives is effected and their cycle time is specified.

- ▶ Axis_IO_Group / Axis_Group_Dummy:

The screenshot shows the 'AxisGroup' dialog box with the 'Module parameters' tab selected. It contains two fields: 'Motion task:' with a dropdown menu, and 'Cycletime:' with a text input field containing '0' and a unit selector set to 'µs'.

- ▶ Axis_Group_CAN:

The screenshot shows the 'AxisGroup' dialog box with the 'Module parameters' tab selected. It contains the same 'Motion task:' and 'Cycletime:' fields as the previous dialog. Below these is a section titled 'specific settings' which includes three fields: 'Controller no.' with a text input field containing '0', 'baud rate:' with a dropdown menu, and 'SYNC producer:' with a dropdown menu set to 'PLC'.

Apart from the specification of the baud rate, the controller number serves to define the number of the CAN interface used (CAN1, CAN2, etc.). If a Logic bus is available (i. e. a CAN master) it always occupies the CAN1 interface. Depending on the industrial PC used, Motion buses can then be placed on the interfaces CAN2 ... CAN4.



For this observe the information in the "**CANopen control technology**" communication manual!

For sync generators different methods can be selected for the synchronisation between drives and the PLC:

- ▶ **PLC:** Here the PLC is the synchronisation master. The user usually defines the motion task as a cyclic task. This task calls the driver which immediately sends a sync telegram. This method is the simplest, however it can cause problems if control systems with a high variation and drives requiring a high accuracy of the sync telegram are used.
- ▶ **1. drive:** This operating mode is not supported.
- ▶ **Sync device:** This operating mode is not supported.

Parameters of the PLC configuration

Parameter	Description
wParam1	CAN channel no. (typically 0)
wParam2	Baud rate in kBit (125, 250, 500, 1000)
wParam3	Sync generator: 0: The PLC generates the sync signal (only possible if the PLC has a high precision) 1: The 1. drive in the AxisGroup generates the sync signal 2: A sync device generates the sync signal (additional hardware required)
wParam4	Not used
dwParam1	Reserved
dwParam2	Reserved
dwParam3	Not used
dwParam4	Reserved

The following parameters can be viewed and altered under the **Module parameters** tab.

Parameter	Description
sTask	String that corresponds to the name of the task in which the data transmission of this axis group is to take place.
dwCycle	Cycle time of the task defined under "sTask" in μs (only has to be specified if the control system does not support any tasks and automatically calls PLC_PRG (default task))
wParam1 ... wParam4	Card/drive-specific parameters of the WORD type
dwParam1 ... dwParam4	Card/drive-specific parameters of the DWORD type

3.1.3 CANDrive

Drive | CANDrive | Module parameters

Drive id: 1

Type: ☒ rotary (modulo) ☐ linear

Scale

65536	Increments	<->	1	Motor rotations
1	Input rotations of gear	<->	1	Output rotations of gear
1	Output rotations of gear	<->	360	SoftMotion units

Settings for rotary drive

Modulo value [SM units]: 360.00000

Cyclic communications data

PLC -> Drive:	☒ POS	☐ VEL	☐ ACC	☐ TOR	☐ CUR
Drive -> PLC:	☒ POS	☐ VEL	☐ ACC	☐ TOR	☐ CUR

Maximum values (CNC: SMC_ControlAxisByPos)

Velocity:	660.00000	SM-units/s
Acceleration:	100.00000	SM-units/s²
Deceleration:	100.00000	SM-units/s²

Velocity ramp type

☐ trapezoid ☒ sigmoidal ☐ parabolic

Jerk: 50000.000 SM-units/s³

In this dialog the drive ID (**Drive ID**) is defined. Furthermore the drive type is selected: linear or rotary (modulo).

In the **Conversion factor** area the conversion between the integer position values received by the drive (first line on the left) and the technical units used in the IEC program (first line on the right) is defined. Set the value 65536 in the **Increments** input field. Here also a gearbox can be taken into consideration (second and third line). In the above example a drive would exactly carry out a mechanical revolution if 360 degrees were programmed.

Depending on the drive type selected, software limit switches can be defined in the area **Settings for linear drive** for linear drives, or the modulo range has to be defined for **rotary drives**.

In the **cyclic communication data** area is it selected which setpoint and actual data are to be transmitted cyclically between the control system (PLC) and the drive.



Note!

ECSxM (motion) Lenze controllers only support the transmission of position values in the cyclic communication data. If these controllers are used, therefore no further cyclic data must be activated.

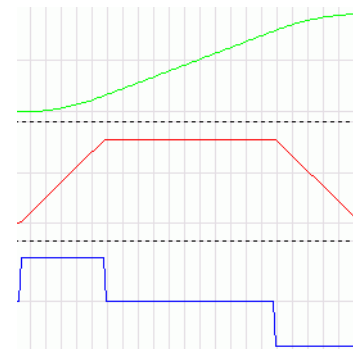
Under **Maximum values** the limit values are set which are used by SMC_ControlBy blocks to determine whether there is a skip (see [SMC_ControlAxisByPos](#) (38)).

In the case of the **Speed ramp type**, the user specifies (if supported by the libraries integrated) the speed profile type for the motion-generating single and master/slave blocks. "Trapezoid" results in a trapezoidal speed profile (constant acceleration in sections), "sigmoidal" in a $\sin^2$ -form speed profile (continuous acceleration), "parabolic" in a continuous trapezoidal acceleration profile, and therefore a parabolic speed profile.

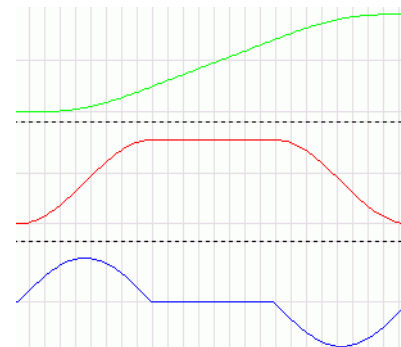
For the profiles "sigmoidal" and "parabolic", the jerk must also be set. This value is set by selecting the element "fRampJerk" in AX_REF. The write access must be carried out from the PLC application.

The following graphics show the effect of the different ramp modes for a positioning process. The position is represented in green, the speed in red, and the acceleration in blue colour.

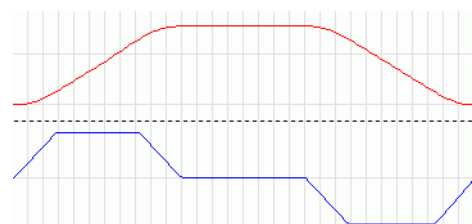
In the case of the **trapezoidal** speed profile it can be seen that steps occur during the acceleration.



In the case of the **sigmoidal** profile, the jerk is evaluated additionally. This value can be changed by selecting the element "fRampJerk" in AX_REF, the write access must be carried out from the der PLC application. A change is not necessary for most of the applications, the preset default value should be kept.



In the case of the **parabolic** speed profile the acceleration adopts a continuous trapezoidal profile the gradient of which is limited by the jerk. For the speed a continuous parabolic profile results. Only in the case of this profile the jerk can really be limited.





Note!

The jerk is only effective during the deceleration if the acceleration is non-zero at this point of time.

Example: The MC_MoveVelocity function block is started and shortly afterwards, the MC_Stop function block is activated. As the drive is still in the acceleration phase at this point of time, the jerk is evaluated. If the set jerk value is too low, this can lead to a significantly longer braking distance.

3.1.3.1 "Module parameters" dialog

All settings of the [CANDrive](#) (22) dialog can also be viewed and altered in the *Module parameters* dialog:

Parameter	Description
wDriveld	ID of the drive in the AxisGroup (WORD)
wControlType	Predefined control and feedback types (WORD); the following options are provided: (<transmitted data> -> <feedback data>) TOR -> --- Torque -> --- VEL -> VEL Speed -> speed VEL -> POS Speed -> position POS -> POS Position -> position POS, VEL -> POS, VEL Position, speed -> speed, position VEL -> --- Speed -> --- CONFIGURABLE Manual configuration via: • wCyclicDataS1, ..S2, ..S3 and • wCyclicDataR1, ..R2, ..R3 (s.u.)
wCyclicDataS1 wCyclicDataS2 wCyclicDataS3 wCyclicDataR1 wCyclicDataR2 wCyclicDataR3	Definition of the transmitted data (..S..) and feedback data (..R..) if wControlType is set to "CONFIGURABLE". Options depend on the corresponding drive element. Basically the following can be set: • Act/SetPosition: position • Act/SetVelocity: Speed • Act/SetTorque: Torque • Act/SetCurrent: Current • Act/SetUserDef: User-specific definition
dwRatioTechUnitsDenom iRatioTechUnitsNum	Denominator and numerator for conversion factor of bus data in technical units [u] (DWORD or INT)
iMovementType	Motion type; options: linear or rotary
fPositionPeriod	Period for rotary axes in technical units Please note that, converted into increments (by "dwRatioTechUnitsDenom" and "iRatioTechUnitsNum"), it must not be greater than 80000000 _{hex} (2147483648 _{dec}).
fSWMaxVelocity	Maximum speed for software check
fSWMaxAcceleration	Maximum acceleration for software check
fSWMaxDeceleration	Maximum deceleration for software check
bSWLimitEnable	Switch on software position check (linear drives only) which causes the axis to be set in the error status if it exits the position window.
fSWLimitNegative	Negative position limit (linear drives only)
fSWLimitPositive	Positive position limit (linear drives only)
eRampType	Speed profile type (trapezoidal, sigmoidal, parabolic)
fRampJerk	Jerk for the speed profiles "sigmoidal" and "parabolic"

3.1.3.2 "CANDrive" dialog

In the *CANDrive* dialog parameters which are to be written to the drive during acceleration can be entered. They can be stored in XML files or loaded from them.

3.2 SM_DriveBasic.lib and automatic code generation

If the **SM_DriveBasic.lib** library is integrated into the IEC1131 application in the »PLC Designer«, structure objects which the IEC program can access are created automatically from the drive image that has been entered in the control configuration using the corresponding parameters. (For this see also [Variables of the AXIS_REF structure](#) (□ 56) and [The structure of LenzeECS_AXIS_REF](#) (□ 50).)

In the IEC1131 application furthermore the manufacturer-specific library that matches the control system used **has to** be integrated with the name **<BusInterfaceName>Drive.lib**. It supports the hardware-specific drive interface function. The BusInterface designation results from the entry selected in the control configuration in the module parameters of the bus interface at 'InterfaceType'. The left part to the first space character of its string designation provides the name (example: "CAN (Peak)" => "CAN" => manufacturer-specific library is called: "CANDrive.lib").

When the application is started, the implicit call of the functions **<BusInterfaceName>DriveExecute_Start** and **<BusInterfaceName>DriveInit** at the task start and **<BusInterfaceName>DriveExecute_End** at the task end provides for the transmission and maintenance of the **AXIS_REF** structure variables. In case that errors occur during the initialisation of the drives, the global variable **g_strBootupError** contains an error description generated by the **<BusInterfaceName>Drive.lib** library.

In addition to its main function, the representation of the drives within the IEC program, the **SM_DriveBasic.lib** library contains some auxiliary blocks.

3.2.1 Mathematical auxiliary blocks

The **SMC_sgn** function returns the sign value of the input, i. e. -1 for a negative, +1 for a positive, and 0 for a zero entry.

The **SMC_fmod** function calculates the modulo value of the input *x* to the period *m*. The return value is always in the interval $[0, m]$.

The **SMC_atan2** function calculates and returns the angle which solves the following equations:

$$\sin() * f = \text{sine and } \cos() * f = \text{cosine}$$

In contrast to the common ATAN function the value range here covers the complete interval $[0; 2\pi]$.

3.2.2 Axis group blocks

3.2.2.1 Cycle time measurement

For diagnostics it is often important to be able to measure the exact cycle time of an axis group, in particular if the task controls communication actively (e. g. if the PLC acts as CAN sync generator). In the axis group data structure the variable <AxisGroup>.bDebug (BOOL) can be found, which controls the time measurement. If it is TRUE, the minimum or maximum time elapsed between two task calls is specified in <AxisGroup>.udiTaskMinTime and <AxisGroup>.udiTaskMaxTime. Furthermore the current and the maximum runtime of the task are recorded in <AxisGroup>.udiTaskRunTime and <AxisGroup>.udiTaskMaxRunTime.

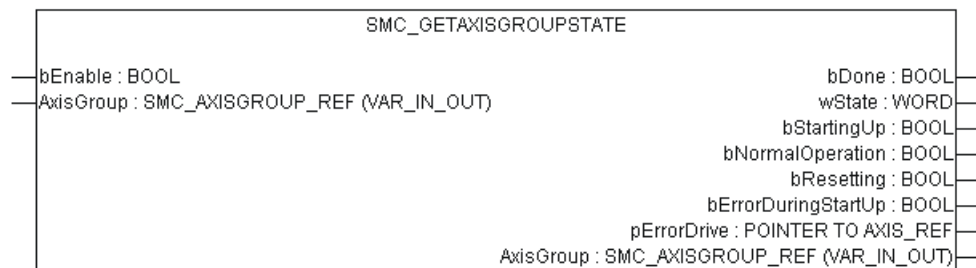
3.2.2.2 SMC_IsAxisGroupReady

This function provides a feedback via a BOOL variable on whether the acceleration which is implicitly carried out at the start of the IEC program for each axis group is completed and whether it is ready for operation with its axes (TRUE), or whether the acceleration has not yet been completed or an error has occurred (FALSE).



3.2.2.3 SMC_GetAxisGroupState

This function block provides details on the status of the axis group:



Inputs/outputs (VAR_IN_OUT)

AxisGroup : SMC_AXISGROUP_REF

Axis group for which information is required.

Inputs (VAR_INPUT)

bEnable: BOOL

If this input is TRUE, the POU supplies information on the status of the axis group.

Outputs (VAR_OUTPUT)

bDone : BOOL

TRUE as soon as valid data on the outputs.

wState : WORD

Internal state variable of the axis.

bStartingUp : BOOL

Axis group is accelerated, i. e. drives are configured. ($0 \leq wState \leq 99$)

bNormalOperation: BOOL

Axis group in normal operation. ($wState = 100$)

bResetting: BOOL

Axis group is currently reinitialised. ($200 \leq wState \leq 210$)

bErrorDuringStartUp: BOOL

An error occurred during acceleration. ($wState \geq 1000$)

pErrorDrive: POINTER TO AXIS_REF

Pointer to the axis causing the error. Only valid if $bErrorDuringStartUp = TRUE$.

By means of this task the erroneous axis can be deleted from the axis group for the runtime by setting the variable `bDisableDriveInAxisGroup`, then the axis group can be reinitialised via `SMC_ResetAxisGroup` and continued with the remaining axes that are working correctly, if the required redundancies within the machine are available

3.2.2.4 SMC_ResetAxisGroup

With this function block a complete axis group can be reinitialised.



Inputs/outputs (VAR_IN_OUT)

AxisGroup : SMC_AXISGROUP_REF

Axis group to be reinitialised.

Inputs (VAR_INPUT)

bExecute: BOOL

If this input is TRUE, the POU starts with the reinitialisation of the axis group.

bKeepRatioSettings: BOOL

If this input is TRUE, the previous gearbox settings (dwRatioTechunitsDenom and iRatioTechUnitsNum), the modulo value (fPositionPeriod), and the axis type (iMovementType, linear/rotary) are maintained and are not replaced by the values set in the control configuration.

Outputs (VAR_OUTPUT)

bDone : BOOL

TRUE if the process has been completed.

bError : BOOL

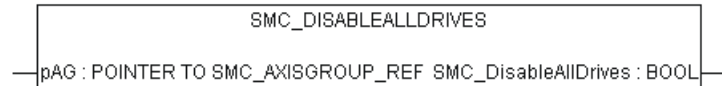
Error occurred.

nErrorID: SMC_ERROR

Description of the error.

3.2.2.5 SMC_DisableAllDrives

With this function the variable bDisableDriveInAxisGroup of each drive is set. This brings about that at the next execution of the SMC_ResetAxisGroup function no drive is provided, but that only the basic communication mechanisms of the axis group are executed.



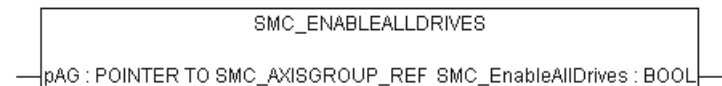
Inputs (VAR_INPUT)

pAG : POINTER TO SMC_AXIS_GROUP

Pointer to the axis group.

Return value of the function: TRUE

3.2.2.6 SMC_EnableAllDrives



In contrast to SMC_DisableAllDrives, by means of this function all drives are switched on again and are started up by calling SMC_ResetAxisGroup.

Inputs (VAR_INPUT)

pAG : POINTER TO SMC_AXIS_GROUP

Pointer to the axis group.

Return value of the function: TRUE

3.2.3 Configuration blocks

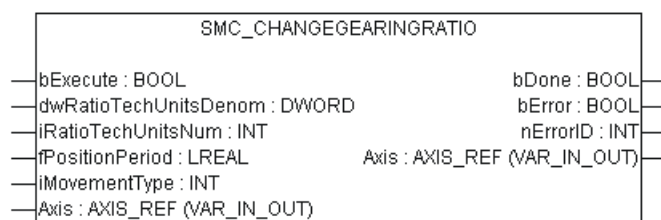
3.2.3.1 SMC_ChangeGearingRatio

By means of this POU the IEC program can change the gearbox ratio and the drive type for the runtime.



Note!

After the execution of this POU the axis group should be restarted via SMC_ResetAxisGroup (bKeepRatioSettings = TRUE), as only like this it can be ensured that all variables are initialised correctly!



Inputs/outputs (VAR_IN_OUT)

Axis : AXIS_REF

Drive the gearbox ratio of which is to be changed.

Inputs (VAR_INPUT)

bExecute: BOOL

The program organisation unit is activated with a rising edge.

dwRatioTechUnitsDenom : DWORD

iRatioTechUnitsNum: DWORD

Conversion ratio between SoftMotion unit and increments

fPositionPeriod: LREAL

Position period, modulo value (for rotary drives only)

iMovementType: INT

0: rotary axis / 1: linear axis.

Outputs (VAR_OUTPUT)

bDone : BOOL

TRUE as soon as the action has been carried out.

bError : BOOL

TRUE if an error occurred.

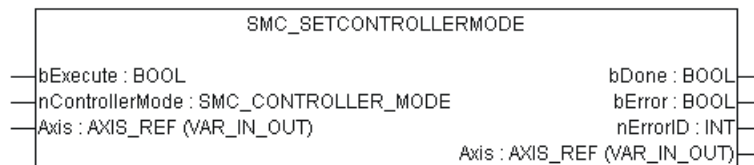
nErrorID: SMC_ERROR

Describes the error.

3.2.4 Control mode blocks

3.2.4.1 SMC_SetControllerMode

By means of this POU a change-over to another control mode can be effected if this is supported by the drive.



Inputs/outputs (VAR_IN_OUT)

Axis : AXIS_REF (VAR_IN_OUT)

Drive the control mode of which is to be changed.

Inputs (VAR_INPUT)

bExecute: BOOL

The program organisation unit is activated with a rising edge.

nControllerMode: SMC_CONTROLLER_MODE

Desired control mode: SMC_torque (torque) SMC_velocity (speed), SMC_position (position), SMC_current (current)

Outputs (VAR_OUTPUT)

bDone : BOOL (VAR_OUTPUT)

TRUE as soon as the action has been carried out.

bError : BOOL (VAR_OUTPUT)

TRUE if an error occurred.

nErrorID : SMC_Error (VAR_OUTPUT)

Describes the error.

3.2.5 Direct setpoint selection

SoftMotion offers the possibility to the user to directly specify setpoints for the position, speed, and torque/force (see [SMC_SetTorque](#) (167)) and write them to the axis. The modules provided for this purpose for instance can be used to write profiles to the axis, which have been self-calculated.

Another important application area is the CNC control where setpoints in X/Y/Z first run through a transformation block and have to be converted to setpoints for the individual axes. In order to then transfer these setpoints to the axis and, if required, carry out limit value monitoring processes, the following POU's are provided.

The POU's SMC_FollowPosition/Velocity write specified setpoints to the axis structure without further check. The POU SMC_CheckLimits checks the current setpoints with regard to the compliance of limit values.

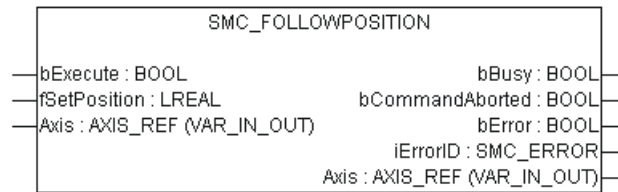
The POU's SMC_ControlAxisByPos/Vel are specifically designed for the CNC and offer additional functions: As long as the limit values are not exceeded, they work like SMC_FollowPosition/Velocity.

However, if a setpoint step-change occurs (e. g. caused by the sequential execution of two CNC programs where the end point and starting point does not correspond to each other), they stop the interpolator, carry out a positioning process to the current set interpolator position and deactivate the inhibit again, and the interpolator continues its path.

This, these blocks are intended to compensate positions setpoint step-changes. They detect such a change by checking the compliance with the peak values (of speed and acceleration). However, the limit values may also be exceeded if the speed of the CNC program or the override value on the interpolator is too high, or the maximum values of the axis in the control configuration were set too low (or not at all). The result would be a jerky run, as the interpolator is always decelerated again as soon as the material speed is so high that an axis limit is exceeded, and the individual axes are traversed to the corresponding position. Then the interpolator would start up again to be decelerated again shortly afterwards.

3.2.5.1 SMC_FollowPosition

This POU writes position setpoints to the axis and during this process does not carry out any checks.



Inputs (VAR_INPUT)

bExecute: BOOL

By a rising edge the block starts to write the setpoints entered in fSetPosition to the axis data structure. It remains active until it is interrupted by another block (e.g. MC_Stop).

fSetPosition: LREAL

Set position in technical units.

Outputs (VAR_OUTPUT)

bBusy : BOOL (Default: FALSE)

With TRUE the block indicates that it is active and that it writes setpoints to the axis.

bCommandAborted : BOOL (Default: FALSE)

If this is TRUE, the block was aborted by another.

bError : BOOL (Default: FALSE)

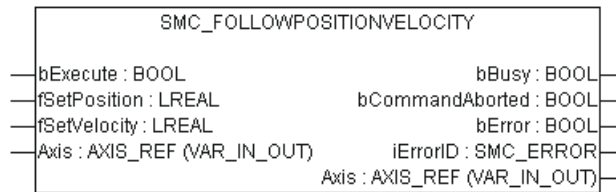
TRUE indicates that an error has occurred in the function block.

iErrorID : SMC_Error (INT)

Error number

3.2.5.2 SMC_FollowPositionVelocity

This block writes position and speed setpoints to the axis and during this process does not carry out any checks. The user has to ensure that the values match in a reasonable manner.



Inputs (VAR_INPUT)

bExecute: BOOL

By a rising edge the block starts to write the setpoints entered in fSetPosition to the axis data structure. It remains active until it is interrupted by another block (e.g. MC_Stop).

fSetPosition: LREAL

Set position in technical units.

fSetVelocity: LREAL

Setpoint speed in technical units per second.

Outputs (VAR_OUTPUT)

bBusy : BOOL (Default: FALSE)

With TRUE the block indicates that it is active and that it writes setpoints to the axis.

bCommandAborted : BOOL (Default: FALSE)

If this is TRUE, the block was aborted by another.

bError : BOOL (Default: FALSE)

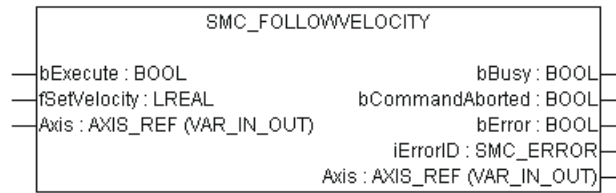
TRUE indicates that an error has occurred in the function block.

iErrorID : SMC_Error (INT)

Error number

3.2.5.3 SMC_FollowVelocity

This POU writes speed setpoints to the axis and during this process does not carry out any checks.



Inputs (VAR_INPUT)

bExecute: BOOL

By a rising edge the block starts to write the setpoints entered in fSetPosition to the axis data structure. It remains active until it is interrupted by another block (e.g. MC_Stop).

fSetVelocity: LREAL

Setpoint speed in technical units per second.

Outputs (VAR_OUTPUT)

bBusy : BOOL (Default: FALSE)

With TRUE the block indicates that it is active and that it writes setpoints to the axis.

bCommandAborted : BOOL (Default: FALSE)

If this is TRUE, the block was aborted by another.

bError : BOOL (Default: FALSE)

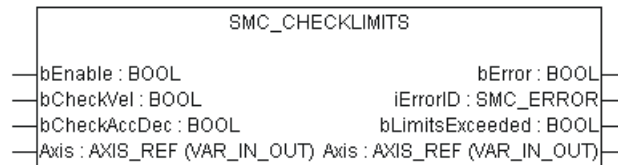
TRUE indicates that an error has occurred in the function block.

iErrorID : SMC_Error (INT)

Error number

3.2.5.4 SMC_CheckLimits

This block checks whether the current setpoints of the drive exceed the peak values set in the control system and indicates this via output bLimitsExceeded.



Inputs (VAR_INPUT)

bEnable: BOOL

During bEnable is set the check is carried out.

bCheckVel: BOOL

If this is set, the setpoint speed is checked.

bCheckAccDec: BOOL

If this is set, the setpoint acceleration and deceleration are checked.

Outputs (VAR_OUTPUT)

bError : BOOL (Default: FALSE)

TRUE indicates that an error has occurred in the function block.

iErrorID : SMC_Error (INT)

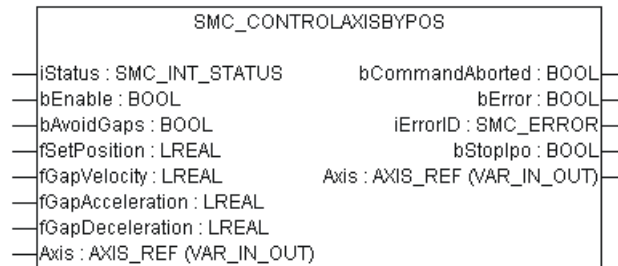
Error number

bLimitsExceeded : BOOL (Default: FALSE)

TRUE means that the current setpoints exceed the values defined in the control configuration and saved under fSWMaxVelocity, fSWMaxAcceleration or fSWMaxDeceleration.

3.2.5.5 SMC_ControlAxisByPos

This block writes setpoint positions in a drive structure and monitors it with regard to step-changes.



Inputs (VAR_INPUT)

iStatus: SMC_INT_STATUS

Status of the interpolation block. Is connected to the output of SMC_Interpolator with the same name.

bEnable: BOOL

If this input is set to "1", the function block starts to control the axis. It remains active until this input is reset to "0".

bAvoidGaps: BOOL

TRUE: Block monitors the position and speed. If the speed exceeds the limit value fSWMaxVelocity (configured in the drive dialog under "Maximum values") saved in the axis, the block sets the output bStoplpo, moves the axis to this position using the parameters fGapVelocity, fGapAcceleration and fGapDeceleration, and deletes the output bStoplpo again.

fSetPosition: LREAL

Set position of the axis. Typically an output of a transformation block.

fGapVelocity, fGapAcceleration, fGapDeceleration: LREAL

Motion parameters for bridging a step-change.

Outputs (VAR_OUTPUT)

bCommandAborted : BOOL (Default: FALSE)

If this is TRUE, the block was aborted by another.

bError : BOOL (Default: FALSE)

TRUE indicates that an error has occurred in the function block.

iErrorID : SMC_Error (INT)

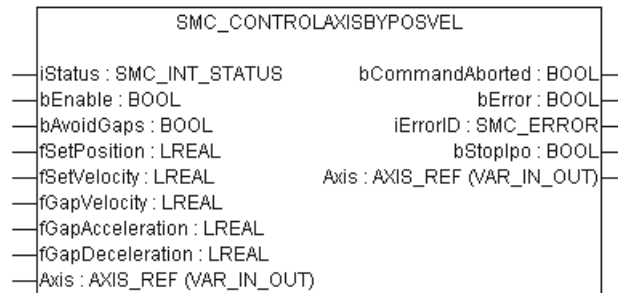
Error number

bStoplpo : BOOL (Default: FALSE)

If TRUE is set the block has detected a speed or position step change and currently is carrying out a positioning process to the new position. Therefore this output should be connected to the EmergencyStop input of the SMC_Interpolator, so that the interpolator waits until the axis is in the correct position.

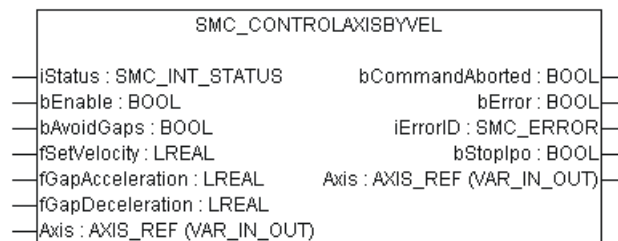
3.2.5.6 SMC_ControlAxisByPosVel

This block has a similar function as SMC_ControlAxisByPos, only that the speed can be specified additionally.

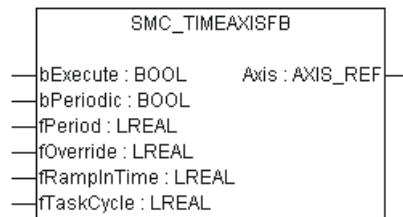


3.2.5.7 SMC_ControlAxisByVel

This block has a similar function as SMC_ControlAxisByPos, only that the axis is not controlled via the position, but via the speed.

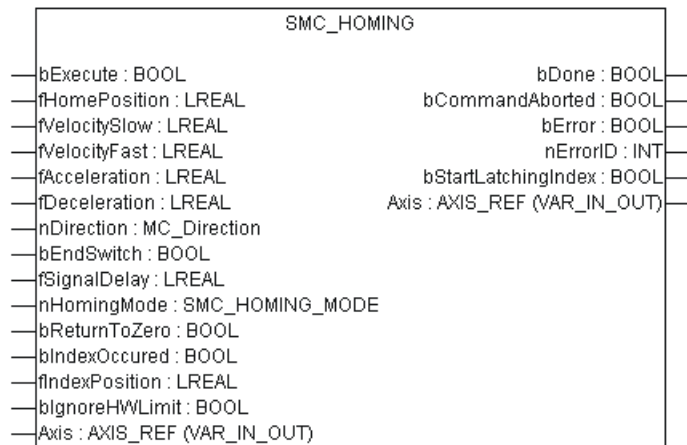


3.2.6 Virtual time axis (SMC_TimeAxisFB)



This function block generates a time axis which it provides in the output Axis (AXIS_REF). With a rising edge in the input bExecute the set position of the time axis starts to increment in seconds, beginning with 0. If the input bPeriodic is set, it starts again with 0 when the time fPeriod is reached. The input fOverride specifies a time multiplier that is set to 1 by default. A 2 would mean that the time would elapse twice as quickly. fRampInTime defines the time the block has to ramp to the new override after the new setpoints have been accepted. In the input fTaskCycle the cycle time of the task from which the block is called has to be specified in seconds.

3.2.7 Homing via digital hardware inputs (SMC_Homing)



This function block is able to carry out the homing of an axis. For this a BOOL value which typically is a hardware input acts as a switch-on.

If the block has been started with a rising edge in bExecute, by the use of the block fVelocityFast it moves the axis in the direction specified by nDirection until bEndSwitch is FALSE, i. e. until the reference switch is closed. Then the axis is decelerated and traversed in the opposite direction with fVelocitySlow. In the position where the reference switch opens (bEndSwitch = TRUE) the home position is set (fHomePosition) and the drive is stopped.

Inputs (VAR_INPUT)

bExecute : BOOL (Default: FALSE)

The drive reference run will be started on a rising edge.

fHomePosition : REAL

Specification of the absolute position on the home position [u].

fVelocitySlow, fVelocityFast : REAL

Setpoint speeds for phases 1 and 2 in [u/s].

fAcceleration, fDeceleration : REAL

Setpoint acceleration and deceleration in [u/s²].

nDirection : MC_Direction (Default: negative)

Direction of the homing: permissible values: positive/negative.

bEndSwitch : BOOL (Default: TRUE)

Reference switch: TRUE (open), FALSE (closed).

fSignalDelay : REAL (Default: 0.0)

Transmission time of the reference switch in s. If a time >0 is specified, the block does not use the position as home position where bEndSwicth became TRUE, but the position that the axis fSignalDelay had a few seconds before.

nHomingMode : SMC_HOMING_MODE (Default: FAST_BSLOW_S_STOP)

► FAST_BSLOW_S_STOP:

- The drive is traversed in the specified direction with the speed fVelocityFast (FAST) until the input bEndSwitch becomes FALSE, then it is stopped and moved in the opposite direction with fVelocitySlow (BSLOW) until bEndSwitch is TRUE again. In this position the home position is set (S) and the process is stopped (STOP).

► FAST_BSLOW_STOP_S:

- In contrast to FAST_BSLOW_S_STOP, after retracting a stop takes place first, and then the home position is set.

► FAST_BSLOW_I_S_STOP:

- In contrast to FAST_BSLOW_S_STOP, after retracting an index pulse (bIndexOccured=TRUE) is set here, and is position flIndexPosition which is set as home position is awaited. Only then a stop takes place.

► FAST_BSLOW_S_STOP/ FAST_BSLOW_STOP_S / FAST_BSLOW_I_S_STOP:

- These modes operate exactly as the ones described above, with the difference that no reverse movement takes place when the reference switch is reached, but the travel movement continues in the same direction. Please note that for safety reasons the input bIgnoreHWLimits in this mode has to be FALSE.

bReturnToZero: BOOL (Default: FALSE)

If this flag is set, the block carries out the positioning process to the zero point after the procedure defined in nHomingMode has been completed.

bIndexOccured: BOOL (Default: FALSE)

Only for nHomingMode FAST_BSLOW_I_S_STOP: Indicates whether index pulse has occurred.

flIndexPosition: REAL (Default: 0.0)

Only for nHomingMode FAST_BSLOW_I_S_STOP: Latched position of the index pulse.

If this input is TRUE, the hardware check of the limit switches is interrupted. Use this option if you use the same physical switch for the hardware limit switch and reference switch.

bIgnoreHWLimit: BOOL (Default: FALSE)

If this input is TRUE, the hardware check of the limit switches is interrupted. Use this option if you use the same physical switch for the hardware limit switch and reference switch.

Outputs (VAR_OUTPUT)

bDone : BOOL (Default: FALSE)

If TRUE is set, the drive is referenced and is at standstill.

bCommandAborted : BOOL (Default: FALSE)

If this is TRUE, the command was aborted by another.

bError : BOOL (Default: FALSE)

TRUE indicates that an error has occurred in the function block.

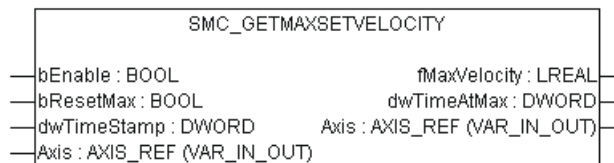
nErrorID: SMC_ERROR

Error number

3.2.8 Diagnostic blocks

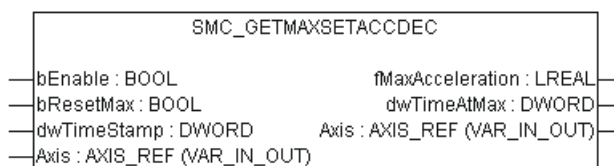
3.2.8.1 SMC_GetMaxSetVelocity

This function block is able to measure the amount (setpoint) speed of an axis. The measurement is carried out if bEnable is TRUE and is reset to 0 as long as bResetMax is TRUE. dwTimeStamp offers the possibility to accept or output any DWORD (e.g. call counter) with a new maximum value.



3.2.8.2 SMC_GetMaxSetAccDec

This block operates similar to SMC_GetMaxSetVelocity and determines the acceleration or deceleration with the greatest amount.



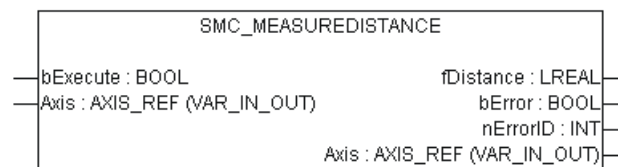
3.2.8.3 SMC_GetTrackingError

This POU measures the current and maximum following error, depending on the dead time which can be caused by communication via a fieldbus and which is specified in cycle numbers (byDeadTimeCycles). As in the case of SMC_GetMaxSetVelocity, a time stamp (dwTimeStamp) can be used to measure the time of the maximum.



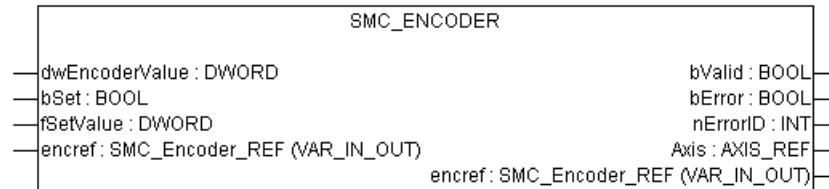
3.2.8.4 SMC_MeasureDistance

If it is used for a rotary axis, this POU measures the path that has been covered and, in doing this, takes the change into consideration.



3.2.9 ENCODER (SMC_Encoder)

The control configuration serves to implement and configure encoders into an axis group. The created data structure of SMC_Encoder_REF type must be processed by an instance of the SMC_ENCODER POU. This outputs an AXIS_REF data structure which can serve as input for all other function blocks (e.g. MC_CamIn, MC_GearIn, MC_TouchProbe) as soon as the bValid output confirms the validity of the data.



Inputs/outputs (VAR_IN_OUT)

encref: SMC_Encoder_REF: VAR_IN_OUT

Pointer to the operating data of the encoder.

Inputs (VAR_INPUT)

dwEncoderValue: DWORD

Actual value of the encoder

bSet: BOOL

Set the value at the EncoderValue input to the SetValue input.

fSetValue: DWORD

Position value to be accepted.

Outputs (VAR_OUTPUT)

bValid: BOOL

TRUE indicates that the data is valid.

bError : BOOL

TRUE indicates that an error has occurred in the function block.

nErrorID: INT

Error number

Axis : AXIS_REF

The POU writes the correspondingly scaled position data of the encoder axis into this axis data structure.

Dead time compensation and filtering for a master/slave connection

The actual positions read-in by an encoder have a certain runtime for control. The setpoints calculated in the control have a certain runtime as well until they get effective in the drive. The SMC_Encoder_POU offers the possibility to compensate these runtimes by an extrapolation (advance calculation) of the read-in encoder values. Here, it is required to filter the position values before extrapolation. A corresponding filter function is also included in the POU.

In the SMC_Encoder_REF data structure, the dead time compensation and filtering of the input signal can be activated by writing the elements fDeadTimeCycle and usiFilterDepth.

fDeadTimeCycle: Number of PLC cycles via which the extrapolation takes place.

Depending on the fieldbus, encoder and drive type used, a value between 5.0 and 7.0 cycles must be entered. The exact value must be detected for the runtime at the machine.

usiFilterDepth: Number of position values via which filtering takes place.

Here, the setting also depends on the quality or accuracy of the read-in actual position and thus on the encoder used. As starting value, 5 position values are sensible here and an optimisation is also required. The max. filter depth is 16 position values.



Note!

A too low filter depth causes a disturbed behaviour of the controlled slave axis.
A too high filter depth causes a strong smoothing of the speed variations of the encoder.

3.2.10 Visualisation templates

For each of the two types of drives (linear/rotary) a visualisation template is contained in the library, which can be linked to an axis structure (AXIS_REF) and visualises the actual position of the drive:

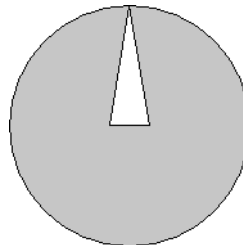
LinDrive



For a linear drive the illustration above is shown. The slide is placed relative to the lower and upper position limit according to its current position and is coloured blue as soon as it is control. The requirement for the use of the template is that the parameters fSWLimitPositive and fSWLimitNegative are set.

The LinDrive_V template represents the drive in vertical form.

RotDrive



For a rotary drive the illustration above is shown. The current position is represented by the position of the arrow and is coloured blue as soon as the drive is controlled. The requirement for the use of the template is that the parameter fPositionPeriod is set.

3.3 Drive element <BusInterfaceName>Drive.lib

- ▶ Drive elements are responsible for communication between the IEC program, in particular the AXIS_REF structures, and the drives. These libraries are usually supplied by the manufacturer of the control system and have to be integrated in the project.
- ▶ DummyDrive.lib is an example of drive element libraries and is supplied with the SoftMotion libraries. Even if this library does not operate real drives, it functions according to the same principle.

3.3.1 LenzeCANdrive.lib

The LenzeCANdrive.lib function library supports the drive types (Drive.wControlType) Servo Drive 9400 StateLine and the ECSxM axis module (Motion):

ECSxM axis module (Motion)

Apart from control and status signals, the process data contain the current set position and actual position.

Servo Drive 9400 StateLine

The process data can be configured. Apart from control and status signals they can contain the following values:

- ▶ Set position and actual position
- ▶ Setpoint speed and actual speed
- ▶ Setpoint torque and actual torque

3.3.1.1 Properties

- ▶ RegulatorOn, DriveStart
- ▶ Error display and acknowledgement
- ▶ Reading and writing of SoftMotion and drive parameters:
 - Access via index 0xaabb_{hex} and subindex 0xcc_{hex} with the length 0xdd_{hex} in bytes (only required for writing), or
 - Use of MC_Read/Write(Bool)parameters with the parameter number ddaabbcc_{hex}, or
 - Use of the FBs LenzeECSReadParameter and LenzeECSWriteParameter to directly access to Lenze codes, or
 - Use of the FBs SMC_ReadCANParameter and SMC_WriteCANParameter to address standard CAN objects via index/subindex.
- ▶ Reading of drive parameters (as string) with LenzeECSReadString
- ▶ Optional ratio factors (dwRatioTechUnitsDenom/iRatioTechUnitsNum)
- ▶ Linear or rotary axes
- ▶ Control modes: position, velocity (9400), torque (9400)
 - Use SMC_SetControllerMode for changing the mode.
- ▶ Drive-internal homing
 - ECSxM: configure C3010, C0935, C0936
Note: During the homing the current position is not reported by the ECS drive!
 - 9400: configure 6098_{hex}, 6099_{hex}, 609A_{hex}
- ▶ Latching: 1 channel (trigger number = 1), ECSxM only
- ▶ ECSxM: depending on the setting in code C3175, the hardware limit switches are monitored by the control system (C3175 = 3) or by the drive.
- ▶ Configuration from one file possible
- ▶ Configuration from dialogs in the PLC configuration possible
- ▶ Supported sync generators (to be set in the PLC configuration, AxisGroup): PLC, 1. drive, sync device

3.3.1.2 Global variables

Variable	Data type	Description
g_udiTimeDistanceSyncSend	UDINT	Time between the transmission of a sync signal and the first PDO (default = 0)

3.3.1.3 The structure of LenzeECS_AXIS_REF

Variable	Data type	Description
byDriveState	BYTE	Status of the drive (internal use)
wStatusWord	WORD	Status word C0150 (ECSxM) / 6041 _{hex} (9400)
wStatusWord2	WORD	Status word2 C0155 (ECSxM)
wControlWord	WORD	Control word C0135 (ECSxM) / 6040 _{hex} (9400)
wStatusWordOld	WORD	Last status word (internal use)
dwSetPosition	DWORD	Target position sent to the drive (read only)
byStatusInfo	BYTE	Bits 8 ... 11 of the status word (ECSxM)
byErrAcknCounter	BYTE	Error counter (internal use)
bHWLimitsActive	BOOL	TRUE if a limit switch is activated (ECSxM)
fLimitSwitchDeceleration	LREAL	Speed deceleration for stopping at the limit switch (if 0, no ramp is available), ECSxM only
mcstopLS	MC_Stop	Internal use
strConfigFile	STRING	Complete file name and path of the configuration file
acit	...	Internal use
txList, rxList, txPDOs, rxPDOs	...	Internal use
byActOpMode	BYTE	Object 6061 _{hex} (9400)
bySetOpMode	BYTE	Object 6060 _{hex} (read only), 9400 only
bOldReference	BOOL	Internal use
byDigitalInputs, byDigitalOutputs	BYTE	Digital inputs (60FD _{hex}) and outputs (60FE _{hex} /1), 9400 only
crap	SMC_CANReadAllParams	Internal use
pParameterlist	POINTER TO CAN_InitTelegram	Parameter list (internal use)
wp	LenzeECSWriteParameter	Written parameters ECSxM (internal use)
rp	LenzeECSReadParameter	Read parameters ECSxM (internal use)

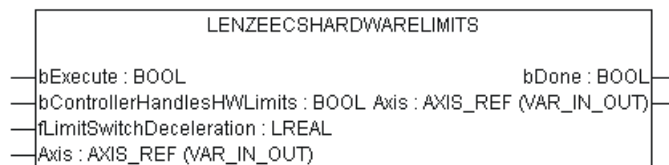
3.3.1.4 Hardware limit switch (ECSxM axis module)

The response behaviour to hardware limit switches is configured via C3175 within the drive. If C3175 $\neq 3$, the drive responds to the hardware limit switches. In this case it could be impossible for the software to move an axis beyond its limit values, as the drive blocks this process. Only in the case of the setting C3175 = 3, not the drive, but the controller is responsible for the hardware limit switches:

If bControllerHandlesHWLimits (default setting) is set, and the axis is operated with a hardware limit switch, the controller stops the axis and transfers it to error stop. After a MC_Reset has been carried out on the drive, it can be moved in the direction leading away from the limit switch, while the other direction is blocked. As soon as the drive moves in the opposite direction, an error is actuated.

Depending on the deceleration value, the stop is carried out by means of a ramp (fLimitSwitchDeceleration > 0) or immediately (other setting/default setting).

In the LenzeECSDrive.lib library a specific FB is provided to set all values required for this purpose:



If bExecute has a rising edge, the new values are set.

3.3.1.5 Configured parameters during start-up

ECSxM axis module (Motion)

Parameter / code	Value set during start-up		Description
	Sync generator: PLC / sync device	Sync generator: 1. drive	
C0043	0	0	Reset error
C0352	0	0	CAN slave
C0353/1	0	0	Addressing mode
C0356/3	Cycle time	Cycle time	PDO cycle
C0356/4	0	0	Delay time PDO
C1120	1	0	Sync mode
C1121	Cycle time	Cycle time	Sync cycle
C0367	128	129	Sync Rx-ID
C0368	128	128	Sync Tx-ID
C1123	Cycle time/2	Cycle time/2	Sync time slot
C0369	0	Cycle time	Sync generator cycle
C0366	1	1	Sync response
C5000	7	7	Operating mode
C0003	1	1	Save parameters
C0135	0	0	Control word

Servo Drive 9400 StateLine

Parameter / index [hex] / code	Value set during start-up		Description
	Sync generator: PLC / sync device	Sync generator: 1. drive	
6040 _{hex}	0		Control word
1006 _{hex} 60C2 _{hex} C01121, C01130	Cycle time		Sync cycle
1005 _{hex}	80 _{hex} (128 _{dec})	40000080 _{hex} (1073741952 _{dec})	Addressing mode
1400 _{hex} ... 1AFF _{hex}			PDO mapping
6092 _{hex}	10000 _{hex} (65536 _{dec})		feedconst_e4

3.3.1.6 CAN communication times

The following values apply to the ECSxM axis module (Motion) and the 9400 Servo Drives StateLine, provided that only one pair of set values and actual values, respectively, is transmitted (e.g. SetPosition, ActPosition). If cyclic data transmission is additionally used, the data traffic via CAN increases. Therefore a smaller number of drives can be activated in the same cycle time.

► Basic utilisation:

Telegram	Data bytes	Bit length	125 kbps	250 kbps	500 kbps	1 Mbps
Sync	0	47	0.376 ms	0.188 ms	0.094 ms	0.047 ms
SDO	8	111	0.888 ms	0.444 ms	0.222 ms	0.111 ms
Total			1.264 ms	0.632 ms	0.316 ms	0.158 ms

► Per drive:

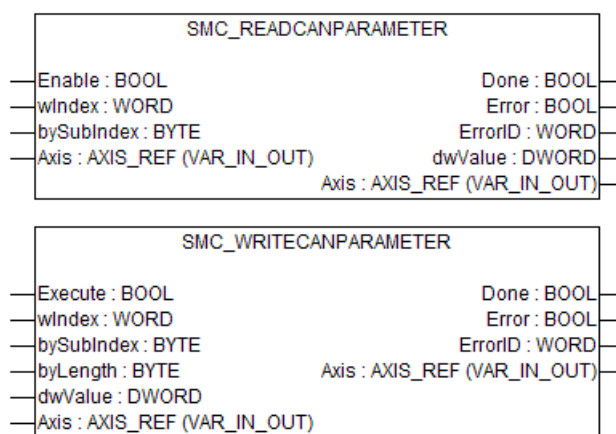
Telegram	Data bytes	Bit length	125 kbps	250 kbps	500 kbps	1 Mbps
Control word, target position	8	111	0.888 ms	0.444 ms	0.222 ms	0.111 ms
Status word, current position	8	111	0.888 ms	0.444 ms	0.222 ms	0.111 ms
Total			1.776 ms	0.888 ms	0.444 ms	0.222 ms

► The following table specifies the maximum number of drives per cycle time in this connection:

Max. number of drives/cycle	125 kbps	250 kbps	500 kbps	1 Mbps
1 ms	0	0	1	3
2 ms	0	1	3	6
3 ms	1	2	5	10
4 ms	1	3	6	13
5 ms	2	4	8	17
6 ms	2	5	10	20
8 ms	3	6	13	27

3.3.2 SM_CAN.lib

- ▶ For each CAN drive connected an individual driver is required.
- ▶ However, if this is set in the cfg file, the baud rate and the number of the CAN controller (starting with 0) can be defined for all CAN drives in the control configuration under the axis group dialog. In order to keep the bus deterministic, either I/Os or drives, but never both at the same time, are operated for each CAN channel. If the 3S-CANopen library is used, it automatically takes the first CAN controllers, and thus a different one can be reserved for the axis group.
- ▶ All CAN libraries created by 3S are based on the SM_CAN.lib library. It contains two blocks which are practical for the user, as they can be used in a simple way to read out or write parameters of the drive: SMC_ReadCANParameter and SMC_WriteCANParameter. With regard to their operating mode they are similar to the blocks MC_ReadParameter and MC_WriteParameter:



Input/output (VAR_IN_OUT)

Axis : AXIS_REF (VAR_IN_OUT)

Drive the parameter of which is to be read out/written.

Inputs (VAR_INPUT)

Enable : BOOL

TRUE: parameter is read continuously.

Execute : BOOL

Rising edge: one-time writing of parameter is activated.

wIndex, bySubIndex : BYTE

Index and subindex of the parameter that is to be read or written.

Outputs (VAR_OUTPUT)

dwValue : DWORD

Value read or value to be written.

Done : BOOL

TRUE: action completed successfully.

Error : BOOL

TRUE: error has occurred.

ErrorID : WORD

Error number according to SMC_Error.

3.4 Variables of the AXIS_REF structure

For each drive created in the »PLC Designer« control configuration a variable of the AXIS_REF structure is created during the compilation process (defined in the SM_DriveBasic.lib, see chapter [3.2](#) (26)). This structure provides the interface between the application and the drive interface. It serves to exchange both cyclic and acyclic data.

Most variables of the structure have no importance for the user, but are used internally by the system. The user should always use function blocks and should never directly access the structure, at least not in the writing mode.

No.	Name	Data type	Init value	Description
1000	nAxisState	INT	Standstill	Axis status: 0: power_off 1: errorstop 2: stopping 3: standstill 4: discrete_motion 5: continuous_motion 6: synchronized_motion 7: homing
1001	wControlType	WORD	PLC config*	Digit showing the components of the structure which are transmitted and received cyclically. 0: defined by param 1002 ... 1008 1: SetTorque 2: SetVelocity,ActVelocity 3: SetVelocity,ActPosition 4: SetPosition,ActPosition 5: SetVelocity,SetPosition,ActVelocity,ActPosition 6: SetVelocity
1002 1003 1004	wCyclicDataS1 wCyclicDataS2 wCyclicDataS3	WORD	PLC-Config* or init-FB of <BusInterfaceName> Drive.lib	Number of the 3S parameters that are to be transmitted per cycle
1006 1007 1008	wCyclicDataR1 wCyclicDataR2 wCyclicDataR3	WORD	PLC-Config* or init-FB of <BusInterfaceName> Drive.lib	Parameter numbers of the 3S parameters that are to be received per cycle
1010	bRegulatorOn	BOOL	FALSE	Connect/switch off controller (power)
1011	bDriveStart	BOOL	FALSE	Activate/deactivate brake
1012	bCommunication	BOOL	FALSE	TRUE: drive responds
1013	wCommunication State	WORD	0	Internal use
1015	bRegulatorRealState	BOOL	FALSE	Status of the controller
1016	bDriveStartRealState	BOOL	FALSE	Status of the brake
1020	wAxisGroupId	WORD	PLC config*	Index of the axis group in the configuration
1021	wDriveId	WORD	PLC config*	Node number of the drive on the fieldbus
1022	iOwner	INT	0	ID number of the current owner (FB)
1023	iNoOwner	INT	0	Number of the previous and current owners

No.	Name	Data type	Init value	Description
1024	bMovedInThisCycle	BOOL	FALSE	Has drive been moved during this IEC cycle
1025	fTaskCycle	REAL	PLC config*	Task cycle time in s
1026	bRealDrive	BOOL	PLC config*	TRUE: generated by config; FALSE: generated by IEC program
1030	bError	BOOL	FALSE	Error has occurred
1031	wErrorID	WORD	0	Error number
1032	bErrorAckn	BOOL	FALSE	Errors confirmed
1035	fbeFBError	ARRAY [0..g_SMC_NUMBER _FB_ERRORS] OF SMC_FBERROR	0	FB error number memory
1051	dwRatioTechUnits Denom	DWORD	PLC config*	Conversion of technical units to increments: denominator
1052	iRatioTechUnits Num	INT	PLC config*	Conversion of technical units to increments: numerator
1053	nDirection	MC_Direction	Positive	-1: negative (fSetVelocity < 0) 1: positive
1054	fScalefactor	REAL	1	Conversion of bus unit to technical unit per unit received on the bus
1055	fFactorVel	REAL	1	Conversion of bus unit in techn. Inherited/s
1056	fFactorAcc	REAL	1	Conversion of bus unit in techn. unit/s ²
1057	fFactorTor	REAL	1	Conversion of bus unit in Nm or N
1058	fFactorJerk	REAL	1	Conversion of bus unit in techn. unit/s ³
1060	iMovementType	INT	1	0: rotary (modulo) 1: linear
1061	fPositionPeriod	REAL	1000	Length of the period for rotary drives in techn.units
1091	byControllerMode	BYTE	3	1: torque control 2: speed control 3: position control
1092	byRealController Mode	BYTE	0	Actual control mode
1100/1	fSetPosition	REAL	0	Specified position in techn. units
1101	fActPosition	REAL	0	Current position in techn. units
1105	fAimPosition	REAL	0	Target position (for some MC_FBs)
1106	fMarkPosition	REAL	0	Internal position marker
1107	fSavePosition	REAL	0	Internal position at cycle start
1110,11	fSetVelocity	REAL	0	Specified speed in techn. units/ sec
1111,10	fActVelocity	REAL	0	Current speed in techn. units/sec
1112,9	fMaxVelocity	REAL	100	Is only used internally
1113	fSWMaxVelocity	REAL	100	Maximum speed for implicit movements in techn. units/sec
1115	bConstantVelocity	BOOL	FALSE	Axis is moving at a constant velocity.
1116	fMarkVelocity	REAL	0	Internal speed mark

No.	Name	Data type	Init value	Description
1117	fSaveVelocity	REAL	0	Internal speed at cycle start
1120	fSetAcceleration	REAL	0	Specified acceleration in techn. units/sec ²
1121	fActAcceleration	REAL	0	Current acceleration in techn. units/sec ²
1122,13	fMaxAcceleration	REAL	100	Maximum acceleration in techn. units/sec ²
1123	fSWMaxAcceleration	REAL	100	Maximum acceleration for implicit movements in techn. units/sec
1125	bAccelerating	BOOL	FALSE	Axis is currently accelerating
1126	fMarkAcceleration	REAL	0	Internal acceleration mark
1127	fSaveAcceleration	REAL	0	Internal acceleration at cycle start
1130	fSetDeceleration	REAL	0	Specified deceleration in techn. units/sec ²
1131	fActDeceleration	REAL	0	Current deceleration in techn. units/sec ²
1132,15	fMaxDeceleration	REAL	100	Maximum deceleration in techn. units/sec ²
1133	fSWMaxDeceleration	REAL	100	Maximum deceleration for implicit movements in techn. units/sec
1135	bDecelerating	BOOL	FALSE	Axis is currently decelerating
1137	fSaveDeceleration	REAL	0	Internal deceleration at cycle start
1140	fSetJerk	REAL	0	Specified jerk in techn. units/sec ³
1141	fActJerk	REAL	0	Current jerk in techn. units/sec ³
1142,16	fMaxJerk	REAL	100	Maximum jerk in techn. units/sec ³
1143	fSWMaxJerk	REAL	100	Maximum jerk for implicit movements in techn. units/sec
1146	fMarkJerk	REAL	0	Internal jerk mark
1147	fSaveJerk	REAL	0	Internal jerk at cycle start
1150	fSetCurrent	REAL	0	Specified amperage (A)
1151	fActCurrent	REAL	0	Current amperage (A)
1152	fMaxCurrent	REAL	100	Maximum amperage (A)
1153	fSWMaxCurrent	REAL	0	Maximum amperage for implicit movements in techn. units/sec
1160	fSetTorque	REAL	0	Specified torque in Nm or N (linear)
1161	fActTorque	REAL	0	Current torque in Nm or N (linear)
1162	fMaxTorque	REAL	0	Maximum torque in Nm or N (linear)
1200,2	fSWLimitPositive	REAL	0	Position limit of positive direction in techn. units
1201,3	fSWLimitNegative	REAL	0	Position limit of negative direction in techn. units
1202	fCaptPosition	REAL	0	Capture position in techn. units
1204	bSWEndSwitchActive	BOOL	FALSE	SW limit switch active

No.	Name	Data type	Init value	Description
1205	bSWLimitEnable	BOOL	FALSE	Activate software limit switch
1206	bHWLimitEnable	BOOL	FALSE	Activate/deactivate hardware limit switch
1207	bCaptureOccurred	BOOL	FALSE	Capture has taken place (confirm by writing)
1208	bStartCapturing	BOOL	FALSE	Start/stop capture for current trigger
1210	bStartReference	BOOL	FALSE	TRUE: start homing run
1211	fReference	REAL	0	Referenzposition
1215	fOffsetPosition	REAL	0	Zero shift
1220	fFirstCapturePosition	REAL	0	Window start position for capture
1221	fLastCapturePosition	REAL	0	Window end position for capture
1222	tiTriggerInput	TRIGGER_REF		Writing of the capture input
1223	bCaptureWindow Active	BOOL	FALSE	Capture limited to window
1230	dwPosOffsetFor Residuals	DWORD	0	Internal variable for remainder considered
1231	dwOneTurn	DWORD	0	Internal variable for remainder considered
1232	fLastPosition	REAL	0	Internal variable for remainder considered
1234	iRestNumerator	INT	0	Internal variable for remainder considered
1235	iTurn	INT	0	Internal variable for remainder considered
1236	dwBusModuloValue	DWORD	0	Internal variable for remainder considered
1237	dwPosOffsetFor ResidualsHoming	DWORD	0	Internal variable for remainder considered
1300	bDisableDriveInAxis Group	BOOL	FALSE	Delete drive from axis group
1301	bErrorDuringStartup	BOOL	FALSE	Is set if error during start-up occurs
	pMS	POINTER TO BYTE	0	Pointer to drive-specific structure <BusInterfaceName>_AXIS_REF

Each AXIS_REF structure variable acts according to the PLCopen specification "Function blocks for motion control", version 1.0.

3.5 Parameterisation of the drive

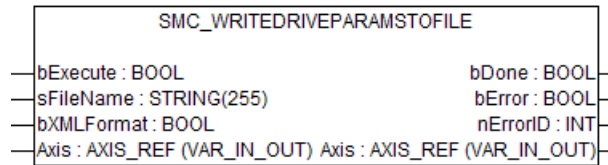
Many important configuration data are saved within the drive. SoftMotion offers the possibility of defining parameter values in the control configuration, which are transferred during the acceleration phase (see chapter [CANDrive \(22\)](#)), however, this is difficult for the normal user, as he usually does not know which parameters are to be transferred and which value they are to receive. Thus for initial commissioning (also in the case of several machines of the same series) and if drives are replaced, often the drive-specific tools are required. Therefore a function is integrated within SoftMotion, which, at least to some extent, carries out this task.

Therefore there are the following variants:

- ▶ The user configures the drive with the tool of the drive manufacturer. He goes online with a SoftMotion project where the drives are registered and calls a POU which reads all drive parameters and saves them in a XML file which the user reads back from the control system and imports into the corresponding configuration dialog. Then all required parameters are saved within the project and are transferred at every start. Neither for the initial commissioning of further machines nor for the replacement of a defective drive the commissioning tool would be required.
- ▶ The user configures the drive with the tool by the drive manufacturer. In his application he provides the possibility to the user that he can save the parameters of the drive in an ASCII file on the control system via an FB. In contrast to A, the parameters are stored within a file on the control system and not within the application.

Both solutions have advantages and disadvantages: In order to alter a parameter later on in the case of A, the project has to be downloaded again. In the case of B the change can be effected, but it has to be observed that for every new commissioning either the parameter files have to be stored on the control system, or the drive commissioning tool has to be used.

3.5.1 SMC_WriteDriveParamsToFile



This POU reads out all configuration parameters of the drive and saves them in a file. As it carries out a data access which possibly blocks the processing of the application for several ms, it usually may not be called in the Motion task, but should be carried out in a task with a lower priority.

The POU is informed by the drive element about the parameters which are to be read out; the driver can either receive this information from the drive itself, or it contains a standard parameter list (for CAN: see global variable list of the standard 3S drivers). For CAN drives an individual list of the same format can be created and allocated to the drive via the following assignment:

```
<Drive>_MS.pParameterlist := ADR(<NeueList>);
```

Input/output (VAR_IN_OUT)

Axis : AXIS_REF

Drive the parameters of which are to be read out.

Inputs (VAR_INPUT)

bExecute: BOOL

With TRUE the POU is started.

sFileName: STRING(255)

File name.

bXMLFormat: BOOL

If this variable is TRUE, the file is created in the XML format (and can then be imported into the drive dialog), otherwise in the text format (which can be stored on the control system and be imported and transmitted during the acceleration by the drive element).

Outputs (VAR_OUTPUT)

bDone : BOOL

Action completed.

bError : BOOL

Error occurred.

nErrorID: INT

Error number

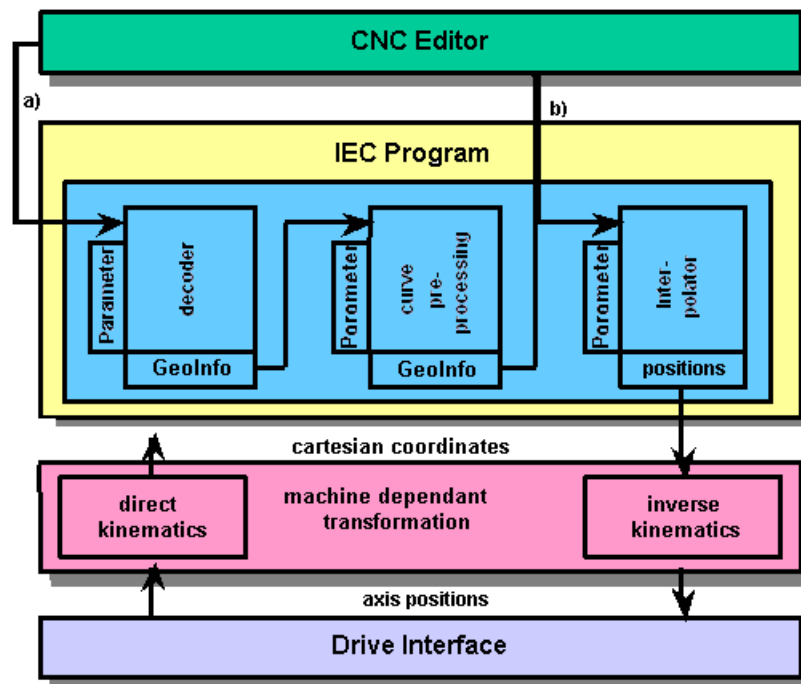
4

CNC editor

**Note!**

The CNC editor will only be available if ...

- The SoftMotion add-on has been installed and
- An automation system with the SoftMotion add-on has been selected in the Target Settings dialog box.



The CNC editor in the »PLC Designer« supports the programming of multidimensional movements of both graphics and textual format based on the CNC language of **DIN 66025**.

In principle, movements with up to 9 dimensions can be programmed, whereby only two dimensions are interpolated in a non-linear fashion. In other words, straight lines, circles, circular arcs, parabolas and splines can be programmed in two dimensions and the other directions are simply interpolated in a linear fashion.

The »PLC Designer« creates a global data structure (CNC data) automatically for every path generated. This structure is available in the IEC program. This can be achieved in various ways:

- The CNC program is stored as an **Array of G-code words** and is decoded at PLC program runtime using a decoder POU in order to make it available for the individual path objects (GEOINFO structure objects). They can then be edited using path preprocessing modules (e.g. tool radius correction) before being interpolated, transformed, and transferred back to the Drive Interface for communication with the hardware. For more information, see the POUs in **SM_CNC.lib**.

- The CNC program is written to a data structure as a list (**OUTQUEUE structure**) of **GEOINFO structure objects** and this means that it can be entered into the interpolator directly. As a result, the decoder and path preprocessing function POU no longer needs to be called as in the case of a). However, it also means that the program cannot be changed at runtime.
- The CNC program is written to the control of the **file system** using either method a) or b). Step-by-step processing is employed for read-out and conversion at runtime. This method is primarily suitable for large programs which the memory is unable to accommodate in full.

4.1 Supported and extended elements of the CNC language of DIN66025

In order to give programmers an easy means of creating geometric paths, SoftMotion supports parts of the CNC language of DIN 66025. As the entire SoftMotion concept is embedded in the much more powerful language of IEC 61131, only the parts of DIN 66025 used to create paths are supported.

Specified structure of a CNC program

```
%  
N<Number> G<Number> ...  
...  
N<Number> G<Number> ...  
  
Example:  
%1 example  
N10 G01 X100 Y100 E100 F100 E-200  
N20 G01 Z40 F20  
N30 G03 X-100 R200 F100  
...
```

A »SoftMotion« CNC program has to start with a percentage sign (%). A program name can appear on the same line but has to be separated by a blank space or a TAB character. The actual CNC program comprises a number of records.

The number of words in a record (line) is not limited.

A word comprises a letter (word identifier) followed by a number (e.g. **G01**; see [Word identifiers](#) (65)). The format is not case-sensitive and leading zeros are ignored (G01 = g1).

The first word in every record represents the number of the record (N<Number>), e.g. "N01", comprising the record's letter and number. The words in a record are separated by blank spaces or TAB characters. The record number is currently of no relevance but is required for reasons of conformity. The remaining words in each record are processed from right to left. With the exception of the travel command (G<Number>, e.g. **G02**; see [Travel commands](#) (66)), when each word is processed, the word number is written to a variable in accordance with its letter and this variable is then accessed by the travel command during subsequent execution.

Each record can only contain one travel command which must appear directly to the right of the record number. If a record does not contain a travel command, the one used in the last record is added automatically.

Every travel command can be understood as a path object (straight line, circular arc, etc.). The velocity at which the path objects are interpolated is determined essentially by the setpoint velocity, acceleration and deceleration. The interpolator must ensure that these limit values are not exceeded. The velocity during the transition from one object to the next is determined in accordance with the following rules:

- ▶ One of the two objects is a positioning object (**G0**): Transition speed = 0
- ▶ The angle between the tangents of the two objects at the transition is greater than the angular tolerance: Transition velocity = 0
- ▶ Otherwise: The transition velocity is the smaller setpoint velocity of the two path objects.

In principle, a travel command initiates interpolation starting from the target position of the last travel command and ending at its own target position. The first travel command starts at the specified start position (specified in the decoder or CNC editor). If no such start position has been defined, it starts at X=0, Y=0, Z=0. It is also possible to use **G92** to set the position in the CNC program. This is possible both at the start of the CNC program (effecting the setting of the start position) and in the middle of the program, where it effects a jump of the setpoint position to the position defined with **G92**. In the event of a series of **G92** commands appearing one after the other, the last takes priority and all commands preceding it are skipped. However, if you wish to ensure that the preceding **G92** positions (for the duration of one cycle) are output, insert a **G1** command in between each of the **G92** commands, using identical coordinates. This technique can be used if the path between these points is not of any relevance and the aim is simply to move the setpoint position to the target position along the quickest possible path. SMC_ControlAxisByPos will then detect a jump to the setpoints, stop the interpolator, and interpolate each axis individually along the quickest possible path.

- ▶ **G92 X100 Y100** (set setpoint position to 100/100)
- ▶ **G1 X100 Y100** (ensure one-time output of position)
- ▶ **G92 X50 Y100** (set setpoint position to 50/100)

Activate record suppression to skip records starting with the "/" character.

Characters appearing inside a pair of round parentheses are read as comments and do not have any effect on the programmed path. Nested comments are not supported.

All numerical values except those in the travel command (**G**<Number>) and the switching point number (**H**<Number>) can be floating point numbers.

Word identifiers

Word identifier	Meaning
D	Tool radius (for correction G40-42 or angle rounding G50-51), or variable value (G36/G37)
E	Max. acceleration (> 0) / deceleration (< 0) [path units/s ²]
F	Velocity at which travel is to take place [path unit/s]
G	Travel command
H	Activate (>0)/deactivate (<0) switching point
I	X coordinate of the circle/ellipse centre (G02/G03/G08/G09), or X coordinate of the parabola-tangent intersection
J	Y coordinate of the circle/ellipse centre (G02/G03/G08/G09), or Y coordinate of the parabola-tangent intersection
K	Direction of the principle ellipse axis in the mathematical sense (0° O, 90° N, ...), or jump condition (G20), or dT1 parameter value (M-function)
L	Absolute switching point position measured from the path object start (> 0) / end (< 0), or jump destination (G20), or dT2 parameter value (M-function)
M	Additional function
O	Relative switching point position [0 ... 1], variable to be changed (G36/G37), or M-parameter data structure (M)
P	Target value of additional axis P
Q	Target value of additional axis Q
R	Circle radius (G02/G03) (alternatively to I,J), or length ratio of the secondary/principal axis (G08/G09) [0 ... 1]
S	Switch on (>0)/switch off (< 0) S-profile for linear axes 3: Z-axis, 7: P-axis, 8: Q-axis, 9: U-axis, 10: V-axis, 11: W-axis
U	Target value of additional axis U
V	Target value of additional axis V
W	Target value of additional axis W
X	X-coordinate of target point
Y	Y-coordinate of target point
Z	Target value of additional axis Z

Travel commands

Travel command	Meaning
G00	Direct movement without tool operation, positioning
G01	Linear (straight) movement with tool operation
G02	Circle (segment) clockwise
G03	Circle (segment) counter-clockwise
G05	Point of a cardinal spline
G06	Parabola
G08	Ellipse (segment) clockwise
G09	Ellipse (segment) counter-clockwise
G20	Conditional jump (to L if K<>0)
G36	Write value (D) to variable (O)
G37	Increment variable (O) by value (D)
G40	End of tool radius correction
G41	Start of tool radius correction to the left of the direction of travel
G42	Start of tool radius correction to the right of the direction of travel
G50	End of path rounding/smoothing
G51	Start of path smoothing
G52	Start of path rounding
G60	End of avoid-loop function
G61	Start of avoid-loop function
G90	Effects the shift of the coordinate system to the following coordinate values (for X/Y/Z/A/B/C/P/Q/U/V/W) as absolute values (standard mode).
G91	Effects the shift of the coordinate system by the following coordinate values (for X/Y/Z/A/B/C/P/Q/U/V/W) as relative values.
G92	Set position without movement
G98	Interpret subsequent values of I/J as absolute values
G99	Interpret subsequent values of I/J as relative to the starting point (default)



Note!

For a correct translation of the project, the **SM_CNC.lib** library must be integrated.

4.1.1 Switching points, H-function

The switching point function (or H-function) provides programmers with a means of using binary path-specific switches. The number of the switching point must always be specified first (H<Number>), followed by the position of the switching point in the object either as an absolute value via word L<Position> or as a relative value via word O<Position>. In the example below, switching point 2 is being deactivated at position X =40/Y=25 (after the first quarter of the object).

```
N90 G1 X20 Y20
N100 G1 X100 Y40 H-2 00.25
```

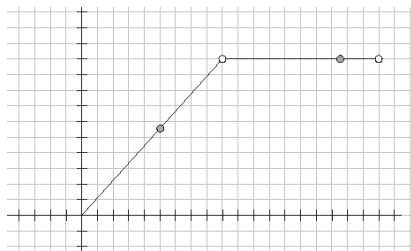


Note!

- Only a limited number of switching point switching actions (MAX_SWITCHES) can be performed for each path object.
- A switching point position can only be inserted in the CNC text editor! It will appear as a green dot on the curve in the graphics-based editor.

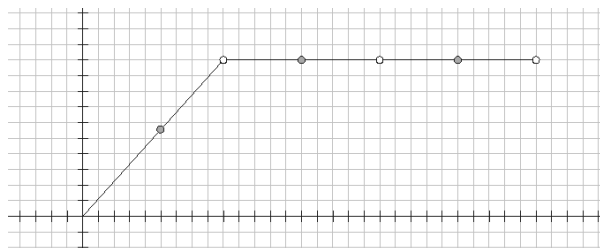
Example of a relative H-function:

```
N1 G1 X100 Y100 H1 00.5 (*switch on H1 after a half path element*)
N2 G1 X200 Y100 H-1 00.75 (*switch off H1 after 3/4 of the path element*)
```



Example of an absolute H-function (the user should know the path length):

```
N0 F200 E1000 E-1000
N1 G1 X100 Y100 H1 L75 (*switch on H1 after 75mm*)
N2 G1 X200 Y100 H-1 L50 (*switch off after 50mm*)
N3 G1 X300 Y100 H2 L-50 (*switch on after 50mm after the end of the path element H2*)
```



4.1.2 Additional function, M-function

Additional functions or M-functions can be used to set a binary output to start a different action. These differ from switching points in that the current position is retained until the M-function is confirmed by the setting of an input. This technique is often used if the subsequent processing of the program depends on other processes. The following line would start M-function 10 and wait for it to be confirmed:

```
N90 M10
```

K and L can be used to enter two parameters which are copied to the corresponding SMC_GeoInfo object. During runtime, the application will have the opportunity to access these values via the SMC_GetMParameters POU whenever the interpolator reaches and waits at an M-function.

The mechanism described below can only be used if the path is being processed in online mode (with SMC_NCDecoder):

The global data structure gSMC_MParameters (type: SMC_M_PARAMETERS) stored in the SM_CNC.lib library can be used to transfer additional parameters. The structure corresponds to parameters dP1..dP8.

To use a user-defined data structure instead of gSMC_MParameters, transfer it via O\$var\$. For example, g_myMParams (type: SMC_M_PARAMETERS) contains the parameters to be transferred:

```
N150 M13 O$g_myMParams
```

4.1.3 Use of variables

Instead of fixed numerical values, variables can also be used. These must be enclosed in \$ characters (e.g. R\$g_fVar\$). In this context, please note that using variables will only work if the program has been compiled as a program variable and is being processed online by the decoder POU. The variables are substituted at the point in time at which the decoder processes the corresponding line.

If the CNC program is compiled as an OutQueue, on the other hand, the variable mechanism will not work, since the path is generated offline and transferred as an uneditable application data structure. In this case - as with offline display - the editor will substitute variables with their initial values.

The use of variables in G-code programs imported online requires some additional preparation (see [SMC_VARLIST structure](#)).

4.1.4 Circular interpolation

The target coordinates (X/Y) need to be defined in order to describe a circular arc. There are two ways of defining the circular curve: enter either the radius (R) of the circular arc or its centre point coordinates (I/J).

If you enter the radius, please note that only arcs spanning fewer than 180° (semicircles) are possible, since this method (except in the case of semicircles) always describes two possible solutions - one greater than and one less than a semicircle. The system always selects the solution less than a semicircle.

Example of a semicircle:

```
N10 G1 X100 Y100  
N20 G2 X200 Y100 R50
```

If you require a semicircle with an opening angle in excess of 180°, you need to define it via (I/J). This method is unique, except in cases where the start and end points of the circle are identical. In such cases this could result in a zero circle or a full circle; the system will insert a full circle.

Please also note that where the method using (I/J) is concerned, it is of crucial importance whether the coordinates I/J (see G98/G99) are entered as relative or absolute values. If I and J are not entered correctly (distance between centre point and start/end point must be identical), it will not be possible to create a circular arc and the system will insert a straight line instead.

Example of the same semicircle specified by means of a relative centre point:

```
N10 G1 X100 Y100  
N15 G99  
N20 G2 X200 Y100 I50 J0
```

Example of the same semicircle specified by means of an absolute centre point:

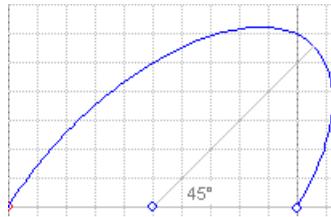
```
N10 G1 X100 Y100  
N15 G98  
N20 G2 X200 Y100 I150 J100
```

4.1.5 Ellipse interpolation

In order to describe an ellipse arc, the target coordinates (X/Y), the ellipse's centre point (I/J), the direction of its main axis (K) and the length ratio between the main and secondary axes (R) must be specified.

Example:

```
N10 G0 X100 Y100  
N15 G98  
N20 G8 X200 Y100 I150 J100 K45 R0.5
```



4.1.6 Spline interpolation

In order to describe splines you simply need to enter the target coordinates of the next spline segment. The spline segment will be calculated automatically by the system in such a way that the end vector of the preceding element and the start vector of the spline, as well as the end vector of the spline and the start vector of the subsequent element, match, thereby ensuring that there will be no kinks on the path.

4.1.7 Conditional jumps

Command **G20** can be used to implement a conditional jump. Enter the line number of the jump destination (L) and the condition for the jump (K). If a jump condition is not specified, the implicit decoder variable will be used automatically. The jump will then be executed automatically whenever the condition has a non-zero value.

4.1.8 Changing variable values

Commands **G36** and **G37** are provided to the user to write or change variables. The variable to be changed is entered using **O\$var\$**. **D** is used to set the value to be written (**G36**) or added (**G37**) to the variable.

The following example sets global variable **g_i** to 5:

```
N1000 G36 O$g_i$ D5
```

The following example leads to lines 1010 and 1020 being traversed five times:

```
N1000 G36 O$g_i$ D5  
N1010 G1 X100 F100 E100 E-100  
N1020 G1 X0  
N1030 G37 O$g_i$ D-1  
N1040 G20 L1010 K$g_i$
```

Please note that this mechanism **will only work if the path is being processed in online mode**, since variables can only be used in online mode. This mechanism will therefore not work in the CNC editor. Instead, you can proceed as follows:

If no variable is specified in **O**, an **implicit decoder variable** (type: INT) is used. This mechanism also works offline in the editor. However, please note that as it only makes one variable available, nested jumps and loops cannot be programmed.

Example:

```
N1000 G36 D5  
N1010 G1 X100 F100 E100 E-100  
N1020 G1 X0  
N1030 G37 D-1  
N1040 G20 L1010
```

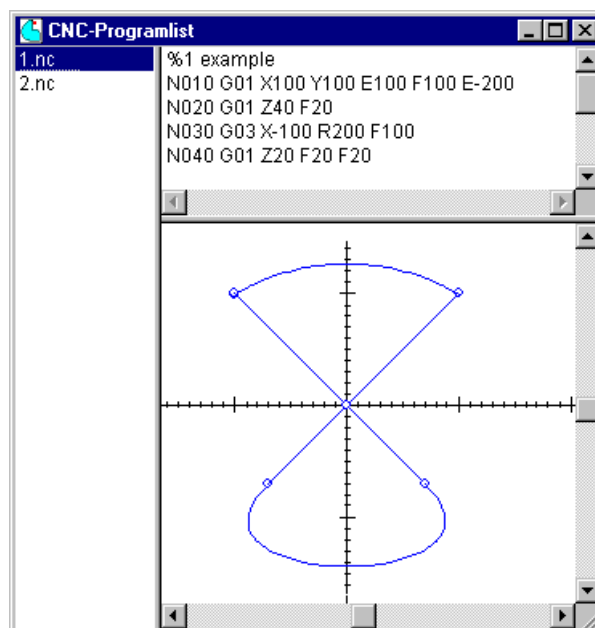
4.2 Starting the CNC editor, inserting and managing CNC programs

The CNC editor is started under "Resources" in the Object Organiser. A tripartite window called "CNC Program List" appears. The names of the existing programs are listed in the left-hand column. The upper right-hand window pane serves as a text editor, where the CNC program can be entered in accordance with DIN 66025. The lower right-hand window pane is used to display the program graphically. The program can be edited in both the text-based and graphics-based sections and every time a change is made in one section, the other is updated automatically.

As long as the CNC editor is active, the **Insert** menu is replaced on the menu bar by the **CNC program** menu.

4.2.1 Creating a new CNC program

Move the focus to the CNC program list column and select the **New CNC program** command from the **CNC Program** menu or from the context menu. The *Program name* dialog appears in which the name of the new program can be entered. By default, CNC programs are named "_CNC<n>", where n represents one of a sequence of consecutive numbers starting at 1. You can edit the program name set by default in the dialog but not enter a name which is already assigned. Close the dialog with **OK** and the new program will appear selected in the list. The first program line is displayed in the text editor: for a new program, "% comment" will appear; the graphic editor is still empty in this case.



The selected program can be edited in both the text editor and the graphic editor.

4.2.2 Rename CNC Program

Select the program from the CNC program list followed by the **Rename CNC program** command from the **CNC Program** menu or from the context menu. The *Program name* dialog appears in which the name can be edited.

4.2.3 Deleting a CNC program

Select the program from the CNC program list followed by the **Delete CNC program** command from the **CNC Program** menu or from the context menu. The program is deleted from the list and the focus moves to the next entry.

4.2.4 Information

Select the program from the CNC program list followed by the **Information** command from the **CNC Program** menu or from the context menu. A dialog containing information about the selected CNC program appears.

4.2.5 Define Queue Size

You can define the size of the OutQueue data buffer here. This can be very useful if there is only limited memory space available for the CNC function blocks in the IEC program, so that not all GeoInfo objects can be stored there and the ring buffer functionality must be used. This can lead to the occurrence of special effects (e.g. deceleration whilst traversing straight paths) which you can simulate and reproduce using this function.

Possible values: 5000 - 100000 bytes. Click **OK** to apply the values.

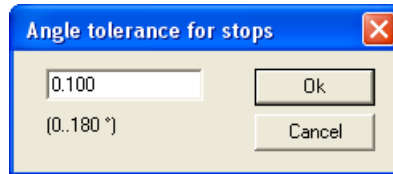
4.2.6 Defining the start position of the CNC program

In the *Starting point* dialog, you can enter path start positions deviating from the default value of 0 for simulation purposes.

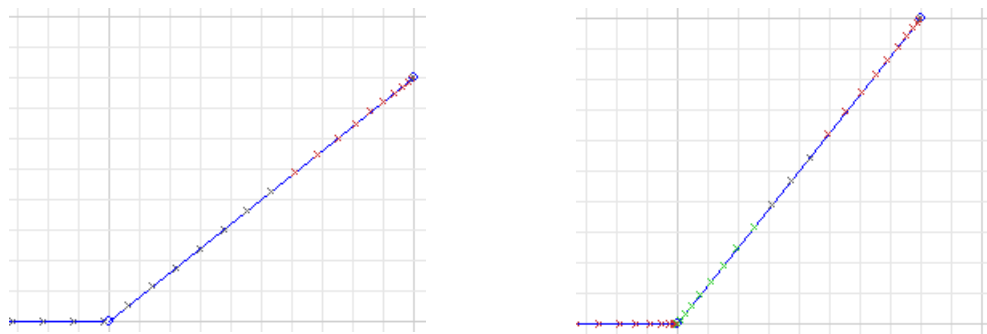
Values can be entered for the following axes: X, Y, Z, P, Q, U, V, W.

4.2.7 Defining the angular tolerance for stopping in the CNC program

In the *Angle tolerance for stops* dialog you can set the path bend sensitivity. The value entered defines the limit angle for the tangents of two path objects in excess of which a stop will be induced.



Angular tolerance 45°



4.2.8 Shifting a CNC program

Select this menu item and enter a displacement vector in the *Translation vector* dialog. The current CNC program will be moved by this vector.

Values can be entered for the following axes: X, Y, Z, P, Q, U, V, W.

4.2.9 Rotating the CNC program

To rotate the current program, select this menu item and enter the rotation angle in the *Rotation angle* dialog. The CNC program will be rotated counter-clockwise around the zero point by the specified angle.

4.2.10 Stretch the CNC program

Enter the stretch factor in the *Stretch factor* dialog. The CNC program will be stretched by the specified factor.

4.2.11 Invert Direction

Select this menu item to invert the path and induce processing in the opposite direction. The positions of switch markers do not change.

4.2.12 Split Object

Select a path object (the object will turn red) to specify a split position (0 start, 1 end) in that object. The object will then be split at the specified position to form two path objects.

Example: Travel command **N10** is split with a split position of 0.5:

```
...  
N0 G01 X123.000000 Y73.550000  
N10 G01 X40.0 Y50.0  
...  
Result: New split position at X=20  
...  
N0 G01 X20.000000 Y40.000000  
N10 G01 X40.000000 Y50.000000  
N20 G01 X123.000000 Y73.550000  
...
```

4.2.13 Read CNC Program from File

Select this menu item to load a CNC program stored in an ASCII file. You can select the required TXT file in the standard dialog for opening a file. Enter the name of a program in the next dialog.

4.2.14 Write CNC Program to File

Select this menu item to write the current CNC program to an ASCII file of your choosing. If the file already exists, the »PLC Designer« will prompt you to confirm your selection.

4.2.15 Import DXF File

Select this menu item to import a DXF file. You can select the required DXF file in the standard dialog for opening a file. Once you have done this, the *DXF import options* dialog will appear in which you can set the following options:

- ▶ One CNC program for the entire DXF file: All paths in the DXF file will be written to a single CNC program.
- ▶ One CNC program for each DXF layer: A separate CNC program will be created for each DXF layer.
- ▶ One CNC program for each coherent segment: A separate CNC program will be created for each coherent path segment. Due to the fact that in a DXF file the individual path objects are not stored in any particular order, the »PLC Designer« tries to connect the objects in a way that results in a coherent path.

4.2.16 Write OutQueue to File

You can use this function to convert the entire CNC program into an OutQueue (i.e. a list of GEOINFO structure objects) and save this list in a file which then can be downloaded to the control's file system and read in during runtime (c). We recommend that you adopt this approach if you have CNC programs which are too big for the control's global data memory or which need to be exchanged without making changes to the »PLC Designer« project.

4.3 CNC text editor

The text editor is located in the upper right-hand pane of the CNC Program List window. You can use it to enter and modify a CNC program in accordance with DIN 66025 (see supported and extended elements of the CNC language, DIN 66025, for more information). The program will be displayed accordingly in the graphic editor and can be modified there. In this case the text part will be updated automatically.

For processing, the commands of the menus **CNC program** and **Extras** are provided; for this, see: [CNC graphic editor](#) (□ 78).

(For this see also the [SoftMotion programming examples](#) (□ 267).)

Press <F2> to open the Input Assistant, where you can select global variables to be inserted into the CNC program. The variables will be displayed in the graphic editor with their initial values (if available).



Note!

Please note that references to global variables in the decoder POU will be evaluated in response to a rising edge in the Execute input.

If you choose to have the »PLC Designer« create a ready-to-use OutQueue structure during compilation (option b)), the reference to the global variable will of course be replaced with the initial value during compilation, thereby making the use of global variables in this context meaningless.

4.4 CNC graphic editor

The graphic editor is located in the lower right-hand pane of the CNC Program List window. As well as visualising the CNC program created in the text editor, the graphic editor also provides a means of modifying the program using the mouse. Any changes made will be updated automatically and immediately in the program text in the upper window pane.

Display

A coordinate system is displayed. There are pitch lines marking the intervals on the coordinate axes and a light grey grid can also shown/hidden using the **Extras→Show grid** command.

Press and hold down the left mouse button to move the CNC program view as required on the screen. Use the mouse wheel and the <Ctrl> key to enlarge the zoom factor.

Positioning objects (G00) are displayed in green and all other elements in blue. The current object identified by the cursor being located on the corresponding code line in the [CNC text editor](#) (□ 77) is displayed in red.

For spline segments (G05) the convex mantle of the cubic polynomials is displayed in light grey.



Note!

A switching point position can only be inserted in the [CNC text editor](#) (□ 77)!
They are displayed in the graphic editor as green dots on the curve.

(For this see also the [SoftMotion programming examples](#) (□ 267)).

4.5 Commands in the "Extras" menu

The **Extras** menu provides commands and options for working with and displaying in the graphic editor. (You can of course also perform programming by entering the corresponding instructions in the text editor.) Corresponding buttons also appear on the tool bar. A tick appears in front of activated options in the 'Extras' menu and the corresponding buttons on the tool bar look like they have been pressed.



Note!

It is entirely possible to select more than one of the Tool Radius Correction, Round Path, Smooth Path and Avoid Loop options. In so doing, you can simulate the effect of path preprocessing POU's connected in series. Only the 'Tool Radius Correction' and 'Round Path'/'Smooth Path' path preprocessing options cannot be activated at the same time.

4.5.1 Select Mode



If this option is activated, you can select a graphic element in the CNC editor by clicking with the mouse. Once an element has been selected, it turns red and the corresponding line is highlighted in the text editor. Click the end point of an element and hold the mouse button down to move the element with the mouse.

4.5.2 Line Insert Mode



If this option is activated, clicking with the mouse within the editor pane will insert a G01 line element. The new element will start behind the currently selected (red) element. The mouse position defines the destination (end point) of the line.

4.5.3 Circle CW Insert Mode



If this option is activated, clicking with the mouse within the editor pane will insert a G02 circle element for clockwise motion. The new element will be inserted behind the currently selected (red) element. The mouse position defines the destination (end point). The radius of the circle is set to 100 by default; if necessary, this can be modified in the text editor.

4.5.4 Circle CCW Insert Mode



If this option is activated, clicking with the mouse within the editor pane will insert a G03 circle element for counter-clockwise motion. The new element will be inserted behind the currently selected (red) element. The mouse position defines the destination (end point). The radius of the circle is set to 100 by default; if necessary, this can be modified in the text editor.

4.5.5 Spline Insert Mode



If this option is activated, clicking with the mouse within the editor pane will insert a spline behind the currently selected (red) element. The mouse position defines the destination (end point).

4.5.6 Fit to Screen



If this option is activated, the visible part of the window will show the complete CNC program.

4.5.7 Renumber Program



This command automatically renumbers the program consecutively, assigning new line numbers (N<Number>) in decimal increments.

4.5.8 Convert Splines/Ellipses to Lines

Splines and ellipses need a lot of computing time for interpolation. In order to reduce this time, this menu item can be used to convert all splines and ellipses in the CNC program into an approximately equivalent number of lines. This means that splines and ellipses can be used in the design of the path but do not have to be computed during runtime.

The command opens a dialog providing two conversion options:

- ▶ **Length dependent**, i.e. one line will be created for every x length units (number x can be inserted in the dialog) of the spline/ellipse, or
- ▶ **Angle dependent**, i.e. the original object will be partitioned in such a way that the resulting lines will include angles less than x (angle x [degrees] can be inserted in the dialog).



Tip!

If the default settings are retained, when the lines are interpolated, there will be a deceleration to zero velocity after each new line segment (this is not the case when splines/ellipses are interpolated). You can avoid this by increasing the angular tolerance accordingly.

4.5.9 Tool Radius Correction



If this option is activated and if the start (G41/G42) and the end (G40) of the tool radius correction and a tool radius (D<Radius>) have been programmed in the CNC program, the path will be displayed corrected accordingly. This menu item corresponds to the SMC_ToolCorr POU in the SoftMotion library. The original path will be displayed in light grey for reference purposes. Positionings on the corrected path are "blind" positions and are therefore displayed in dark yellow.

4.5.10 Round Path



If this option is activated, the rounded path will be displayed, showing the effect the SMC_RoundPath POU from the SoftMotion library has on the programmed path. In order to achieve this, the start (G52), end (G50) and rounding radius (D<Radius>) must be set in the CNC program. The original path will be displayed in light grey for reference purposes.

4.5.11 Smooth Path



If this option is activated, the programmed path will be displayed, showing the effect of the SMC_SmoothPath POU from the SM_CNC.lib SoftMotion library. It creates a smoothed path by means of a cubic polynomial (spline segment). In order to achieve this, the start (G51), end (G50) and rounding radius (D<Radius>) must be set in the CNC program. The original path will be displayed in light grey for reference purposes.

4.5.12 Avoid Loop



If this option is activated, the path created when the path crosses itself and is cut off at this point (thereby avoiding loops) will be displayed. This menu item corresponds to the SMC_AvoidLoop POU in the SoftMotion library.

In order to achieve this, the start (G61) and end (G60) must be set in the CNC program. The original path will be displayed in light grey for reference purposes.

4.5.13 Suppress Record



If this option is activated, all lines in the text editor starting with "/" will be ignored.

4.5.14 Show Interpolation Points



If this option is activated, interpolation points will be displayed on the basis of a cycle of 100 ms; in other words, the tool positions will be indicated by small grey crosses every 100 ms. This provides a means of approximately estimating the velocity characteristics (fast = long intervals, slow = short intervals).

4.5.15 Set epsilon values for zero

The internal check of a value x for zero must be replaced by an examination for $x < e$ due to the inaccuracy of a floating point calculation. The size of the e -value may need to be adapted (e.g. if 32-bit floating point values are used instead of 64-bit values, or when importing a CNC program with limited accuracy). If you need to adapt the value, use the *Zero tolerance values* dialog which appears when you select this command.



Note!

For standard applications, modification of the tolerance values should not be necessary and should in fact be avoided!

4.6 Automatic structure filling in the CNC editor

When the IEC program is compiled, a global variables directory called "CNC Data" is created automatically. The CNC programs are stored in this folder in data structures of the same names.

There are three possible variants A ... C, which can be set separately for each CNC program in the CNC editor's **CNC program** menu:

A. Create program variable on compile

- This option corresponds to a): The CNC program is saved in an SMC_CNC_REF structure located in the SM_CNC.lib library. This structure variable has to be processed by the decoder and path preprocessing POU's and cannot be entered directly in the interpolator POU.
- Although this option uses more online computing power, it does enable variables to be used in the CNC program and supports modulations of the CNC program by means of the PLC program.

B. Create OutQueue on compile

- This option corresponds to b): The CNC program is saved in an SMC_OUTQUEUE structure located in the SM_CNC.lib library. This structure variable can be entered directly in the interpolator POU.
- Although no variables can be used in the path, this option has the advantage of minimising online resource requirements.

C. Do not compile

- This option corresponds to c): You have saved the program as an ASCII file or an OutQueue file in the control's file system and you want to import it during runtime using one of the [CNC function blocks](#) (250) For this reason it should not be added to the IEC data.

5 CAM editor



Note!

The CAM editor will only be available if ...

- The SoftMotion add-on has been installed and
- An automation system with the SoftMotion add-on has been selected in the Target Settings dialog box.

The SoftMotion cam profiler (CAM editor) is integrated in the »PLC Designer« programming interface. It can be used to program electronic cams and tappet control units in graphic and tabular format, which the "PLC Designer" will use to automatically generate global data structures »PLC Designer«(CAM data) during project compilation. These structures can be accessed by the IEC program.

Standardised PLCOpen POU's are used for cam conversion in the IEC program (SM_PLCOpen.lib library).

(See also the [SoftMotion programming examples](#) (p. 267))

5.1 Definition of a cam for SoftMotion

In simple terms, a cam describes the functional dependence of one value (slave) on another (master).

To describe this dependence, the master axis is divided into various intervals. The »PLC Designer« provides two options for each interval [a, b] to facilitate the functional mapping of the master axis on the slave axis:

- ▶ **Line:** The dependence is described by a linear image. In this interval the first derivative (velocity) is constant, in accordance with the slope of the line; the second derivative (acceleration) is 0.
- ▶ **Fifth order polynomial:** In this interval the dependence is described by a fifth order polynomial. Thus the first and the second derivative become fourth and third order polynomials.

The functions in these intervals must be concatenated in such a way that at the point of transition from one function to the next, both the function value and at least the first and second derivatives are consistent.

Individual interpolation points and lines can be set in the CAM editor. The remaining intermediate intervals will be filled up automatically by the editor with fifth order polynomials, thereby meeting requirements in respect of consistency and differentiation.

The function value, the first derivative (velocity, in this case constant) and the second derivative (acceleration, in this case always 0) are defined along a line. A point however can be defined with any first and second derivative.

Users can also place tappets, i.e. binary position switches, on cams.

5.2 Starting the CAM editor

The cam editor is started from the **Resources** tab. A window appears, which is divided into three areas. As long as no cam is created, the window is empty. Otherwise there is a tree with all cams that are programmed in the left column, and in the top part of the window the function dependency within the master-slave axis is represented graphically. For the bottom part of the window three representation modes can be selected. The commands for them can be found in the **Extras** menu (see general editor settings): representation of the first (blue curve, speed) and the second (green curve, acceleration) derivative, representation of a table with all cam elements (points/lines), or representation of a table with all cams. The tables can be edited.

5.3 Creating a cam

Select the **New cam** command from the **Insert** menu or shortcut menu and carry out the required settings in the *Cam properties* dialog box which is opened.

Name of cam

A name for the new cam

Type

A cam contains cam elements as well as tappets; a tappet control unit (tappet table) only contains tappets.

Scaling master axis

The scaling of the master axis is defined here. If 360° is selected, the Minimum, Maximum, Step and Unit settings will be made automatically (0, 360, 20, °); otherwise you can set them manually.

Scaling slave axis

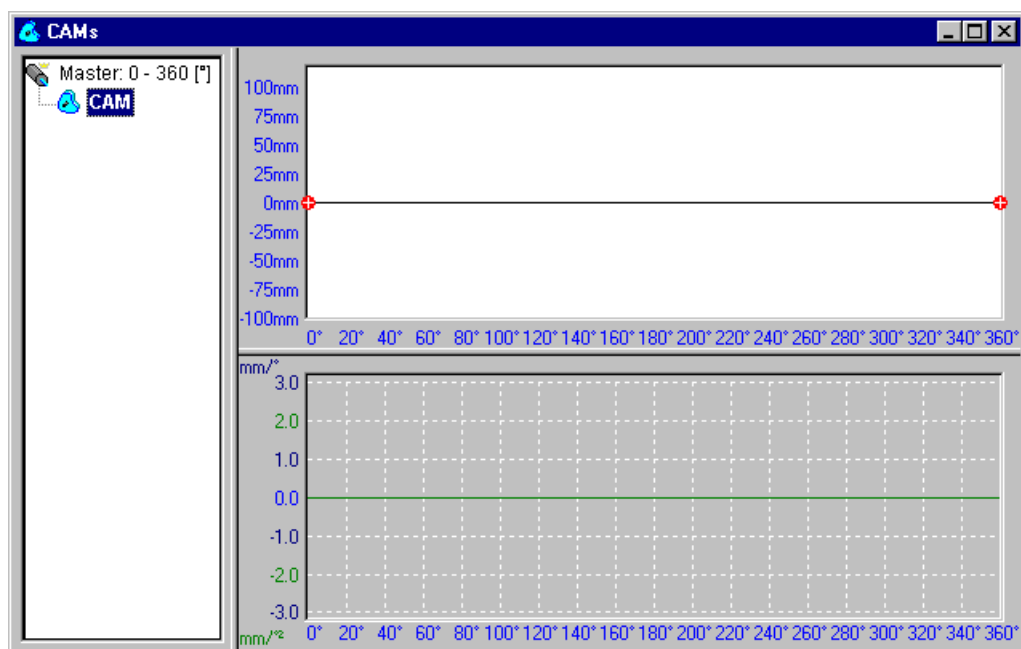
The scaling of the slave axis is defined here. The default settings are shown in the screenshot above.

Properties

Select "periodic" to ensure that the function values and the first and second derivatives will match at the cam's start and end points. Any modifications made to the end point during editing of the cam will then be ignored.

Close the dialog with **OK** to confirm your settings.

The name of the cam you have just created will now appear in the left-hand column (cam list) of the CAM editor window. As long as this entry is selected, the cam will be displayed in the editor and can be edited (see [Editing a cam](#) (88)). The new cam is displayed in the two right-hand window panes. You will see the horizontal blue master axes, the blue position axis (slave) in the upper window pane and the velocity (dark blue) and acceleration (green) scales in the lower window pane. The following screenshot shows the default settings in the Properties dialog:



In order to edit the settings the Properties dialog for the currently selected cam can be reopened at any time by selecting the **Settings** command from the **Extras** menu or shortcut menu or by double-clicking the entry in the cam list.

Cam tree

A cam tree listing all cams and tappet control units appears in the left-hand pane of the window. These elements always are sorted so that all elements which have the same master scaling, i.e. all elements which potentially refer to the same axis, have the same "father".

5.4 Editing a cam

Select the cam you wish to edit in the left-hand column of the editor. Click the entry with the mouse to select it (blue highlighting will be applied) and display it in the editor windows.

To display one or more other cams with the same "father2" (see [Cam tree](#) (87)), press and hold down the <Ctrl> key and click with the mouse to select them.

5.4.1 General editor settings

Edit mode can be selected in the **Insert** menu or via the corresponding button on the toolbar:

	Select element
	Insert point
	Insert line
	Insert cam

Editing the general settings of the cam

In order to alter the settings carried out during the insertion of a new cam in the *Cam properties* dialog, the **Settings** command in the **Extras** menu is selected.

Altering the display of the curve

For this purpose the commands **Complete curve** and **Display peak values** in the **Extras** menu are provided.

The display mode for the lower part of the window can be selected in the **Extras** menu or via the corresponding button on the tool bar:

	Display speed/acceleration In the bottom part of the window the first (blue) and second (green) derivative of the cam are visualised.
	Cam as table The bottom part of the window shows the individual elements (points/lines) of the cam and their properties in an editable table.
	Cams as table The bottom part of the window shows the individual cams and their properties in an editable table.

5.4.2 Editing properties of individual cam elements

The attributes of individual elements can be modified in the Properties CAM Element dialog or by selecting and moving the element in the editor window.

5.4.2.1 Editing a point / line

Double-click the element to open the *Cam element properties* dialog in which the following properties of an element can be edited by means of numeric inputs:

Dialog object	Properties
Element type	Line/Point: The type can be changed here. If a point is changed into a line, then it will automatically be assigned a certain length; if a line is changed into a point, the coordinates of the start point of the line will be applied automatically.
Master start, Master end	Start and end values (lines only) on the x axis (for unit see <i>Cam properties</i> dialog (Extras→Settings)).
Slave start, Slave end	Start and end values (lines only) on the x axis (for unit see Extras→Settings).
Speed	(Points only)
Acceleration	(Points only)

5.4.2.2 Editing a cam

Double-click the element to open the *Tappet element properties* dialog in which the following settings can be made.

Activation

The tappet is activated, i.e. the Boolean variable in the program (tappet bit) which is assigned to the tappet group ID (see below) is set to TRUE when the cam is run through; one of the three options below can be set to define when exactly this should happen.

Cam track	Information
Positive pass	Only if the cam is passed through from left to right; after confirming with Accept the green arrow above the tappet symbol will point to the right.
Negative pass	Only if the cam is passed through from right to left; after confirming with Accept the green arrow above the tappet symbol will point to the left.
Each pass	At each pass of the cam; after confirming with Accept the green arrow above the tappet symbol will point to the left and to the right.

Action

One of the following options can be set to define the effect activating the tappet should have on the action linked to it in the project:

Action	Information
on	The action will be started (the "tappet bit" will be set to TRUE); the tappet symbol will turn green.
off	The action will be stopped (the "tappet bit" will be set to FALSE); the tappet symbol will turn red.
invert	If the action is currently active, it will be stopped; if it is currently inactive, it will be started (the tappet bit is inverted). The tappet symbol will turn yellow.
timed in	The action will be started with values entered in the Delay and Duration fields; the tappet symbol will turn cyan.

Group ID:

Identification number (INT) of the tappet which serves to reference the tappet in the project; several tappets can have the same group ID and thus be "grouped" so that the action assigned to them operates the same digital switch.

Delay [μs]:	Period of time which must elapse following a pass of a tappet element before the action assigned to it in the project is started (time at the end of which the tappet bit should be set to TRUE) (only if action = timed in).
Duration [μs]:	Entry indicating how long the action assigned to the tappet in the project should remain active once it starts (how long the tappet bit should remain set to TRUE) (only if action = timed in).
Master position:	X-position of tappet
Slave position:	Y-position of tappet, cannot be edited since it is defined by the cam characteristic

The settings in the properties dialogs can be confirmed by clicking **OK** or **Apply**. The cam curve display in the editor will be updated accordingly. **OK** will also close the dialog, whereas it will remain open with **Apply**.

5.4.2.3 Selecting and shifting in the editor window

Click an element with the mouse to select it. To move an element, press and hold down the mouse button and drag the cursor to the desired position. This will cause the corresponding values in the associated properties dialog to be modified:

Point	If you click to select a point in the cam editor, a small red square will appear representing the gradient (velocity). Move this square to modify the gradient of the point. The gradient will be displayed with the aid of an auxiliary tangent. The point itself can also be moved.
Line	If you click to select a line in the cam editor, small red squares will appear at the end points. Click again to select one of the end points and move it to modify the gradient or click the line to move it without modifying the gradient.
Tappet	If you click to select a tappet in the cam editor the frame of the tappet square will turn red. The tappet can be moved along the cam curve by dragging the mouse pointer accordingly.

5.4.3 Commands in the "Extras" menu

5.4.3.1 Settings



This command opens the *Cam properties* dialog box that was displayed during the creation of the cam and where the scaling and units can be modified.

5.4.3.2 Show Complete Cam

If this option is activated (a tick will appear in front of the menu item or the button in the tool bar will look like it has been pressed), the fifth order polynomials linking the elements (points, lines, tappets) will also be displayed on the cam. Otherwise only the individual elements will be visible.

5.4.3.3 Show Bounds

If this option is activated (a tick will appear in front of the menu item or the button in the tool bar will look like it has been pressed), the cam's bound values (maximum/minimum) will also be displayed in addition to the cam and its derivatives.

5.4.3.4 Transmission options

This menu item opens a dialog by means of which the transmission ratio of the cam can be set.

Essentially, there are three compilation modes.

polynomial

During compilation MC_CAM_REF structure variables are created. For each segment, they contain the description of the fifth order polynomial which describes the cam. Structures of this type are used as inputs for the MC_CamIn POU. The structure is part of the SM_DriveBasic.lib library.

Compile options CAM				
Compile option:				
☐	don't compile			
☐	polynomial			
☒	equidistant point table			
☐	element optimized point table			
Elements:	256	Data type:	UINT	
Master scaling:				
	Start:	End:	Unit:	
Editor:	0.000	360.000	1 [°]	
PLC:	0.000	64800.000	180.000	
Slave scaling:				
	Start:	End:	Unit:	Interval:
Editor:	0.000	360.000	1 [°]	360.000
PLC:	0.000	64800.000	180.000	64800.000

equidistant point table

An SMC_CAMTable_<Data type>_<Element number>_1 type interpolation point table is created on the basis of the settings towards the bottom of the dialog. The position array of the dialog contains the slave values of the cam with reference to the master values, which are distributed evenly on the domain of the master axis. The first value in the table refers to the slave position at the master minimum of the cam. For non-periodic cams, the last value refers to the slave position at the master maximum. For periodic cams this value does not need to be rewritten, because it is the same as that at the master minimum; for this reason the intervals are made slightly narrower and the last value in the table describes the slave position at $\text{Master.End} - (\text{Master.End} - \text{Master.Start}) / \text{Number of elements}$.

element optimised point table

A two-dimensional (not usually equidistant) SMC_CAMTable_<Data type>_<Element number>_2 type interpolation point table is created on the basis of the settings towards the bottom of the dialog. The table it contains describes pairs of master and associated slave positions. Partitioning is applied so that elements with constant velocity (lines) are only assigned one interpolation point each at the start and end respectively. The remaining interpolation points are distributed as evenly as possible across the remainder of the cam.

do not compile

No global variables are created for the cam. This option is used primarily if the cam needs to be loaded from the file system during runtime, for example because it needs to be changed without affecting the active »PLC Designer« project.

Scaling is only relevant for interpolation point tables. You can define the scaling of the master and the slave axis by assigning either start and end values or start and end units.

5.4.3.5 Write Cam to File

This menu item opens a file selection dialog in which a *.CAM file can be defined to which the current cam will be written. This file can be read during runtime and converted to a standard data structure by the SMC_ReadCAM function block. The cam will be stored in polynomial, equidistant or element-optimised format in accordance with the currently set compile option.

5.4.3.6 Read Cam from File

This menu item can be used to reimport a CAM file *. into the »PLC Designer«. Once the required file has been selected, the *Cam properties* dialog will appear in which a name for the cam must be entered and the scaling of the slave axis must be adapted.

Due to the fact that during the creation of a cam only the information needed for the execution of the cam is output, a read cam may differ from the original..

5.4.3.7 Export Cam as ASCII Table

This command can be used to export the current cam as an ASCII text file. The number of points can be specified. The start and end point always will be included as the first and last point respectively. A text file with the following structure will be created:

<Master position>;<Slave position><CR><LF>

The resulting text file can, for example, be imported into other programs and used for drive dimensioning.

5.4.3.8 Import Cam from ASCII Table

This function can be used to import cam tables stored as ASCII files in the following format:

<Master position>;<Slave position>

Please note that the points are sorted in ascending order relative to the master position.

Once the cam has been imported successfully, the user can make the corresponding name, interval and scaling settings in the properties dialog. After this, the number of interpolation points can be reduced.

5.4.4 Commands in the "Insert" menu

5.4.4.1 Select element



This menu item can be used to switch select mode on and off. As long as select mode is activated (a tick appears in front of the menu item and the icon in the tool bar will look like it has been pressed), you can select a cam element by clicking the desired element (point, line, tappet) with the left mouse button. The corresponding position marks (small red squares) will then appear and the element can be edited.

5.4.4.2 Insert point



Select this command in the **Insert** menu or press the button in the tool bar to add a new point to the cam as follows: Move the mouse pointer to the position of the new interpolation point. Press the left mouse button to insert the point; a horizontal tangent will appear. Release the left mouse button to apply the point (it will appear as a solid red circle with cross hair) and switch to "Select Element" mode automatically.

If you hold the left mouse button pressed whilst inserting the new point, you can modify the gradient of the tangent (velocity) directly by moving the mouse.

5.4.4.3 Insert line



Select this command in the **Insert** menu or from the menu bar to add a line to the cam as follows: Move the mouse pointer to the place where you would like the line to start and click the left mouse button. The start point is identified by a small red square. Hold down the left mouse button and move the mouse to the place where you would like the line to end (a small red square will appear here too). The line's end point must not be positioned to the left of the start point or to the right of the next defined point. Release the left mouse button to apply the end point and switch to "Select Element" mode automatically.

5.4.4.4 Insert cam



Select this command in the **Insert** menu or from the tool bar to add a tappet to the cam as follows: Move the mouse pointer to the place where you would like the tappet to be inserted and press the left mouse button. The tappet will be appended to the cam in the X-position selected with the mouse pointer. Keep holding down the left mouse button to move the tappet along the cam with the mouse. Release the left mouse button to apply the position and switch to "Select Element" mode automatically.

5.5 Online mode of the cam editor

The cam editor is provided with an online mode. By this cams can be modified with regard to the runtime.

If the »PLC Designer« HMI is used, the online editor can be opened via the internal command (internal CAM).

The cams are positioned on the control system as a global data structure. As is generally known, without a new download only the content, but not the size of data structures can be modified. Thus the editing options of the editor in the online mode are limited in the following way:

1. Points can be shifted to any position
2. Lines can be modified in any way
3. Tappets can be shifted to any position
4. Cam settings (scaling, etc.) can be altered
5. New points/lines or tappets cannot be added
6. New cams cannot be generated

The following additional menu commands are provided in the online mode:

► Extras→Read cam from control system

- In the online mode the cam editor displays the currently available cam. If it has been changed during a previous online session or by the application it has not got to correspond to the cam that is actually positioned on the control system.
- By means of this command the current cam is read from the control system via the monitoring mechanism and is adjusted. Like this the cam shown in the editor is adapted to the cam that is actually available on the control system.
- Please note that this function can only be used if "polynomial" has been used as compilation mode, as only in this case the complete information on the cam is available in the runtime system. The option of retrieving the original points from a point table is not offered here.

► Extras→Write cam into control system

- After the cam has been modified, the user can rewrite it into the control system using this command.
- During this process it is not considered whether the cam is possibly executed at this moment, which can result in skips. The user has to ensure that the cam modification is carried out in a safe state.
- Please also note that changes carried out are reset by a control reset. If you want to save the changes in the control system permanently, you have to program this yourself (e.g. by using SMC_WriteCAM or by copying the cam into the remanent memory).

5.6 Using cams

The SM_PLCOpen.lib library contains POU's for executing cams.

This section contains descriptions of how to use cams, addresses cam concatenation in particular and provides a more detailed description of the various parameters (periodicity, offset, etc.):

5.6.1 How POU parameters work

Editor: Cam Element Properties

- Periodic, - Interval

These settings are only relevant for the creation of a cam; they have no effect on how a cam is processed in an application.

The "Periodic" option determines whether the cam can be processed without jumps, i.e. if it can be processed periodically. If this option is activated, the start and end points will have the same slave position or the difference between their positions will be a multiple of the value entered under "Interval".

Even if "Periodic" is not activated, cams can still be processed consecutively, but in this case the user must make sure that the transitions meet the relevant consistency requirements.

5.6.1.1 MC_CamTableSelect

Periodic

This input determines whether a cam is processed again if the master position leaves the cam range.

If this input is FALSE, the EndOfProfile output will be set at the CamIn POU at the end of the cam and the slave will be stopped at the last position programmed by the cam. Please note that if the master re-enters the valid range the slave will still be controlled according to the cam. In other words, the cam action will not be terminated when the defined master range is exited.

If the input is TRUE, the cam will be processed consecutively, i.e. the cam will be shifted in the master direction to its previous end position.

This means that the master interval (width) of a cam now bears no relation to the position interval of the master drive. Please note, therefore, that the assignment 'slave position = cam (master position)', i.e. the definition of the slave position by the master position via the cam, will only be applicable in the first cam cycle but not after that if the width of the cam differs from the master interval.

MasterAbsolute

If this input is TRUE, the cam will be started at the current master position. This point can therefore also be located in the middle of the cam. If the point is outside the range specified by the cam, an error will be reported.

If this input is FALSE, the cam will be shifted to the current position, i.e. the zero point of the cam will henceforth be the current master position. Cams whose master range does not include 0 cannot be used with this mode, because an error would be reported (master out of specified range).

SlaveAbsolute

This input relates to the StartMode input of MC_CamIn. The following table describes how it affects the start mode.

CamIn.StartMode	absolute	relative	ramp_in	ramp_in_pos	ramp_in_neg
CamTableSelect.SlaveAbsolute					
TRUE	absolute	relative	ramp_in absolute	ramp_in_pos absolute	ramp_in_neg absolute
FALSE	relative	relative	ramp_in relative	ramp_in_pos relative	ramp_in_neg relative

See CamIn.StartMode for a more detailed description of the modes [MC_CamIn](#) (98).

5.6.1.2 MC_CamIn

MasterOffset, - MasterScaling

These two settings apply the following formula to transform the master position and use the transformed position to evaluate the cam:

$$x = \text{MasterPosition} * \text{MasterScaling} + \text{MasterOffset}$$

MasterScaling > 1 increases the speed at which the cam is processed (compressed), < 1 stretches the cam.

SlaveOffset, - SlaveScaling

These two settings shift and stretch a cam in the slave direction. The cam is first stretched and then shifted.

SlaveOffset > 1 will increase the size of the slave movement (stretching the cam); < 1 compresses the cam.

StartMode : absolute/relative/ramp_in/ramp_in_pos/ramp_in_neg

"Relative" means that when a cam is started it will be shifted by the current slave position. This only makes sense if the slave setpoint position according to the cam is at the start point 0, otherwise a jump will occur.

"Absolute" means that the cam is evaluated independently of the current slave position. If the slave is in a position other than that resulting from the cam and the current master position, a jump will occur.

The "ramp_in" option compensates potential jumps at the start of a cam by executing a compensating movement (according to the set limitations VelocityDiff, Acceleration, Deceleration). In this context, "ramp_in_pos" (as long as it is a rotary slave) will only compensate in a positive direction and "ramp_in_neg" in a negative direction. For linear slaves the compensation direction is generated automatically, i.e. "ramp_in_pos" and "ramp_in_neg" are treated like "ramp_in".

5.6.2 Switching between cams

In principle, switching between cams is possible at any time. Please note the following: on a rising edge, the slave setpoint position is strictly determined by the current master position (with master offset and master scaling applied) and the cam used, and it is only modified by means of slave offset and slave scaling. If the master or slave interval of the cam and the actual position interval of the drives are different, this will result in further offsets of the master and slave position for each cam cycle/drive revolution completed. Hence, the simple function equation.

$$\text{Slave position} = \text{cam}(\text{master position})$$

no longer applies, and is replaced by:

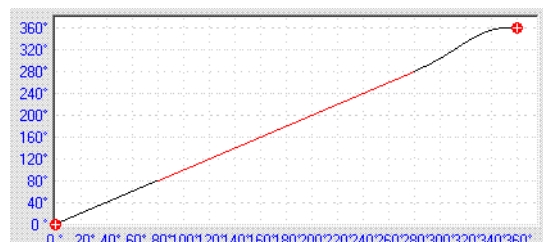
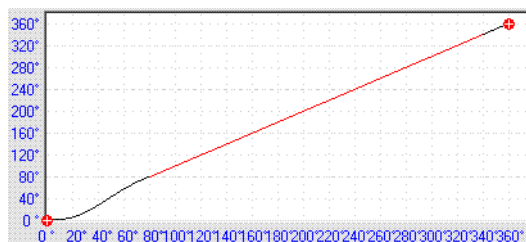
$$\text{Slave position} = \text{cam}(\text{master position} + \text{Offset1}) + \text{Offset2}$$

where Offset1 and Offset2 can change at the end of each cam.

As a result, if the MC_CamIn cam POU is restarted (rising edge at bExecute), interpreting this as a completely new action, it will delete its memory and consequently its Offset1 and Offset2, and use the original equation. This can produce a completely different slave setpoint. For this reason, you should only restart the MC_CamIn FB if you are about to process a different cam.

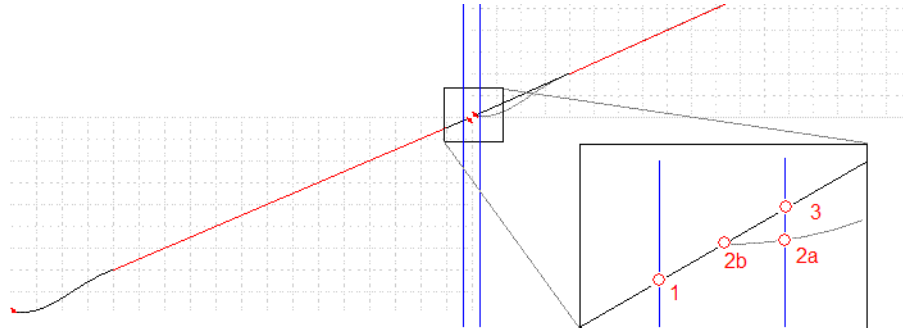
When switching between cams, please note:

Example involving a switch from Cam1 to Cam2:



(1) The velocities and accelerations of the end point of the last cam and the new cam should match in order to avoid jumps and jerks. This is the case in the example: the end point of Cam1 and the start point of Cam2 have identical velocity and acceleration values.

(2a) If the new cam has been configured so that the slave position starts at 0, the cam can be started relatively, but in this case the previous cam also has to be started as a non-periodic cam (MC_CamTableSelect). This is because if the cam was started periodically, the following interpolation points could be calculated:



The transition between Cam1 and Cam2 is displayed here. Point 1 (first blue line) is still located on Cam1 and is processed completely normally. The next time the POU is called, the master (second blue line) has strayed beyond the end point of Cam1. It is now important that Cam1 was processed as a non-periodic cam, because slave position 2b will be calculated. If Cam1 had been processed periodically, position 2a, which would result if Cam1 needed to be repeated, would be calculated.

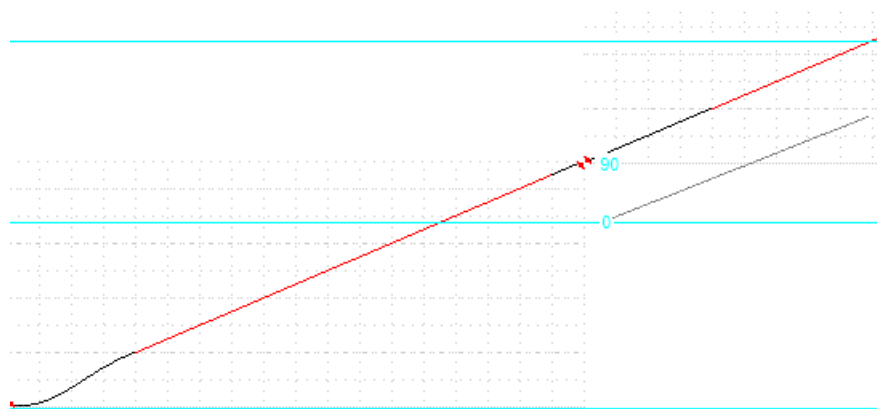
EndOfProfile is set simultaneously, regardless of how the cam is processed (periodic or non-periodic).

This output can now be used to start Cam2. For this purpose the MC_CamIn instance is called with Execute=FALSE, and then again in the same cycle with Execute=TRUE and the new CamTableID cam, resulting in an output of "3" to the slave drive.

(2b) If a cam needs to be started absolutely, please note the following: at the start of the cam the slave must be in the appropriate position in order to avoid a jump. In the context of cam switching this will usually be the case if the interval of the cam is the same as the slave interval.

In the example described above Cam2 could be started in absolute mode if the position intervals of master and slave are 360° , matching the range of the cam. In this case it is of no importance whether the cam starts periodically or non-periodically.

This would not be possible if the slave interval was 270° , for example (marked by the light blue lines).



In this case, the slave would be in position 90 at the point of transition between Cam1 and Cam2. Starting Cam2 in absolute mode at this point would produce the grey curve and a jump would occur. However, programming a corresponding slave offset of 90 would provide a possible workaround.

5.7 Cam data structures

During compilation of the project a global variable list (**Resources** tab in the »PLC Designer«) named CAM Data is created from the cams programmed in the CAM editor. The description of each individual cam is written to a structure (one structure per cam) corresponding to the settings in the 'Build Options' dialog and is thereby made accessible to the IEC program and the cam POU. For successful project compilation, the corresponding structure definitions must be available in the IEC program.

(For this see also the [SoftMotion programming examples](#) (□ 267).)

Furthermore, an `_SMC_CAM_LIST` (ARRAY OF POINTER TO `MC_CAM_REF`) data structure is also created, referring to the individual cams via pointers.

Of course, the corresponding data structures can also be created and/or completed from the IEC program during runtime. For this reason they are described in more detail below. Structure variables which are only of relevance internally are not listed.

5.7.1 MC_CAM_REF

This data structure represents a generic cam and contains the following elements:

wCAMStructID: WORD

The POU uses this variable, which always has a fixed value, to check whether the data structure entered is an `MC_CAM_REF` structure.

xStart, xEnd: LREAL

Cam domain. Master start and end position.

byType: BYTE

This variable describes the cam type, i.e. how the cam is represented.

- ▶ 1: equidistant, 1-dimensional table comprising slave positions
- ▶ 2: non-equidistant, 2-dimensional table comprising pairs of master/slave points
- ▶ 3: polynomial description of individual points comprising master position, slave position, slave velocity and slave acceleration (XYVA)

byVarType:BYTE (only for byType=1 or byType=2)

Variable type of which the cam table is comprised:

1: INT

2: UINT

3: DINT

4: UDINT

5: REAL

6: LREAL

nElements:INT

Number of elements, according to type slave positions, master/slave positions or XYVA points

byInterpolationQuality:BYTE (only for byType=1 or byType=2)

Fine interpolation degree: 1: linear (default), 3: cubic

pce: POINTER TO BYTE

Pointer to the actual data element; depending on type:

Type	
1 (equidistant)	SMC_CAMTable_<VarType>_<nElements>_1
2 (non-equidistant)	SMC_CAMTable_<VarType>_<nElements>_2
3 (XYVA)	ARRAY OF SMC_CAMXYVA

nTappets: INT

Number of tappet actions

pt: POINTER TO SMC_CAMTappet

Pointer to an ARRAY OF SMC_CAMTappet

strCAMName:STRING

Name of cam

5.7.2 SMC_CAMXYVA

An XYVA cam consists of an array of SMC_CAMXYVA. Each variable of that array describes a point on the cam using dX (master position), dY (slave position), dV (first derivative dY/dX ; corresponds to the slave velocity at constant master velocity 1) and dA (second derivative d^2Y/dX^2 ; corresponds to the slave acceleration at constant master velocity 1). The start and end points of the cam must always be included.

5.7.3 SMC_CAMTable_<Variable type>_<Number of elements>_1

An equidistant cam table is described in this data structure. The individual slave positions are stored in Table: ARRAY[0..<Number of elements>-1] OF <Variable type>. The start and end points of the cam must always be included.

The variables fEditorMasterMin, fEditorMasterMax, fTableMasterMin, fTableMasterMax describe an additional scaling of the tables, whereby the range of definitions/values is stored in SoftMotion units (fEditorMaster, fEditorSlave) and scaled on table units (fTableMaster, fTableSlave).

5.7.4 SMC_CAMTable_<Variable type>_<Number of elements>_2

A non-equidistant cam table is stored in Table: ARRAY[0..<Number of elements>-1] OF ARRAY[0..1] OF <Variable type>. In contrast to the equidistant format, the first element is the master position and the second is the slave position.

5.8 Example: manually generated cam

This example illustrates how to create a cam in the IEC program without using the editor:

Variable declaration:

```
CAM: MC_CAM_REF:=(  
  byType:=2, (* non-equidistant *)  
  byVarType:=2, (* UINT *)  
  nElements:=128,  
  xStart:=0,  
  xEnd:=360);  
Table: SMC_CAMTable_UINT_128_1:=(  
  fEditorMasterMin := 0, fEditorMasterMax := 360,  
  fTableMasterMin := 0, fTableMasterMax := 65536,  
  fEditorSlaveMin := 0, fEditorSlaveMax := 360,  
  fTableSlaveMin := 0, fTableSlaveMax := 65536);
```

Program part:

```
(* Create cam (a line for example); unique *)  
FOR i:=0 TO 127 DO  
  Table.Table[i][0]:=Table.Table[i][1]:=REAL_TO_UINT(i / 127.0 * 65536);  
END_FOR  
  
(* Link pointers, this must be done in every cycle!!! *)  
CAM.pce := ADR(Table);
```

The cam created now can be entered in the MC_CamTableSelect POU and its output can in turn be reused for MC_CamIn.

6 The SM_PLCopen.lib library

The program organisation units provided in the "SM_PLCopen.lib" library are based on the PLCopen specification "Function blocks for motion control, Version 1.0".

This description is also based on that specification. Program organisation units based on the PLCopen standard are called MC_<POU name>, whilst 3S-specific program organisation units are named SMC_<POU name>.

The function blocks that are fully programmed in IEC 1131-3 can be divided into three categories:

- ▶ Program organisation units for the general operation, monitoring and parameterisation of individual drives
- ▶ Program organisation units for the independent motion control of individual drives
 - These program organisation units help to move single axes autonomously in different ways.
- ▶ Program organisation units for the motion control of a drive (slave) in accordance with another drive (master). These program organisation units enable cams, electronic gears and phase shifts to be realised.

The library also contains visualisation templates for all key program organisation units. These templates are linked to an instance of the corresponding program organisation unit and represent its inputs and outputs. They are particularly useful when it comes to creating and testing applications.

Preconditions

This library is based on the "SM_DriveBasic.lib" library, which contains, inter alia, the definition of the AXIS_REF structure.

6.1 PLCOpen specification

We recommend that you read the PLCOpen specification "Function blocks for motion control, Version 1.0" as well as this description. The main points are summarised below:

Program organisation units are activated in two ways:

- A. Enable input: If the program organisation unit has an enable input (such as MC_ReadParameter), it will remain active for as long as the enable is set to TRUE.
- B. Execute input: The program organisation unit is activated on a rising edge (transition from FALSE to TRUE) of the Execute input and will not become inactive again until it has completed its motion, another program organisation unit has taken control of the axis (AXIS_REF) or a new rising edge has occurred at the Execute input and the motion has thus been restarted. Please also note that all input variables will only be read on a rising edge.

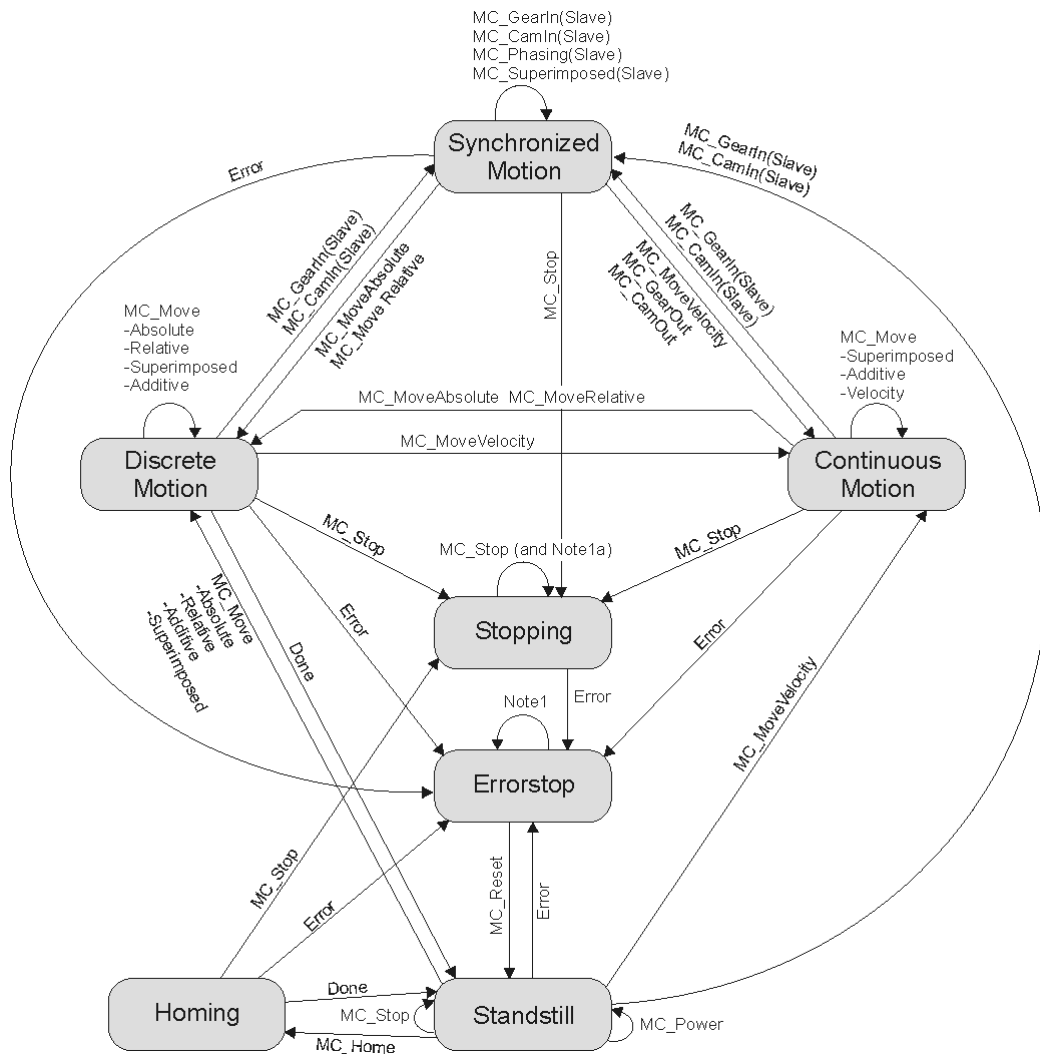
The program organisation units indicate either the validity of the outputs (e.g. MC_ReadStatus) or the completion of the motion (e.g. MC_MoveAbsolute) by means of the Done output (or another similar output).

A program organisation unit that generates a motion and is interrupted by another will indicate this by setting its CommandAborted output.

Once the Done output has been set, the outputs of the program organisation units started via Execute will remain unchanged for as long as the Execute input is set. They will be cleared on a falling edge. If a falling edge has been detected prior to termination, the outputs will be set for one cycle and cleared in the next one.

All program organisation units that generate motion need the controller enable to have been set and the brake released in the corresponding axis, otherwise they will report an error.

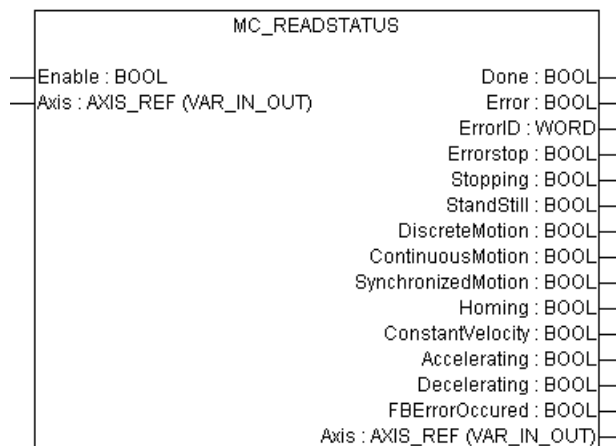
The possible state transitions of an axis are shown in the following chart. Each state change shown here is set in motion by the call of a PLCopen function block.



6.2 POU's for the motion control of individual axes

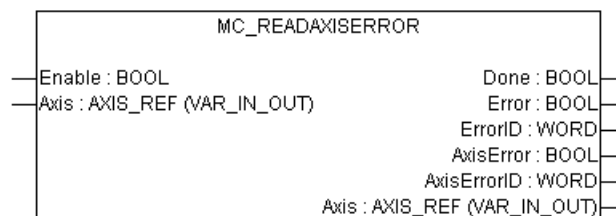
6.2.1 MC_ReadStatus

This program organisation unit from SM_PLCOpen.lib provides some selected states relating to an axis.



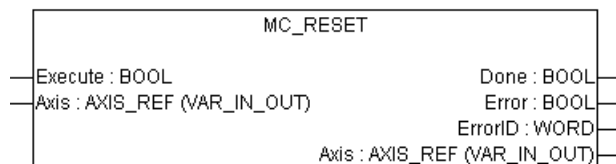
6.2.2 MC_ReadAxisError

This program organisation unit from SM_PLCOpen.lib returns general errors that have occurred on the drive.



6.2.3 MC_Reset

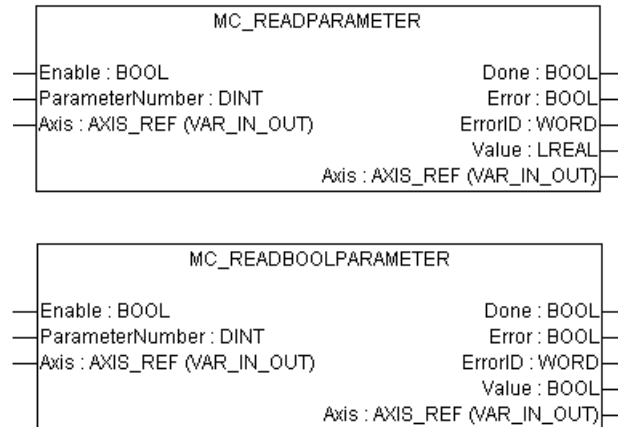
This program organisation unit from SM_PLCOpen.lib resets the axis state (SMC_AXIS_STATE) from error_stop to standstill.



6.2.4 MC_ReadParameter / MC_ReadBoolParameter

These program organisation units from SM_PLCOpen.lib enable some standard parameters to be read from the drive structure. Some of their numbers are specified by PLCOpen and described in the SoftMotion Drive Interface.

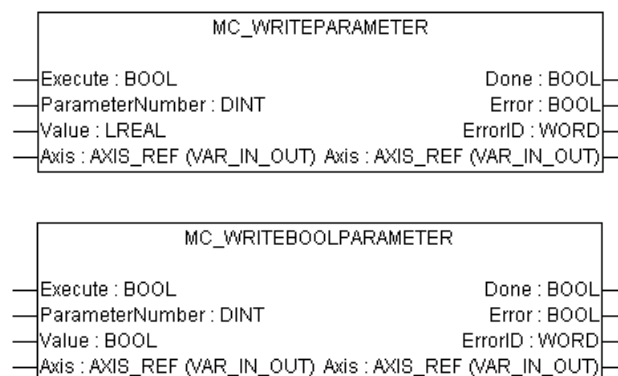
These program organisation units can also be used to read manufacturer-specific data from the drive. A document relating to the particular drive library (XXXDrive.lib) describes the coding of the drive-specific parameter numbers.



6.2.5 MC_WriteParameter / MC_WriteBoolParameter

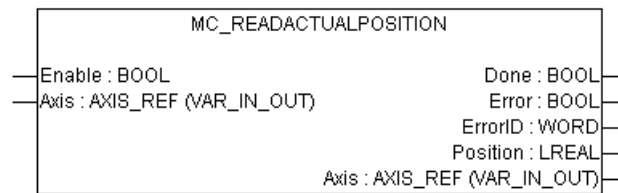
These program organisation units enable some standard parameters of the drive structure to be set. Some of their numbers are specified by PLCOpen and described in the SoftMotion Drive Interface.

These program organisation units can also be used to send manufacturer-specific data to the drive. A document relating to the particular drive library (XXXDrive.lib) describes the coding of the drive-specific parameter numbers.



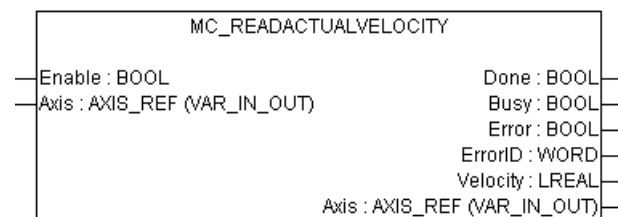
6.2.6 MC_ReadActualPosition

This program organisation unit provides the current drive position.



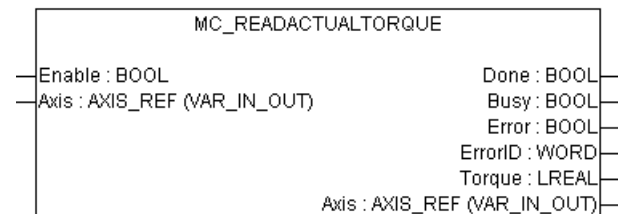
6.2.7 MC_ReadActualVelocity

This program organisation unit provides the current drive velocity.



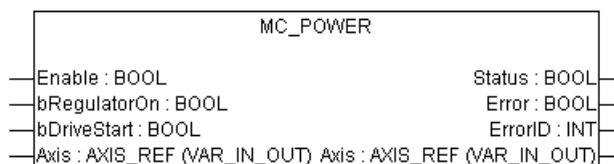
6.2.8 MC_ReadActualTorque

This program organisation unit provides the current drive torque or force.



6.2.9 MC_Power

This program organisation unit from SM_PLCopen.lib controls the ON/OFF switch (power) and the status of the drive brakes. If a drive has not been switched on in this way, if no controller enable has been set or if the brake has not been released, motion control will not be possible on this drive.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Enable : BOOL (default: FALSE)

As long as this variable is TRUE, the function block is active.

bRegulatorOn : BOOL (default: FALSE)

Switches the regulator on/off.

bDriveStart : BOOL (default: FALSE)

Applies/releases the brake in the drive.

Outputs (VAR_OUT)

Status : BOOL (default: FALSE)

Indicates whether the drive is currently being regulated (TRUE) or not (FALSE).

Error : BOOL (default: FALSE)

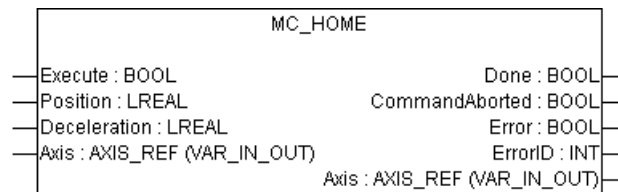
TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT);

Error number

6.2.10 MC_Home

This program organisation unit from SM_PLCOpen.lib starts a reference run in the drive. This is manufacturer-specific and can only be started via the Drive Interface. Once the drive indicates that the run has been completed successfully, the "Done" output is set.



Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The drive reference run will be started on a rising edge.

Position : REAL

Indicates the absolute position of the drive when the reference signal is detected.

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

If this is TRUE, the reference run has been completed and the drive is at standstill again.

CommandAborted : BOOL (default: FALSE)

If this is TRUE, the command was aborted by another.

Error : BOOL (default: FALSE)

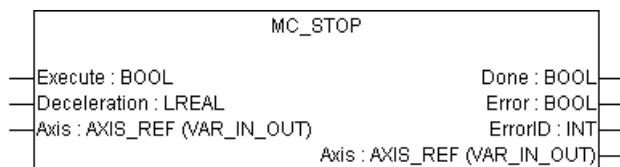
TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

6.2.11 MC_Stop

This program organisation unit from SM_PLCOpen.lib decelerates the axis to velocity 0. This program organisation unit cannot be interrupted and blocks the axis for as long as the Execute input is set and the axis has not yet been stopped completely.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge, i.e. the deceleration will begin.

Deceleration : REAL

Deceleration value [μ/s^2]

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the drive has stopped.

Error : BOOL (default: FALSE)

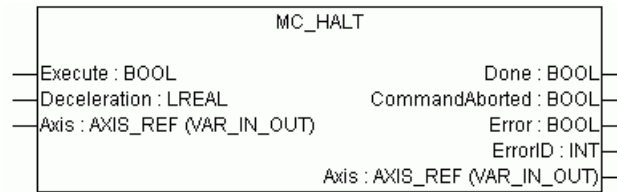
TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

6.2.12 MC_Halt

This POU of SM_PLCopen.lib brakes the axis to the speed 0. This POU can be interrupted and does not block the axis like the POU MC_Stop.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge, i.e. the deceleration will begin.

Deceleration : REAL

Deceleration value [u/s^2]

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the drive has stopped.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis.

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

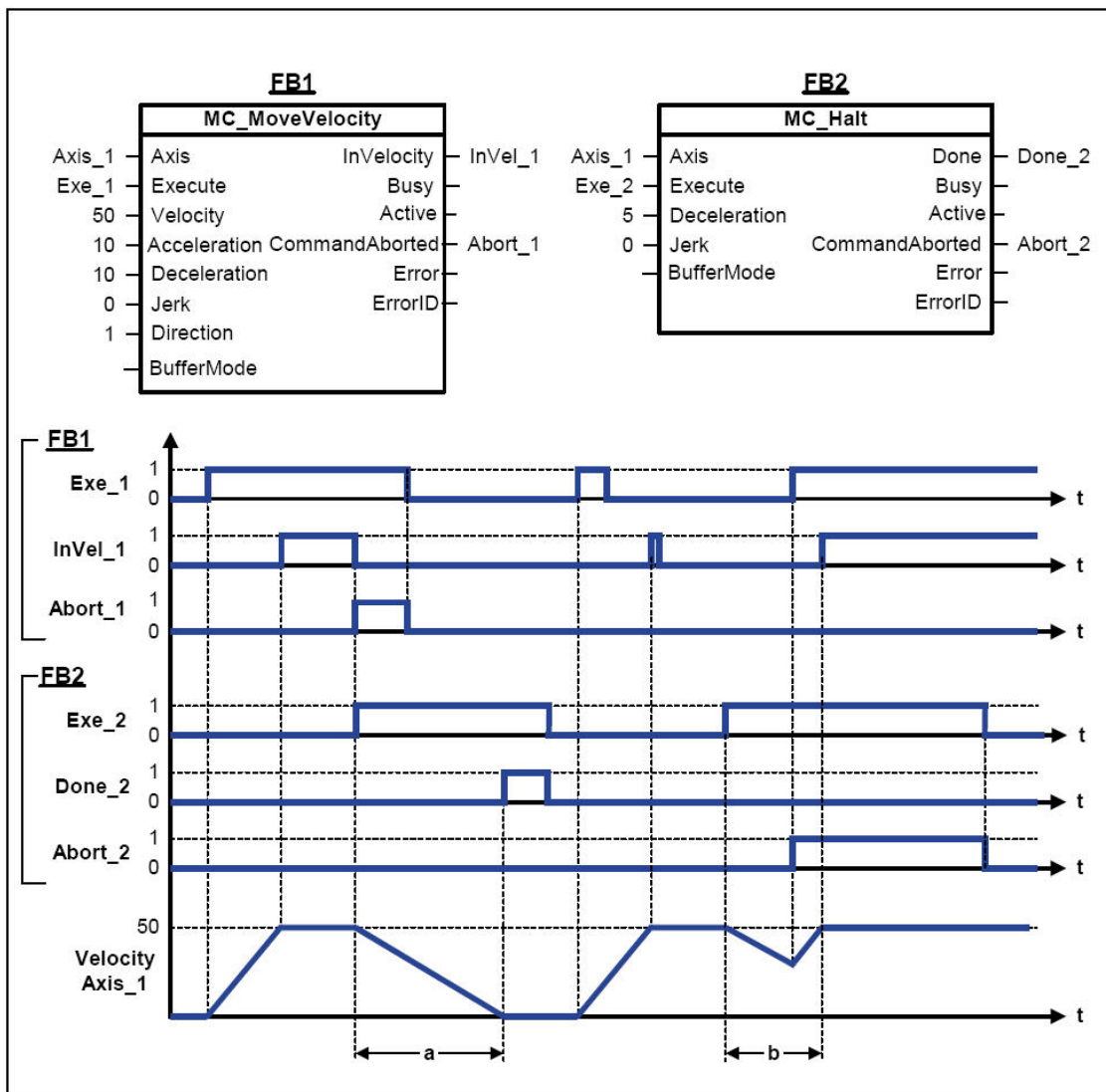
ErrorID : SMC_Error (INT)

Error number

Example: The example below shows the behavior in combination with a MC_MoveVelocity:

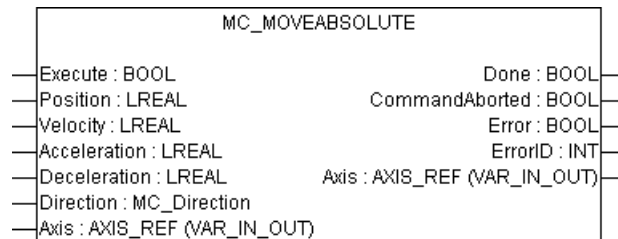
A rotating axis is ramped down with FB MC_Halt.

Another motion command overrides the MC_Halt command. MC_Halt allows this, in contrast to MC_Stop. The axis can accelerate again without reaching standstill.



6.2.13 MC_MoveAbsolute

This program organisation unit from SM_PLCOpen.lib moves the axis to an absolute position in accordance with the defined velocity, deceleration and acceleration values. In the case of a linear axis, the direction value is of no significance; in the case of rotating axes it determines the direction of rotation.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

Position : REAL

Target position for the motion (technical unit [u])

Velocity : REAL

Value of the setpoint velocity that does not necessarily have to be reached [u/s]

Acceleration : REAL

Desired acceleration [u/s²]

Deceleration : REAL

Desired deceleration [u/s²]

nDirection : MC_Direction (default: shortest)

This enumeration value specifies the desired direction, which is only relevant for rotating axes: see SM_DriveBasic.lib. Permissible values: current (current direction of motion), positive, negative, shortest (shortest distance as viewed from the current position), fastest (direction that would complete the motion in the fastest time possible).

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the desired position has been reached.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

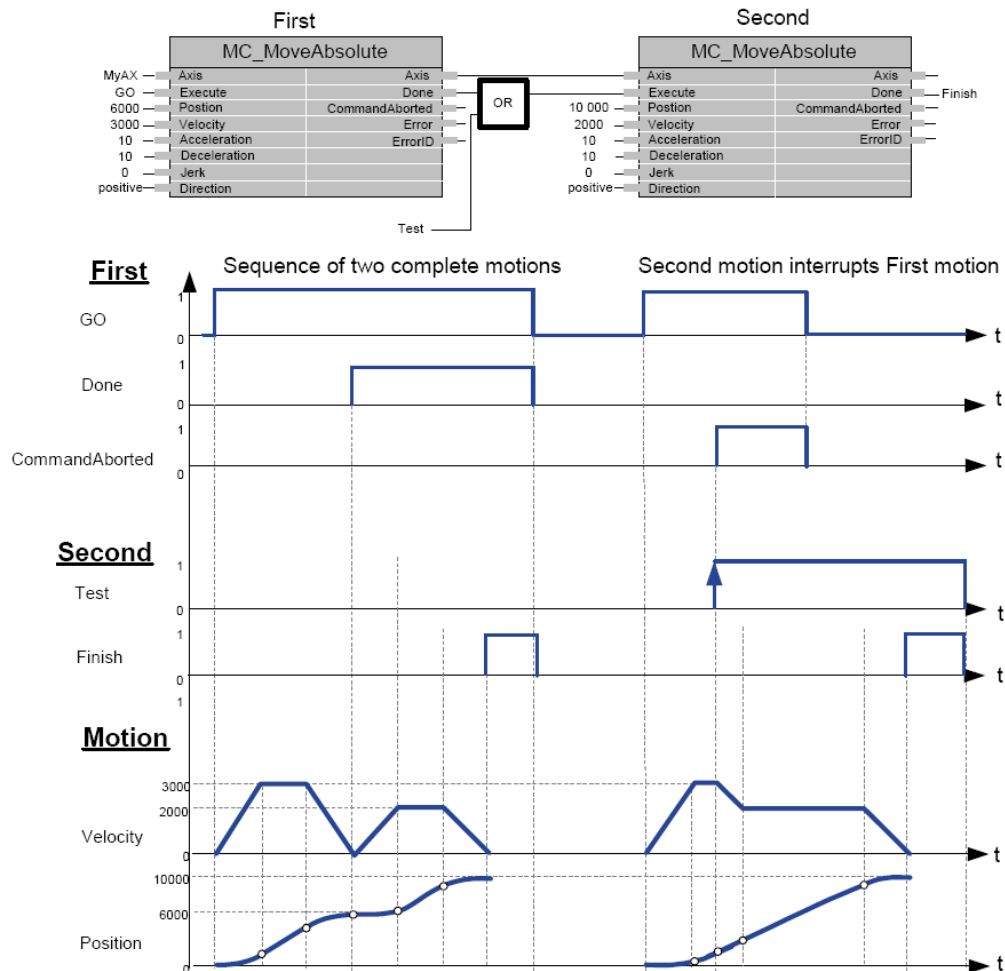
ErrorID : SMC_Error (INT)

Error number

Example:

The following figure shows two examples based on the combination of two instances of the Function Block (instance "First" and "Second"):

MoveAbsolute - Example

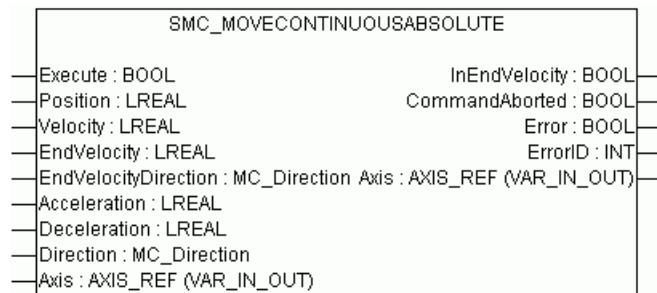


1. The left part of timing diagram illustrates the case if the Second Function Block is called after the First one. If First reaches the commanded position of 6000 (and the velocity is 0) then the output Done causes the Second FB to move to the position 10000.
2. The right part of the timing diagram illustrates the case if the Second move Function Block starts the execution while the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB moves directly to the position 10000 although the position of 6000 is not yet reached.

6.2.14 SMC_MoveContinuousAbsolute

This POU moves the axis to a fixed position according to the specified values for speed, acceleration, and deceleration. In contrast to MC_MoveAbsolute, the user can additionally define a final speed that the axis has at this position, which the POU displays by the output InEndVelocity. If the POU is not interrupted by another POU, it keeps on moving the axis continuously (axis status = move_continuous) with the final speed defined.

By corresponding specifications (path required to bring the axis to the final speed < distance between the position at the start and target position) it can happen that the axis at first carries out a movement in the opposite direction.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

Position : REAL

Target position for the motion (technical unit [u])

Velocity : REAL

Value of the setpoint speed (>0) that does not necessarily have to be reached [u/s]

EndVelocity : REAL

Value of the final setpoint speed which the axis has to have after it has covered the distance [u/s].

EndVelocityDirection : MC_Direction

Direction of the final setpoint speed (positive, negative or current (direction of rotation of the axis at the start of the movement)).

Acceleration : REAL

Desired acceleration [u/s²]

Deceleration : REAL

Desired deceleration [u/s²]

Direction : MC_Direction (Default: shortest)

This enumeration value specifies the desired direction, which is only relevant for rotating axes: see SM_DriveBasic.lib. Permissible values: current (current direction of motion), positive, negative, shortest (shortest distance as viewed from the current position).

Outputs (VAR_OUT)

InEndVelocity : BOOL (Default: FALSE)

TRUE indicates that the desired distance has been traversed and the specified speed has been reached.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

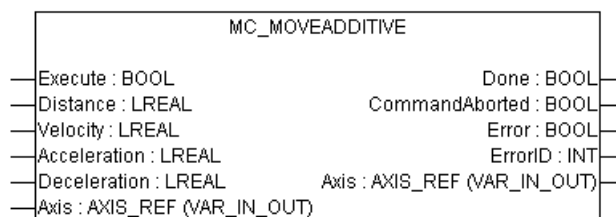
ErrorID : SMC_Error (INT)

Error number

6.2.15 MC_MoveAdditive

This program organisation unit from SM_PLCOpen.lib has two different modes of action, depending on the current state of the axis:

- ▶ discrete_motion
 - The distance will be added to the target position of the program organisation unit currently working on the axis and the newly defined target position will be approached according to the specified parameters.
- ▶ continuous_motion or standstill:
 - The distance will be covered as measured from the current position in accordance with the specified parameters.

**Input/output (VAR_IN_OUT)**

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

Distance : REAL

Relative distance for the motion (technical unit [u])

Velocity : REAL

Value of the setpoint velocity that does not necessarily have to be reached [u/s]

Acceleration : REAL

Desired acceleration [u/s²]

Deceleration : REAL

Desired deceleration [u/s²]

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the desired distance has been reached.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

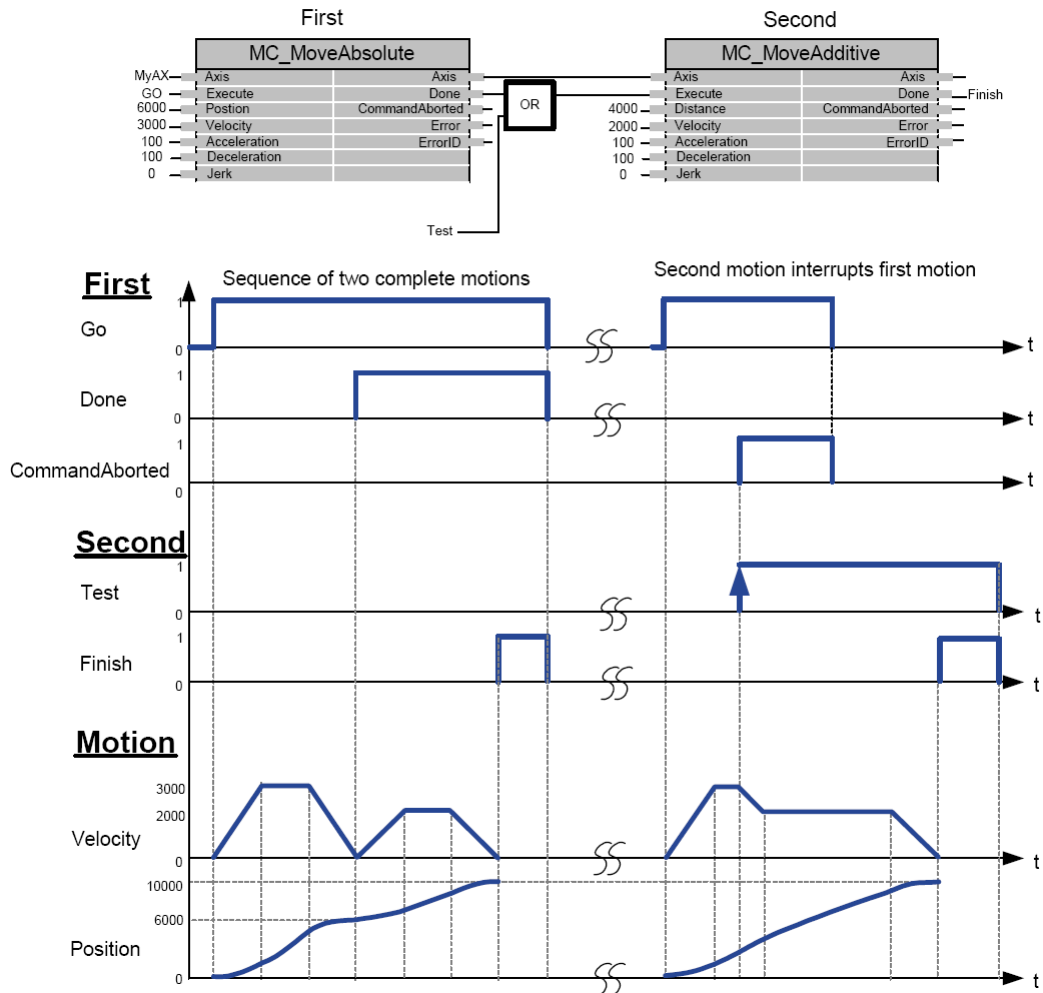
ErrorID : SMC_Error (INT)

Error number

Example:

The following figure shows two examples of the combination of two Function Blocks while the axis is in Discrete Motion state:

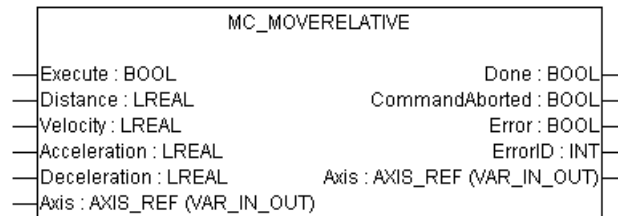
MoveAdditive - Example



1. The left part of timing diagram illustrates the case if the Second Function Block is called after the First one. If First reaches the commanded distance 6000 (and the velocity is 0) then the output 'Done' causes the Second FB to move to the distance 10000.
2. The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution while the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB adds on the previous commanded position of 6000 the distance 4000 and moves the axis to the resulting position of 10000.

6.2.16 MC_MoveRelative

This program organisation unit from SM_PLCopen.lib moves the axis by a relative distance in accordance with the defined velocity, acceleration and deceleration values. The distance may be positive or negative.

**Input/output (VAR_IN_OUT)**

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

Distance : REAL

Relative distance for the motion (technical unit [u])

Velocity : REAL

Value of the setpoint velocity that does not necessarily have to be reached [u/s]

Acceleration : REAL

Desired acceleration [u/s²]

Deceleration : REAL

Desired deceleration [u/s²]

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the desired distance has been reached.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

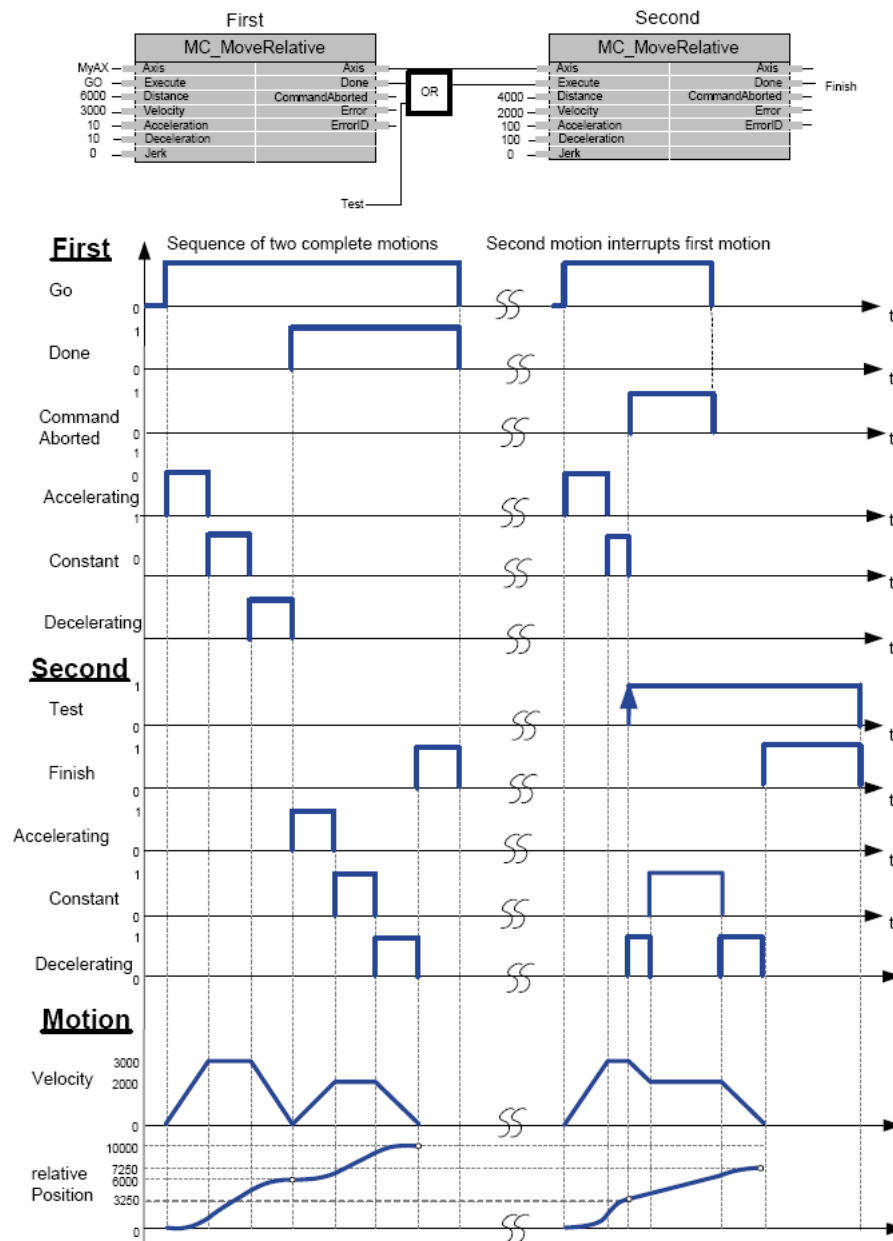
ErrorID : SMC_Error (INT)

Error number

Example:

The following figure shows the example of the combination of two relative move Function Blocks:

MoveRelative - Example

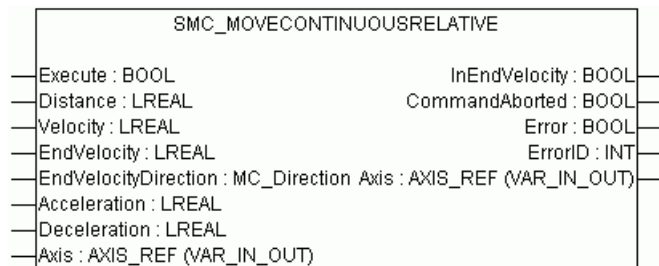


1. The left part of timing diagram illustrates the case if the Second Function Block is called after the First one. If First reaches the commanded distance 6000 (and the velocity is 0) then the output Done causes the Second FB to move to the distance 10000.
2. The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution while the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB adds on the actual position of 3250 the distance 4000 and moves the axis to the resulting position of 7250.

6.2.17 SMC_MoveContinuousRelative

This POU moves the axis by a relative distance according to the specified values for speed, acceleration, and deceleration. The distance can be positive or negative. In contrast to MC_MoveRelative, the user can additionally define a final speed that the axis has after the distance specified has been traversed, which the POU displays by the output InEndVelocity. If the POU is not interrupted by another POU, it keeps on moving the axis continuously (axis status = move_continuous) with the final speed defined.

By corresponding specifications (path required to bring the axis to the final speed < distance) it can happen that the axis at first carries out a movement in the opposite direction.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

Distance : REAL

Relative distance measured from the position of the axis at the time of the rising edge at Execute, after which the final speed must have been reached.

Velocity : REAL

Value of the setpoint speed (>0) that does not necessarily have to be reached [u/s]

EndVelocity : REAL

Value of the final setpoint speed which the axis has to have after it has covered the distance [u/s].

EndVelocityDirection : MC_Direction

Direction of the final setpoint speed (positive, negative or current (direction of rotation of the axis at the start of the movement)).

Acceleration : REAL

Desired acceleration [u/s²]

Deceleration : REAL

Desired deceleration [u/s²]

Outputs (VAR_OUT)

InEndVelocity : BOOL (Default: FALSE)

TRUE indicates that the desired distance has been traversed and the specified speed has been reached.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

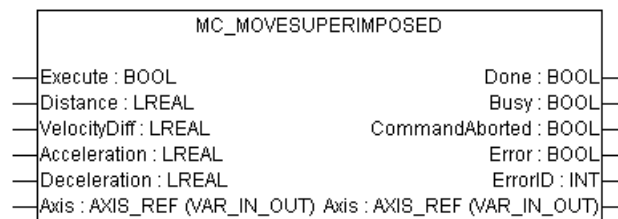
Error number

6.2.18 MC_MoveSuperImposed

Where applicable, this program organisation unit from SM_PLCopen.lib executes a motion in addition to the currently active motion, which moves the axis across a defined distance. The specified velocity, acceleration and deceleration values must be regarded as relative values, i.e. they are independent of the underlying motion.

The program organisation unit that was originally active will not be interrupted by MC_MoveSuperImposed. If it is interrupted by another program organisation unit whilst MC_MoveSuperImposed is active, the MC_MoveSuperImposed motion will be aborted.

Please note that MC_MoveSuperImposed must be called after the program organisation unit generating the underlying motion.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

Distance : REAL

Relative distance for the motion (technical unit [u])

VelocityDiff : REAL

Maximum velocity difference in relation to the current motion that does not necessarily have to be reached [u/s]

Acceleration : REAL

Desired acceleration [u/s²]

Deceleration : REAL

Desired deceleration [u/s²]

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the desired distance has been covered.

Busy : BOOL (default: FALSE)

TRUE indicates that the superimposed motion is currently being executed.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis.

Error : BOOL (default: FALSE)

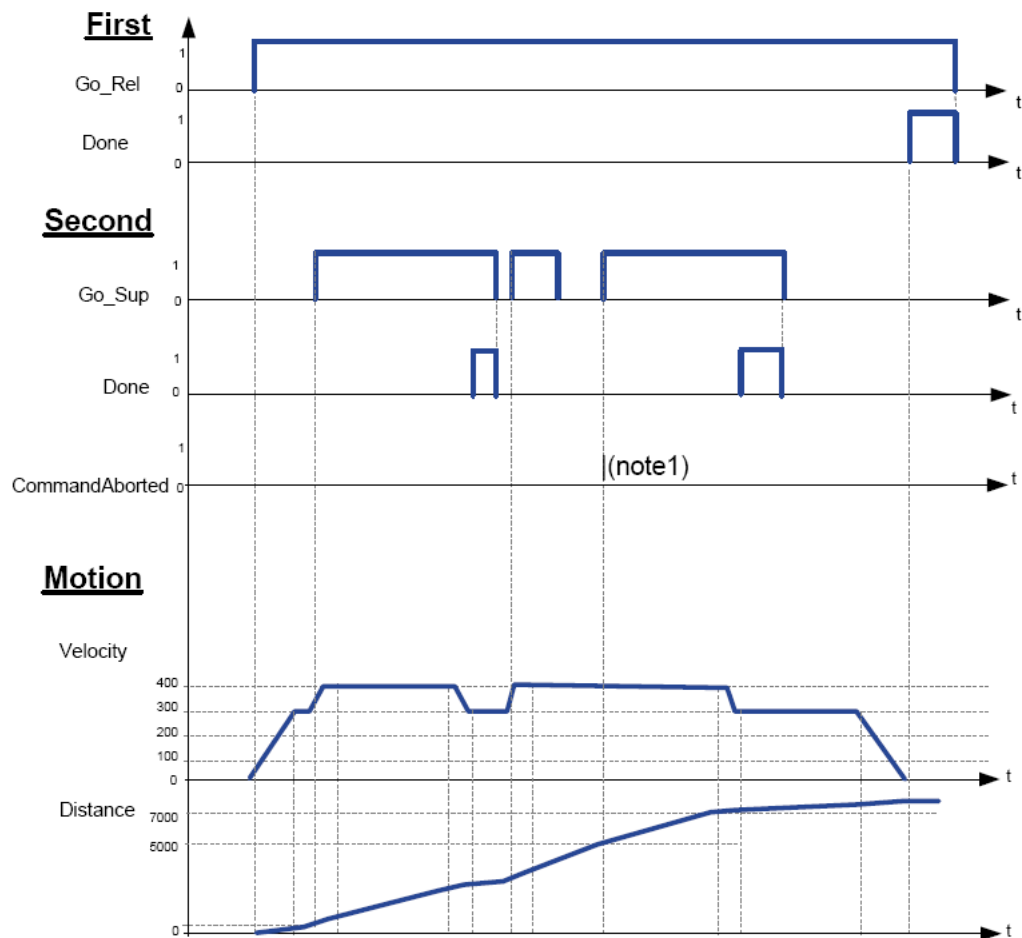
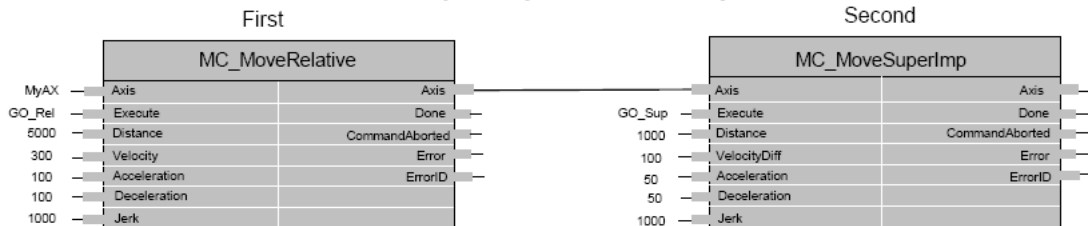
TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

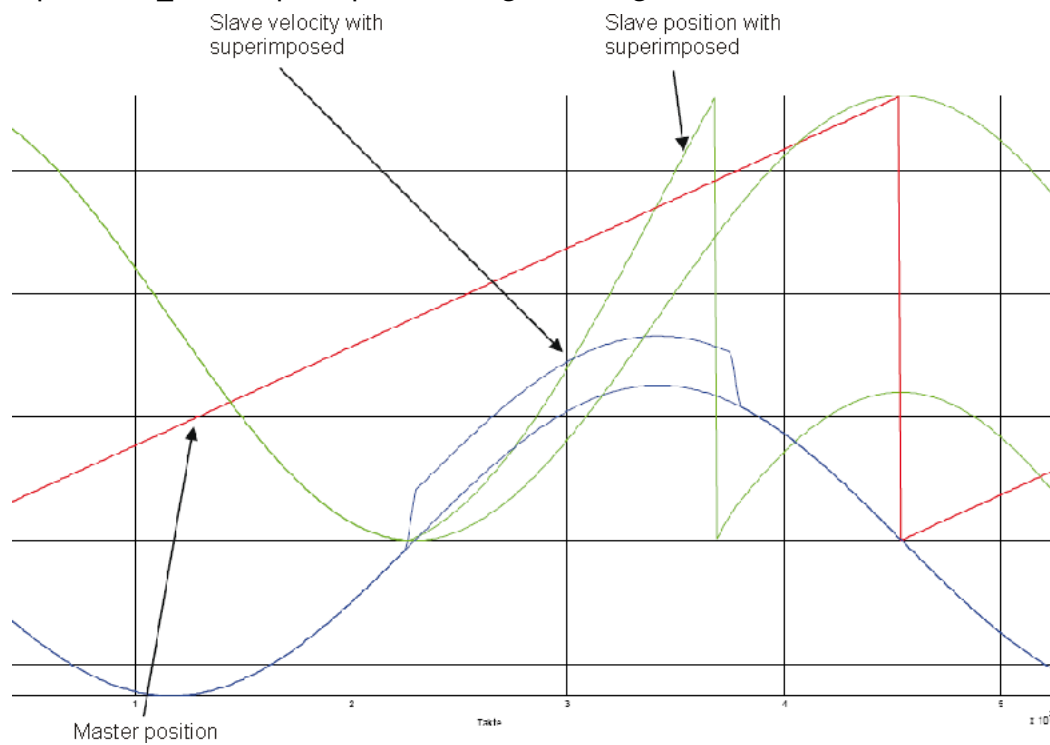
Example:

MoveSuperimposed - Example



Note: the CommandAborted is not visible at the point marked with "(note1)", because the new command works on the same instance. The end position is between 7000 and 8000, depending on the timing of the aborting of the second command set for the MC_MoveSuperimposed.

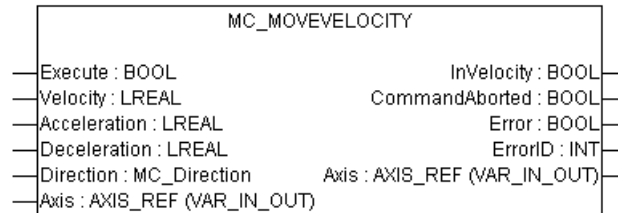
Example of MC_MoveSuperImposed during Camming:



Note: at Slave velocity, the double line shows the effect of MoveSuperImposed while in synchronized motion during Camming. The same is valid for the related slave position.

6.2.19 MC_MoveVelocity

This program organisation unit from SM_PLCOpen.lib causes the axis to move continuously at a defined velocity. In order to reach this velocity, MC_MoveVelocity uses the programmed acceleration and deceleration values. The target velocity is always positive. The Direction input variable defines the direction.



Input/output (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

Velocity : REAL

Value of the maximum velocity that does not necessarily have to be reached [u/s]

Acceleration : REAL

Desired acceleration [u/s²]

Deceleration : REAL

Desired deceleration [u/s²]

Direction : MC_Direction (default: current)

This enumeration value specifies the desired direction (see SM_DriveBasic.lib). Permissible values: current, positive, negative.

Outputs (VAR_OUT)

InVelocity : BOOL (default: FALSE)

TRUE indicates that the defined velocity has been reached.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

Error : BOOL (default: FALSE)

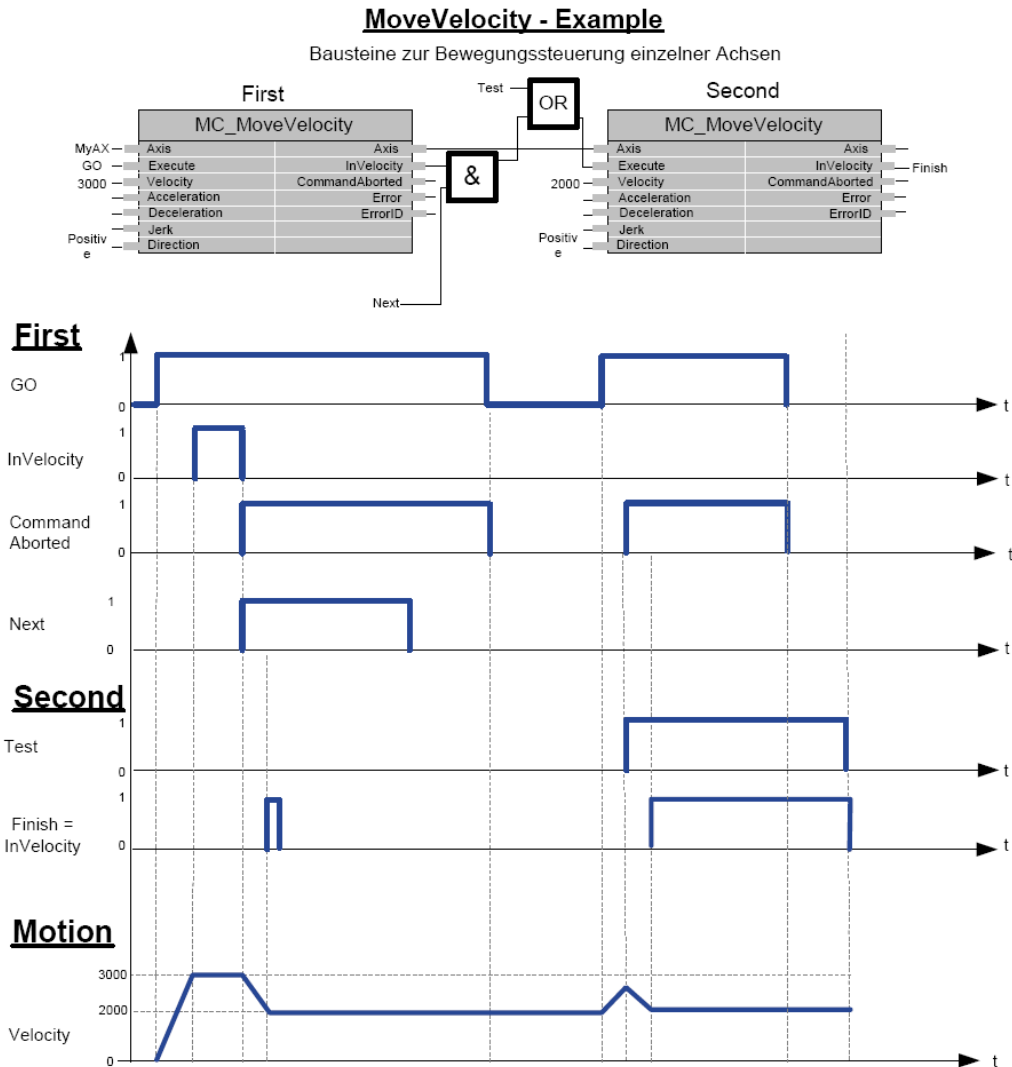
TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

Example:

The following figure shows two examples of the combination of two MoveVelocity Function Blocks:



1. The left part of timing diagram illustrates the case if the Second Function Block is called after the First one is completed. If First reaches the commanded velocity 3000 then the output First.InVelocity AND the signal Next cause the Second FB to move to the velocity 2000.
2. The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution while the First FB is not yet InVelocity. The following sequence is shown: The First motion is started again by Go at the input First.Execute. While the First FB is still accelerating to reach the velocity 3000 the First FB will be interrupted and aborted because the Test signal starts the Run of the Second FB. Now the Second FB runs and decelerates the velocity to 2000.

6.2.20 MC_PositionProfile

This program organisation unit from SM_PLCOpen.lib follows a defined position profile. A variable of the MC_TP_REF structure must be defined and filled in.

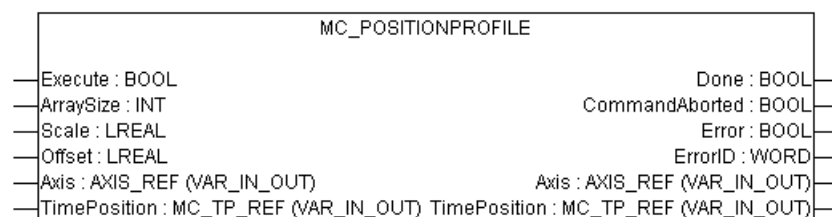
MC_TP_REF variables

Variable	Type	Initial value	Description
Number_of_pairs	INT	0	Number of profile position points
IsAbsolute	BOOL	TRUE	Absolute or relative positions
MC_TP_Array	ARRAY[1..100] OF SMC_TP		Position points

MC_TP variables

Variable	Type	Initial value	Description
delta_time	TIME	T#0s	Period of time between arrival at the previous position point and this one
position	REAL	0	(Absolute/relative) position of the position point

The program organisation unit uses the defined positions to create a double, continuously differentiable curve composed of cubic polynomials.



Inputs/outputs (VAR_IN_OUT)

Axis : AXIS_REF

The structure that is filled with axis data in the Drive Interface (SM_DriveBasic.lib) is shown here.

TimePosition : MC_TP_REF

Information on time and position values, see above.

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit is activated on a rising edge.

ArraySize : INT

Size of the array (max. 1..100)

Scale : REAL (default: 1)

General scaling factor for the profile

Offset : REAL

General offset for the profile [u]

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the desired distance has been covered.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

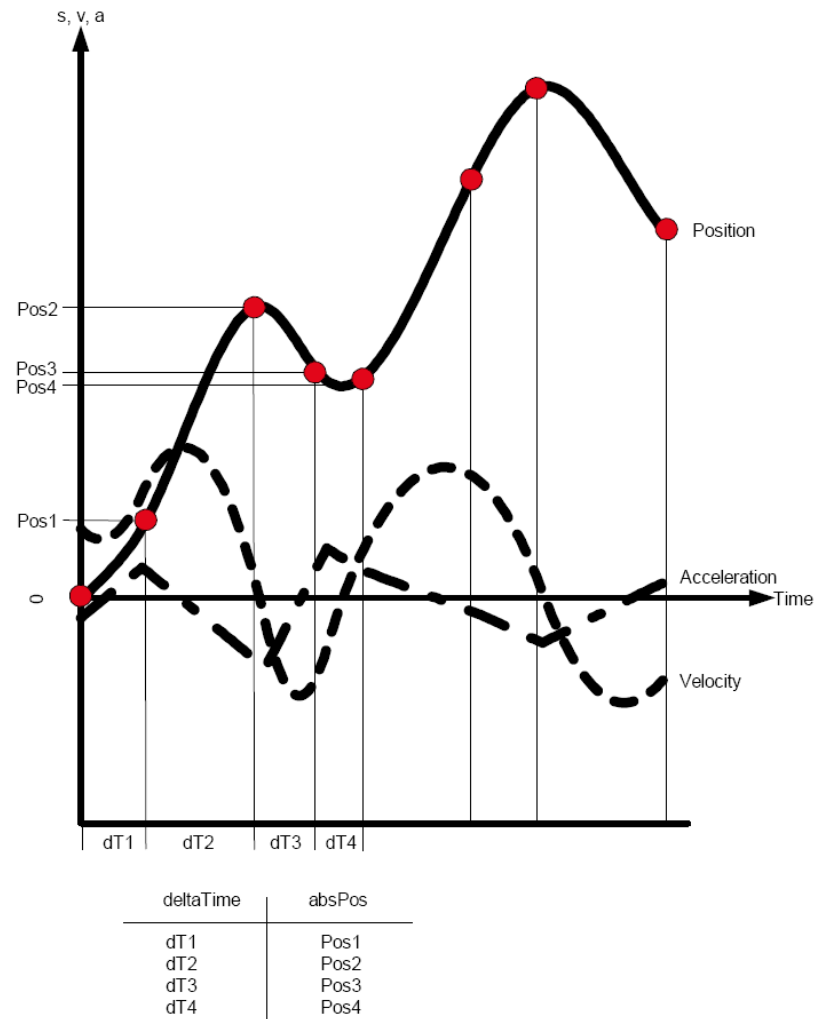
Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

Example of Time / Position Profile:



Note: The Time / Velocity and Time / Acceleration Profiles are similar to the Position Profile, with sampling points on the Velocity or Acceleration lines.

6.2.21 MC_VelocityProfile

This program organisation unit from SM_PLCopen.lib functions in a similar way to the MC_PositionProfile program organisation unit. Here, however, the position points are defined by their velocities in the input variables of the MC_TV_REF structure.

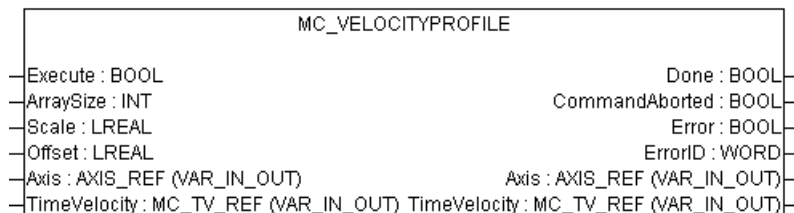
MC_TV_REF variables

Variable	Type	Initial value	Description
Number_of_pairs	INT	0	Number of profile position points
IsAbsolute	BOOL	TRUE	Absolute or relative positions
MC_TV_REF	ARRAY[1..100] OF MC_TV_REF		Position points

MC_TV variables

Variable	Type	Initial value	Description
delta_time	TIME	T#0s	Period of time between arrival at the previous position point and this one
velocity	REAL	0	(Absolute/relative) velocity of the position point

The program organisation unit uses the defined position points to create a continuously differentiable curve composed of parabolas. The axis position is calculated from the start position and the integral over velocity.



6.2.22 MC_AccelerationProfile

This program organisation unit from SM_PLCOpen.lib functions in a similar way to the MC_PositionProfile program organisation unit. Here, however, the position points are defined by their accelerations in the input variables of the MC_TA_REF structure.

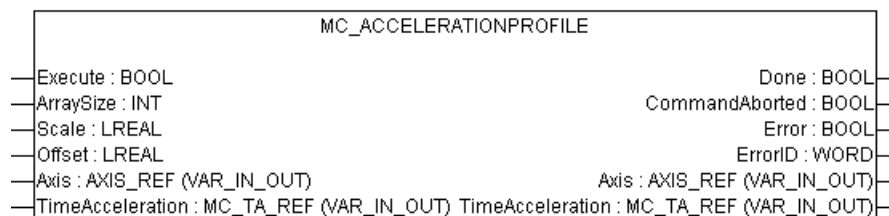
MC_TA_REF variables

Variable	Type	Initial value	Description
Number_of_pairs	INT	0	Number of profile position points
IsAbsolute	BOOL	TRUE	Absolute or relative positions
MC_TA_Array	ARRAY[1..100] OF SMC_TA		Position points

MC_TA variables

Variable	Type	Initial value	Description
delta_time	TIME	T#0s	Period of time between arrival at the previous position point and this one
Acceleration	REAL	0	(Absolute/relative) acceleration of the position point

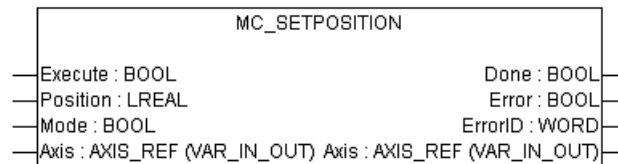
The program organisation unit uses the defined position points to create a continuous curve composed of lines. The velocity of the curve results from the velocity at the start of the profile and the integral over acceleration. The axis position is calculated from the start position and the integral over velocity.



6.2.23 MC_SetPosition

This program organisation unit shifts the zero point of the axis so that ...

- In absolute mode (mode = FALSE; default), the value defined via the Position input will become the current set position.
- In relative mode (mode = TRUE), the current set position will be shifted by the Position value.



Basically, the program organisation unit can be called at any time. However, please note that in the case of a path-controlled motion, if the set positions are fed to the program organisation unit directly, via SMC_ControlAxisByPos, for example, a set position jump may result.

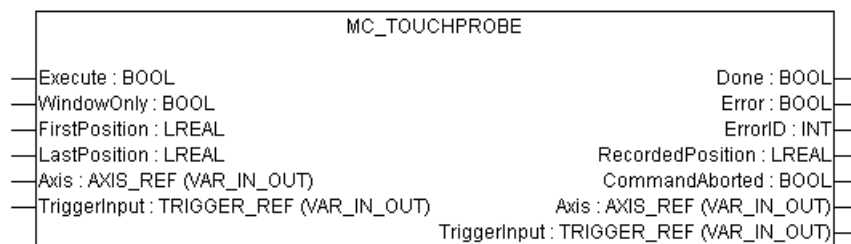
6.2.24 MC_TouchProbe



Note!

This function block is only supported by the **ECSxM (Motion) axis module**.

This program organisation unit is used to detect the precise position of the drive via a fast input. As this generally has to be achieved faster than in a normal PLC cycle, in many cases either the drive is charged with this function or it is executed via interrupts or the like, independently of the PLC cycle.



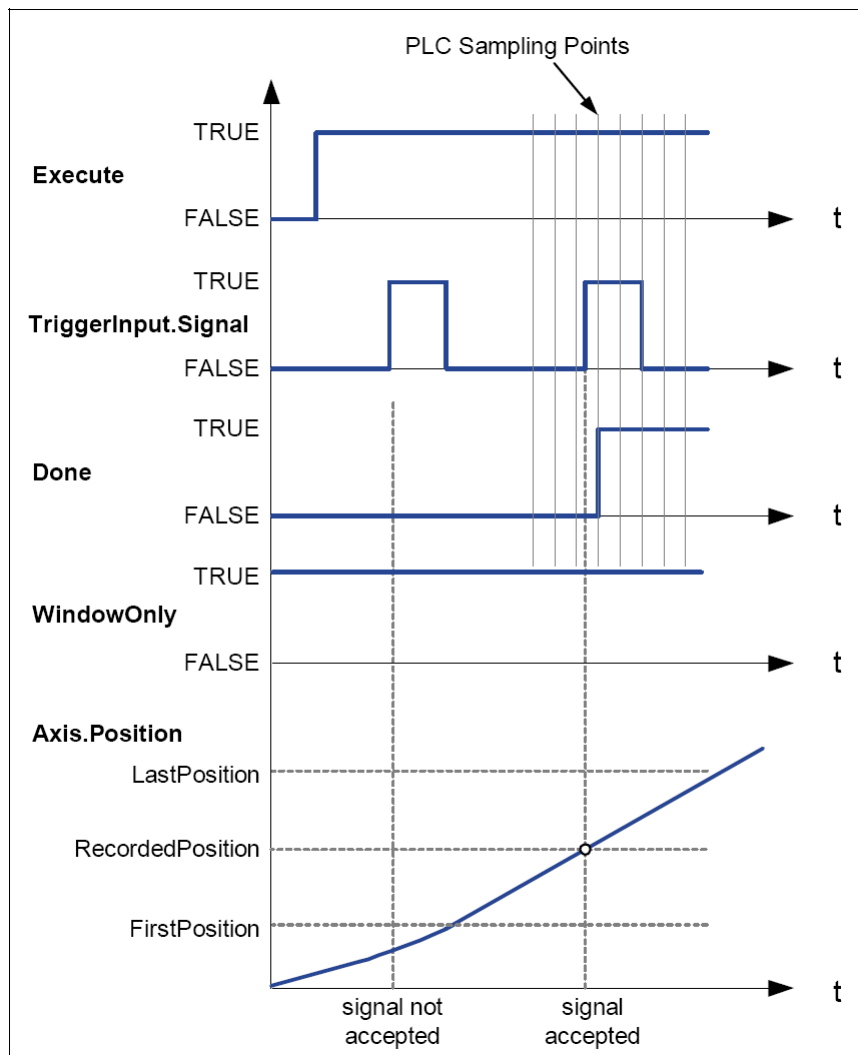
The input "TriggerInput" is of type TRIGGER_REF and describes the trigger input more detailed:

Variable	Type	Initial value	Description
bFastLatching	BOOL	TRUE	Fast latching via Drive Interface (TRUE) or latching in PLC cycle (FALSE)
iTriggerNumber	INT	-1	Only for bFastLatching = TRUE: trigger number; depending on Drive Interface
bInput	BOOL	TRUE	Only for bFastLatching = TRUE: input signal; TRUE initiates latching
bActive	BOOL	TRUE	Internal variable

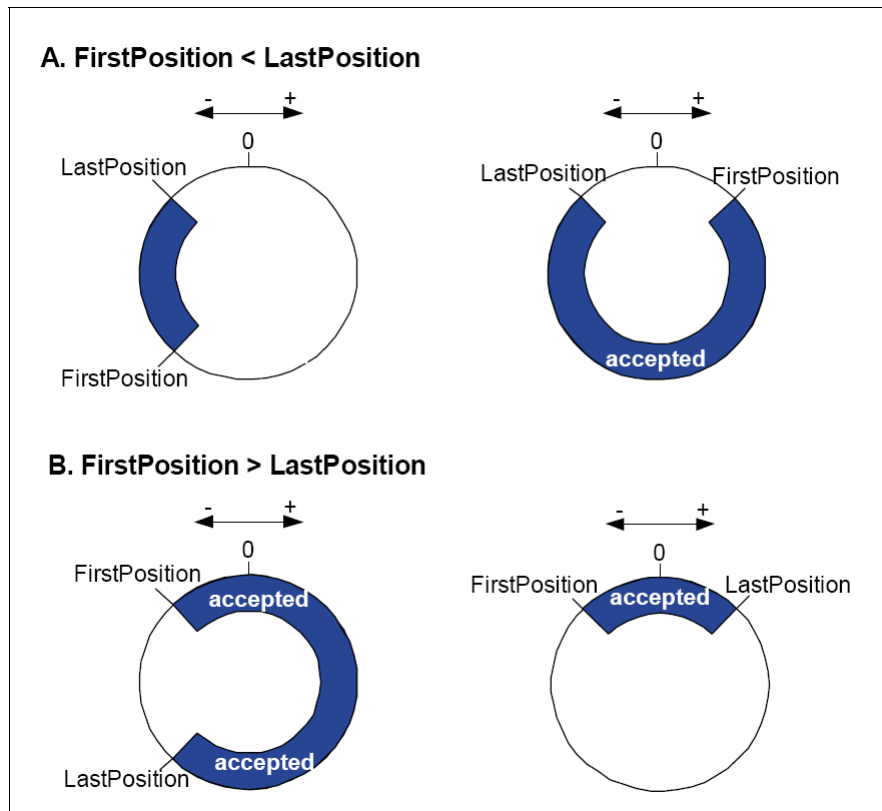
The window function, activated and defined via WindowOnly, FirstPosition, LastPosition, must be supported by the Drive Interface; if this is not the case, it will return an error.

The program organisation unit is independent of the axis state and remains active until a position is latched or the process is aborted via MC_AbortTrigger.

Example: Timing example TouchProbe



Example: Examples of windows, where trigger events are accepted (for modulo axes.)



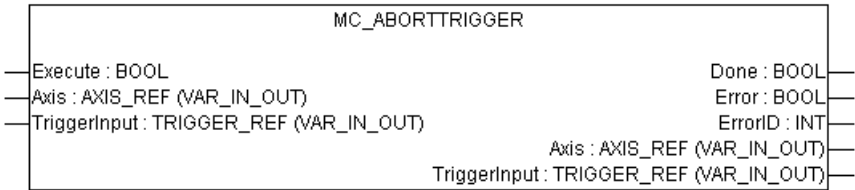
6.2.25 MC_AbortTrigger



Note!

This function block is only supported by the **ECSxM (Motion) axis module**.

This program organisation unit aborts a latching procedure executed at the TriggerInput.



6.3 POU for the synchronised motion control (multi-axis)

6.3.1 MC_CamTableSelect

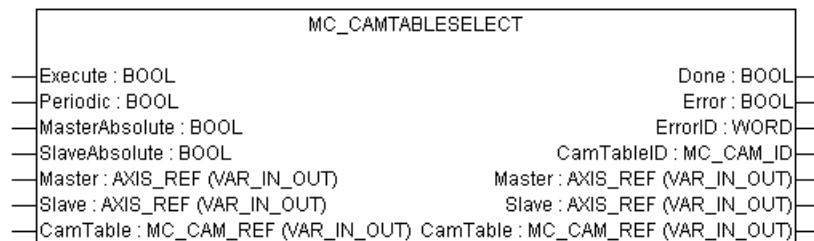
This program organisation unit from SM_PLCOpen.lib is used to select a cam, determine the master and slave axes and make certain presettings. The CamTableID object, which is available as an output, will be passed onto the MC_CamIn cam program organisation unit at a later time.

The master axis can be virtual, i.e. it does not have to physically exist. If the bRegulatorOn variable is set to TRUE, the setpoints will be used, otherwise the actual values will be used.

The cam to be traversed can either be programmed manually in an object of the MC_CAM_REF structure or created by means of the cam editor integrated in the »PLC Designer«.

The Periodic variable determines whether or not the cam will be started again from the beginning once it has been completed.

The MasterAbsolute and SlaveAbsolute variables define whether the cam mapping of the master axis to the slave axis is to refer to absolute values or increments.



Inputs/outputs (VAR_IN_OUT)

Master : AXIS_REF

Master axis

Slave : AXIS_REF

Slave axis

CamTable : MC_CAM_REF

Description of the cam

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit selects a new cam on a rising edge.

Periodic : BOOL (default: TRUE)

Periodic/non-periodic cam

MasterAbsolute : BOOL (default: TRUE)

Cam refers to absolute/relative master position (relative to the position on a rising edge in Execute of CAMIn).

SlaveAbsolute : BOOL (default: TRUE)

Cam refers to absolute/relative slave position (relative to the position on a rising edge in Execute of CAMIn).

Outputs (VAR_OUT)

Done : BOOL (default: FALSE)

TRUE indicates that the desired distance has been covered.

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

CAMTableID : MC_CAM_ID

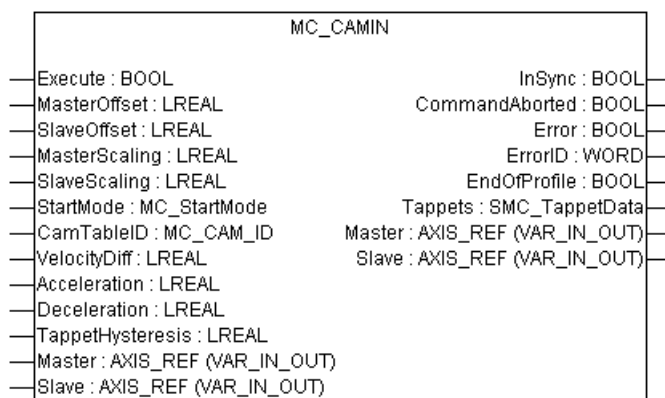
Output that describes the cam. Serves as an input for the input of the same name in MC_CamIn.

6.3.2 MC_CamIn

This program organisation unit from SM_PLCopen.lib implements a cam selected in MC_CAMTABLESELECT.

The start mode can be defined as well as the offsets and scalings. Please note that the ramp_in, ramp_in_pos and ramp_in_neg modes can only be used if the VelocityDiff, Acceleration and Deceleration parameters have been specified. If the slave's actual value and the cam start value do not match at the start, these modes will ensure that the slave's setpoint is constantly fed to the cam.

This program organisation unit also provides an additional function: it detects tappets and can pass tappet information on to one or more SMC_GetTappetValue function POUs via the Tappets output (see SMC_GetTappetValue). Please note that the CamIn program organisation unit can detect a maximum of three tappets at the same time per cycle. The SMC_CAMRegister program organisation unit is not subject to this restriction.



Inputs/outputs (VAR_IN_OUT)

Master : AXIS_REF

Master axis

Slave : AXIS_REF

Slave axis

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

The program organisation unit starts the motion on a rising edge.

MasterOffset : LREAL (default: 0)

Additional offset to master position

SlaveOffset : LREAL (default: 0)

Additional offset to slave position

StartMode : MC_StartMode (absolute/relative/ramp_in/ramp_in_pos/ramp_in_neg)
(default: absolute)

Cam either refers to the actual position relatively (relative) or absolutely, without (absolute) or with slow ramping on the shortest path (ramp_in), in the positive (ramp_in_pos) or negative (ramp_in_neg) direction.

CAMTableID : MC_CAM_ID

Output of MC_CamTableSelect

Velocity, Acceleration, Deceleration: LREAL (default: 0)

Additional velocity, acceleration, deceleration for ramp_in mode

TappetHysteresis: LREAL (default: 0)

Width of the hysteresis band around the tappets

Outputs (VAR_OUT)

InSync : BOOL (default: FALSE)

TRUE indicates that the motion is on the cam.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

EndOfProfile : BOOL

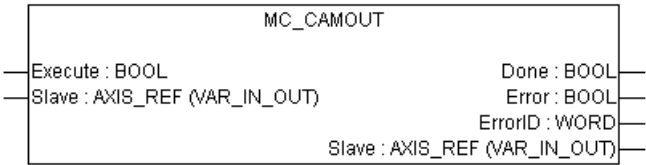
Indicates the end of a cam. With periodic cams this output will be pulsed.

Tappets : SMC_TappetData

Tappet output. The individual tappet positions will be finally evaluated by the SMC_GetTappetValue program organisation unit.

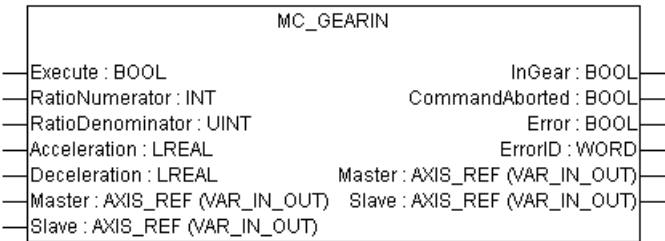
6.3.3 MC_CamOut

This program organisation unit from SM_PLCopen.lib disengages the slave drive from the master and continues to drive the slave at the current velocity.



6.3.4 MC_GearIn

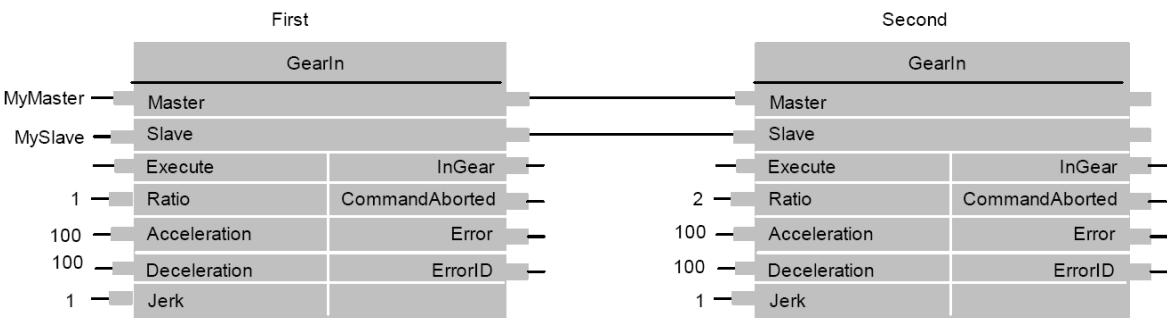
This program organisation unit from SM_PLCopen.lib couples the slave axis to the master axis. The velocity of the slave axis is, thus, f times that of the master axis. The value of factor f results from the quotient of the RatioNumerator and RatioDenominator input parameters.

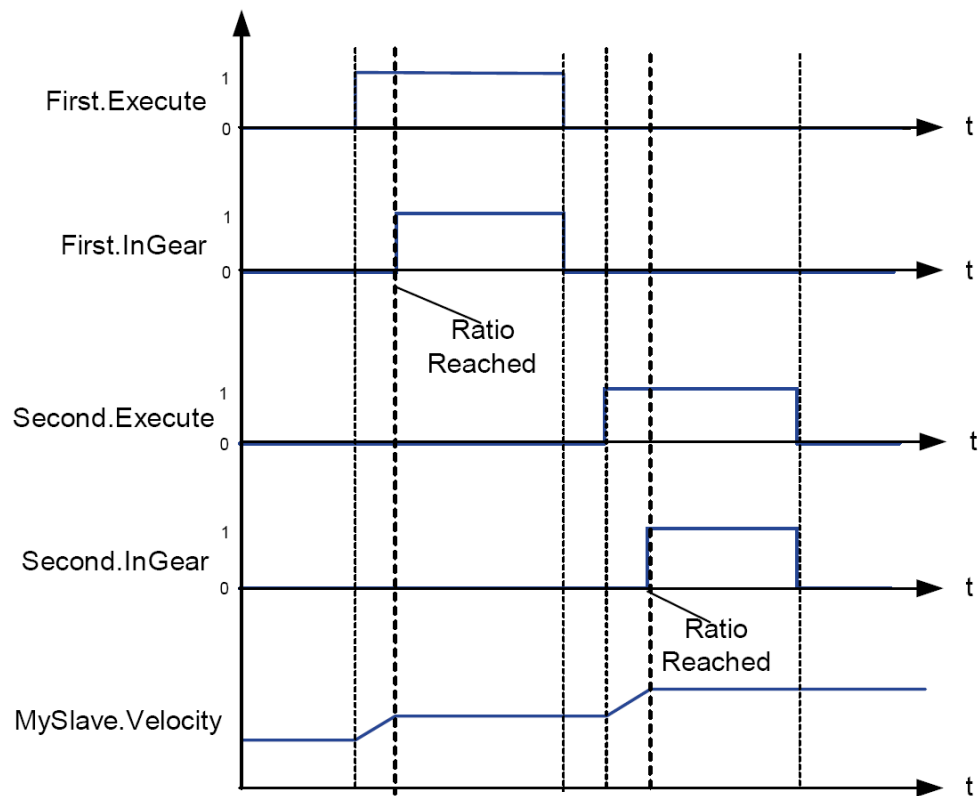


The program organisation unit uses the Acceleration and Deceleration values to accelerate and decelerate the slave axis until its velocity has the required ratio to that of the master axis. Once this has been achieved, the velocity of the slave axis is derived from the master axis.

If the bRegulatorOn variable of the master axis is set to TRUE, the velocity setpoints will be used, otherwise the actual values will be used.

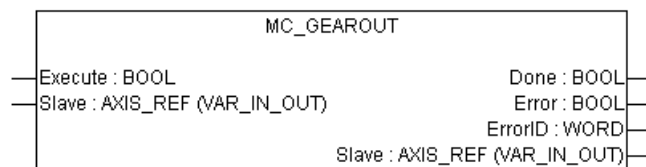
Example: Gear timing diagram





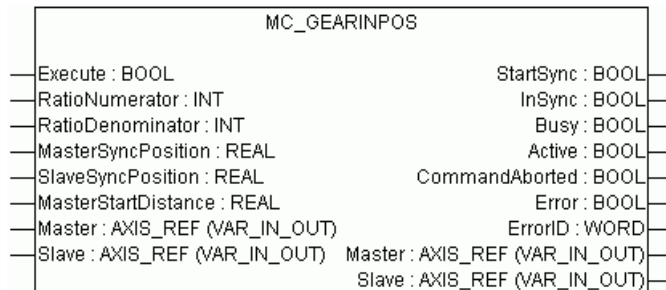
6.3.5 MC_GearOut

This program organisation unit from SM_PLCOpen.lib disengages the slave drive from the master and continues to drive the slave at the current velocity.



6.3.6 MC_GearInPos

This POU of SM_PLCopen.lib couples the slave axis to the master axis in a position and speed-synchronous manner. Via inputs positions can be defined, from which the slave axis is coupled in and is in the position specified.



Inputs/outputs (VAR_IN_OUT)

Master : AXIS_REF

Master axis

Slave : AXIS_REF

Slave axis

Inputs (VAR_IN)

Execute : BOOL (default: FALSE)

At the rising edge the POU starts the coupling process. If MasterStartDistance 0, the coupling process is started immediately and the slave axis accelerates via polynomial function. At the master position set the slave axis then is speed-synchronous and is placed in the slave position specified. If MasterStartDistance is greater 0, the slave axis waits until the master is within the coupling window and then starts with the acceleration. Depending on the slave speed before the coupling process, the synchronous speed, and the master or slave position specified, or the coupling path, the axis can also move opposite to the actual direction of rotation for a short time, as only like this the conditions specified can be maintained.

RatioNumerator: INT (Default: 1)

By this the numerator of a ration can be set.

RatioDenominator: INT (Default: 1)

By this a denominator of the ration can be defined.

MasterSyncPosition: REAL (Default: 0)

From this position the slave is coupled in completely.

SlaveSyncPosition: REAL (Default: 0)

The position of the slave when the master is in the MasterSyncPosition.

MasterStartDistance: REAL (Default: 0)

The desired coupling path, or if 0, then the accelertion process is started immediately.

Outputs (VAR_OUT)

StartSync: BOOL (Default: FALSE)

The output is TRUE until the slave axis is being coupled in. As soon as the slave is coupled in completely, StartSync becomes FALSE and InSync becomes TRUE.

InSync : BOOL (default: FALSE)

This output reports TRUE if the slave is coupled in.

Busy : BOOL (default: FALSE)

Indicates that the POU is not completed.

Active: BOOL (Default: FALSE)

Indicates that the function block controls the axis.

CommandAborted : BOOL (default: FALSE)

The motion that had started was interrupted by another function block acting on the same axis (exception: MoveSuperImposed).

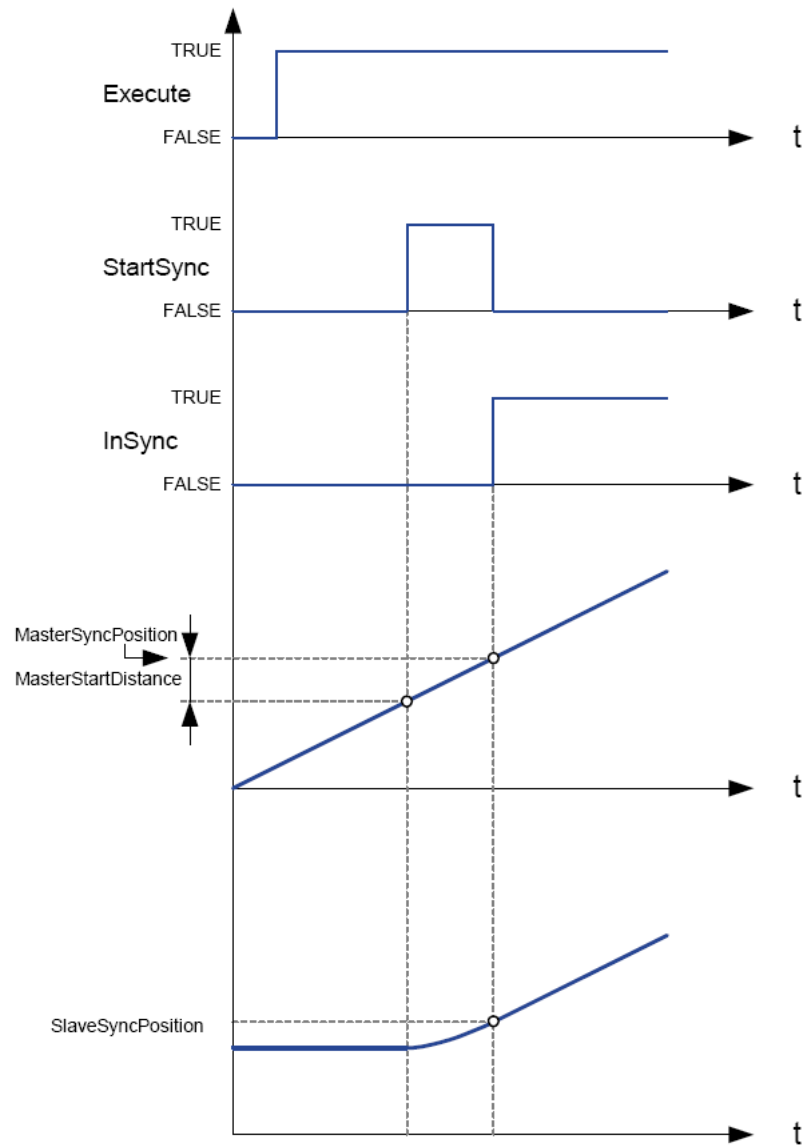
Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

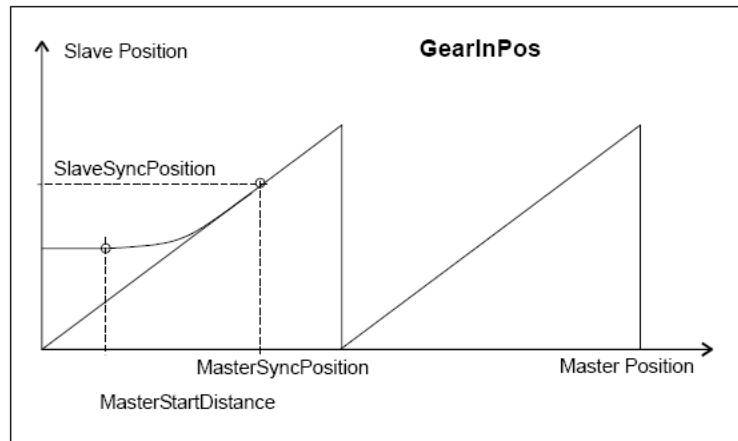
ErrorID : SMC_Error (INT)

Error number

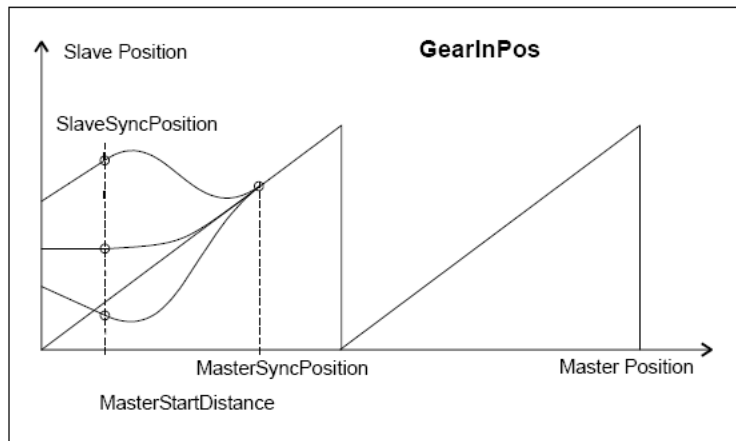
Example: Timing Diagram of MC_GearInPos



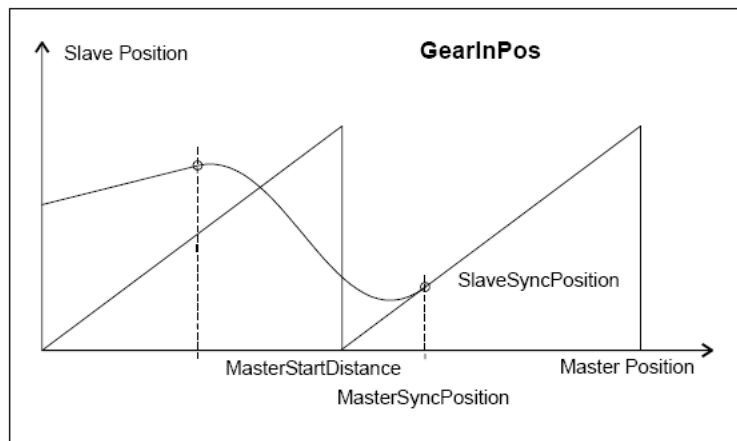
Example: Different examples of MC_GearInPos



Pic1



Pic2



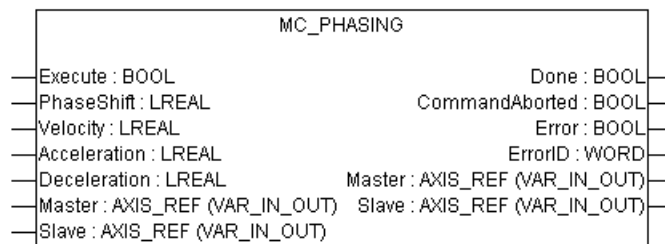
Pic3

6.3.7 MC_Phasing

This program organisation unit from SM_PLCOpen.lib maintains a constant distance between the master and slave axes. In this case, the master and slave will have the same velocity and acceleration, of course.

To achieve this, the slave axis will be accelerated or decelerated until it has the same velocity as the master axis. Once this velocity has been reached, an additional motion (similar to MC_MoveSuperImposed) will be executed on the master axis, with this motion effecting the required phase shift.

The MC_Phasing program organisation unit remains active until it is interrupted by another program organisation unit.



6.4 Additional program organisation units

6.4.1 SMC_GetCamSlaveSetPosition

This program organisation unit calculates the current set position of an axis (slave), if this axis is coupled to the motion of a different (master) axis via a cam. However, neither axis is moved or affected by this.

This program organisation unit is used if a slave axis is to be moved to the set position resulting from a cam prior to being connected to that cam.

The program organisation unit calculates the corresponding value within a cycle, so no Done output is required.

Inputs/outputs (VAR_IN_OUT)

CAMTableID : MC_CAM_ID

Cam; output of MC_CamTableSelect

Master : AXIS_REF

Master axis

Slave : AXIS_REF

Axis for which the cam set position is calculated

Inputs (VAR_IN) of the program organisation unit (any not listed correspond to those for MC_CamIn):

Enable : BOOL

Activates the program organisation unit.

Outputs (VAR_OUT)

fStartPosition : LREAL

Calculated set position for the slave

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

6.4.2 SMC_CAMBounds

This program organisation unit calculates the maximum setpoints for the position, velocity and acceleration/deceleration of the slave, if the slave is absolutely bound to a master via a specified cam and the master is moved at a defined maximum velocity and acceleration/deceleration.

This program organisation unit is particularly useful, inter alia, if a cam is created or modified online and you want to check compliance with the maximum values in advance.

Inputs (VAR_IN)

bExecute: BOOL

Activates the calculation.

dMasterVelMax : LREAL (default: 1)

Maximum velocity (absolute value) of the master [u/s]

dMasterAccMax : LREAL (default: 0)

Maximum acceleration/deceleration (absolute value) of the master [u/s²]

dMasterScaling, dSlaveScaling : LREAL (default: 1)

Master/slave scaling factor used for the cam

Inputs/outputs (VAR_IN_OUT) of the program organisation unit:

CAM: MC_CAM_REF

Cam

Outputs (VAR_OUT)

bDone : BOOL

TRUE indicates that the calculation is complete.

bError : BOOL

TRUE indicates an error in the function block.

nErrorID : SMC_Error (INT)

Error number

dMaxPos, dMinPos : LREAL

Maximum/minimum slave position value [u]

dMaxVel, dMinVel : LREAL

Maximum/minimum (may also be negative) slave velocity value [u/s]

dMaxAccDec, dMinAccDec : LREAL

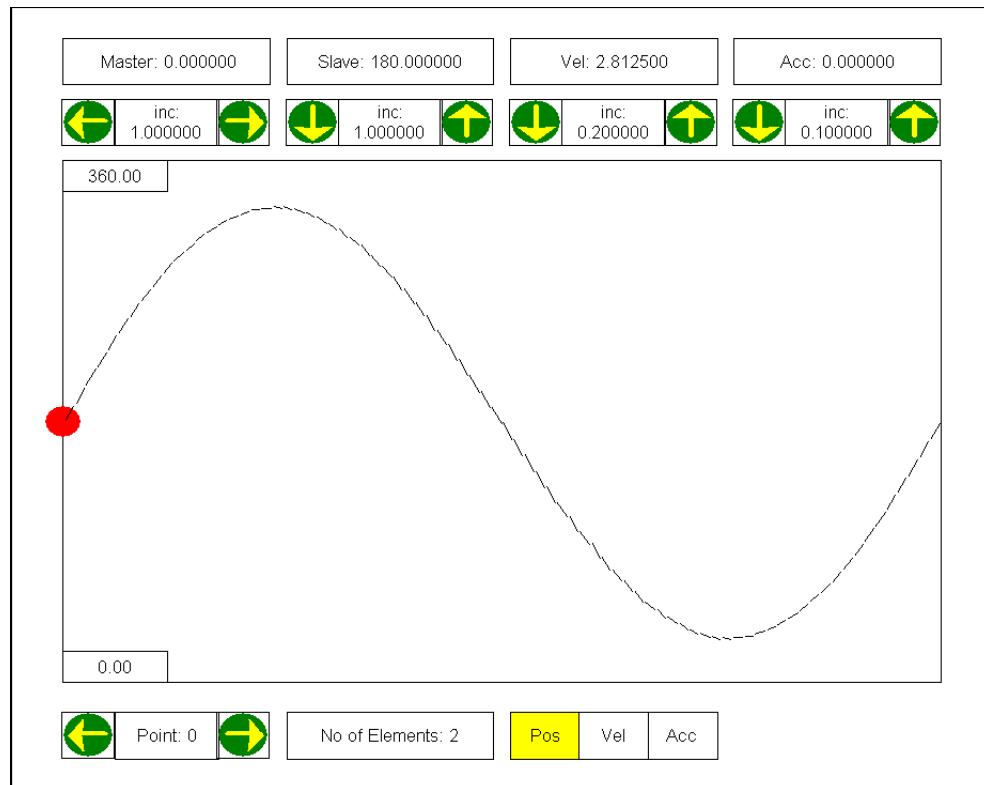
Maximum/minimum (may also be negative) slave acceleration/deceleration value [u/s²]

6.4.3 SMC_CAMEditor / SMC_CAMVisu

An online cam editor can be generated using these program organisation units.

SMC_CAMEditor must be called in the SoftMotion task, whereas SMC_CAMVisu should be called in a slower task of lower priority.

Both program organisation units should be connected to the corresponding visualisation template (SMC_CAMEditor) that represents the defined cam and allows the user to modify it, even during its runtime.



The red circle marks the current cam point, which can be changed using the arrows at the bottom left. The buttons at the bottom right can be used to select whether the position, velocity or acceleration will be displayed. The arrows at the top can be used to shift the master/slave position, slave velocity and slave acceleration by the specified increment.

Inputs/outputs (VAR_IN_OUT) of the SMC_CAMEditor program organisation unit

CAM: MC_CAM_REF

Cam to be displayed and modified

Inputs (VAR_IN) of the POU SMC_CAMEditor

Enable : BOOL (default: FALSE)

TRUE activates the program organisation unit.

dYPeriod : LREAL

Slave period (for periodic cams)

bPeriodic : BOOL (default: TRUE)

TRUE for periodic cam, otherwise FALSE

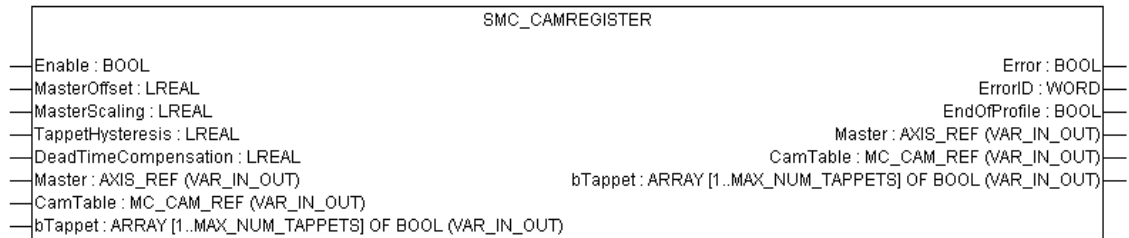
Inputs/outputs (VAR_IN_OUT) of the SMC_CAMEditor program organisation unit

ce:SMC_CAMEditor

SMC_CAMEditor instance

6.4.4 SMC_CAMRegister

This program organisation unit represents a tappet control unit. It works (like MC_CamIn) on an MC_CAM_REF structure, negating the actual cam information and only reading the tappet information.



Inputs/outputs (VAR_IN_OUT)

Master : AXIS_REF

Shows the axis structure that will switch the tappets.

CamTable : MC_CAM_REF

Description of a (possibly empty) cam containing the description of the tappets.

bTappet : ARRAY [1..MAX_NUM_TAPPETS] OF BOOL

Tappet bits

Inputs (VAR_IN)

Enable : BOOL (default: FALSE)

The program organisation unit starts to switch the tappets on TRUE.

MasterOffset : REAL

Offset to the master position

MasterScaling : REAL (default: 1)

General scaling factor for the master axis

TappetHysteresis : REAL

Hysteresis around tappets

DeadTimeCompensation : REAL

Dead time in seconds. The expected position of the master axis will be calculated in advance using linear extrapolation.

Outputs (VAR_OUT)

Error : BOOL (default: FALSE)

TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

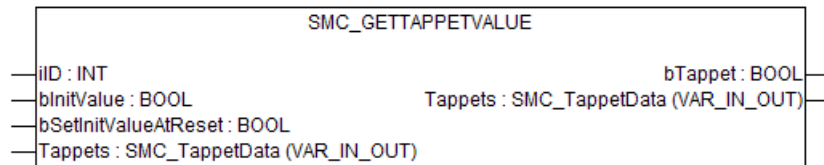
Error number

EndOfProfile : BOOL

This output will be set to TRUE on the transition from the end of the profile to the start.

6.4.5 SMC_GetTappetValue

This program organisation unit evaluates the Tappets output of the MC_CamIn program organisation unit and contains the current tappet status.



Inputs/outputs (VAR_IN_OUT)

Tappets : SMC_TappetData

Inputs (VAR_IN)

iID : INT

Group ID of the tappet to be evaluated

bInitValue : BOOL

Initial tappet value, assigned on the first call

bSetInitValueAtReset : BOOL

TRUE: When the CamIn FB is restarted, the value of the tappet is set to bInitValue.

FALSE: When the CamIn FB is restarted, the tappet value is retained.

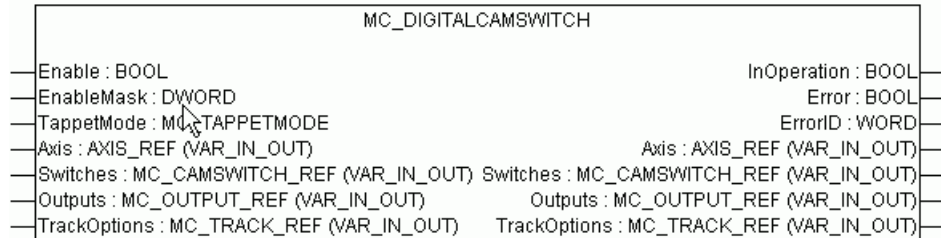
Outputs (VAR_OUT)

bTappet : BOOL (default: FALSE)

Tappet value

6.4.6 MC_DigitalCamSwitch

Like SMC_CamRegister, this POU represents a cam group. Up to 32 outputs can be switched. Movements in all directions are permissible.

**Inputs/outputs (VAR_IN_OUT)**

Axis : AXIS_REF

Shows the axis structure that will switch the tappets.

Switches: MC_CAMSWITCH_REF

Here the switching positions are defined.

The structures are as follows:

```

TYPE MC_CAMSWITCH_REF :
  STRUCT
    NoOfSwitches: BYTE;
    CamSwitchPtr: POINTER TO MC_CAMSWITCH_TR;
  END_STRUCT
END_TYPE

NoOfSwitches specifies the number of switching positions, CamSwitchPtr is a pointer to an array of the type
MC_CAMSWITCH_TR.

TYPE MC_CAMSWITCH_TR :
  STRUCT
    TrackNumber: INT;
    FirstOnPosition: LREAL;
    LastOnPosition: LREAL;
    AxisDirection: INT;
    CamSwitchMode: INT;
    Duration: TIME;
  END_STRUCT
END_TYPE

```

The TrackNumber specifies the output number. Several position ranges per output are possible.

FirstOnPosition specifies the switch-on position of the output, LastOnPosition the switch-off position.

AxisDirection 0: Output is switched in both directions, 1 only positive direction, 2 only negative direction.

CamSwitchMode : The value 0 means that the tappets are calculated in a position-dependent manner, the value 1 switches to the time-based mode. Now only the value of FirstOnPosition is used, and the output then remains on TRUE for a time set (duration).

Example:

```
Positions:
ARRAY [1...4] OF MC_CAMSWITCH_TR:=TrackNumber:=1, FirstOnPosition:=200,
LastOnPosition:=300, AxisDirection:=2, CamSwitchMode:=0),

(TrackNumber:=2, FirstOnPosition:=100, LastOnPosition:=300, CamSwitchMode:=0),
(TrackNumber:=3, FirstOnPosition:=10, CamSwitchMode:=1, Duration:=T#2000ms),
(TrackNumber:=1, FirstOnPosition:=0, LastOnPosition:=100, AxisDirection:=1);
```

- ▶ In positive direction output 1 is switched on between positions 0 and 100, and in negative direction between positions 200 and 300.
- ▶ Output 2 is switched on between 100 and 300 in all directions.
- ▶ Output 3 is switched on for 2 seconds in all directions at position 10.

Outputs: MC_OUTPUT_REF

Here the 32 tappet outputs can be determined.

```
TYPE MC_OUTPUT_REF :
  ARRAY [0...31] OF BOOL;
END_TYPE
```

TrackOptions: MC_TRACK_REF

Via the TrackOptions further settings can be carried out for each tappet output.

```
TYPE MC_TRACK_REF :
  ARRAY [0...31] OF MC_TRACK_TR;
END_TYPE
```

Each element within the array is used for the corresponding output.

```
TYPE MC_TRACK_TR :
  STRUCT
    OnCompensation: LREAL;
    OffCompensation: LREAL;
    Hysteresis: LREAL;
  END_STRUCT
END_TYPE
```

- ▶ OnCompensation: A delay (positive value) or an early switch-on process (negative value) can be set. The time is specified in seconds.
- ▶ OffCompensation: A delay (positive value) or an early switch-off process (negative value) can be set. The time is specified in seconds.
- ▶ Hysteresis: Additionally a hysteresis can be set. It avoids a permanent switching of the output, e. g. if a servo controller is placed exactly on the switching position and moves around the position a bit because of the position control.

Inputs (VAR_IN)

Enable : BOOL (default: FALSE)

The program organisation unit starts to switch the tappets on TRUE.

EnableMasks: DWORD (Default: FFFFFFFF_{hex})

Here the outputs can be switched on or off. If the bit is 1, the corresponding output is active. The lowest bit is assigned to the lowest output. The default value switches on all tappet outputs. A value of 1 would only activate the first output.

TappetMode: MC_TappetMode (Default: tp_mode_auto)

Via the TappetMode the position data of the tappets can be calculated. In the setting tp_mode_auto the set position is used if the axis is switched on. Otherwise the actual position is used for calculation. With the settings tp_mode_demandposition and tp_mode_actualposition always the set position or actual position is used.

Outputs (VAR_OUT)

InOperation: BOOL (Default: FALSE)

True indicates that the POU is active and the tappets are calculated.

Error : BOOL (default: FALSE)

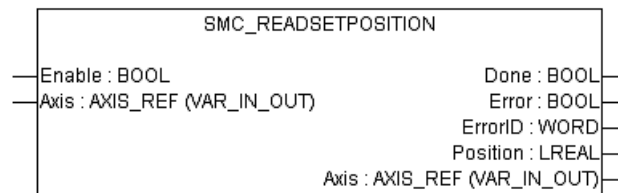
TRUE indicates an error in the function block.

ErrorID : SMC_Error (INT)

Error number

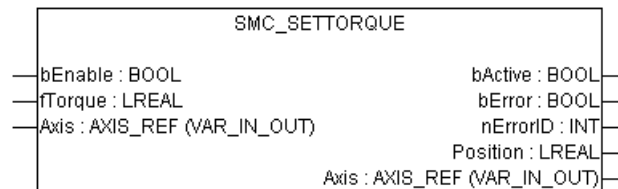
6.4.7 SMC_ReadSetPosition

This program organisation unit reads the current set position of the drive.



6.4.8 SMC_SetTorque

This program organisation unit can be used to create a target torque, provided that the drive is in the "torque" controller mode.



7 The SM_CNC.lib library

This POU library provides program organisation units and functions for IEC programs that have been realised using SoftMotion. It contains the motion functionality and is integrated in the IEC program. It includes, inter alia, the program organisation units listed below, which execute the motions designed offline in the editors and can also realise the paths planned online in the IEC program. This involves accessing internal structures that describe the individual path objects.

Path object	Information
SMC_NCDecoder	Decodes the path programmed in the CNC editor in order to obtain structure objects
SMC_ToolCorr	Path preprocessing: tool radius correction
SMC_AvoidLoop	Path preprocessing: avoids loops in the path
SMC_SmoothPath	Path preprocessing: smoothes the path using splines
SMC_RoundPath	Path preprocessing: rounds the path using circular arcs
SMC_CheckVelocities	Checks the final velocities of the segments
SMC_Interpolator	Conversion of the decoded and, if necessary, preprocessed path objects into discrete points
Auxiliary functions	For moving and rotating a created path
Global variables	Set certain internal parameters
Structures	SMC_POSINFO, SMC_GEOINFO, SMC_VECTOR3D and SMC_VECTOR6D for saving positions, path segments and vectors
Structure	SMC_OUTQUEUE for managing GEOINFO objects in a list of a defined size

The program organisation units and structures are described in the following chapters.



Note!

Please note that the CNC editor can compile a CNC program in two different ways: as a program variable (SMC_CNC_REF) that has to run through the decoder and, if applicable, path preprocessing POU's and as OUTQUEUE that can be passed on to the interpolator directly.

7.1 SMC_NCDecoder

The SMC_NCDecoder program organisation unit (SM_CNC.lib) converts a CNC program created in the CNC editor into a list of SoftMotion GEOINFO structure objects. One program line is decoded per cycle.

Input/output (VAR_IN_OUT)

ncprog: SMC_CNC_REF

The CNC program (structure SMC_CNC_REF of SM_DriveBasic.lib) is shown in this IN_OUT variable. This program may either have been generated by the IEC program or programmed in the CNC editor.

Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the decoding procedure if a rising edge is detected at this input.

bAppend: BOOL

If this input is set to FALSE, the DataOut queue will be cleared on each reset. If it is TRUE, new data will be written at the end of the DataOut queue.

bStepSuppress: BOOL

If this input is set to TRUE, lines of the CNC program starting with "/" will be ignored. If it is FALSE (default), they will be processed regardless.

piStartPosition

Position of the point to be moved at the beginning of the path. Here, the current setpoint positions of the axes should always be used. Use of the actual positions can lead to an additional revolution, especially in the case of modulo axes.

nSizeOutQueue: UDINT

Here, the POU is informed of the size of the data buffer into which the list of GEOINFO structure objects is written. The buffer must be five times bigger than a GEOINFO structure. If this is not the case, the SMC_NCDecoder does not perform any actions. It is permissible to set the value but it is only permissible to alter it subsequently during a reset. It is advisable to create the actual buffer as described below e.g. as an example: Array of SMC_GeoInfo. The corresponding buffer size then results from sizeof(Example).

pbyBufferOutQueue: POINTER TO BYTE

This input must point to the first byte of the memory area created for the OUTQUEUE structure. This memory area must be at least as large as defined in nSizeOutQueue. It is typically assigned in the declaration part of the IEC program by means of an array of SMC_GeoInfo (e.g. BUF: ARRAY[1...50] OF SMC_GEOINFO; for a buffer that can store 50 path elements). In respect of this value as well, it is permissible to set the value but it is only permissible to alter it subsequently during a reset.

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the program has been fully processed. Thereafter SMC_NCDecoder will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

If applicable, this enum output will describe an error that occurred during decoding. Following an error, processing will be stopped until a reset is performed.

poqDataOut: POINTER TO SMC_OUTQUEUE

This output points to an SMC_OUTQUEUE structure that manages decoded SMC_GEOINFO objects.

iStatus: SMC_DEC_STATUS (INT)

This enum variable shows the current state of the program organisation unit. Possible states:

WAIT_PROG	0	Program not yet found
READ_WORD	1	Word read
PROG_READ	2	End of program reached

7.2 SMC_GCodeViewer

This POU can be used in combination with the SMC_NCDecoder. It collects and stores the textual representation (G code) of the individual GeoInfo objects and outputs it again in the form of an ARRAY OF STRING, which can be output by means of a table in the visualisation, for example. Via the input, which is typically connected to the interpolator output iActObjectSourceNo, the POU receives information about which row has already been dealt with and deletes it. In this way, a display element of the current G code can be created.

Please note that the output of this POU does not correspond to the G-code that is stored in the editor or text file, as here for instance no blank lines or comments are accepted. Variables that might be used are already replaced.

The POU is called in the task context of the SMC_NCDecoder programm organisation unit.

A buffer of sufficient size has to be transmitted to it. The number of the SMC_GCodeViewer_DATA objects at least has to be as great as the sum of all SMC_GeoInfo objects which are created in the buffers of the SMC_NCDecoder POU's and the path preprocessing POU's.

Input/output (VAR_IN_OUT)

GCodeText: SMC_GCODE_TEXT

SMC_GCODE_TEXT is structured as follows:

```
TYPE SMC_GCODE_TEXT :  
STRUCT  
    str: STRING(80);  
    iLineNumber: INT;  
    bNewLine: BOOL;  
    bClearList: BOOL;  
END_STRUCT  
END_TYPE
```

Data source. Is linked to the output of the SMC_NCDecoder POU's of the same name.

Inputs (VAR_IN)

bEnable: BOOL

The POU is active if this input is set to TRUE.

iActObjectSourceNo: INT

Line number that is currently edited by the interpolator.

uiNumberOfBufferEntries: UINT

Size of the array transferred in pBuffer.

pBuffer: POINTER TO SMC_GCODEVIEWER_DATA;

Address of the array that has been reserved as buffer for the POU.

Outputs (VAR_OUT)

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

This Enum output possibly describes an error that has occurred. After an error the data preparation is stopped until the POU is started again by a new bEnable.

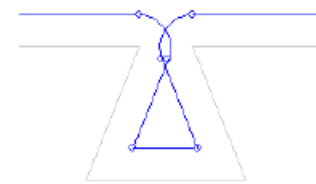
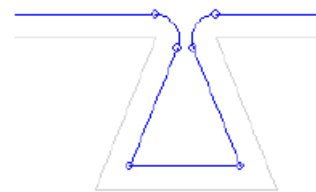
asGCode: ARRAY[0...c_uiLines] OF STRING

Text lines containing the current segment of the G-code. The first element contains the line which is currently traversed by the interpolator.

7.3 SMC_ToolCorr

The SMC_ToolCorr program organisation unit (SM_CNC.lib) can be used for path preprocessing: it creates a shifted path based on an originally defined path. In the shifted path, each point of each path object is a definable distance from its original point and its direct neighbours (tool radius correction). The shifted path thus ensures that each of its points is a fixed distance from the original path. A typical application is the milling of a programmed outline using a milling drill of a defined diameter. In order to compensate for the radius of the drill, the milling drill must then follow an appropriately shifted path - which can be created with the help of the SMC_ToolCorr program organisation unit:

The following restriction must be taken into account: If the outline and the drill radius are selected in such a way that crossover points would result within the shifted path, i.e. the desired outline would be destroyed were the shifted path traversed, this will not be taken into consideration (see sketch on the right). To prevent this, use the SMC_AvoidLoop program organisation unit.



The SMC_ToolCorr program organisation unit functions as follows:

All SMC_GEOINFO objects found in the input OUTQUEUE structure are checked one after the other. If bit 1 (bit 2) of the Intern_Mark variable is set in one of these objects, displacement of the path in the direction of travel to the left (right) around the current tool radius is started from that point. In order to ensure a continuous path, a positioning object (MoveType = 100) is inserted, or if such a positioning object precedes the object, it is directly shifted to the starting point of the moved path. Every additional object is then displaced as well as long as bit 0 of Intern_Mark has been set, which leads to the tool radius correction being aborted. Here as well, a constant continuation of the path is ensured by means of a positioning object. Once tool radius correction has been started, it must be ended before displacement in the respective other direction can be started. The SMC_NCDecoder sets these bits as a reaction to commands **G41/G42/G40**. In other words, tool radius correction is carried out for all objects that are within the commands **G41** and **G40** or **G42** and **G40**.

Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the tool radius correction if a rising edge is detected at this input.

bAppend: BOOL

If this input is set to FALSE, the DataOut queue will be cleared on each reset. If it is TRUE, new data will be written at the end of the DataOut queue.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object that contains the SMC_GEOINFO objects of the path to be displaced, typically to the DataOut output of the preceding POU (e.g. the SMC_NCDecoder).

dToolRadius: LREAL

This input variable contains the value which, when added to the relevant ToolRadius value of the SMC_GEOINFO object, gives the tool radius by which the path is to be shifted. It may be changed online, which means that you can define the value offline (by means of the SMC_GEOINFO structure) and modulate it online. Please note, however, that changing the tool radius whilst processing a POU to be shifted will cancel tool path correction and should, therefore, be avoided. The tool radius can be changed during a reset or during a phase when the program organisation unit is definitely not in the process of shifting a POU (status = TC_ORIG). Default value: 0.

nSizeOutQueue: UDINT

Here, the POU is informed of the size of the data buffer into which the list of GEOINFO structure objects is written. The buffer must be five times bigger than a GEOINFO structure. If this is not the case, the SMC_NCDecoder does not perform any actions. It is permissible to set the value but it is only permissible to alter it subsequently during a reset. It is advisable to create the actual buffer as described below e.g. as an example: Array of SMC_GeoInfo. The corresponding buffer size then results from sizeof(Example).

pbyBufferOutQueue: POINTER TO BYTE

This input must point to the first byte of the memory area created for the OUTQUEUE structure. This memory area must be at least as large as defined in nSizeOutQueue. It is typically assigned in the declaration part of the IEC program by means of an array of SMC_GeoInfo (e.g. BUF: ARRAY[1...50] OF SMC_GEOINFO; for a buffer that can store 50 path elements). In respect of this value as well, it is permissible to set the value but it is only permissible to alter it subsequently during a reset.

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the input data from DataIn has been fully processed. Thereafter the program organisation unit will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

If an error occurs, this variable will contain the error number.

poqDataOut: POINTER TO SMC_OUTQUEUE

This output points to an SMC_OUTQUEUE structure that contains the shifted path as a list of SMC_GEOINFO objects.

iStatus: SMC_TC_STATUS (INT)

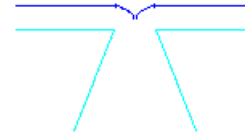
This enum variable shows the current state of the program organisation unit. Possible states:

TC_ORIG	0	No tool radius correction on the current object
TC_RIGHT	1	Shift objects to the right
TC_LEFT	2	Shift objects to the left
TC_END	4	Object processing terminated

7.4

SMC_AvoidLoop

The SMC_AvoidLoop program organisation unit (SM_CNC.lib) can be used for path preprocessing: it creates a loopless copy of a defined path. If an intersection is detected in the original path, the path will be cut at this point, the loop will be removed and the residual path will be followed. This results in a loopless, continuous path.



This POU might be used to address the problem described in the section on the SMC_ToolCorr program organisation unit, for example.

The SMC_AvoidLoop program organisation unit functions as follows:

The program organisation unit runs through all the SMC_GEOINFO objects found in the input SMC_OUTQUEUE structure. If bit 7 of the Intern_Mark variable is set in one of these objects, the "avoid loop" function will be activated. The program organisation unit will now start to check whether the current object intersects with the subsequent SMC_GEOINFO objects that come before an SMC_GEOINFO object in which bit 6 of the Intern_Mark variable is set, thus deactivating the avoid loop function. If no intersections are found, the object will be copied to the output SMC_OUTQUEUE unchanged, otherwise the first of the intersecting objects will be cut at the intersection point, the SMC_GEOINFO objects positioned between the intersecting objects will be removed and the new path will be resumed at the intersection point, starting with the second of the two intersecting objects. SMC_NCDecoder will set bit 6 and bit 7 of Intern_Mark as a response to commands **G61/G60**.

A decisive factor here is how many objects are contained in the input SMC_OUTQUEUE, i.e. how it has been dimensioned. If a loop contains more objects than can be stored in the SMC_OUTQUEUE, that particular loop will not be detected.

Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the tool radius correction if a rising edge is detected at this input.

bAppend: BOOL

If this input is set to FALSE, the DataOut queue will be cleared on each reset. If it is TRUE, new data will be written at the end of the DataOut queue.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object that contains the SMC_GEOINFO objects of the path, typically to the DataOut output of the preceding POU (e.g. the SMC_NCDecoder). It must be of a corresponding size.

nSizeOutQueue: UDINT

Here, the POU is informed of the size of the data buffer into which the list of GEOINFO structure objects is written. The buffer must be five times bigger than a GEOINFO structure. If this is not the case, the SMC_NCDecoder does not perform any actions. It is permissible to set the value but it is only permissible to alter it subsequently during a reset. It is advisable to create the actual buffer as described below e.g. as an example: Array of SMC_GeoInfo. The corresponding buffer size then results from sizeof(Example).

pbyBufferOutQueue: POINTER TO BYTE

This input must point to the first byte of the memory area created for the OUTQUEUE structure. This memory area must be at least as large as defined in nSizeOutQueue. It is typically assigned in the declaration part of the IEC program by means of an array of SMC_GeoInfo (e.g. BUF: ARRAY[1...50] OF SMC_GEOINFO; for a buffer that can store 50 path elements). In respect of this value as well, it is permissible to set the value but it is only permissible to alter it subsequently during a reset.

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the input data from DataIn has been fully processed. Thereafter the program organisation unit will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

If an error occurs, this variable will contain the error number.

poqDataOut: POINTER TO SMC_OUTQUEUE

This output points to an SMC_OUTQUEUE structure that manages the SMC_GEOINFO objects of the loopless path.

iStatus: SMC_AL_STATUS (INT)

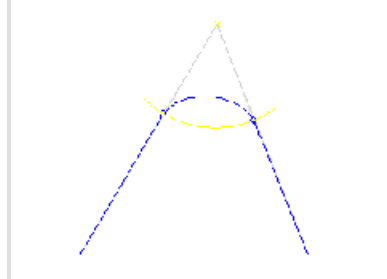
This enum variable shows the current state of the program organisation unit. Possible states:

AL_OFF	0	Avoid loop function deactivated
AL_ON	1	Avoid loop function activated
AL_END	2	Object processing terminated

7.5 SMC_SmoothPath

The SMC_SmoothPath program organisation unit (SM_CNC.lib) may be used for path preprocessing: it rounds path angles, thus creating a smooth path. This is needed for applications where the accuracy of the path is not as important as the velocity and where, therefore, angles which force the velocity to be reduced to 0 must be avoided.

For this purpose, the path will be cut at a defined distance to the angle and a spline will be inserted. The distance is specified by the SMC_GEOINFO structure object of the first object to be smoothed as well as by one of its inputs. The sum of both values is taken as the radius of a circle which has its centre point in the path angle and which will intersect with the surrounding objects.



The SMC_SmoothPath program organisation unit functions as follows:

All SMC_GEOINFO objects found in the input OUTQUEUE structure are checked one after the other. If bit 4 of the Intern_Mark variable is set in one of these objects, angle smoothing starts from that point until bit 3 of Intern_Mark is set in one of the subsequent objects. SMC_NCDecoder sets these bits as a response to commands **G51/G50**. In other words, path smoothing will be performed for all objects located within commands **G51** and **G50**.

Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the tool radius correction if a rising edge is detected at this input.

bAppend: BOOL

If this input is set to FALSE, the DataOut queue will be cleared on each reset. If it is TRUE, new data will be written at the end of the DataOut queue.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object that contains the SMC_GEOINFO objects of the unrounded path; typically it points to the DataOut output of the preceding program organisation unit (e.g. SMC_NCDecoder).

dEdgeDistance: LREAL

This input variable contains the value which, when added to the relevant ToolRadius value of the SMC_GEOINFO object, gives the distance to an angle at which the corresponding objects will be cut and replaced by a spline. It may be changed online, which means that you can define the value offline (by means of the SMC_GEOINFO structure) and modulate it online. Default value: 0.

dAngleTol: REAL

This input describes the angle tolerance value up to which a path bend should not be smoothed.

nSizeOutQueue: UDINT

This variable contains the size of the data buffer to which the list of smoothed GEOINFO structure objects will be written. This buffer must be at least five times as large as an SMC_GEOINFO structure, i.e. around 2 KB. If this is not the case, the SMC_SmoothPath program organisation unit will not be able to perform any actions. The value can be set but can only be subsequently modified during a reset.

poqBufferOutQueue: POINTER TO BYTE

This input must point to the first byte of the memory area created for the OUTQUEUE structure. This memory area must be at least as big as defined in nSizeOutQueue. It is typically assigned in the declaration part of the IEC program by means of a byte array (e.g.. BUF: ARRAY[1...10000] OF BYTE; for a 10,000 byte memory area. In this case as well, this value can be set but can only be subsequently modified during a reset.

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the input data from DataIn has been fully processed. Thereafter the program organisation unit will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

If an error occurs, this variable will contain the error number.

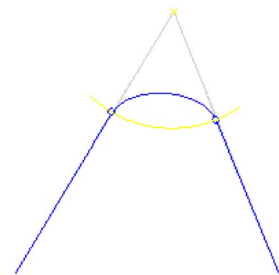
poqDataOut: POINTER TO SMC_OUTQUEUE

This output points to an SMC_OUTQUEUE structure that manages rounded SMC_GEOINFO objects.

7.6 SMC_RoundPath

The SMC_RoundPath program organisation unit (SM_CNC.lib) is very similar to SMC_SmoothPath. It rounds angles that occur at the junction of two lines by means of circular arcs.

For this purpose the path will be cut at a distance "r" to the angle and a spline will be inserted. The distance is specified by the SMC_GEOINFO structure object of the first object to be rounded as well as by the dRadius input. The value of dRadius is dominant, i.e. only if it is 0 will the value of the object be taken into account. If the set value is greater than half the length of one of the two SMC_GEOINFO objects, the half length will be used instead.



The SMC_RoundPath program organisation unit functions as follows:

All SMC_GEOINFO objects found in the input SMC_OUTQUEUE structure are checked one after the other. If bit 5 of the Intern_Mark variable is set in one of these objects, angle smoothing starts from that point until bit 3 of Intern_Mark is set in one of the subsequent objects. SMC_NCDecoder sets these bits as a response to commands **G52** and **G50**. In other words, angle smoothing will be performed for all objects located within commands **G52** and **G50**.

Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the tool radius correction if a rising edge is detected at this input.

bAppend: BOOL

If this input is set to FALSE, the DataOut queue will be cleared on each reset. If it is TRUE, new data will be written at the end of the DataOut queue.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object that contains the SMC_GEOINFO objects of the unrounded path; typically it points to the DataOut output of the preceding program organisation unit (e.g. SMC_NCDecoder).

dRadius: LREAL

This input variable contains the value that gives the distance to an angle at which the corresponding objects will be cut and replaced by a circular arc. It may be changed online, which means that you can adapt this value to the path conditions. Default value: 0.

dAngleTol: REAL

This input describes the angle tolerance value up to which a path bend should not be smoothed.

nSizeOutQueue: UDINT

This variable contains the size of the data buffer to which the list of smoothed GEOINFO structure objects will be written. This buffer must be at least five times as large as an SMC_GEOINFO structure. If this is not the case, the ISMC_NCDdecoder does not perform any actions. It is advisable to create the actual buffer as described below, e.g. as an example : Array of SMC_GeoInfo. The corresponding buffer size then results from sizeof(Example)

pbyBufferOutQueue: POINTER TO BYTE

This input must point to the first byte of the memory area created for the OUTQUEUE structure. This memory area must be at least as large as defined in nSizeOutQueue. It is typically assigned in the declaration part of the IEC program by means of an array of SMC_GeoInfo (e.g. BUF: ARRAY[1...50] OF SMC_GEOINFO; for a buffer that can store 50 path elements). In respect of this value as well, it is permissible to set the value but it is only permissible to alter it subsequently during a reset.

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the input data from DataIn has been fully processed. Thereafter the program organisation unit will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ ERROR (INT)

If an error occurs, this variable will contain the error number.

poqDataOut: POINTER TO SMC_OUTQUEUE

This output points to an SMC_OUTQUEUE structure that manages rounded SMC_GEOINFO objects.

7.7 SMC_CheckVelocities

This POU checks the speeds of the individual path segments. It must always be called directly before the interpolator if the OutQueue has been created not by the Editor but in the IEC program (e.g. by the SMC_NCDecoder).

This function's main task is to check the path for bends and to reduce the velocity to zero at such bends.

Inputs (VAR_IN)

bExecute: BOOL

The check will start on a rising edge.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object that contains the SMC_GEOINFO objects of the path, typically to the poqDataOut output of the preceding POU (e.g. SMC_NCDecoder/SMC_SmoothPath).

dAngleTol: REAL

This input describes the angle tolerance value up to which no stop should be executed at a path bend.

Outputs (VAR_OUT)

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

If applicable, this enum output will describe an error.

poqDataOut: POINTER TO SMC_OUTQUEUE

This output points to the SMC_OUTQUEUE structure object that contains the path with the permissible velocity values and will now be fed to the interpolator directly.

7.8 SMC_LimitCircularVelocities

This program organisation unit checks the individual elements of the OutQueue and limits the path velocities of circular elements as a function of their radii.

When moving at a constant velocity (v) across a transition from a line to a circle with a radius " r ", the path acceleration will jump from 0 to the value [], in accordance with the particular amount. In order to limit this acceleration jump to the value a , the velocity of the circular arc at the transition must not exceed [] accordingly.

This program organisation unit checks the transitions of two elements (line to circular arc, circular arc to line and circular arc to circular arc) and adapts the final velocity of the first element so that the acceleration jump does not exceed the value $dMaxAccJump$.

Furthermore, the program organisation unit limits the path acceleration on circles to the value $dMaxAcc$, by reducing the path velocity of the circle accordingly.

Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the tool radius correction if a rising edge is detected at this input.

bAppend: BOOL

If this input is set to FALSE, the DataOut queue will be cleared on each reset. If it is TRUE, new data will be written at the end of the DataOut queue.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object that contains the SMC_GEOINFO objects of the path to be changed, typically to the DataOut output of the preceding POU (e.g. the SMC_NCDecoder).

dMaxAcc: LREAL

This input variable contains the maximum permissible acceleration value for circular arcs. A value of 0 means that this check will not be performed.

dMaxAccJump: LREAL

This input variable contains the maximum acceleration jump (a) for a transition between two objects. A value of 0 means that this check will not be performed.

nSizeOutQueue: UDINT

Here, the POU is informed of the size of the data buffer into which the list of GEOINFO structure objects is written. The buffer must be five times bigger than a GEOINFO structure. If this is not the case, the SMC_NCDecoder does not perform any actions. It is permissible to set the value but it is only permissible to alter it subsequently during a reset. It is advisable to create the actual buffer as described below e.g. as an example: Array of SMC_GeoInfo. The corresponding buffer size then results from `sizeof(Example)`.

pbyBufferOutQueue: POINTER TO BYTE

This input must point to the first byte of the memory area created for the OUTQUEUE structure. This memory area must be at least as large as defined in nSizeOutQueue. It is typically assigned in the declaration part of the IEC program by means of an array of SMC_GeoInfo (e.g. BUF: ARRAY[1...50] OF SMC_GEOINFO; for a buffer that can store 50 path elements). In respect of this value as well, it is permissible to set the value but it is only permissible to alter it subsequently during a reset.

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the input data from DataIn has been fully processed. Thereafter the program organisation unit will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ ERROR (INT)

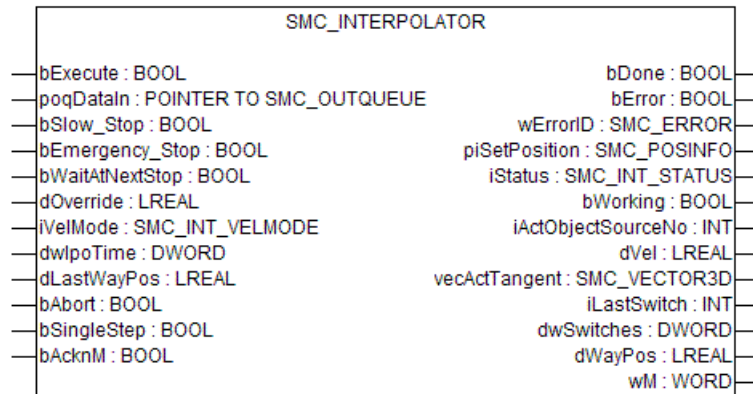
If an error occurs, this variable will contain the error number.

poqDataOut: POINTER TO SMC_OUTQUEUE

This output points to an SMC_OUTQUEUE structure that manages new SMC_GEOINFO objects.

7.9 SMC_Interpolator

The SMC_Interpolator program organisation unit (SM_CNC.lib) is used to convert an existing continuous path described by GEOINFO objects into discrete path position points, taking the defined velocity profile and time pattern into account. These position points will then typically be transformed by the IEC program (into drive axis positions, for example) and sent to the drives via the Drive Interface.



Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the interpolation if a rising edge is detected at this input.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object that contains the SMC_GEOINFO objects of the path to be interpolated; typically it points to the poqDataOut output of the preceding program organisation unit SMC_CheckVelocities.

bSlow_Stop: BOOL

If this input has been set to FALSE (default), the path is travelled on without interruption. TRUE causes the SMC_Interpolator to reduce the speed to 0 in accordance with the set speed profile (byVelMode s. u.) and the maximum deceleration of the current GEOINFO object (dDecel s.u.) and to wait until Slow_Stop is set to FALSE again.

bEmergency_Stop: BOOL

This input is set to FALSE by default. If it becomes TRUE, SMC_Interpolator will execute an immediate stop, i.e. the current position will be retained. The velocity will thus be immediately set to 0.

bWaitAtNextStop: BOOL

If this input is set to FALSE (default), the path will be traversed without stopping. On TRUE, SMC_Interpolator will retain the position at the next regular stop, i.e. at position points where the velocity is 0, typically at path angles, and will trigger a delay until bWaitAtNextStop is reset to FALSE.

dOverride: LREAL

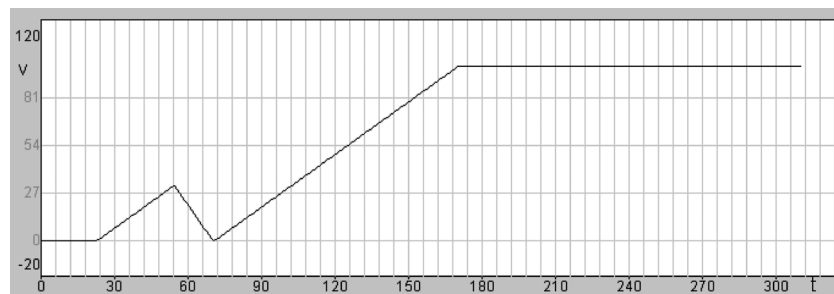
With the help of this variable, it is possible to make use of the override. Values are greater than 0.01 are permissible. The override is multiplied in relation to the set speed of the individual objects and thus make it possible to increase or reduce the set speed. The effect of an override of 1 (default), for example, is that the programmed set speeds are executed whereas an override of 2 would double them.

Although the override can be modified at any time, any change will only be applied if no acceleration or deceleration is currently in progress.

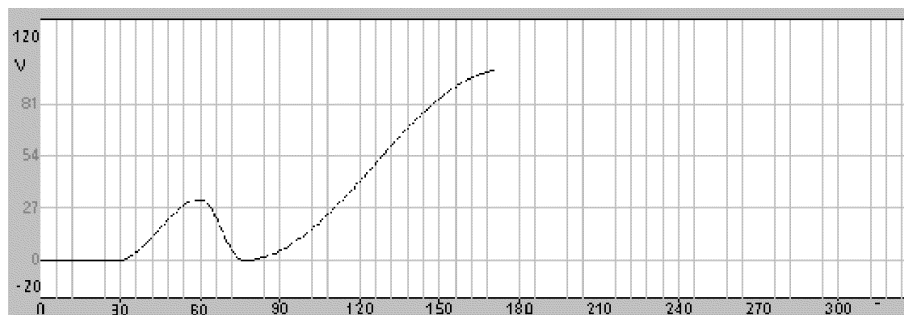
iVelMode: SMC_INT_VELMODE

This input defines the velocity profile. The value "TRAPEZOID" (default) produces a trapezoidal velocity profile, whilst "SIGMOID" produces an S-shaped, or sigmoidal, profile.

Example of a trapezoidal velocity profile (Vel_Mode = 0):



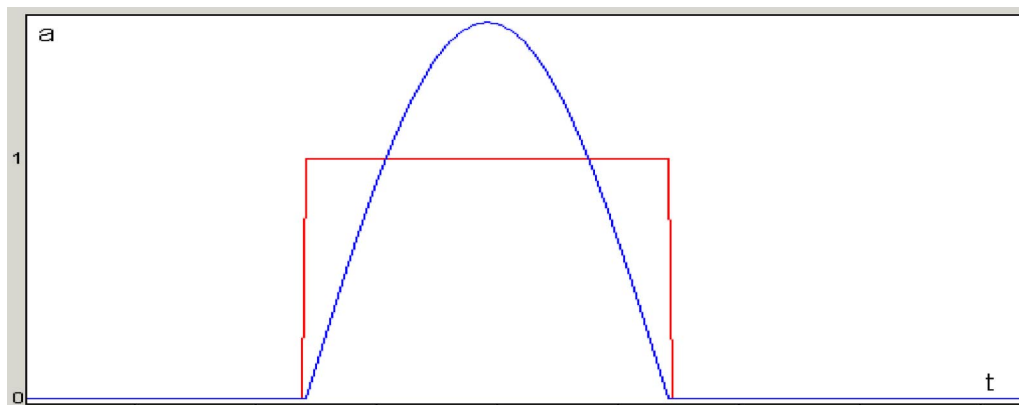
Example of a sigmoidal velocity profile (Vel_Mode = 1):



In the examples shown above, the maximum acceleration (Accel) value is lower than the maximum deceleration (Decel). This causes the different gradients of the velocity curve at acceleration and deceleration.

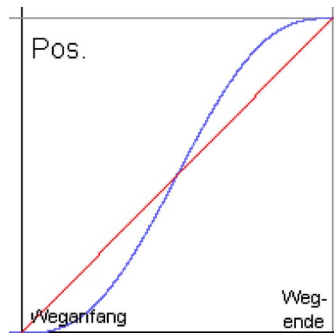
The advantage of a sigmoidal velocity profile is that, in contrast to the trapezoidal version, the associated acceleration is continuous, thereby relieving strain, especially for heavy machines. The trade off for this is a slightly increased calculating time.

Because the S-shaped speed profile (blue) is such that a changeover to the trapezoidal profile (red) does not result in any change of the time needed to complete travelling on the path, the careful increasing of acceleration at the beginning and end must be balanced out by higher acceleration in the middle. Here, it must be ensured that the maximum acceleration or braking programmed in the Geo objects is not exceeded by more than a maximum factor of $p/2$:



As in the case of the override, an online change is not accepted until an acceleration or braking phase has been concluded.

In order to interpolate the additional axes in sigmoidal form (blue, see sketch) rather than linearly (red), the corresponding bits must be set in the wSProfile variable of piStartPos for the current path object, irrespective of the byVelMode input of SMC_Interpolator. Now the additional axis will not be interpolated linearly relative to the path length in the X, Y, Z space, but as a polynomial function of this path length, resulting in a sigmoidal profile for the axis position, which has a velocity and acceleration of zero at the start and end of a path segment.



dwlpoTime: DWORD

This input, which must be set for each call, contains the cycle time in μsec .

dLastWayPos: LREAL

This input allows the user to measure the length of the path that is covered by the interpolator. The dWayPos output is the sum of dLastWayPos and the distance covered in the current cycle. If dLastWayPos = 0, dWayPos shows the length of the current path segment. If dLastWayPos is set to the same value as the dWayPos output, dWayPos will always be incremented by the current path segment and the total length of the path covered is obtained. During this process, dLastWayPos can be (re)set to 0 or another value at any time.

bAbort: BOOL

This input aborts processing of an outline.

bSingleStep: BOOL

This input causes the interpolator to stop at each transition between two path objects (including at transitions between identical tangents) for the duration of one cycle. If bSingleStep is set to TRUE during the traversing procedure, the interpolator will stop at the end of the next object it can reach without exceeding the defined deceleration value.

If the interpolator is to stop at the next possible stop position (i.e. at points where the velocity is 0), bWaitAtNextStop must be used.

bAcknM: BOOL

This input can be used to acknowledge a pending additional option (M option). If the input is TRUE, this option will be cleared and path processing resumed.

bQuick_Stop: BOOL

If this input is TRUE, according to the speed profile set (byVelMode) and the deceleration which is defined by the maximum of dQuickDeceleration and the deceleration programmed in the path, the interpolator decreases the speed to 0 until bQuick_Stop is reset to FALSE. Default: FALSE

dQuickDeceleration: LREAL

Value for the deceleration, which is used for bQuick_Stop.

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the input data from DataIn has been fully processed. Thereafter the program organisation unit will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

If applicable, this enum output will describe an error that occurred during interpolation. Following an error, processing will be stopped until a reset is performed.

piSetPosition: SMC_POSINFO

Set_Position contains the set position calculated in accordance with the specifications made. Set_Position is an SMC_POSINFO structure and not only contains the Cartesian coordinates of the path position point to be approached, but also the position of the additional axes.

iStatus: SMC_INT_STATUS (INT)

This enum variable shows the current state of the program organisation unit. Possible states:

IPO_UNKNOWN	0	Internal state. State may not occur after a complete SMC_Interpolator pass.
IPO_INIT	1	Initialisation state; DataIn is not and has not yet been full.
IPO_ACCEL	2	Program organisation unit is currently accelerating.
IPO_CONSTANT	3	Program organisation unit is currently moving at a constant velocity.
IPO_DECEL	4	Program organisation unit is currently decelerating.
IPO_FINISHED	5	Processing of the GEOINFO list is finished. Any other GEOINFO objects that arrive in DataIn subsequently will not be processed.
IPO_WAIT	6	Program organisation unit is waiting because one of the situations below has occurred: Emergency_Stop = TRUE Slow_Stop = TRUE and Vel = 0 Wait_At_Next_Stop = TRUE and Vel = 0

bWorking: BOOL

This output assumes the value TRUE once list processing has begun and before it has been completed (IPO_ACCEL, IPO_CONSTANT, IPO_DECEL or IPO_WAIT). Otherwise Working is FALSE.

iActObjectSourceNo: INT

The SourceLine_Nr value of the GEOINFO object in the DataIn queue that is currently being traversed can be found here. If SMC_Interpolator is not (or no longer) working (Working = FALSE), -1 appears here.

dVel: LREAL

This variable contains the current velocity that results if an object moves from the previous space coordinate to Set_Position within the time set in Ipo_Time.

vecActTangent: SMC_VECTOR3D

This structure contains the direction of the path specified in the Set_Position. If Vel = 0, Act_Tangent is also filled with zeros.

iLastSwitch: INT

This variable contains the number of the most recently passed switch. If several switches are passed in one cycle, only the last one will be shown.

dwSwitches: DWORD

This DWORD contains the current switching state of all switches from 1 to 32. Bit 0 of the DWORD represents auxiliary mark 1, bit 31 for auxiliary mark 32. In contrast to iLastHelpMark, like this can be ensured that no auxiliary marks are left out.

dWayPos: LREAL

See the information about the dLastWayPos input for a description.

SMC_Interpolator will calculate and output the subsequent path position point on each call, taking the specified parameters, the velocity history and the last path position into account. Once the first GEOINFO object has been processed, it will be cleared from the poqDataIn SMC_OUTQUEUE structure.

wM: WORD

If the interpolator passes an M object, i.e. a line describing an additional option, this output will be set to the corresponding value and the program organisation unit will wait until it is acknowledged by the bAcknM input.



Note!

The SMC_OUTQUEUE variable is empty at the end of an outline. If you want to traverse the same outline again, you will either have to convert the CNC program into an SMC_OUTQUEUE structure via the decoder and path preprocessing POU's, or use the SMC_RESTOREQUEUE function. The latter is only possible if the OUTQUEUE buffer is large enough to hold the entire outline.

7.10 SMC_GetMParameters

If the interpolator is currently on an M option, this program organisation unit can be used to query the parameters that have been set for this M option (K, L, O).

Inputs (VAR_IN)

bEnable: BOOL

Program organisation unit is active if this input is set.

Inputs/outputs (VAR_IN_OUT) of the program organisation unit:

Interpolator: SMC_Interpolator

Interpolator instant

Outputs (VAR_OUT)

bMActive: BOOL (default: FALSE)

TRUE if an M option is currently pending.

dK, dL: LREAL (default: 0)

M parameters specified via words K and L

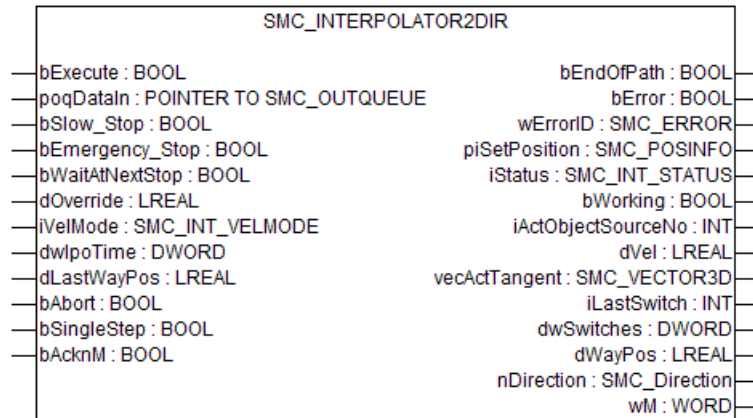
MParameters: SMC_M_PARAMETERS

M parameter that has been specified by means of the global data structure gSMC_MParameters or by means of the variable transferred by O.

7.11 SMC_Interpolator2Dir

In terms of its function and how its inputs and outputs are assigned, this program organisation unit is the same as the SMC_Interpolator function POU. The only difference is that SMC_Interpolator2Dir can also interpolate a path in reverse.

For this purpose, a negative value is assigned to the dOverride input, whereupon SMC_Interpolator2Dir interpolates in negative direction. The analogue speed input of a handwheel can be at this input so that the user can travel forwards and backwards at any speed.



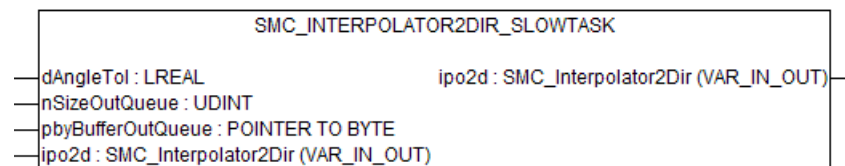
Additional inputs and outputs (VAR_IN_OUT)

nDirection: SMC_Direction

This program organisation unit shows the direction in which it is currently moving. Possible values: IPO_positive, IPO_negative and IPO_standstill.

The following preconditions must be met in order for this program organisation unit to be used:

- The entire path must fit in poqDataIn. As the program organisation unit must move forwards and backwards alternately, the complete path must be available in the memory.
- An additional program organisation unit, SMC_Interpolator2Dir_SlowTask, is called:



This program organisation unit is responsible for generating the reverse path. It has been split off from SMC_Interpolator2Dir, and thus can be swapped to a lower-priority task in the case of lower-performance automation systems that are being stretched to the limit.

Inputs (VAR_IN)

dAngleTol: LREAL

Angle tolerance for the reverse path. Typically identical to the angle tolerance of the original path.

nSizeOutQueue: UDINT, pbyBufferOutQueue: POINTER TO BYTE

Size of and pointer to data buffer in which the reverse path is to be stored. This buffer must be large enough to hold the complete path.

lpo2d: SMC_Interpolator2Dir

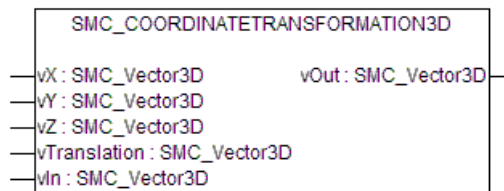
SMC_Interpolator2Dir instance for which the reverse path is to be generated

7.12 Function blocks for spatial transformation processes

Please also see the chapters "[The SM_Trafo.lib library](#)" (216) and "[Spatial transformations](#)" (238). SM_CNC.lib provides the following modules for spatial transformation:

7.12.1 SMC_CoordinateTransformation3D

This program organisation unit calculates the coordinates of a position (available in the old coordinate system) with reference to the new coordinate system. For this purpose the coordinate transformation for the new coordinate system is defined in relation to the old one on the basis of the translation vector and the new unit vectors.



Inputs (VAR_IN)

vX, vY, vZ: SMC_Vector3D

Unit vectors of the new coordinate system with reference to the old coordinate system.

vTranslation: SMC_Vector3D

Translation vector: Vector from the old coordinate origin to the new one with reference to the old coordinate system

vIn: SMC_Vector3D

Position to be transformed

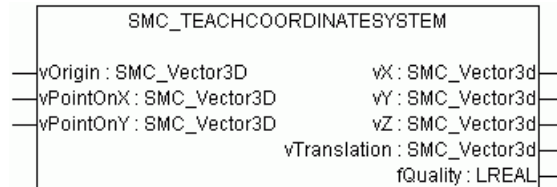
Outputs (VAR_OUT)

vOut: SMC_Vector3D

Transformed position

7.12.2 SMC_TeachCoordinateTransformation

This POU of SM_Trafo.lib helps the user to calculate the vectors of a new coordinate system which he wants to "teach". As inputs the source of the new coordinate system and a random point are transferred to the x and y axis, respectively. As outputs the POU provides the translation vector and the unit vectors of the new coordinate system. Furthermore it calculates the quality of the input parameters (angle between x and y axis should be 90°).



Inputs (VAR_IN)

vOrigin: SMC_Vector3D

Source position of the new coordinate system.

vPointOnX, vPointOnY: SMC_Vector3D

Outputs (VAR_OUT)

Position of a point on the X or Y axis.

vX, vY, vZ: SMC_Vector3D

Unit vectors of the new coordinate system.

vTranslation: SMC_Vector3D

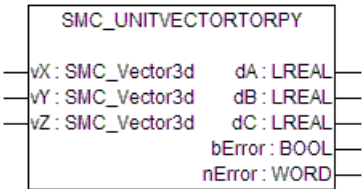
Translation vector: Vector from the old coordinate source to the new one with reference to the old coordinate system.

fQuality: LREAL

Quality of the input parameters; value from 0 to 1, where 1 = optimum (vX, vY are orthogonal), and 0 = vX and vY are invalid

7.12.3 SMC_UnitVectorToRPY

Program organisation unit for calculating the RPY angle (with reference to the old coordinate system) from the unit vectors of the new coordinate system



Inputs (VAR_IN)

vX, vY, vZ: SMC_Vector3D

Unit vectors of the new coordinate system with reference to the old coordinate system.

Outputs (VAR_OUT)

dA, dB, dC: LREAL

RPY angle in radian measure

bError : BOOL

Invalid input values

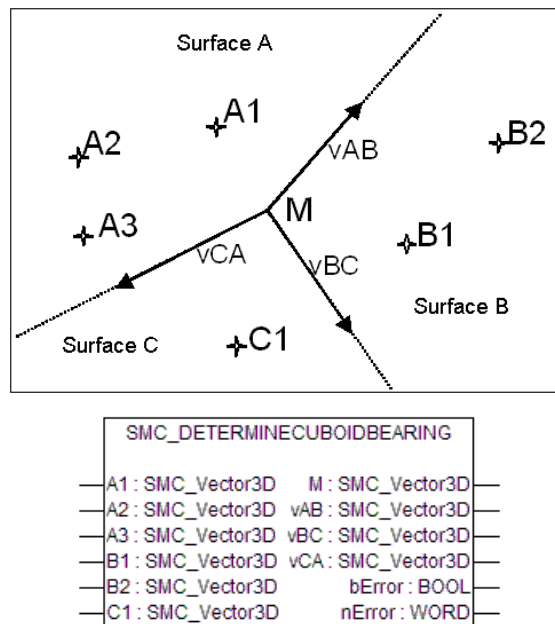
nError: WORD

Error description

Value	Information
0	No error (bError = FALSE)
1	Vectors do not have length 1.
2	Vectors are not perpendicular to one another.
3	No right-hand system

7.12.4 SMC_DetermineCuboidBearing

Program organisation unit for determining the position of a cuboid (corner point, edge alignment) in space by specifying 6 (3/2/1) points:



Inputs (VAR_IN)

A1, A2, A3: SMC_Vector3D

Three points on a lateral surface A of the cuboid, which must not lie on a straight line

B1, B2: SMC_Vector3D

Two points on different lateral surfaces of the cuboid, the projections of which onto surface A must not be identical

C1: SMC_Vector3D

Points on another lateral surface of the cuboid

Outputs (VAR_OUT)

M: SMC_Vector3D

Corner point of the cuboid

vAB, vBC, vCA: SMC_Vector3D

Unit vectors on the edge lines of the cuboid

bError : BOOL

Invalid input values

nError: WORD

Error description

Value	Information
0	No error (bError = FALSE)
1	A1, A2, A3 lie on a straight line.
2	Projections of B1 and B2 onto surface A are identical.

7.13 Extension of the interpolation process from 2.5-D to real 3-D

The motion libraries contained in the PLC motion target system from version 7.01.xx onwards enable the user to traverse three-dimensional path objects.

The main differences compare to the 2.5-dimensional implementation are as follows:

- ▶ In the 3D mode, the material speed and acceleration refer to the motion in the X/Y/Z coordinates, whereas up to now they have referred to the X/Y coordinates (with the exception of the mere Z motion).
- ▶ In the 2.5D case, the motions in the additional axes (including Z), are always proportional to the path covered in X/Y. In the 3D case this continues to apply to the additional axes, but the X, Y and Z dimensions have an equal status. Thus, all supported motion profiles can also be executed in Z direction.

Advantages for the user

- ▶ Now, circles can be programmed in any position in the space, the limitation to the X/Y main plane is omitted.
- ▶ By defining a plane or a normal vector to this plane, the system automatically positions the circular arc or helix into the space so that start and final point are connected; the circular component is always vertical to the defined normal direction.
- ▶ The corners are now rounded three-dimensionally in the space as well. This serves to reduce the processing times of an NC program considerably.
 - SMC_RoundPath: Rounding of two adjacent lines by 3D circular arc (see page 180).
 - SMC_SmoothPath: Smoothing of two adjacent objects by 3D spline (see page 178).
 - Extension: The inserted spline contour elements are now calculated via a 5th order function. In contrast to the splines used in 3rd order, the corners are run through smoother. Jumps during acceleration are avoided.

Switching on/off the 3D function

The 3D function is switched on via the plane selection:

G16 I J K	Plane with normal vector I J K
G17	X/Y plane
G18	Z/X plane
G19	Y/Z plane

The 3D function is switched off via G15.

7.14 Auxiliary functions and function blocks for path rotations, translations and scalings

The function blocks SMC_RotateQueue2D and SMC_TranslateQueue3D contained in SM-CNC lib rotate or shift the path stored in an SMC_OutQueue.

The path to be rotated or shifted is handed over in the form of a pointer to the SMC_OUTQUEUE structure object by means of the DataIn input variable.

The bEnable input which has been initialised with FALSE prevents the path from being rotated or shifted until it is set to TRUE. All GEOINFO elements are then processed by poqDataIn. As soon as bEnable becomes FALSE, the blocks do not carry out any more changes.

The bReset input, which has also been initialised with FALSE, causes the GEOINFO objects currently located in poqDataIn not to be processed but only causes those arriving from now onwards to be rotated or shifted.

7.14.1 SMC_ROTATEQUEUE2D

The path contained in poqDataIn will be rotated around the Z axis by the angle in degrees given in dPhi. A positive angle effects a positive rotation in the mathematical sense (counterclockwise).

7.14.2 SMC_TRANSLATEQUEUE3D

The path contained in poqDataIn is shifted by the vector transferred in vec of the structure type SMC_VECTOR3D (see [SMC_VECTOR3D](#) (□ 206)).

7.14.3 SMC_SCALEQUEUE3D

The path contained in poqDataIn will be stretched by the fScaleFaktor factor.

As a sudden modification of the characteristic variables of the rotation/translation (dPhi, vec) during processing might cause a jump in the path, any changes to the inputs of these variables will not be taken into account until the SMC_OUTQUEUE for poqDataIn is empty or a bReset is carried out.

In order to effect a rotation in the (XY) plane around a point other than (00), i.e. to reach point (XpYp), use the following sequence: a translation according to the vector (-Xa-Yp°0), a rotation about the desired angle dPhi and another translation according to the vector (XyYp°0).

7.15 Settings via global variables

Certain internal variables and constants are defined in "SoftMotion_CNC_Globals" (see **Resources** under the SM_CNC.lib library entry). Some of these variables and constants can be changed:

Examination for zero (change epsilon values for zero):

- ▶ g_fSMC_CNC_EPS (epsilon value for accurate examination for zero)
- ▶ g_fSMC_CNC_EPS_RELUCTANT (epsilon value for tolerant examination for zero)

7.16 Structures of SM_CNC.lib

The following section describes a selection of SM_CNC.lib structures, which are used by the SM_CNC.lib program organisation units for saving positions, path segments (path objects) and vectors: SMC_POSINFO, SMC_GEOINFO, SMC_VECTOR3D, SMC_VECTOR6D. Furthermore, the SMC_OUTQUEUE structure helps you to manage GEOINFO objects in a list of a defined size.

7.16.1 SMC_POSINFO

The SMC_POSINFO structure contained in SMC_CNC.lib stores the coordinates and position of the additional axes for a particular position point.

```

TYPE SMC_POSINFO:
STRUCT
  iFrame_Nr:INT;
  wAuxData:WORD;
  wSProfile:WORD;
  dX:LREAL;
  dY:LREAL;
  dZ:LREAL;
  A:LREAL;
  B:LREAL;
  dC:LREAL;
  dA1:LREAL;
  dA2:LREAL;
  dA3:LREAL;
  dA4:LREAL;
  dA5:LREAL;
  dA6:LREAL;
END_STRUCT
END_TYPE

```

Variables dX, dY and dZ contain the position in the coordinate system, dA1 to dA6 the position of the additional axes. The user can store further information that is not relevant to SoftMotion program organisation units in iFrame_Nr. dA, dB and dC are not currently used.

wAuxData describes bit by bit which of the position axes are to be calculated by SMC_Interpolator. wAuxData will be initialised with 111_{bin}, i.e. the X, Y and Z axis will be interpolated. If the first bit is set, the dX position will be calculated, bit 7 causes dA2 to be processed, for example.

In the same way, wSProfile describes whether the additional axes (all of them apart from the X and Y axis) are to be interpolated linearly (FALSE) or sigmoidally (TRUE) by the interpolator. Bit 2 represents the Z axis, bit 6 P, bit 7 Q, bit 8 U, bit 9 V, and bit 10 W.

7.16.2 SMC_GEOINFO

The SMC_GEOINFO structures in SM_CNC.lib each store a path object. A path object is a segment of the programmed path which, due to its geometric form, can be stored in the following structure in its entirety:

```

TYPE SMC_GEOINFO:
STRUCT
  iObjNo:INT;
  iSourceLineNo:INT;
  iMoveType:MOVTYPE;
  piStartPos:SMC_POSINFO;
  piDestPos:SMC_POSINFO;
  dP1:LREAL;
  dP2:LREAL;
  dP3:LREAL;
  dP4:LREAL;
  dP5:LREAL;
  dP6:LREAL;
  dP7:LREAL;
  dP8:LREAL;
  dT1:LREAL;
  dT2:LREAL;
  dToolRadius:LREAL;
  dVel:LREAL;
  dVelEnd:LREAL;
  dAccel:LREAL;
  dDecel:LREAL;
  dLength:LREAL;
  byInternMark:BYTE;
  dHelpPos: ARRAY[0..MAX_IPOSWITCHES] OF LREAL;
  iHelpID: ARRAY[0..MAX_IPOSWITCHES] OF INT;
END_STRUCT
END_TYPE

```

iObjNo: INT

Any object number can be stored in this integer value. It has no relevance for the actual path description.

iSourceLineNo: INT

This integer value typically shows the source code line number of the CNC program. It has no relevance for the actual path description.

iMoveType: MOV_TYP (INT)

The MOV_TYP enum type contains the following permissible values and describes the object type:

LIN	1	Straight motion (G01)
CLW	2	Clockwise circle (G02)
CCLW	3	Counterclockwise circle (G03)
SPLINE	5	Spline, parabola (G05, G06)
ELLCLW	8	Clockwise ellipse (G08)
ELLCCLW	9	Counterclockwise ellipse (G09)
LINPOS	100	Straight positioning (G00)
INITPOS	110	Blind positioning (start point not yet known; will be continuously added to by SMC_Interpolator)
MCOMMAND	120	Additional option, M option

piStartPos: SMC_POSINFO

SMC_POSINFO structure that describes the exact start position (will be ignored if Move_Type = INITPOS).

piDestPos: SMC_POSINFO

SMC_POSINFO structure that contains the exact end position.

dP1, ..., dP8: LREAL

These variables store further information describing the path, depending on the Move_Type:

LIN LINPOS	Not relevant, as all information is already contained in Start_Pos and Dest_Pos.
CLW CCLW	P1: X coordinate of the circle centre P2: Y coordinate of the circle centre P3: Circle radius
SPLINE	Spline parameter
ELLCLW ELLCCLW	P1: X coordinate of the circle centre P2: Y coordinate of the circle centre P3: X component of the main axis 1 vector P4: Y component of the main axis 1 vector P5: Length of the main axis P6: Length of the sub-axis P7: Direction of the main axis (in degrees) P8: Ratio P6/P5
INITPOS	Not relevant

dT1, dT2: LREAL

These variables contain the start and end positions of the parameter. Depending on the Move_Type, this means:

LIN	Not relevant, as all information is already contained in Start_Pos and Dest_Pos.
CLW ELLCLW	- T1: Start angle in mathematical sense (degrees): (0 = east, 90 = north, 180 = west, 360 = south) - T2: Opening angle of the circle, arc length in degrees (e.g.: 90 = quarter circle, 180 = half circle)
CCLW ELLCCW	T1: Start angle in mathematical sense (degrees): (0 = east, 90 = north, 180 = west, 360 = south) T2: Negative opening angle of the circle (e.g.: -90 = quarter circle, -180 = half circle)
SPLINE	Start and end value of parameter t (see description of a spline). Default: 0 and 1.
INITPOS	Not relevant

dToolRadius: LREAL

Within this variable the information required for the path preprocessing is saved (see [SMC_ToolCorr](#) (□ 173)). The entry has no relevance if corresponding path preprocessing blocks (SMC_ToolCorr, SMC_RoundPath) are not passed through.

dVel, dVelEnd: LREAL

These two variables, which must always be assigned, contain information relating to the velocity profile in this object. dVel describes the setpoint velocity to be achieved, dVelEnd describes the velocity that must be reached at the end of the object (see [SMC_Interpolator](#) (□ 185)), both in path units/s.

dAccel, dDecel: LREAL

The maximum permissible acceleration and deceleration in path units/s² are stored in dAccel and dDecel. Both variables are pre-assigned a value of 100.

dLength: LREAL

This variable, which must always be assigned, contains the length of the object in path units.

byIntern_Mark: BYTE

The start and end of path preprocessing actions are stored here as follows:

Bit 0 set	Stop tool radius correction after this object
Bit 1 set	Start tool radius correction to the left of this object
Bit 2 set	Start tool radius correction to the right of this object
Bit 3 set	Stop path rounding/smoothing after this object
Bit 4 set	Start path smoothing at this object
Bit 4 set	Start path rounding at this object
Bit 6 set	Stop avoiding loops after this object
Bit 7 set	Start avoiding loops at this object

dHelpPos: ARRAY[0...MAX_IPOSWITCHES] OF LREAL,

iHelpID: ARRAY[0...MAX_IPOSWITCHES] OF INT:

These variables contain the relative position (0: object start, 1: object end; similar to G code: O) and the ID (see G code: H) of the auxiliary switches.

If the current object is an MCOMMAND, iHelpID[0] will contain the number of the M option.

7.16.3 SMC_VECTOR3D

The SMC_VECTOR3D structure in SMC_CNC.lib will store a three-dimensional vector:

```
TYPE SMC_VECTOR3D:
STRUCT
  dX:LREAL;
  dY:LREAL;
  dZ:LREAL;
END_STRUCT
END_TYPE
```

7.16.4 SMC_VECTOR6D

The SMC_VECTOR6D structure in SMC_CNC.lib will store a six-dimensional vector:

```
TYPE SMC_VECTOR6D:
STRUCT
  dX:LREAL;
  dY:LREAL;
  dZ:LREAL;
  dA:LREAL;
  dB:LREAL;
  dC:LREAL;
END_STRUCT
END_TYPE
```

7.16.5 SMC_OUTQUEUE

The SMC_OUTQUEUE structure in SM_CNC.lib helps you to manage GEOINFO objects in a list of a defined size.

```

TYPE SMC_OUTQUEUE :
STRUCT
  wOUTQUEUEStructID: WORD;
  pbyBuffer: POINTER TO BYTE;
  nSize: UDINT;
  nReadPos: UDINT;
  nWritePos: UDINT;
  bFull: BOOL;
  bEndOfList: BOOL;
  byGenerator: BYTE;
END_STRUCT
END_TYPE

```

The program organisation units use the wOUTQUEUEStructID variable, which has a fixed value, to check internally whether the structure variable provided is of type SMC_OutQueue or not.

The byGenerator variable describes the originator of the queue. The interpolator uses this information to check whether the SMC_CheckVelocities program organisation unit has been processed last as prescribed. The following values are defined:

Originator	Value
SMC_NCDecoder	1
SMC_AvoidLoop	10
SMC_LimitCircularVelocity	11
SMC_RoundPath	12
SMC_SmoothPath	13
SMC_ToolCorr	14
SMC_RotateQueue2D	30
SMC_ScaleQueue3D	31
SMC_TranslateQueue3D	32
SMC_CheckVelocities	254
CNC editor	255

The SoftMotion library provides the following functions for handling an SMC_OUTQUEUE structure object:

BOOL SMC_RESTOREQUEUE(Enable: BOOL, POQ: POINTER TO SMC_OUTQUEUE)

This function restores an already interpolated structure or a structure processed elsewhere. This is only possible if the list at POQ can contain the complete path.

POINTER TO SMC_OUTQUEUE SMC_APPENDOBJ(POQ: POINTER TO SMC_OUTQUEUE, PGI: POINTER TO SMC_GEOINFO)

This function appends the GEOINFO object present at PGI to the end of the list at POQ, as long as OQ has been initialised correctly and has not yet been completely assigned. If successful, the function returns a pointer to the new list element, otherwise it returns 0.

BOOL SMC_DELETEOBJ(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

This function deletes the nth object from the list at POQ, whereby counting is started at 0. If n - 1 is greater than the number of stored GEOINFO objects, nothing happens and the return value is FALSE, otherwise it is TRUE.

UINT SMC_GETCOUNT(POQ: POINTER TO SMC_OUTQUEUE)

This function returns the number of objects saved at POQ.

POINTER TO SMC_GEOINFO GETOBJ(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Provided POQ has been correctly initialised and an nth object exists, this function supplies a pointer to the nth GEOINFO object from the list at POQ.

POINTER TO SMC_GEOINFO GETOBJFROMEND(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Provided POQ has been correctly initialised and at least n + 1 elements exist, this function supplies a pointer to the nth (calculated from the end) GEOINFO object from the list at POQ; for n = 0, therefore, the last list element is returned.

Structure initialisation

The SMC_NCDecoder, SMC_SmoothPath, SMC_RoundPath, SMC_AvoidLoop and SMC_ToolCorr SoftMotion program organisation units, which provide a pointer to an internally managed OutQueue structure, will automatically initialise the structure. The SMC_SmoothPath, SMC_RoundPath, SMC_ToolCorr, SMC_AvoidLoop and SMC_Interpolator program organisation units need a pointer to a correct OutQueue list in the form of an input. If this list is programmed and filled "manually", correct initialisation also needs to be performed manually. The first two variables (buffer, size) must be set. It is also strongly recommended that the above functions are used when working with an SMC_OUTQUEUE and that, if at all possible, the other variables are not modified following initialisation.

Structure components

pbyBuffer: POINTER TO BYTE

This variable contains the address of a coherent memory area, which is allocated for saving GEOINFO objects. This memory must be assigned in the IEC program and its address then written to this variable. The assignment in the declaration part can be performed by means of a byte array, for example (BUF: ARRAY[1..10000] OF BYTE; for a 10,000 byte memory area).

nSize: UDINT

The size of the memory area reserved for the buffer must appear here.

nReadPos: UDINT

This variable contains the address of the first list element relative to the first byte of the reserved memory area.

nWritePos: UDINT

This variable contains the address of the first free byte following the list relative to the first byte of the reserved memory area.

bFull: BOOL

If the list is full, with space only left for three further GEOINFO structures (safety buffer), this variable will be set to TRUE by the APPENDOBJ function. DELETEOBJ will reset it to FALSE if elements are deleted from the list.

bEndOfList: BOOL

The SoftMotion program organisation units that have an OutQueue structure as an input always wait until the queue is completely full before processing it, in order to avoid a data underrun (SMC_Interpolator), for example. As the OutQueue will no longer be full when it comes to the last SMC_GEOINFO objects of a path, this variable must be set to TRUE once the last SMC_GEOINFO object has been stored, in order to keep processing going. If the list then becomes empty and needs to be filled again, EndOfList must be reset to FALSE.

7.16.6 SMC_CNC_REF

Parsed G code files are managed in this data structure:

```
TYPE SMC_CNC_REF :  
STRUCT  
  wCNCREFStructID: WORD := 16#BA56;  
  nElements: UDINT;  
  diReadPos: UDINT := 0;  
  udiBuffer: UDINT := 16#FFFFFFFF;  
  pgc: POINTER TO SMC_GCODE_WORD := 0;  
  piStartPosition: SMC_POSINFO;  
  strProgramName: STRING := '';  
  bRestart: BOOL;  
END_STRUCT  
END_TYPE
```

The program organisation units use the wCNCREFStructID variable, which has a fixed value, to check internally whether the structure variable provided is of type SMC_CNC_REF or not.

The pgc variable points to the first SMC_GCODE_WORD. nElements contains the number of SMC_GCODE_WORD structures at pgc.

The start position of the CNC program is stored in piStartPosition, its name is stored in strProgramName.

The diReadPos and udiBuffer variables are used internally.

The bRestart variable is set by the SMC_NCDecoder program organisation unit if jumps are used in the CNC program (G20). It tells the originator of the data structure that all pointers should be reinitialised and the data structure regenerated.

7.16.7 SMC_GCODE_WORD

G code words are stored in this data structure:

```

TYPE SMC_GCODE_WORD :
STRUCT
  byLetter:BYTE:=0;
  fValue:LREAL:=0;
  diValue:DINT:=0;
  pAdr:POINTER TO BYTE:=0;
  byVarType:BYTE:=0;
END_STRUCT
END_TYPE

```

byLetter contains the ASCII code of the letter of the word, fValue and diValue contain the value of this letter as a floating point and integer number. If a variable is used instead of a fixed value, pADR contains a pointer to this variable and byVarType contains the variable's type:

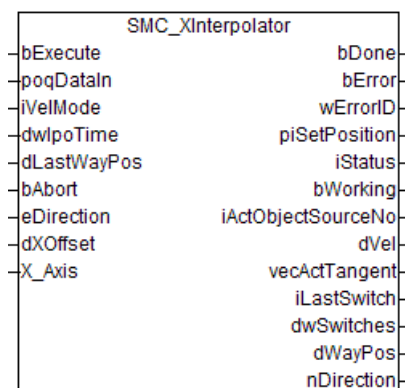
Value	Variable type
1	INT
2	BYTE
3	WORD
4	DINT
5	DWORD
6	REAL
14	SINT
15	USINT
16	UINT
17	UDINT
22	LREAL

7.17 Path cams with SMC_XInterpolator

The SMC_XInterpolator POU implements a mixture of cam and CNC. Imagine you want to cut a certain shape (described by a G code) from a workpiece, whereby the workpiece - from another process - is to be moved (e.g. along the X axis) and the other axes (Y, Z, etc.) are to be controlled according to the current position of the workpiece (X) and the stipulation from the path contour.

The workpiece motion always follows the X direction (other cases can be mapped by means of a rotation).

The SMC_XInterpolator program organisation unit has the following inputs and outputs:



Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit executes a reset and starts the interpolation if a rising edge is detected at this input.

poqDataIn: POINTER TO SMC_OUTQUEUE

This input points to the SMC_OUTQUEUE structure object, which contains the SMC_GEOINFO objects of the path to be interpolated.

dLastWayPos: LREAL

This input allows the user to measure the length of the path that is covered by the interpolator. The dWayPos output is the sum of dLastWayPos and the distance covered in the current cycle. If dLastWayPos = 0, dWayPos shows the length of the current path segment. If dLastWayPos is set to the same value as the dWayPos output, dWayPos will always be incremented by the current path segment and the total length of the path covered is obtained. During this process, dLastWayPos can be (re)set to 0 or another value at any time.

bAbort: BOOL

This input aborts processing of an outline.

eDirection: MC_Direction

This input specifies whether the workpiece will be moved along the X axis in a positive (positive) or negative (negative) direction. No other values are permitted.

dXOffset: LREAL

Offset to the position of the X axis

X_Axis: AXIS_REF

X axis, position of the workpiece

Outputs (VAR_OUT)

bDone : BOOL

This variable is set to TRUE as soon as the input data from DataIn has been fully processed. Thereafter the program organisation unit will not execute any more actions until a reset has been performed. If the bExecute input is FALSE, bDone will also be reset to FALSE.

bError : BOOL

This value will be set to TRUE if an error occurs.

wErrorID: SMC_ERROR (INT)

If applicable, this enum output will describe an error that occurred during interpolation. Following an error, processing will be stopped until a reset is performed.

piSetPosition: SMC_POSINFO

Set_Position contains the set position calculated in accordance with the specifications made. Set_Position is an SMC_POSINFO structure and not only contains the Cartesian coordinates of the path position point to be approached, but also the position of the additional axes.

iStatus: SMC_INT_STATUS (INT)

This enum variable shows the current state of the program organisation unit. Possible states:

State	Value	Information
IPO_UNKNOWN	0	Internal state. State may not occur after a complete SMC_Interpolator pass.
IPO_ACCEL	2	Program organisation unit is currently accelerating.
IPO_CONSTANT	3	Program organisation unit is currently moving at a constant velocity.
IPO_DECEL	4	Program organisation unit is currently decelerating.
IPO_FINISHED	5	Processing of the GEOINFO list is finished. Any other GEOINFO objects that arrive in DataIn subsequently will not be processed.

bWorking: BOOL

This output assumes the value TRUE once list processing has begun and before it has been completed (IPO_ACCEL, IPO_CONSTANT, IPO_DECEL or IPO_WAIT). Otherwise Working is FALSE.

iActObjectSourceNo: INT

The SourceLine_Nr value of the GEOINFO object in the DataIn queue that is currently being traversed can be found here. If SMC_Interpolator is not (or no longer) working (Working = FALSE), -1 appears here.

dVel: LREAL

This variable contains the current velocity that results if an object moves from the previous space coordinate to Set_Position within the time set in lpo_Time.

vecActTangent: SMC_VECTOR3D

This structure contains the direction of the path specified in the Set_Position. If Vel = 0, Act_Tangent is also filled with zeros.

iLastSwitch: INT

This variable contains the number of the most recently passed switch. If several switches are passed in one cycle, only the last one will be shown.

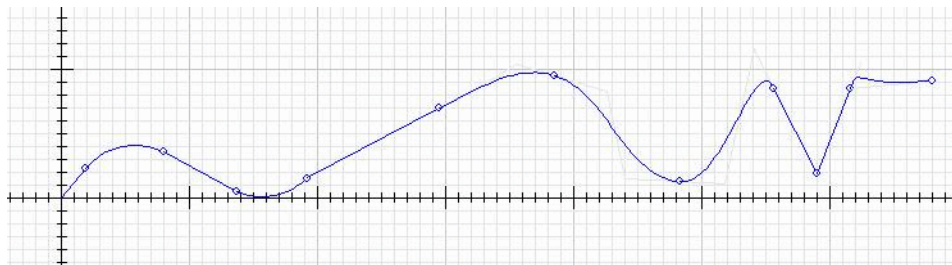
dwSwitches: DWORD

This DWORD contains the current switching state of all switches from 1 to 32. Bit 0 of the DWORD represents auxiliary mark 1, bit 31 for auxiliary mark 32. In contrast to iLastHelpMark, like this can be ensured that no auxiliary marks are left out.

dWayPos: LREAL

See the information about the dLastWAYPos input for a description.

When active, the XInterpolator looks for the position (on the stipulated path) that matches the current X position and then outputs the corresponding point on the path in piSetPosition. In order to enable this to take place smoothly, a clear path position must always exist for an X position. The following path, for example, would be permissible:



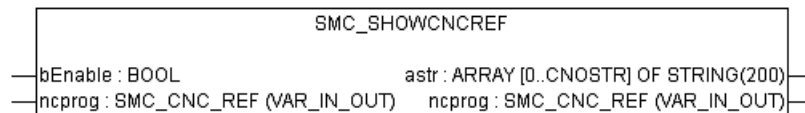
8 The SM_CNCDiagnostic.lib library

This library contains tools that can be very useful during the commissioning phase as they help the application programmer to display data that is exchanged between program organisation units.

8.1 Function blocks for analysing SMC_CNC_REF data

8.1.1 SMC_ShowCNCREF

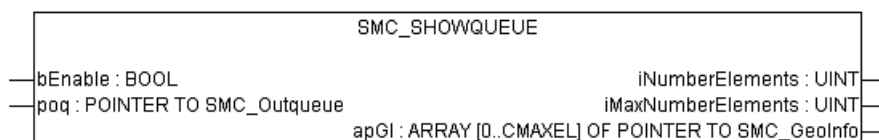
This function block can display the first ten lines of a CNC program, which is available in the form of an SMC_CNC_REF data structure, as text strings (DIN 66025). An array of string (cnostr) containing text lines is created as an output. The VISU_SMC_ShowCNCRef visualisation template can display these outputs.



8.1.2 Function blocks for analysing SMC_OutQueue data

8.1.3 SMC_ShowQueue

This function block provides the first ten SMC_GeoInfo objects of an OutQueue in the form of an ARRAY OF POINTER TO SMC_GeoInfo. Some important elements thereof can then be displayed by the VISU_SMC_ShowQueue visualisation template, including the object number, line number, object type, start position (X/Y/Z), end position (X/Y/Z), target velocity and end velocity.



9 The SM_Trafo.lib library

This POU library is an extension to SM_CNC.lib and provides program organisation units that help the IEC programmer to transform geo coordinates into machine coordinates and to control the axes.

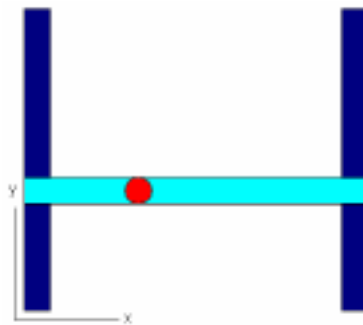
The library contains some program organisation units that control the drives with setpoints whilst also being able to monitor the setpoints and detect jumps.

It contains other program organisation units for the mathematical forwards and reverse transformation of some common kinematics. Instances of the forwards transformation POUs can also be connected to the visualisation templates provided in the library, thus enabling immediate and simple visualisation.

9.1 Transformation function blocks

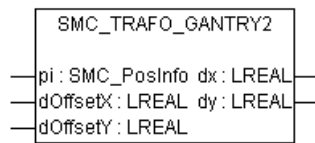
The program organisation units that refer to specific kinematics belong together in pairs, where the program organisation unit starting SMC_TRAFO_<Kinematics> performs a reverse calculation and SMC_TRAFOF_<Kinematics> a forwards calculation. Each instance of an SMC_TRAFOF_<Kinematics> program organisation unit can be connected to a visualisation template named SMC_VISU_<Kinematics>.

9.1.1 Gantry systems



As no transformations have to be performed for gantry systems, the program organisation units simply add offsets to the x, y and z axes.

9.1.1.1 SMC_TRAFO_Gantry2



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Output of the interpolator

dOffsetX, dOffsetY: LREAL

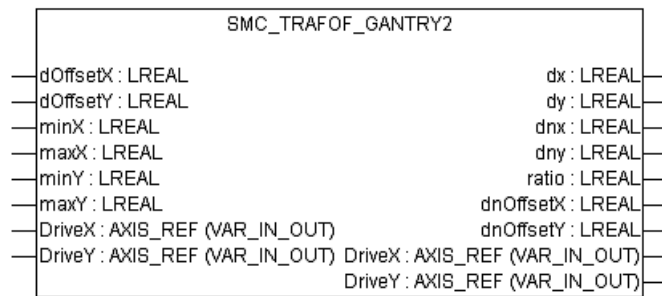
Offset for x and y axis

Outputs (VAR_OUT)

dx, dy: LREAL

Setpoints for x and y axis

9.1.1.2 SMC_TRAFOF_Gantry2



Inputs/outputs (VAR_IN_OUT)

DriveX, DriveY: AXIS_REF

x, y axis

Inputs (VAR_IN)

dOffsetX, dOffsetY: LREAL

Offset for x and y axis. Same values as described for SMC_TRAFO_Gantry2

minX, maxX, minY, maxY: LREAL

Motion range (for visualisation)

Outputs (VAR_OUT)

dx, dy: LREAL

x, y position in geo coordinates

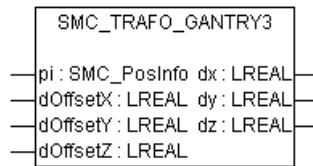
dnx, dny, dnOffsetX, dnOffsetY: LREAL

Standardised x and y position [0...1] and offsets (for visualisation).

ratio: LREAL

Ratio of x interval and y interval (for visualisation)

9.1.1.3 SMC_TRAFO_Gantry3



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Output of the interpolator

dOffsetX, dOffsetY, dOffsetZ: LREAL

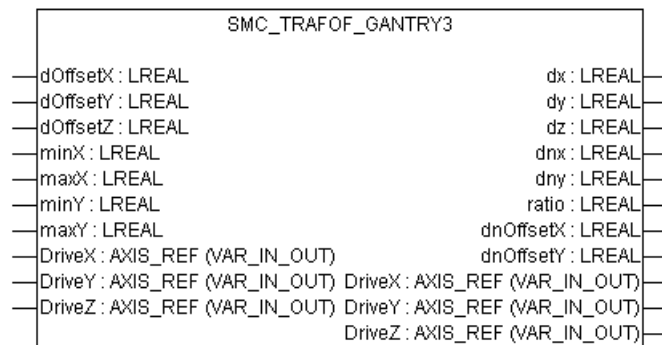
Offset for x, y and z axis

Outputs (VAR_OUT)

dx, dy, dz: LREAL

Setpoints for x, y and z axis

9.1.1.4 SMC_TRAFOF_Gantry3



Inputs/outputs (VAR_IN_OUT)

DriveX, DriveY, DriveZ: AXIS_REF

x, y, z axis

Inputs (VAR_IN)

dOffsetX, dOffsetY, dOffsetZ: LREAL

Offset for x, y and z axis. Same values as described for SMC_TRAFOF_Gantry3

minX, maxX, minY, maxY: LREAL

Motion range (for visualisation)

Outputs (VAR_OUT)

dx, dy, dz: LREAL

x, y, z position in geo coordinates

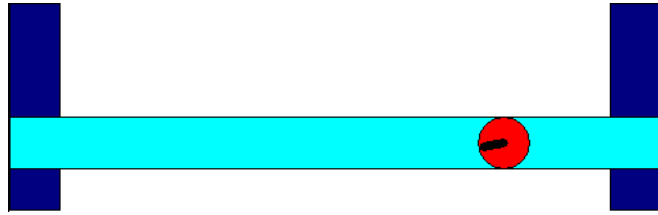
dnx, dny, dnOffsetX, dnOffsetY: LREAL

Standardised x and y position [0..1] and offset (for visualisation).

ratio: LREAL

Ratio of x interval and y interval (for visualisation)

9.1.1.5 GantryCutter



The `SMC_TRAFO<n>_Gantry<n>` program organisation units also exist as `SMC_TRAFO<n>_GantryCutter<n>`. These POU's contain transformations for gantry systems with one rotating axis that is controlled in such a way that it points along the current path tangent. They contain the rotating axis (`DriveR`), which must be created as a rotary axis with period 360, an offset (`dOffsetX`) and the direction of rotation (`iDirectionR`) as additional inputs. The reverse transformation POU also requires the vector of the current path tangent (`v`), which is an output of the interpolator.

9.1.1.6 SMC_TRAFOV_Gantry

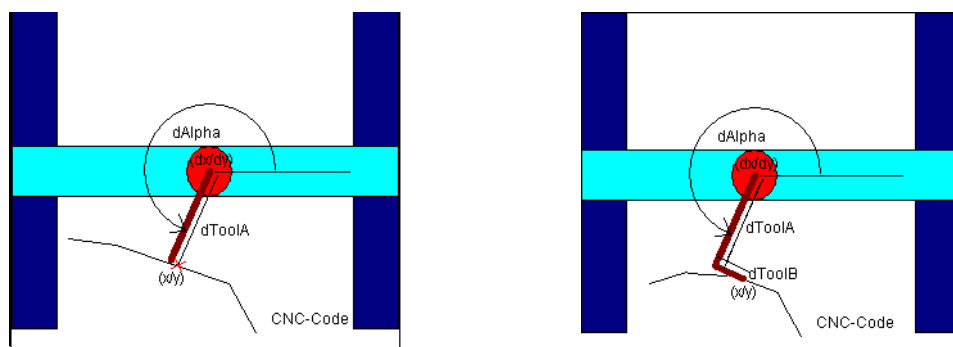
Some of the reverse transformations described above are available in a version whereby the path velocity and direction are also used as control variables for the axes. These start "`SMC_TRAFOV_`" instead of "`SMC_TRAFO_`" and they need the path tangent (`v`) and velocity (`dVel`) from the interpolator as additional inputs. They show the target velocities (`dvx/dvy/dvz`) as well as the set positions. The advantage of this method is that the following error in the drive can be minimised by means of velocity feedforward control, provided that the drive supports this mode. Each axis should, therefore, be controlled by means of the `SMC_ControlAxisByPosVel` program organisation unit.

9.1.2 Gantry systems with tool offset

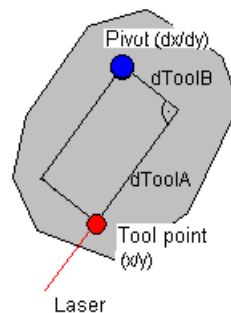
There are machine gantries whose xyz position does not match the tool's point of action, as this is not positioned axially on the z axis, but is offset. If the z axis cannot be rotated, this will be a constant xy offset, which can be fed as such into the standard gantry transformation.

However, if a rotary axis that rotates around z is involved, this will not be a constant offset; in this case, the offset will depend on the position of the C axis.

You must determine whether the tool is to be approached as a line (if the vector between the tool's point of action and axis and the desired tool orientation match -> SMC_TRAFO_Gantry2Tool1) or as a parallelogram or a right-angled triangle (SMC_TRAFO_Gantry2Tool2):

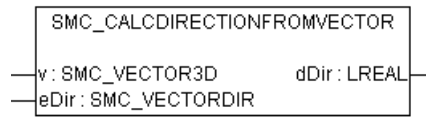


In this example the tool cannot be approached as a line, but must be approached as a right-angled triangle:



In principle, the path can be appropriately modulated by means of SMC_ToolCorr instead of via a one-dimensional transformation with tool offset (tool can be approached as a line). The difference between the two methods is the velocity of the tool point. If SMC_ToolCorr is used, the velocity of the rotation point is controlled according to the specifications made in the CNC program (F, E) and the velocity of the tool point may vary. However, if this transformation is used, the velocity of the tool point is determined by the CNC program.

The following function can be used to calculate the tool orientation (dAlpha):

9.1.2.1 SMC_CalcDirectionFromVector**Inputs (VAR_IN)**

v: SMC_VECTOR3D

The input vector v is typically the vecActTangent interpolator output.

eDir: SMC_VECTORDIR

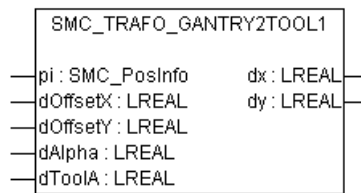
Input eDir specifies whether the direction is to be calculated at a tangent to the path (SMC_tangential), opposite (SMC_opp_tangential) or orthogonally to the path (SMC_orthogonal_r (to the right of the path tangent) or SMC_orthogonal_l (to the left of the path tangent)).

Outputs (VAR_OUT)

dDir: LREAL

Output dDir is in angular degrees and remains constant for phases in which the interpolator is at a standstill (v is zero vector). eDir is mostly used as a setpoint (SMC_ControlAxisByPos) for the directional axis and as input dAlpha for the transformation.

9.1.2.2 SMC_TRAFO_Gantry2Tool1



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Output of the interpolator

dOffsetX, dOffsetY: LREAL

Offset for x and y axis

dAlpha: LREAL

Orientation of the tool in angular degrees

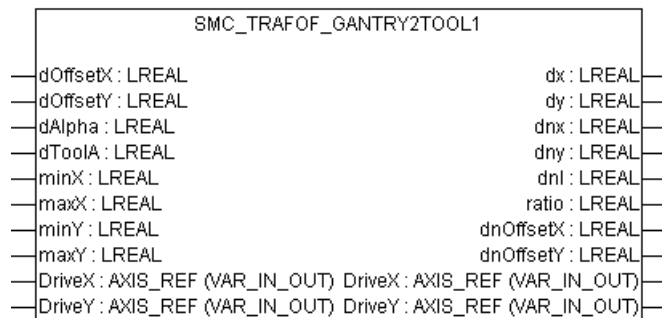
dToolA: LREAL

Length of the tool; distance between the rotation point and tool point

Outputs (VAR_OUT)

dx, dy: LREAL

Setpoints for x and y axis

9.1.2.3 SMC_TRAFOF_Gantry2Tool1**Inputs/outputs (VAR_IN_OUT)**

DriveX, DriveY: AXIS_REF

x, y axis

Inputs (VAR_IN)

dOffsetX, dOffsetY: LREAL

Offset for x and y axis. Same values as described for SMC_TRAFOF_Gantry2

dAlpha: LREAL

Orientation of the tool in angular degrees

dToolA: LREAL

Length of the tool; distance between the rotation point and tool point

minX, maxX, minY, maxY: LREAL

Motion range (for visualisation)

Outputs (VAR_OUT)

dx, dy: LREAL

x, y position in geo coordinates

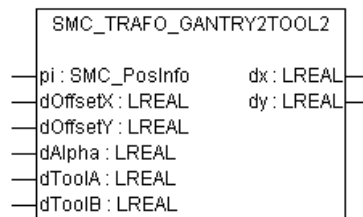
dnx, dny, dnl, dnOffsetX, dnOffsetY: LREAL

Standardised x and y position [0..1], tool length and offsets (for visualisation).

ratio: LREAL

Ratio of x interval and y interval (for visualisation)

9.1.2.4 SMC_TRAFO_Gantry2Tool2



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Output of the interpolator

dOffsetX, dOffsetY: LREAL

Offset for x and y axis

dAlpha: LREAL

Orientation of the tool in angular degrees

dToolA, dToolB: LREAL

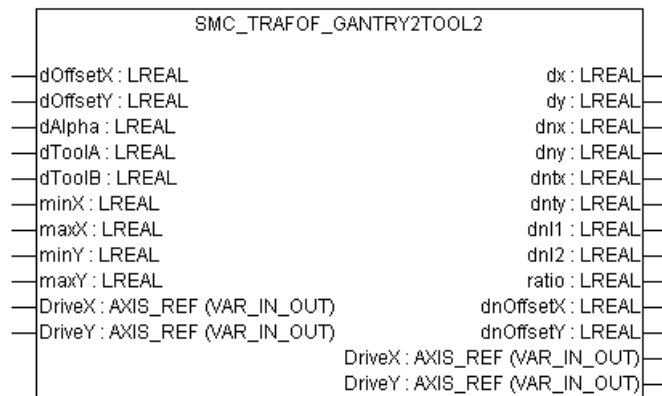
Leg lengths of the right-angled triangle located between the rotation point and the tool point. dToolA is the length of the leg at a tangent to the path, dToolB that of the leg that is orthogonal to the path.

If dToolB is positive, the tool point (x/y) is shifted to the left (viewed in the tool direction), otherwise it is shifted to the right.

Outputs (VAR_OUT)

dx, dy: LREAL

Setpoints for x and y axis

9.1.2.5 SMC_TRAFOF_Gantry2Tool2**Inputs/outputs (VAR_IN_OUT)**

DriveX, DriveY: AXIS_REF

x, y axis

Inputs (VAR_IN)

dOffsetX, dOffsetY: LREAL

Offset for x and y axis. Same values as described for SMC_TRAFO_Gantry2

dAlpha: LREAL

Orientation of the tool in angular degrees

dToolA, dToolB: LREAL

Length of the tool; distance between the rotation point and tool point

minX, maxX, minY, maxY: LREAL

Motion range (for visualisation)

Outputs (VAR_OUT)

dx, dy: LREAL

x, y position in geo coordinates

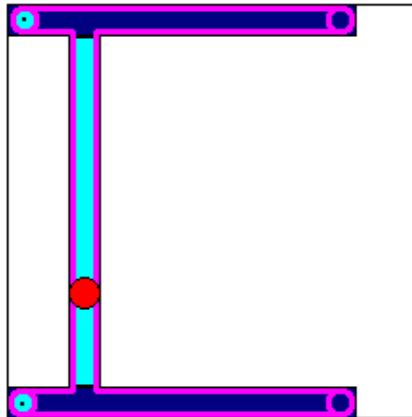
dnx, dny, dnl1, dnl2, dnOffsetX, dnOffsetY: LREAL

Standardised x and y position [0..1], tool length and offsets (for visualisation).

ratio: LREAL

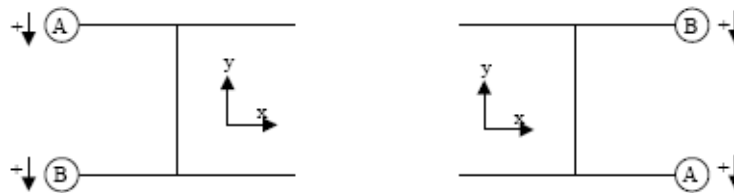
Ratio of x interval and y interval (for visualisation)

9.1.3 H gantry system with stationary drives



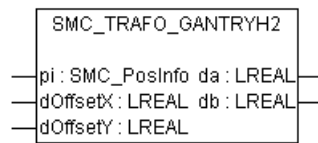
This kinematic system is similar to the gantry systems described above, but the drives are mounted stationary and move the slide and the Y axis via a belt that is looped around several times (shown in pink in the diagram).

The transformation is suitable for the drive configurations below; other configurations can be achieved by interchanging x and y:



Please note that a special reference run is necessary for this transformation. If you want to bring about a motion in the Y direction, drives A and B must be moved in parallel; for a pure X motion they must be counter-rotated. If the reference position has been found, the x and y values calculated by the forwards transformation FB are used as offsets (dOffsetX and dOffsetY).

9.1.3.1 SMC_TRAFO_GantryH2



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Output of the interpolator

dOffsetX, dOffsetY: LREAL

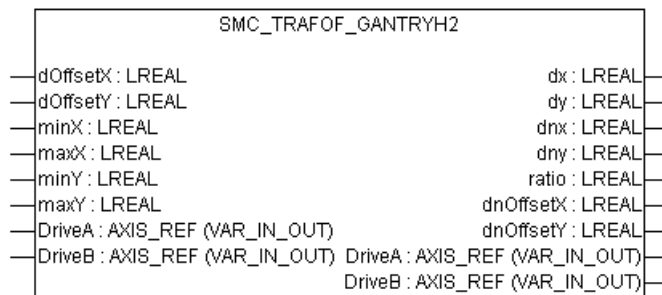
Offset for x and y axis

Outputs (VAR_OUT)

da, db: LREAL

Setpoints for A and B axis

9.1.3.2 SMC_TRAFOF_GantryH2



Inputs/outputs (VAR_IN_OUT)

DriveA, DriveB: AXIS_REF

A, B axis

Inputs (VAR_IN)

dOffsetX, dOffsetY: LREAL

Offset for x and y axis. Same values as described for SMC_TRAFOF_GantryH2

minX, maxX, minY, maxY: LREAL

Motion range (for visualisation)

Outputs (VAR_OUT)

dx, dy: LREAL

x, y position in geo coordinates

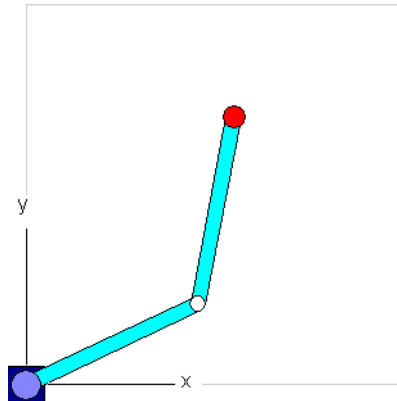
dnx, dny, dnOffsetX, dnOffsetY: LREAL

Standardised x and y position [0...1] and offsets (for visualisation).

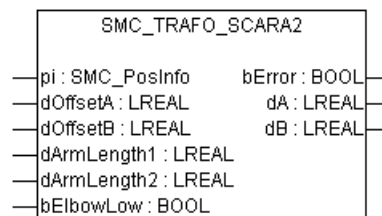
ratio: LREAL

Ratio of x interval and y interval (for visualisation)

9.1.4 Two-jointed Scara systems



9.1.4.1 SMC_TRAFO_Scara2



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Output of the interpolator

dOffsetA, dOffsetB: LREAL

Offset for A and B axis

dArmLength1, dArmLength2: LREAL

Length of the first and second arm

bElbowLow: BOOL

Elbow down (TRUE) or up (FALSE)

Outputs (VAR_OUT)

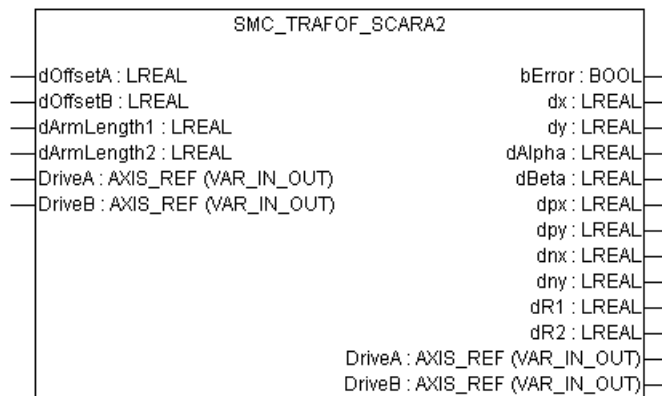
bError BOOL

TRUE: Impermissible values

da, db: LREAL

Axis position of A or B axis

9.1.4.2 SMC_TRAFOF_Scara2



Inputs/outputs (VAR_IN_OUT)

DriveA, DriveB: AXIS_REF

A, B axis

Inputs (VAR_IN)

dOffsetA, dOffsetB: LREAL

Offset for A and B axis. Same values as described for SMC_TRAFO_Scara2

dArmLength1, dArmLength2: LREAL

Length of the first and second arm

Outputs (VAR_OUT)

bError BOOL

TRUE: Impermissible values

dx, dy: LREAL

x, y position in geo coordinates

dAlpha, dBeta: LREAL

Joint angle (axis positions without offset) (for visualisation)

dpx, dpy: LREAL

Standardised position of the first joint [-1...1] (for visualisation)

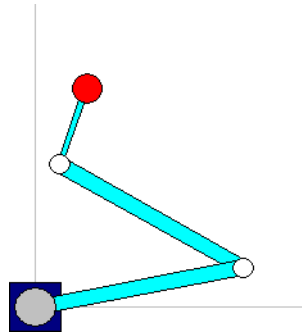
dnx, dny: LREAL

Standardised position of the manipulator [-1...1] (for visualisation)

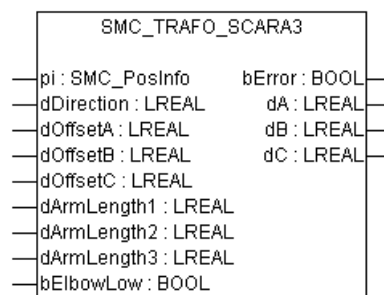
dR1, dR2: LREAL

Relative arm lengths $dR1 + dR2 = 1$. [0...1] (for visualisation)

9.1.5 Three-jointed Scara systems



9.1.5.1 SMC_TRAFO_Scara3



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Output of the interpolator

dDirection: LREAL

Direction angle of the last joint in degrees (0° W, 90° N)

dOffsetA, dOffsetB, dOffsetC: LREAL

Offset for A, B and C axis

dArmLength1, dArmLength2, dArmLength3: LREAL

Length of the arms

bElbowLow: BOOL

Elbow (first and second joint) down (TRUE) or up (FALSE)

Outputs (VAR_OUT)

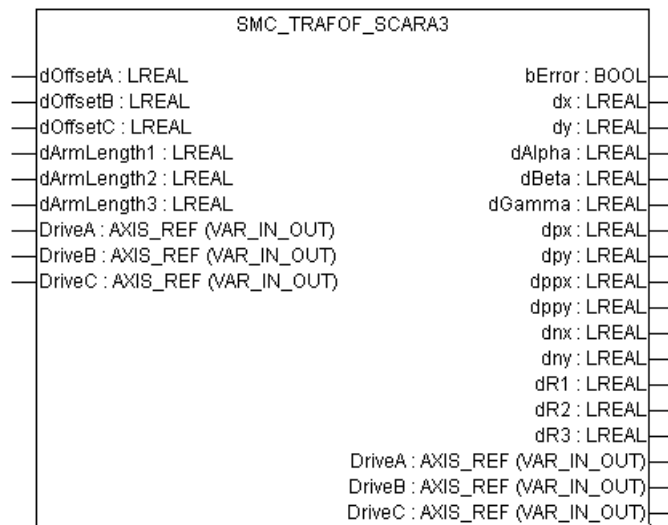
bError : BOOL

TRUE: Impermissible values

dA, dB, dC: LREAL

Axis position of A, B, C axis

9.1.5.2 SMC_TRAFOF_Scara3



Inputs/outputs (VAR_IN_OUT)

DriveA, DriveB, DriveC: AXIS_REF

A, B and C axis

Inputs (VAR_IN)

dOffsetA, dOffsetB, dOffsetC: LREAL

Offset for A, B and C axis. Same values as described for SMC_TRAFO_Scara3

dArmLength1, dArmLength2, dArmLength3: LREAL

Length of the arms

Outputs (VAR_OUT)

bError BOOL

TRUE: Impermissible values

dx, dy: LREAL

x, y position in geo coordinates

dAlpha, dBeta, dGamma: LREAL

Joint angle (axis positions without offset) (for visualisation)

dpx, dpy: LREAL

Standardised position of the first joint]-1...1[(for visualisation)

dppx, dppy: LREAL

Standardised position of the first joint]-1...1[(for visualisation)

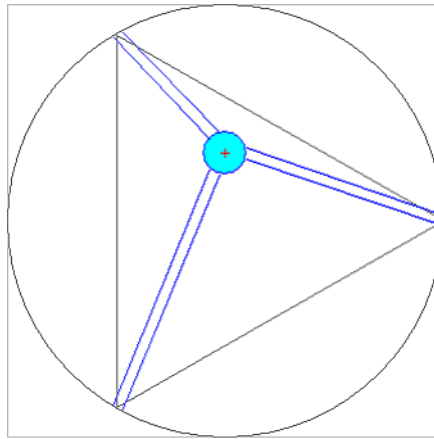
dnx, dny: LREAL

Standardised position of the manipulator]-1...1[(for visualisation)

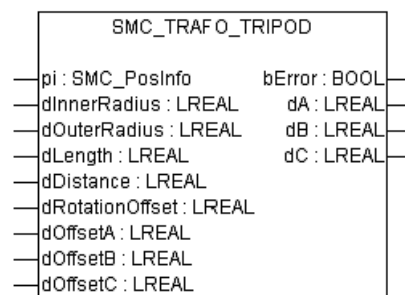
dR1, dR2, dR3: LREAL

Relative arm lengths $dR1 + dR2 + dR3 = 1$]0...1[(for visualisation)

9.1.6 Parallel kinematics



9.1.6.1 SMC_TRAFO_Tripod



Inputs (VAR_IN)

pi: SMC_PosInfo

Target position vector. Position of the centre of the inner ring. Output of the interpolator

dInnerRadius: LREAL

Radius of the inner ring

dOuterRadius: LREAL

Radius of the outer ring

dLength: LREAL

Strut lengths

dDistance: LREAL

Distance between two connected struts at the outer and inner ring

dRotationOffset: LREAL

Direction in which axis A is facing when viewed from the origin (0/0), in degrees (mathematical sense)

dOffsetA, dOffsetB, dOffsetC: LREAL

Offset of the individual axes

Outputs (VAR_OUT)

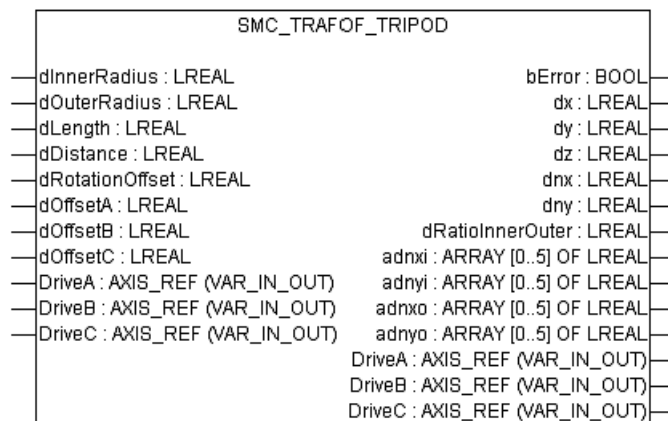
bError BOOL

TRUE: Impermissible values. Exit workspace

dA, dB, dC: LREAL

Axis position of A, B, C axis

9.1.6.2 SMC_TRAFOF_Tripod



Inputs/outputs (VAR_IN_OUT)

DriveA, DriveB, DriveC: AXIS_REF

A, B and C axis

Inputs (VAR_IN)

dInnerRadius: LREAL

Radius of the inner ring

dOuterRadius: LREAL

Radius of the outer ring

dLength: LREAL

Strut lengths

dDistance: LREAL

Distance between two connected struts at the outer and inner ring

dRotationOffset: LREAL

Direction in which axis A is facing when viewed from the origin (0/0), in degrees (mathematical sense)

dOffsetA, dOffsetB, dOffsetC: LREAL

Offset of the individual axes

Outputs (VAR_OUT)

bError BOOL

TRUE: Impermissible values

dx, dy, dz: LREAL

x, y, z position of the centre of the inner ring in geo coordinates

dnx, dny: LREAL

Standardised position of the manipulator (for visualisation)

dRatioInnerOuter: LREAL

Ratio of the radius of the inner ring to that of the outer ring (for visualisation)

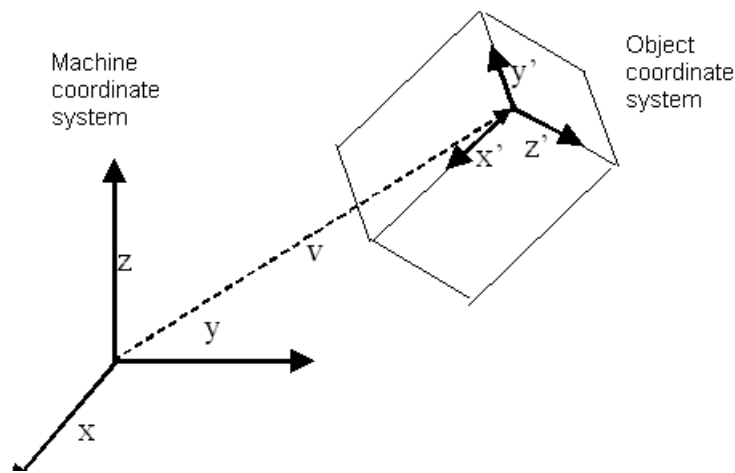
adnxi, adnyi, adnxo, adnyo: ARRAY[0...5] OF LREAL

Standardised start and end position of the struts (for visualisation)

9.1.7 Spatial transformations

In some cases the machine and workpiece have different coordinate systems. The transformation POU's of the SM_Trafo.lib library transform the Cartesian coordinates of the tool point into machine coordinates. Prior to this, however, if the workpiece is not oriented exactly in accordance with the CNC program, it may be necessary to transform the path coordinates calculated by the interpolator before passing them on to the machine transformation.

Imagine a standard gantry (X/Y/Z). The tool point of the gantry must be moved across the surface of a workpiece that is positioned transversely in space:



There are several options available for describing the relationship between two coordinate systems. A coordinate transformation always consists of a spatial translation and a spatial rotation. The translation is described by a three-dimensional vector, the rotation either by three angles (e.g. YawPitchRoll) or by the three unit vectors of the new (object) coordinate system x' , y' , z' .

If the method using the three rotation angles is selected, these can be defined according to the Roll/Pitch/Yaw (RPY) convention, for example. In this case the new coordinate system results from the old one by means of a rotation around various axes. The RPY (α , β , γ) can operate in two ways, both of which provide the same result:

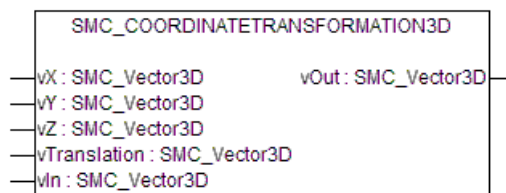
1. Starting from the coordinate system (x, y, z) , rotate the coordinate system around the z axis by angle β in a mathematically positive direction. This will produce the new coordinate system $(x_1, y_1, z_1=z)$. Now fix axis y_1 of the new coordinate system and rotate the system around β , which will produce $(x_2, y_2=y_1, y_2)$. Finally, rotate this coordinate system around x_2 by angle α , thus producing $(x'=x_2, y', z')$.
2. Starting from the coordinate system (x, y, z) , rotate the coordinate system around the x axis by angle α . Rotate the resulting coordinate system $(A_\alpha=x, y_\alpha, z_\alpha)$ around the original y axis (not y_α !) by β ($x_\beta, y_\beta, z_\beta$) and then around the original z axis by γ , which will produce (x', y', z') .

The SM_CNC.lib library provides the following POU's for spatial transformations:

- ▶ SMC_CoordinateTransformation3D
- ▶ SMC_TeachCoordinateTransformation
- ▶ SMC_UnitVectorToRPY
- ▶ SMC_DetermineCuboidBearing

9.1.7.1 SMC_CoordinateTransformation3D

This program organisation unit calculates the coordinates of a position (available in the old coordinate system) with reference to the new coordinate system. For this purpose the coordinate transformation for the new coordinate system is defined in relation to the old one on the basis of the translation vector and the new unit vectors.



Inputs (VAR_IN)

vX, vY, vZ: SMC_Vector3D

Unit vectors of the new coordinate system with reference to the old one

vTranslation: SMC_Vector3D

Translation vector. Vector from the old coordinate origin to the new one with reference to the old coordinate system

vIn: SMC_Vector3D

Position to be transformed

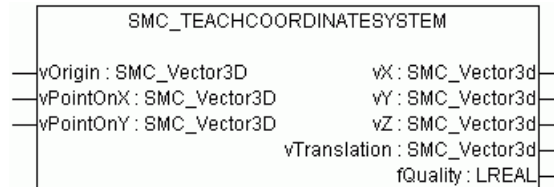
Outputs (VAR_OUT)

vOut: SMC_Vector3D

Transformed position

9.1.7.2 SMC_TeachCoordinateSystem

This POU of SM_Trafo.lib helps the user to calculate the vectors of a new coordinate system which he wants to "teach". As inputs the source of the new coordinate system and a random point are transferred to the x and y axis, respectively. As outputs the POU provides the translation vector and the unit vectors of the new coordinate system. Furthermore it calculates the quality of the input parameters (angle between x and y axis should be 90°).



Inputs (VAR_IN)

vOrigin: SMC_Vector3D

Source position of the new coordinate system.

vPointOnX, vPointOnY: SMC_Vector3D

Position of a point on the x or y axis.

Outputs (VAR_OUT)

vX, vY, vZ: SMC_Vector3D

Unit vectors of the new coordinate system.

vTranslation: SMC_Vector3D

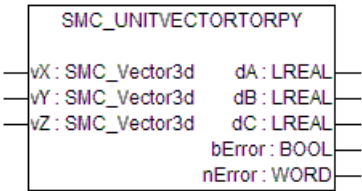
Translation vector. Vector from the old coordinate origin to the new one with reference to the old coordinate system

fQuality: LREAL

Quality of the input parameters; value from 0 to 1, where 1 = optimum (vX, vY are orthogonal), and 0 = vX and vY are invalid.

9.1.7.3 SMC_UnitVectorToRPY

Program organisation unit for calculating the RPY angle (with reference to the old coordinate system) from the unit vectors of the new coordinate system



Inputs (VAR_IN)

vX, vY, vZ: SMC_Vector3D

Unit vectors of the new coordinate system with reference to the old one

Outputs (VAR_OUT)

dA, dB, dC: LREAL

RPY angle in radian measure

bError : BOOL

Invalid input values

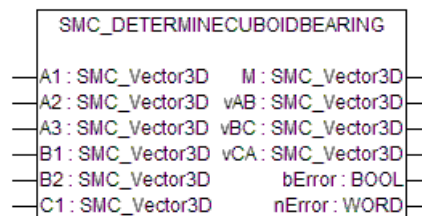
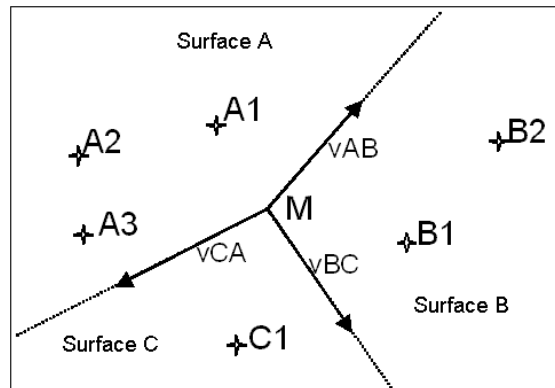
nError: WORD

Error description

Value	Information
0	No error (bError = FALSE)
1	Vectors do not have length 1
2	Vectors are not perpendicular to one another
3	No right-hand system

9.1.7.4 SMC_DetermineCuboidBearing

Program organisation unit for determining the position of a cuboid (corner point, edge alignment) in space by specifying 6 (3/2/1) points:



Inputs (VAR_IN)

A1, A2, A3: SMC_Vector3D

Three points on a lateral surface A of the cuboid, which must not lie on a straight line

B1, B2: SMC_Vector3D

Two points on different lateral surfaces of the cuboid, the projections of which onto surface A must not be identical

C1: SMC_Vector3D

Points on another lateral surface of the cuboid

Outputs (VAR_OUT)

M: SMC_Vector3D

Corner point of the cuboid

vAB, vBC, vCA: SMC_Vector3D

Unit vectors on the edge lines of the cuboid

bError : BOOL

Invalid input values

nError: WORD

Error description

Value	Information
0	No error (bError = FALSE)
1	A1, A2, A3 lie on a straight line
2	Projections of B1 and B2 onto surface A are identical

10 The SM_Error.lib library

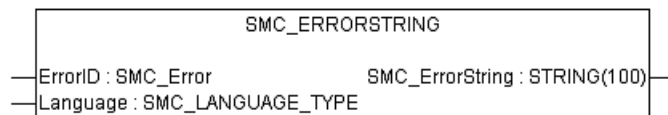
This library must be available in every project, as it contains all error definitions. It contains all errors generated by SoftMotion function POU's and can display them as strings.

Basically, the application programmer should note that two types of error may occur in his or her program. On the one hand, there could be drive errors, which are errors occurring in the drive (i.e. following errors, missing power supply, etc.). On the other hand, there could be POU errors, which are errors reported by program organisation units via the Error and ErrorID outputs and which can often be traced back to incorrect parameterisation.

Drive errors must be read via MC_ReadAxisError or MC_ReadParameter and, where appropriate, deleted using MC_Reset. Drive errors are drive-specific and are not standardised.

If necessary, POU errors can be converted into strings by means of the functions in SM_Error.lib. As these errors could occur on all SoftMotion program organisation units and would have to be collated by the application, an additional function has been implemented in the AXIS_REF data structure, allowing a list of the errors that have occurred most recently to be stored. The FBErrorOccured output of MC_ReadStatus can be used to check if a POU error has occurred and, if so, which one occurred most recently. The SMC_ReadFBError function block returns the error number of the first error to have occurred and the SMC_ClearFBError function clears it.

10.1 SMC_ErrorString



The SMC_ErrorString function returns a string representing the error, in accordance with the ErrorID (SMC_Error) and Language (SMC_LANGUAGE_TYPE (English, German)) inputs.

10.2 The enumeration SMC_Error

The SMC_Error enumeration contains all error numbers generated by SoftMotion FBs:

Error	POU	Enum value	Description
0	All	SMC_NO_ERROR	No error
1	Drive Interface	SMC_DI_GENERAL_COMMUNICATION_ERROR	Communication error
2	Drive Interface	SMC_DI_AXIS_ERROR	Axis error
10	Drive Interface	SMC_DI_SWLIMITS_EXCEEDED	Position outside of permissible range (SWLimit)
11	Drive Interface	SMC_DI_HWLIMITS_EXCEEDED	Hardware limit switch active
12	Drive Interface	SMC_DI_LINEAR_AXIS_OUTOFRANGE	The position limits of the axis have been exceeded
13	Drive Interface	SMC_DI_HALT_OR_QUICKSTOP_NOT_SUPPORTED	The "Halt" or "Quick stop" drive status is not supported
20	All program organisation units that generate motion	SMC_REGULATOR_OR_START_NOT_SET	Controller enable not set or brake applied
30	Drive Interface	SMC_FB_WASNT_CALLED_DURING_MOTION	Program organisation unit that generates motion was not called again before the end of the motion
31	All program organisation units	SMC_AXIS_IS_NO_AXIS_REF	Entered AXIS_REF variable is not of type AXIS_REF
32	All program organisation units that generate motion	SMC_AXIS_REF_CHANGED_DURING_OPERATION	Entered AXIS_REF variable has been exchanged whilst the program organisation unit was active
50	SMC_Homing	SMC_3SH_INVALID_VELOCITY_VALUES	Invalid velocity or acceleration values
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HWLIMIT	Mode requires limit switches to be used (for safety reasons)
70	SMC_SetControllerMode	SMC_SCM_NOT_SUPPORTED	Mode not supported
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLER_MODE	Axis not in correct controller mode
80	SMC_ResetAxisGroup	SMC_RAG_ERROR_DURING_STARTUP	Error during axis group start-up
90	SMC_ChangeGearingRatio	SMC_CGR_ZERO_VALUES	Impermissible values
91	SMC_ChangeGearingRatio	SMC_CGR_DRIVE_POWERED	Gearing parameters may not be changed while the drive is being controlled
110	MC_Power	SMC_P_FTASKCYCLE_EMPTY	The axis does not contain any information relating to the cycle time (fTaskCycle = 0)
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	No errors on axis
121	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER	Axis does not perform an error reset
122	MC_Reset	SMC_R_ERROR_NOT_RESETTABLE	Error could not be reset
123	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER_IN_TIME	Communication with the axis failed
130	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	Parameter number unknown

Error	POU	Enum value	Description
131	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_REQUESTING_ERR OR	Error during transmission to the drives. See error number in the ReadDriveParameter POU instance (SM_DriveBasic.lib)
140	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_PARAM_INVALID	Parameter number unknown or writing not allowed
141	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_SENDING_ERROR	Error in the transmission to the drives. Check if the call in the Motion Task is programmed. See error numbers of the function block WriteDriveParameter (SM_DriveBasic.lib)
170	MC_Home	SMC_H_AXIS_WASNT_STAN DSTILL	Axis was not in the "standstill" state
171	MC_Home	SMC_H_AXIS_DIDNT_START _HOMING	Error starting the homing action
172	MC_Home	SMC_H_AXIS_DIDNT_ANSW ER	Communication error
173	MC_Home	SMC_H_ERROR_WHEN_STO PPING	Error stopping after homing. Deceleration set?
180	MC_Stop	SMC_MS_UNKNOWN_STOP PING_ERROR	Unknown stopping error
181	MC_Stop	SMC_MS_INVALID_ACCDEC _VALUES	Impermissible velocity or acceleration values
182	MC_Stop	SMC_MS_DIRECTION_NOT_ APPLICABLE	Direction = shortest/fastest not applicable
183	MC_Stop	SMC_MS_AXIS_IN_ERRORST OP	Drive is in "errorstop" state. Stop cannot be executed
184	MC_Stop	SMC_BLOCKING_MC_STOP_ WASNT_CALLED	An instance of MC_Stop which is blocking the axis (Execute=TRUE) was not called. Please call MC_Stop(Execute=FALSE) first.
201	MC_MoveAbsolute	SMC_MA_INVALID_VELACC_ VALUES	Impermissible velocity or acceleration values
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTI ON	Direction error
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_ VALUES	Impermissible velocity or acceleration values
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTI ON	Direction error
251	MC_MoveAdditive	SMC_MAD_INVALID_VELAC C_VALUES	Impermissible velocity or acceleration values
252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECT ION	Direction error
276	MC_MoveSuperImposed	SMC_MSI_INVALID_VELACC _VALUES	Impermissible velocity or acceleration values
277	MC_MoveSuperImposed	SMC_MSI_INVALID_DIRECTI ON	Direction error
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC _VALUES	Impermissible velocity or acceleration values
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_ APPLICABLE	Direction = shortest/fastest not applicable
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	Incorrect array size
326	MC_PositionProfile	SMC_PP_STEPOMS	Step time = t#0s
350	MC_VelocityProfile	SMC_VP_ARRAYSIZE	Incorrect array size

Error	POU	Enum value	Description
351	MC_VelocityProfile	SMC_VP_STEP0MS	Step time = t#0s
375	MC_AccelerationProfile	SMC_AP_ARRAYSIZE	Incorrect array size
376	MC_AccelerationProfile	SMC_AP_STEP0MS	Step time = t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	Trigger already active
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WINDOW	Drive Interface does not support the window function
402	MC_TouchProbe	SMC_TP_COMM_ERROR	Communication error
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIED	Trigger already unoccupied
426	SMC_MoveContinuousRelative	SMC_MCR_INVALID_VELOCITY_VALUES	Impermissible velocity or acceleration values
427	SMC_MoveContinuousRelative	SMC_MCR_INVALID_DIRECTION	Direction error
451	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_VELOCITY_VALUES	Impermissible velocity or acceleration values
452	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_DIRECTION	Direction error
453	SMC_MoveContinuousAbsolute	SMC_MCA_DIRECTION_NOT_APPLICABLE	Direction = fastest not applicable
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	Cam does not contain any tappets
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	Tappet group ID exceeds MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACCESSES	More than 32 accesses to one CAM_REF
625	MC_CamIn	SMC_CI_NO_CAM_SELECTED	No cam selected
626	MC_CamIn	SMC_CI_MASTER_OUT_OF_SCALE	Master axis outside of the permissible range
627	MC_CamIn	SMC_CI_RAMPIN_NEEDS_VELOCITY_VALUES	Velocity and acceleration values must be specified for the ramp_in function
628	MC_CamIn	SMC_CI_SCALING_INCORRECT	fEditor/TableMasterMin/Max scaling variables incorrect
640	SMC_CAMBounds, SMC_CamBounds_Pos	SMC_CB_NOT_IMPLEMENTED	Function block for this cam data format not implemented.
675	MC_GearIn	SMC_GI_RATIO_DENOM	RatioDenominator = 0
676	MC_GearIn	SMC_GI_INVALID_ACC	Impermissible acceleration
677	MC_GearIn	SMC_GI_INVALID_DEC	Impermissible deceleration
725	MC_Phase	SMC_PH_INVALID_VELOCITY_DEC	Impermissible velocity, deceleration or acceleration values
726	MC_Phase	SMC_PH_ROTARYAXIS_PERIOD0	Rotating axis with fPositionPeriod = 0
750	All program organisation units that use MC_CAM_REF as an input	SMC_NO_CAM_REF_TYPE	Entered cam is not of type MC_CAM_REF
775	MC_GearInPos	SMC_GIP_MASTER_DIRECTION_CHANGE	Master axis changes direction of rotation during coupling in
1001	SMC_Interpolator	SMC_INT_VEL_ZERO	Path cannot be traversed, as setpoint velocity = 0
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	Very last path object has Vel_End > 0

Error	POU	Enum value	Description
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	Warning: GEOINFO list processed in DataIn, but the end of the list is not set. Reason: Forgotten to set <i>EndOfList</i> for the queue in <i>DataIn</i> or SMC_Interpolator is faster than the program organisation units generating the path
1004	SMC_Interpolator	SMC_INT_VEL_NONZERO_AT_STOP	Velocity at stop position > 0
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSIONS	Too many SMC_Interpolator recursions. SoftMotion error
1006	SMC_Interpolator	SMC_INT_NO_CHECKVELOCITIES	Input OutQueue DataIn has not processed the SMC_CheckVelocities program organisation unit last
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	Internal/numerical error
1008	SMC_Interpolator	SMC_INT_VEL_ACC_DEC_ZERO	Speed, acceleration or deceleration zero or too small.
1050	SMC_Interpolator2Dir	SMC_INT2DIR_BUFFER_TOO_SMALL	Data buffer too small
1051	SMC_Interpolator2Dir	SMC_INT2DIR_PATH_FITS_NOT_IN_QUEUE	Path does not fit completely into the queue
1080	SMC_Interpolator	SMC_WAR_INT_OUTQUEUE_TOO_SMALL	Warning: OutQueue <i>DataIn</i> has been inadequately dimensioned. Stops may not be able to be performed.
1081	SMC_Interpolator	SMC_WAR_END_VELOCITIES_INCORRECT	Warning: Inconsistent end velocities
1100	SMC_CheckVelocities	SMC_CV_ACC_DEC_VEL_NONPOSITIVE	Impermissible velocity, deceleration or acceleration values
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	Impermissible acceleration value
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	Impermissible acceleration value
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_EMPTY	Data underrun. Queue has been read and is empty.
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMALL	Buffer too small
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	Buffer elements are of the wrong type
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	Current line of the interpolator could not be found
1500	All function POUs that use SMC_CNC_REF	SMC_NO_CNC_REF_TYPE	Entered CNC program is not of type SMC_CNC_REF
1501	All function POUs that use SMC_OUTQUEUE	SMC_NO_OUTQUEUE_TYPE	Entered OutQueue is not of type SMC_OUTQUEUE
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXIST	File does not exist
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	No buffer created
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMALL	Buffer too small
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	Data underrun. Buffer has been read and is empty.
2004	SMC_ReadNCFile	SMC_RNCF_VAR_COULDNT_BE_REPLACED	Placeholder variable could not be replaced
2005	SMC_ReadNCFile	SMC_RNCF_NOT_VARLIST	Input pvl is not pointing to an SMC_VARLIST object

Error	POU	Enum value	Description
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXIST	File could not be opened
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	No buffer defined
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMALL	Buffer too small
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	Unexpected end of file
2100	SMC_AxisDiagnosticLog	SMC_ADL_FILE_CANNOT_BE_OPENED	File could not be opened
2101	SMC_AxisDiagnosticLog	SMC_ADL_BUFFER_OVERRUN	Buffer overrun; call WriteToFile more frequently
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXIST	File could not be opened
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	Saved cam too large
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	Incorrect compilation mode
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	Incorrect file version
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	Unexpected end of file
3001	SMC_WriteDriveParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	Parameter selection channel occupied
3002	SMC_WriteDriveParamsToFile	SMC_WDPF_CANNOT_CREATE_FILE	File could not be created
3003	SMC_WriteDriveParamsToFile	SMC_WDPF_ERROR_WHEN_READING_PARAMS	Error when reading parameters
3004	SMC_WriteDriveParamsToFile	SMC_WDPF_TIMEOUT_PREPARING_LIST	Timeout when preparing the parameter list

11 The SM_FileFBs.lib library

This library provides program organisation units relating to file functionality.



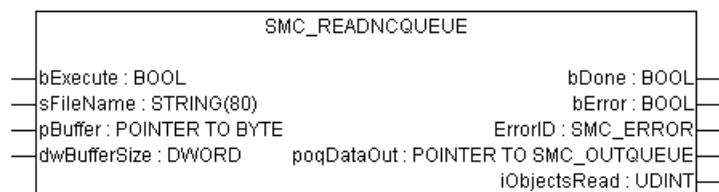
Note!

The use of the SM_FileFBs.lib library requires the 3S system libraries **SysLibFile.lib** and **Standard.lib**!

11.1 CNC function blocks

11.1.1 SMC_ReadNCQueue

This program organisation unit reads in an OutQueue file from the control file system, which has been created by the CNC editor, and provides an OutQueue structure that is typically processed by the interpolator.



Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit starts to read the queue in on a rising edge.

sFileName: STRING(80)

File path and name

pBuffer: POINTER TO BYTE

Pointer to a sufficiently large, available data area reserved in the IEC application

dwBufferSize: DWORD

Size of the data area in bytes

Outputs (VAR_OUT)

bDone : BOOL

Set once the queue has been read in completely

bError : BOOL

TRUE: Error has occurred

ErrorID: SMC_ERROR

Error number

poqDataOut: POINTER TO SMC_OUTQUEUE

Pointer to the queue that has been read in

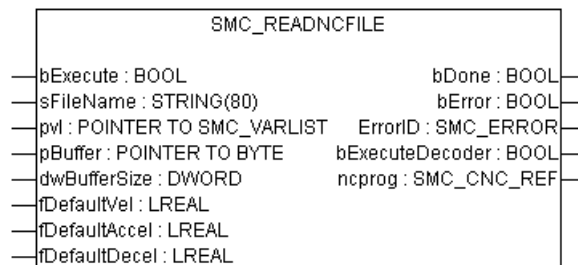
iObjectsRead: UDINT

Number of SMC_GeoInfo objects that have so far been read in and written to the queue

11.1.2 SMC_ReadNCFile

This program organisation unit reads in an NC ASCII file from the control file system so that it can be made available to the SMC_NCDecoder program organisation unit. In this way, a CNC program can be read in and implemented during runtime.

If no default velocity/acceleration/deceleration is defined in the CNC program (G-code file without Exx/Exx/E-xx information), you must select them via the three inputs fDefaultVel, fDefaultAccel and fDefaultDecel. Otherwise an error will be indicated.

**Inputs (VAR_IN)**

bExecute: BOOL

The program organisation unit starts to read the program in on a rising edge.

sFileName: STRING(80)

File path and name

pvl: POINTER TO SMC_VARLIST

Pointer to an SMC_VARLIST object ([SMC_VARLIST structure](#)). If no variables are to be used in the CNC program, this input will not be assigned.

pBuffer: POINTER TO BYTE

Pointer to a sufficiently large, available data area reserved in the IEC application

dwBufferSize: DWORD

Size of the data area in bytes

fDefaultVel: LREAL

Selection of a velocity.

fDefaultAccel: LREAL

Selection of an acceleration.

fDefaultDecel: LREAL

Selection of a deceleration.

Outputs (VAR_OUT)

bDone : BOOL

Set once the program has been read in completely

bError : BOOL

TRUE: Error has occurred

ErrorID: SMC_ERROR

Error number

bExecuteDecoder: BOOL

Signal that is to trigger the Execute input of the SMC_NCDecoder program organisation unit

ncprog: SMC_CNC_REF

CNC program. Input of the subsequent SMC_NCDecoder program organisation unit

11.1.3 SMC_VARLIST structure

IEC 1131-3 does not make provision for acquiring the value of a variable from its symbolic name, which is available, for example, as a string. This must be possible, however, if the variable functionality available to the user with the "Create variable on compile" compilation option is also to be used when reading in the CNC program from a file. The SMC_VARLIST structure can be used to achieve this. The structure consists of the wNumberVars variable, which contains the number of variables used, and a pointer, psvVarList, which points to the first element of an array of SMC_SingleVar containing the type and a pointer to the variable. An SMC_SingleVar object contains the character string strVarName, which in turn contains the name of the variable as it appears in the CNC program, in capital letters. Besides that it also provides a pointer to the variable and its type.

Example: The CNC program that is read from a file using SMC_ReadNCFile contains two variables, g_rTestX (REAL) and g_byCommand (BYTE):

```
N0 G$g_byCommand$ X$g_rTestX$
```

Now define the following variables:

```
g_byCommand: BYTE;
g_rTest: REAL;
asv: ARRAY[0..1] OF SMC_SingleVar :=
  (strVarName:='G_BYCOMMAND', eVarType:=SMC_TYPE_BYTE),
  (strVarName:='G_rTESTX', eVarType:=SMC_TYPE_REAL);
vl: SMC_VarList:=(wNumberVars:=2);
```

Prior to calling the SMC_ReadNCFile program organisation unit whose pvl input will then be fed with ADR(vl), you must set the pointer to the variables:

```
asv[0].pAdr:=ADR(g_byCommand);
asv[1].pAdr:=ADR(g_rTest);
```

You must also link SMC_VarList to SMC_SingleVar:

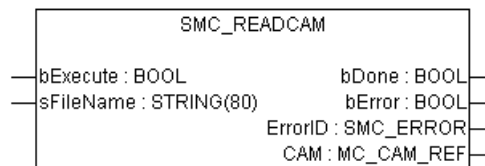
```
vl.psvVarList := ADR(asv[0]);
```

If a variable cannot be replaced, an error will be output and the program organisation unit will abort.

11.2 CAM function blocks

11.2.1 SMC_ReadCAM

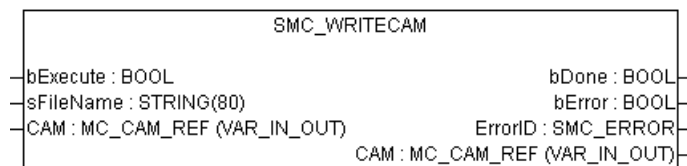
This program organisation unit is used to load a cam that has been created with the CAM editor and saved in a *.CAM file during runtime and to make it available to the MC_CamTableSelect and MC-CamIn program organisation units.



The size of cam that can be loaded is limited by the gc_SMC_FILE_MAXCAMEL (number of elements) and gc_SMC_FILE_MAXCAMTAP (number of tappet switch actions) global constants.

11.2.2 SMC_WriteCAM

This POU of the SM_FileFBs.lib is used to save a cam which was created by means of the CAM editor in a *.CAM file. (See also [SMC_ReadCAM](#).)

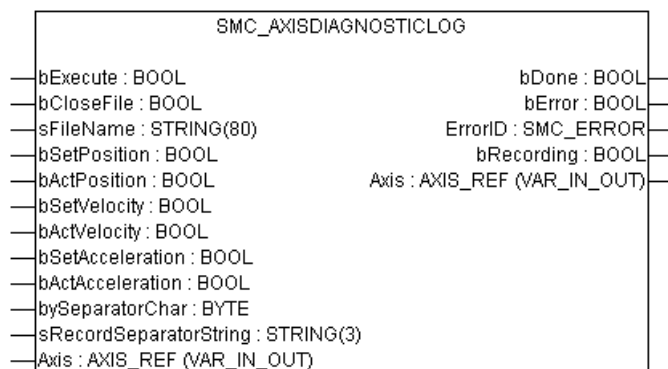


11.3 Diagnostic function blocks

11.3.1 SMC_AxisDiagnosticLog

This program organisation unit can be used to write selected axis values to a file cyclically. The resulting file is ideal for diagnostic use.

As it usually takes quite some time to write data to a data medium, this program organisation unit stores the collected data in a 5 KB buffer and the data is not written until the WriteToFile POU action is called. This call should, therefore, be moved to a slower (around 50 ms) task of lower priority to ensure that the actual motion task is not obstructed and that the drive behaviour is not disturbed. If the buffer overflows, the program organisation unit will generate an error.



Inputs/outputs (VAR_IN_OUT)

Axis: AXIS_REF;

Axis to be watched

Inputs (VAR_IN)

bExecute: BOOL

The program organisation unit starts on a rising edge. If a file of the same name already exists, it will be overwritten.

bCloseFile: BOOL

The program organisation unit closes the file as soon as this input assumes the value TRUE.

sFileName: STRING(80)

File path and name

bSetPosition, bActPosition, bSetVelocity, bActVelocity, bSetAcceleration, bActAcceleration: BOOL

These inputs determine whether the associated axis values will be saved to the file or not.

bySeparatorChar: BYTE (default: TAB)

ASCII code of the character to be written between two values of the same data item

sRecordSeparatorString: STRING(3) (default: ,R\$N')

String to be written at the end of a data item

Outputs (VAR_OUT)

bDone BOOL

TRUE: Logging terminated, file closed

bError : BOOL

TRUE: Error has occurred

ErrorID: SMC_ERROR

Error number

bRecording: BOOL

Program organisation unit is recording.

12 The L_SMC_DriveUtil.lib library (Lenze specifications)

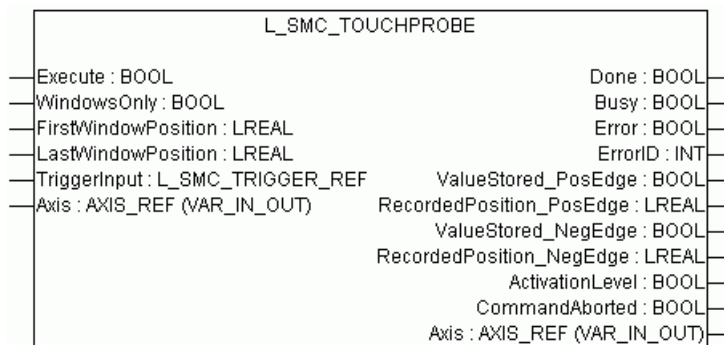
This module library contains specific Lenze blocks.

12.1 L_SMC_TouchProbe

This block serves to detect the position of the drive in a highly precise manner via a touch probe input at the drive. In contrast to the previously available function block **MC_Touchprobe**, the command to be executed and the return information of measured values are effected via process data (PDO). The advantage of this is that the data can be transmitted from and to the drive in a considerably quicker way than this is possible in the case of a control via service data (SDO).

Use of **MC_Touchprobe** and **L_SMC_Touchprobe**:

Function block	ECSxM axis module	Servo Drive 9400 CiA402
MC_Touchprobe	Only CAN • Control via SDO	Is not supported.
L_SMC_Touchprobe	Only CAN • Control via PDO	CAN and EtherCAT • Control via PDO



Inputs (VAR_INPUT)

Execute : BOOL

By setting a new positive edge, all outputs of the block are reset to the initial values.

WindowsOnly: BOOL

Activate window evaluation: All measured values outside the window defined are masked out.

FirstWindowsPosition: LREAL

Lower limit of the window in SoftMotion units [SM units]

LastWindowPosition: LREAL

Upper limit of the window in SoftMotion units [SM units]

Axis: AXIS_REF

Selection of the drive

TriggerInput: L_SMC_Trigger_REF

The 'Triggerinput' variable is of the L_SMC_TRIGGER_REF type with the following elements:

Variable	Type	Initial value	Description
TouchProbeSelect	Byte	1	TP1 or TP2 selection 1: TP_channel1 -> Servo Drive 9400 - DigIn 7 or zero pulse of the position encoder 2: TP_channel2 -> Servo Drive 9400 - DigIn 8 or zero pulse of the position encoder
TouchProbeMode	Byte	1	Selection of the operating mode 1: Single measurement - Execute edge activates the function until Done = 1 (TRUE) or abort by FB L_SMC_AbortTrigger. 2: Continuous measurement - Positive Execute edge activates the function until the negative Execute edge (Execute = FALSE) is reached, or until abort by FB L_SMC_AbortTrigger.
TouchProbeSignal	Byte	1	Selection of zero pulse or digital input 1: Digital input 2: Zero pulse of the position encoder
TouchProbeEdge	Byte	1	Configuration of the edge evaluation 1: Triggering with positive edge 2: Triggering with negative edge 3: Triggering with positive or negative edge (logic in the FB) 4: Triggering with positive and negative edge (The sequence of the edges is not relevant (logic in the FB).) 5: Triggering with positive and negative edge (The sequence of the edges is relevant (logic in the drive).) 6: Triggering with negative and positive edge (The sequence of the edges is relevant (logic in the drive).)

Outputs (VAR_OUTPUT)

Done : BOOL

In the "Single measurement" operating mode: TRUE= measuring procedure completed.

In the "Continuous measuring" operating mode: Without function

(See [TouchProbeMode](#) (258) variable.)

Busy: BOOL

Block activated, waiting for trigger.

Error : BOOL

Error identification: An error with regard to the command to be executed or during communication with the drive has occurred.

ErrorId: INT

Error number

ValueStored_PosEdge: BOOL

Measured value saved after triggering to positive edge.

RecordedPosition_PosEdge: LREAL

Measured value during triggering with the positive edge in SoftMotion units [SM units], i. e. the measured value is provided to the PLC application in the same unit with which the axis is programmed.

ValueStored_NegEdge: BOOL

Measured value saved after triggering to negative edge.

RecordedPosition_NegEdge: LREAL

Measured value during triggering with the negative edge in SoftMotion units [SM units], i. e. the measured value is provided to the PLC application in the same unit with which the axis is programmed.

ActivationLevel: BOOL

Level of the TP input at the time of the TP function activation in the drive. By evaluation of the ActivationLevel during the command to be executed "positive or negative edge", the application can detect the type of the first active edge.

CommandAborted: BOOL

Block aborted via L_SMCAbortTrigger. The outputs "Busy" and "Done" are reset.

Preparation: Creation of the suitable configuration of the drive

- ▶ ECSxM axis module
 - No special treatment:
Insert a "Drive" in the control configuration of the »PLC Designer« below the "Axis_Group_CAN".
- ▶ Servo Drive 9400 CiA402
 - Control via **CAN**:
Insert a "Drive_9400_TP" in the control configuration of the »PLC Designer« below the "Axis_Group_CAN".
 - Control via **EtherCAT**:
Select a corresponding mapping in the »EtherCAT Configurator« which contains the touch probe data as process data.

Restricted functional range for the control via CAN

Due to the limited bandwidth of the CAN bus, only the following operating modes of the **L_SMC_Touchprobe** function block are supported:

- ▶ TouchProbeSelect (TP1 or TP2 selection): Only "TP1" permissible.
- ▶ TouchProbeMode (selection of the operating mode): Only "Single measurement" permissible.
- ▶ TouchProbeSignal (selection of the zero pulse or digital input): No restriction
- ▶ TouchProbeEdge (configuration of the edge evaluation): Only "Triggering with positive edge" permissible.



Note!

If an **ECSxM axis module** is used, the measured values are transmitted via the so-called "monitor data" of the PDOs (See "Monitor data" section in the documentation for the ECSxM axis module).

The selection of the monitor data via code **C3181** is set to the value "7" once during run-up of the control system (nPosLatchDiff).

Error messages

The following error messages are stored:

Error number	Description
411	TouchProbeSelect incorrect: TP channel not available • TouchProbeSelect > 1 for ECSxM • TouchProbeSelect > 2 for Servo Drive 9400
412	TouchProbeMode incorrect: • TouchProbeMode > 1 for ECSxM (Only 'Single measurement' is supported.) • TouchProbeMode > 2 for Servo Drive 9400
413	TouchProbeEdge incorrect: If "TouchProbeSignal = zero pulse" is selected, only the 'positive edge' edge evaluation is possible • TouchProbeMode > 1 for ECSxM (Only 'Single measurement' is supported.) • TouchProbeMode > 2 for Servo Drive 9400
414	TouchProbeEdge incorrect: Requested edge evaluation is not available • TouchProbeEdge > 2 for ECSxM (Only positive or negative edge possible.) • TouchProbeEdge > 4 for Servo Drive 9400 (Type of edge evaluation is not supported.)
415	TP channel not free: TP channel has already been activated via another block.
416	Touch probe is not possible for the currently selected process data mapping.
417	Internal error / TP channel not free: TP channel has already been activated via another block.
418	Internal error: Timeout during command to be executed of the drive
420	Servo Drive 9400: Incorrect PDO mapping, no touch probe data contained.
421	ECSxM: Selected channel is not supported, only channel "1" permissible
422	ECSxM: Selected mode is not supported, only "Single measurement" permissible
423	ECSxM: Selected edge evaluation is not supported, only "Triggering with positive edge" permissible

12.2 Function blocks for EtherCAT

12.2.1 L_SMC_SetTorqueLimit

The torque limit values of a drive in positive and in negative direction of rotation can be defined by the control. So far, the limit values were set via SDO access. For controlling the drives via an EtherCAT fieldbus, it is possible to transfer these values as a part of the cyclic process data. The advantage of this is that the values can be transmitted to the drive in a considerably quicker way. In order to be able to make use of this advantage, a corresponding mapping must be selected in the EtherCAT configurator, which contains these torque limit values as process data.

The torque limit values are accessed from the PLC via the function block L_SMC_SetTorqueLimit:



Inputs (VAR_IN)

Execute : BOOL

Activates the function.

SetTorqueLimitPos: BOOL

Orders a positive torque limit value.

TorqueLimitPos_Value: LREAL

Positive torque limit value in the range of 0 ... 100 % of the rated torque

SetTorqueLimitNeg: BOOL

Orders a negative torque limit value.

TorqueLimitNeg_Value: LREAL

Negative torque limit value in the range of 0 ... 100 % of the rated torque

Axis : AXIS_REF

Here the axis structure to which these torque limit values apply is transferred.

Outputs (VAR_OUT)

bDone : BOOL

TRUE indicates that the selection has been acknowledged.

bError : BOOL

TRUE indicates that an error has occurred in the function block.

nErrorID SMC_Error: INT

Error number

Error messages

The following error messages are stored:

Error message	Description
ERROR_NO_LENZE_AXIS_REF: INT := 16#200	Function is not supported by this drive.
ERROR_TORQUE_LIMIT_NOT_VALID: INT := 16#210	Value range exceeded. Only torque limit values from 0 ... 200 % are permissible.

► Example 1: The PLC wants to set the torque limit value positively to 50 %.

```
SetTorqueLimitPos = TRUE  
TorqueLimitPos_Value: 50.0  
SetTorqueLimitNeg = FALSE  
TorqueLimitNeg_Value = xxx  
Execute = True
```

The drive sets the positive torque limit value to 50 % of the rated torque; the negative torque limit value that is valid within the drive is not changed.

► Example 2: The PLC wants to set both torque limit values to 100 %.

```
SetTorqueLimitPos = TRUE  
TorqueLimitPos_Value: 100.0  
SetTorqueLimitNeg = TRUE  
TorqueLimitNeg_Value = 100.0  
Execute = True
```

The drive sets both torque limit values to 100 % of the rated torque.

12.2.2 L_SMC_AbortTrigger



Note

This block currently only supports EtherCAT axes.



Inputs (VAR_INPUT)

Execute : BOOL

By setting a new positive edge, all outputs of the block are reset to the initial values.

Axis : AXIS_REF

Selection of the drive

TriggerInput: L_SMC_TRIGGER_REF

Structure for configuring the measurement

Outputs (VAR_OUTPUT)

Done : BOOL

Function block L_SMC_TouchProbe was aborted.

Error : BOOL

Error identification: An error during the command to be executed has occurred.

ErrorId: INT

Error number

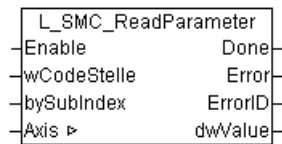
Error messages

The following error messages are stored:

Error number [hex]	Description
0x0001	Error in the command to be executed of the block: • Input data do not correspond to the activated TP command to be executed on this channel. If no request is active on the TP channel, no error message is generated. • TouchProbeSelect > 1 for ECSxM • TouchProbeSelect > 2 for Servo Drive 9400

12.2.3 L_SMC_ReadParameter

This block serves to read out codes of a Motion drive. The code and subindex are specified directly at the block. The conversion to the corresponding parameter number is carried out internally.



Inputs (VAR_INPUT)

Enable : BOOL

Activates the function.

wCodeStelle: WORD

Specification of the code that is to be read.

bySubIndex: BYTE

Specification of the subindex that is to be read.

Axis : AXIS_REF

Here the axis structure for which the code is to be read is transferred.

Outputs (VAR_OUTPUT)

Done : BOOL

TRUE indicates that the selection has been acknowledged.

Error : BOOL

TRUE indicates that an error has occurred in the function block.

ErrorID: INT

Error number

dwValue : DWORD

Value read

Error messages

The following error messages are stored:

SMC_RP_PARAM_UNKNOWN:=130

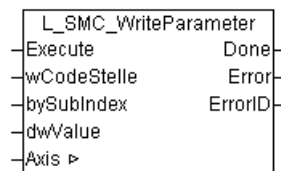
Unknown code

SMC_RP_REQUESTING_ERROR:=131

Read request was aborted, e.g. because the drive has not responded within the time specified.

12.2.4 L_SMC_WriteParameter

By means of this block, codes of a Motion drive are written. The code and subindex are specified directly at the block. The conversion to the corresponding parameter number is carried out internally.



Inputs (VAR_INPUT)

Execute : BOOL

Activates the function.

wCodeStelle: WORD

Specification of the code which is to be written to.

bySubIndex: BYTE

Specification of the subindex which is to be written to.

dwValue : DWORD

Value which is to be written.

Axis : AXIS_REF

Here the axis structure for which the code is to be written is transferred.

Outputs (VAR_OUTPUT)

Done : BOOL

TRUE indicates that the selection has been acknowledged.

Error : BOOL

TRUE indicates that an error has occurred in the function block.

ErrorID: INT

Error number

Error messages

The following error messages are stored:

SMC_WP_PARAM_UNKNOWN:=140

Unknown code

SMC_WP_SENDING_ERROR:=141

Write request was aborted, e.g. because the drive has not responded within the time specified.

13 SoftMotion programming examples

To control a drive hardware with a »PLC Designer« project via the SoftMotion function you have to set up and commission a **CANopen (Motion) bus**:

Step	Task	Lenze software to be used
1.	Create PLC program	»PLC Designer«
2.	Create Motiontask	»PLC Designer«
3.	Create control configuration	»Engineer« or »Global Drive Control« (depending on the device used)
4.	Commission field devices	»PLC Designer«
5.	Create program code for controlling the Motion drives	»PLC Designer«
6.	Prepare restart	»PLC Designer«
7.	Optimisation of the signal propagation delay (optionally)	»PLC Designer«



Detailed information on the commissioning of the **CANopen (Motion) bus** can be found in the "**CANopen control technology**" communication manual.

Programming examples

- ▶ [Single-axis motion control in ST](#) (📖 268)
- ▶ [Single-axis motion control in CFC with visualisation template](#) (📖 270)
- ▶ [Cam drive control with the aid of a virtual time axis](#) (📖 272)
- ▶ [Alternating cams](#) (📖 274)
- ▶ [Drive control with the aid of the CNC editor](#) (📖 275)
- ▶ [Direct creation of the OutQueue](#) (📖 275)
- ▶ [Online decoding with the use of variables](#) (📖 278)
- ▶ [Online path preprocessing](#) (📖 281)
- ▶ [Dynamic SoftMotion programming](#) (📖 283)

13.1 Single-axis motion control in ST



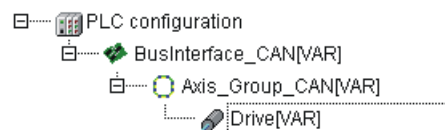
Tip!

See also the example project *PLCopenSingle.pro* provided with SoftMotion.

This example shows how a drive can be controlled by means of the function blocks that have been standardised in accordance with PLCopen.

The SM_PLCopen.lib library must also be linked to the project as well as the Drive Interface libraries.

A linear drive named "Drive" is defined in the PLC configuration:



In the task configuration, call the "Ipo" program, which generates a motion on this axis.

This program must then be created as follows, whereby a program in ST language is set up and populated in the Object Organiser:

Before you can start to move the drive, you must ensure that it has been found and initialised by the driver. If this is the case, you must set the controller enable for the drive and, if applicable, release the brake. This function is performed by the MC_Power program organisation unit:

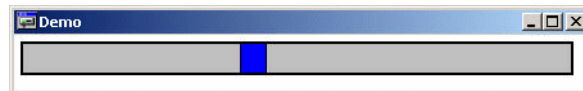
```
PROGRAM Ipo
VAR
    Init: BOOL := FALSE;
    Power: MC_Power;
END_VAR
IF NOT Init THEN
    Power(Enable:=TRUE, bRegulatorOn:=TRUE, bDriveStart:=TRUE, Axis:=Drive);
    Init:= Power.Status;
ELSE
END_IF
```

You can now control the drive in the ELSE branch of the first IF condition. In this example, this should be done using the MC_MoveAbsolute positioning POU. First, declare an instance of this module as local variable named MoveAbsolute and a target position p of type REAL, which will be initialized with "100". Now call this POU instance in each cycle with the required values. When the programmed position has been reached, the program organisation unit's Done output will be set and you must set the Execute input to FALSE if you want to start a new motion, as the program organisation unit will only start a new motion on a rising edge:

```
... (* Continuation of the program shown above *)
ELSE
    MoveAbsolute(Execute:=TRUE, Position:=p, Velocity:=100, Acceleration:=100,
    Deceleration:=100, Axis:=Drive);
    IF MoveAbsolute.Done THEN
        MoveAbsolute(Execute:=FALSE, Axis:=Drive);
    END_IF
END_IF
```

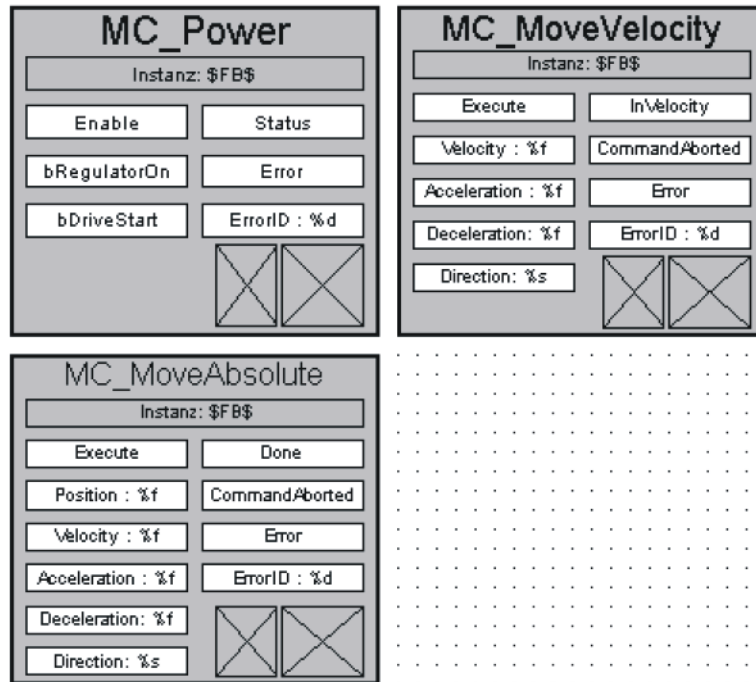
You can now compile the program without errors, go online and start the program. If you are monitoring the Drive.ActPosition actual position with a watch list or the sampling trace, these will show you how the drive is moving towards this position. If you force the value of p, the axis will move towards the new target position as soon as the last one has been reached.

Visualisation templates are available for drives in the SM_DriveBasic.lib library to enable you to view the motion by means of graphics. To use these templates, first go offline, create a new visualisation, insert a visualisation object and select "LinDrive" from the list that appears. Then double-click the newly created object and enter the name of the drive structure (here: Drive) as a replacement under "Visualisation", "Placeholder", "AXISREF". The visualisation configured in this way will display the position of the drive:





Also create an operation visualisation by using the POU templates available in the libraries and connecting them to the POU instances by means of the placeholder concept:



You can now compile the project without errors, log into the control and start the project. Press the Execute input of MoveVelocity to start the drive rotating. Press Execute of MoveAbsolute to position the drive at the set position, where it will only be rotated in the positive direction in accordance with the setting Direction: positive.

This will cause the MoveVelocity program organisation unit to be interrupted. We recommend that you play around with these program organisation units by direct editing of the modules within the visualisation: testing different velocities and accelerations, and also that you test the direction modes (positive/negative/current/shortest/fastest) of MoveAbsolute.

13.3 Cam drive control with the aid of a virtual time axis



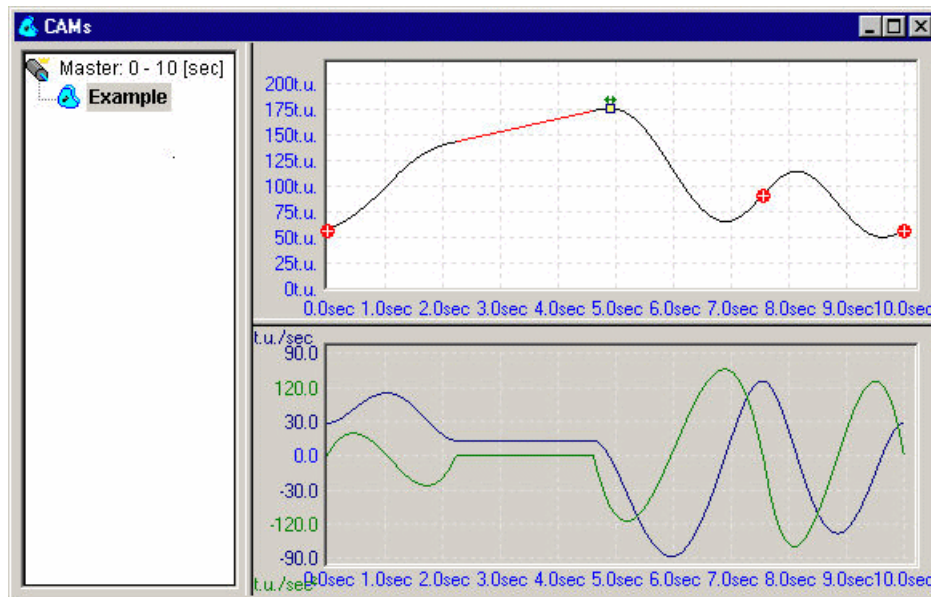
Tip!

See also the example project *PLCopenMulti.pro* provided with SoftMotion.

Precondition: the libraries DriveBasic.lib and SM_PLCopen.lib are incorporated.

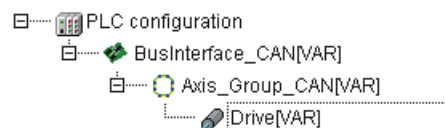
The following example shows how a periodic cam can be implemented on a linear drive. It also demonstrates the tappet function.

1. First create any periodic cam in the "PLC Designer" cam editor. This cam must refer to a master axis covering a range from 0 to 10 seconds and contain at least one inverted tappet with ID1.



The screenshot shows the 'Tappet properties' dialog box. The 'Activate with:' dropdown is set to 'each pass'. The 'Action:' dropdown is set to 'on'. The 'Group ID:' is set to '1'. The 'Delay [μs]:' is set to '0'. The 'Duration [μs]:' is set to '0'. The 'Master position:' is set to '41.333'. The 'Slave position:' is set to '180.000'. The 'OK', 'Apply', and 'Cancel' buttons are visible on the right side.

2. A linear drive named "Drive" is defined in the Drive Interface (PLC configuration):



3. The "Ipo" program is created in FBD and contains the calls of the program organisation units below:

```
PROGRAM Ipo
VAR
    Power: MC_Power;
    TimeAxis : SMC_TimeAxisFB;
    TableSelect: MC_CamTableSelect := (SlaveAbsolute:=TRUE);
    CamIn: MC_CamIn:=(StartMode:=ramp_in, VelocityDiff:=100, Acceleration:=100,
    Deceleration:=100);
    Tappet: SMC_GetTappetValue;
END_VAR
```

After the Power program organisation unit for the slave axis, the first program organisation unit to be called will be TimeAxis. A period (fPeriod) of 10 seconds is specified for this POU, as the cam is also configured for this time. The task cycle time must be entered manually. TableSelect will select the required cam and CamIn will implement it. The Tappet program organisation unit checks the position of the tappet. As the tappet has been programmed "inverted", it will switch every 10 seconds.

The program can now be compiled and started on the control.

To check the set or actual position, create another visualisation, which you can use to check the individual program organisation units and the position of the axes.

Please note that as well as being a virtual time axis, the cam master can also be any AXIS_REF data structure. Setpoints are taken into account for drives that are being controlled, with actual values being taken into account for drives that are not being controlled.

13.4 Alternating cams



Tip!

See also the provided example project *PLCopenMultiCAM.pro*.

Precondition: the libraries DriveBasic.lib and SM_PLCopen.lib are incorporated.

This example shows how a cam motion can be realised with two alternating cams. It has been programmed in ST and executes the same actions as the previous example. At the end of the first cam, the MC_CamIn program organisation unit sets the bEndOfProfile output, which will activate the other MC_CamTableSelect program organisation unit and cause MC_CamIn to be restarted.

13.5 Drive control with the aid of the CNC editor

The basic structure of a possible "PLC Designer" IEC program that can implement paths designed in the CNC editor is described below.

As described in the chapter "[CNC editor](#)" (62), there are two options for compiling and using a CNC program. The first example shows the direct creation of an OutQueue, the second involves the use of variables to decode the program online. The third example shows how a path preprocessing POU can also be used online.

13.5.1 Direct creation of the OutQueue

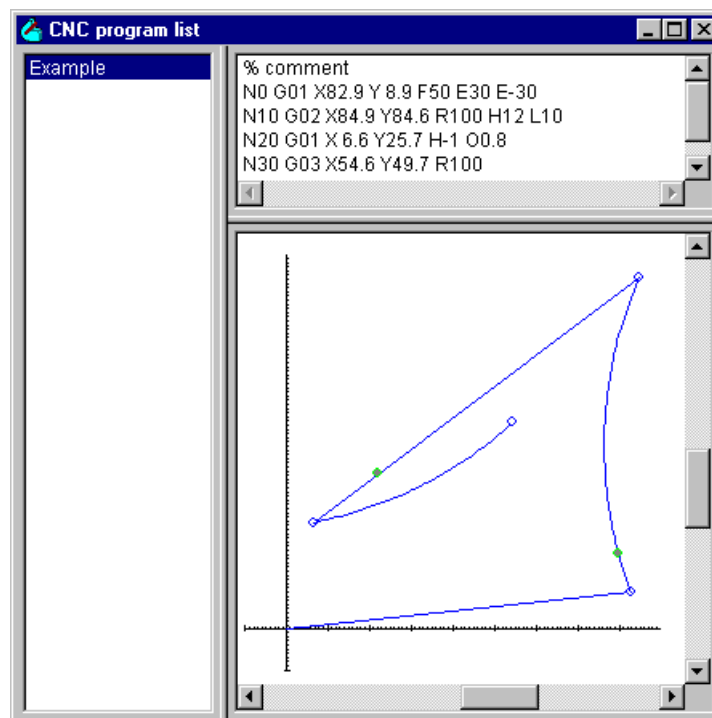


Tip!

See also the example project *CNCdirect.pro* provided with SoftMotion.

1. Creating the CNC program in the CNC editor:

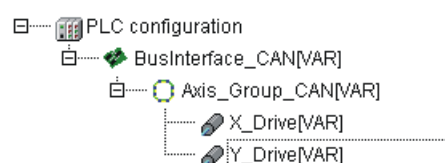
- As described in 0, create a sample program that runs between x out of [0,100] and y out of [0,100]. Also define velocities and accelerations for the path and set two path switch points on the path.



Select "Create OutQueue on compile" as the compilation mode.

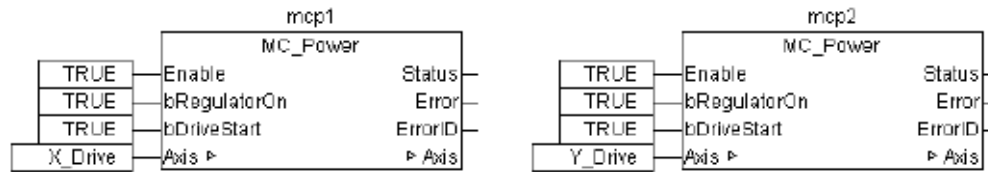
2. Drive Interface, PLC configuration:

- Define a drive structure with two linear drives; the maximum velocity, etc. must also be set.

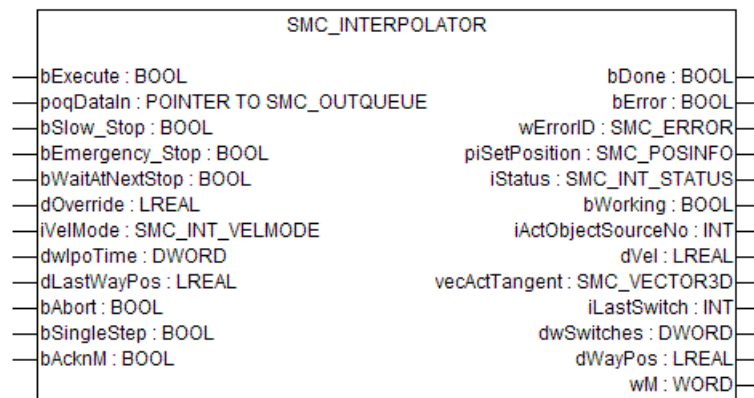


3. Creating the IEC program IPO (CFC):

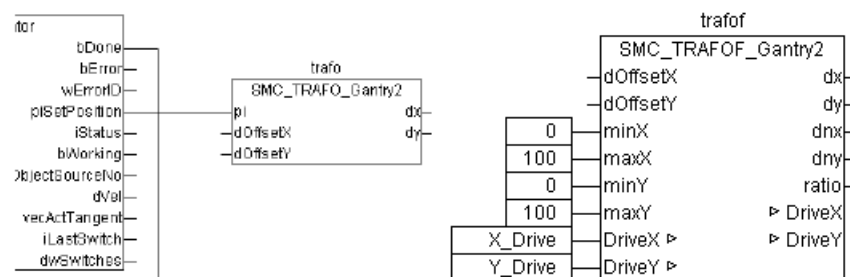
- You first have to activate the drives via the MC_Power program organisation unit.



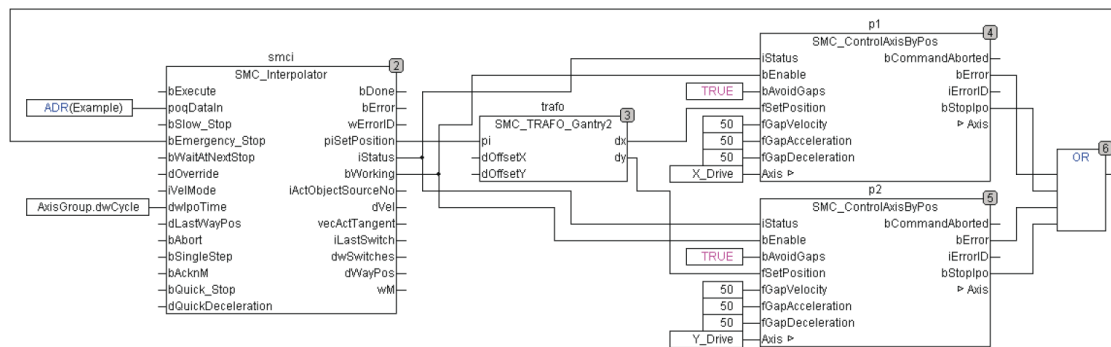
- Another essential element is the SMC_Interpolator program organisation unit. This receives the address of the created CNC program as its poqDataIn input. The IEC task cycle also has to be written to dwlpoTime. You can either enter this as a constant value at the dwlpoTime input or use the dwCycle variable of the axis group structure from the PLC configuration. The latter has the advantage that, if you change the task cycle time, the correct time will be used for the interpolator input automatically.



- In this example, you want to control a gantry system. For this reason, insert an instance of the forwards and reverse transformation POU's from the SM_Trafo.lib library. The forwards transformation POU receives the drives as its inputs (the Z drive is assigned the dummy variable of type AXIS_REF, which is not otherwise used); the reverse transformation POU must receive the set position of the interpolator.



- The outputs of the program organisation unit, i.e. the axis coordinates, now have to be written to the drives, which is what the SMC_ControlAxisByPos function blocks are for. As this application does not guarantee that the interpolator outputs will be continuous (e.g. the path may end at a different point to where it starts), you must activate functions that avoid gaps (bAvoidGaps, fGapVelocity, fGapAcceleration, fGapDeceleration), connect the Stoplpo output to the interpolator's bEmergency_Stop and connect the iStatus interpolator output to the corresponding inputs of the axis control program organisation units.



- Please note that the program organisation units must be in the correct order, particularly when programming with CFC.

4. Creating the operation and test interface:

- Create a new visualisation and insert two visualisation objects (the interpolator template and the transformation template) into it. These are connected to the corresponding POU instances (here: lpo.smci or lpo.trafof) by means of the placeholder concept.

5. Commissioning:

- The program created in this way can be compiled without errors and started. It will execute the CNC program as soon the Execute input of the interpolator is set. Once it has been processed completely, a new rising edge will cause the program to run again.
- Please also note the function of the path switches, which are also displayed in the visualisation of the interpolator program organisation unit.

13.5.2 Online decoding with the use of variables



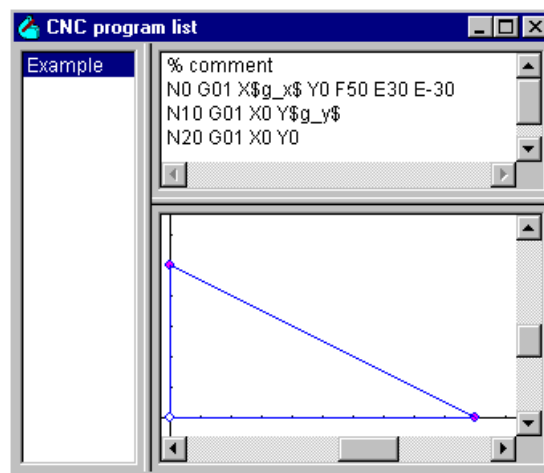
Tip!

See also the example project *CNConline.pro* provided with SoftMotion.

6. Creating the CNC program in the CNC editor:

- As in the previous example, we create a CNC program and use two global variables `g_x` and `g_y` for it, e. g.:

```
VAR_GLOBAL  
  g_x: REAL:=100;  
  g_y: REAL:=50;  
END_VAR
```



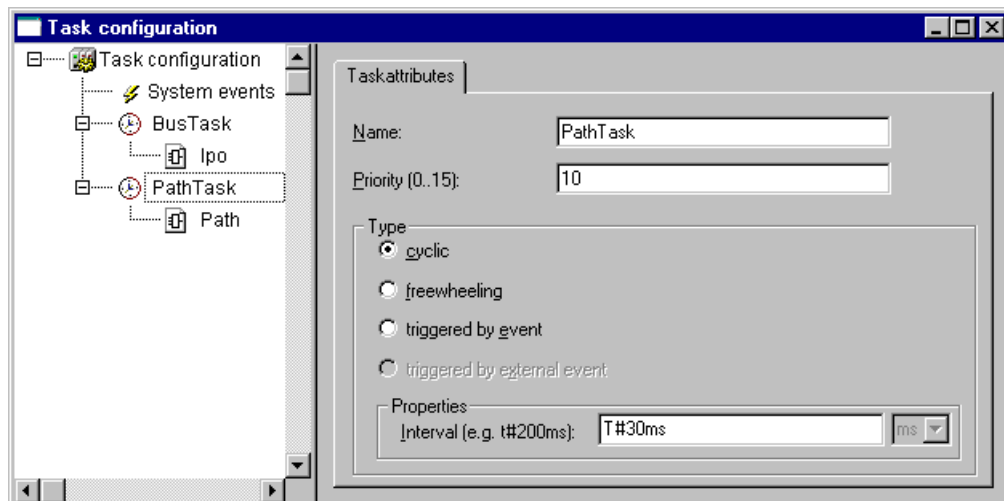
- As compilation mode, however, this time we select "Generate program variable during compilation", as we use variables in our program.

7. Drive Interface, PLC configuration:

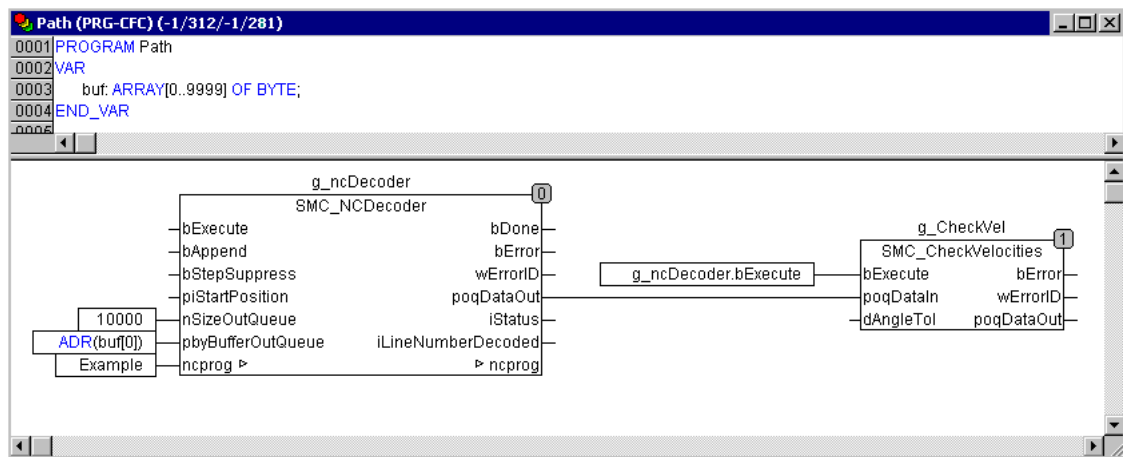
- The drive structure corresponds to that of the previous example.

8. Creating the IEC program:

- As we have selected the other compilation mode this time, we have to carry out the decoding and path preprocessing in the IEC program. As this process which is rather runtime-consuming does not have to be effected in the interpolator cycle (at every decoder call one path object is generated, which typically lasts for many interpolator calls), it is often swapped out into a task with a lower priority that is called less frequently:



- The mechanism behind this works as follows: Initially approximately one GEOINFO object per cycle is generated within the slow task, which is saved in the OUTQUEUE structure of the decoder block. If this OUTQUEUE is full, the blocks of the slow task pause until the OUTQUEUE is no longer full, i. e. until the quick task has processed the first GEOINFO object and deletes it from the OUTQUEUE. Then the blocks of the slow task are reactivated and fill the OUTQUEUE structure. In the quick task one path position from the OUTQUEUE structure which the input DataIn of the interpolator points to is calculated and processed per cycle. As a GEOINFO object usually consists of several path positions, it takes several cycles until the first GEOINFO object is processed and is deleted automatically by the interpolator. As the processing of a GEOINFO object obviously requires more cycles than the creation, the slow task actually can be called less frequently than the quick one. However, the task times have to be selected in a way that in the last OUTQUEUE of the slow task always a sufficient number of GEOINFO objects is stored, so that there is no data underrun. This occurs if no GEOINFO objects are provided to the interpolator from DataIn and the path end has not yet been reached. In this case the interpolator brakes and remains at standstill until new data elements are available again.
- In the Path program the decoding of the CNC program for OutQueue and the speed check take place:



- The interpolating part of the IEC program nearly remains identical, only with the difference that the data input of the interpolator does not correspond to the CNC program name (ADR(Example)) like before, but consists of the OutQueue output of the path preprocessing blocks (g_CheckVel.poqDataOut).

9. Creating the operation and test interface:

- For visualising the previous example it is reasonable to add templates of the new blocks (SMC_NCDecoder and SMC_CheckVelocities). Furthermore it should be possible to alter the global variables g_x and g_y, in order to be able to check their function during commissioning later on.

10. Commissioning:

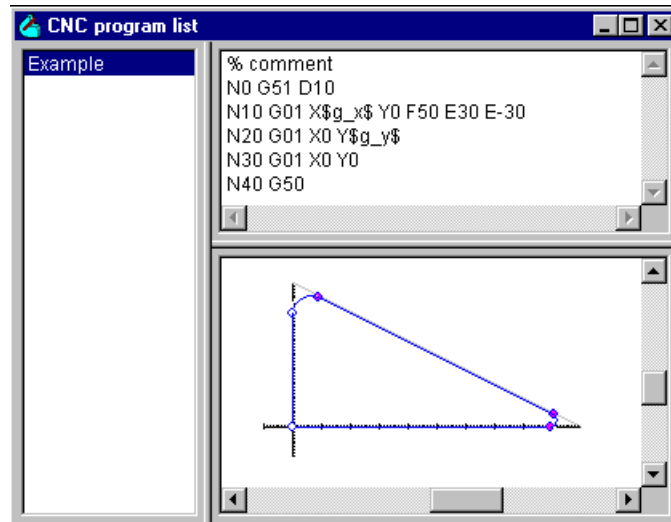
- The program created can be compiled and started in an error-free manner and executes the CNC program after the execute inputs of the decoder and the interpolator have been set. If the values of the global variables are changed, they are accepted when the decoder is restarted, and the path adapts itself accordingly. Please also note the function of the append input of the decoder.

13.5.3 Online path preprocessing

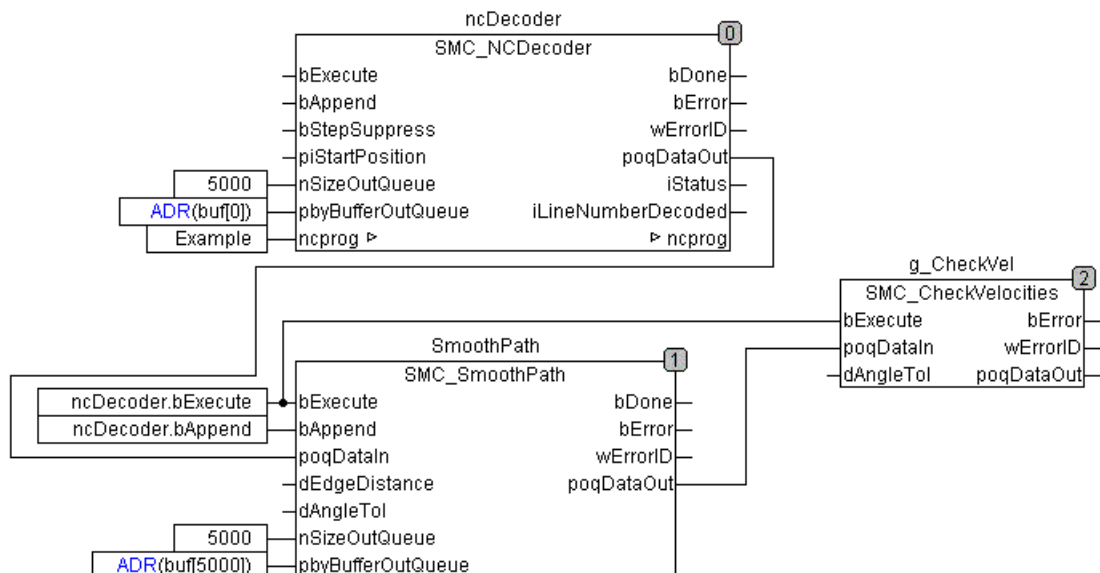
**Tip!**

See also the example project *CNCprepro.pro* provided with SoftMotion.

Let us extend the previous example by a path preprocessing function: The corners of the last program are to be smoothed by splines. This is carried out by the SMC_SmoothPath block. The CNC program has to be bracketed by the words G51/G50, then the following results:



If we did not use variables, we could compile the program as Queue in this form and directly enter it in the interpolator; however, as we are using variables, we have to carry out the decoding and the smoothing of the corners ourselves. Therefore we define a new block of type SMC_SmoothPath and call it after the decoder:



As always, the data input of the interpolator block has to be set to the output poqDataOut of the CheckVelocities block.

This program can be compiled in an error-free manner and, in contrast to the previous program, does no longer stop in the reference points of the CNC program, as the reference points of the path have been rid of bends by the path preprocessing.

13.6 Dynamic SoftMotion programming



Tip!

See also the example project *CNCDynamicPath.pro* provided with SoftMotion.

It is one of the advantages of SoftMotion that the programmer and user do not only have an impact on the processing of a path, but that they can also generate or alter the processing during the program is running. In order to achieve this, the programmer just has to **replace the decoder block by an individual path generator**, and he can nevertheless operate the path preprocessing, and in particular the interpolator as usual.

To replace the decoder block an **OUTQUEUE structure object has to be created** in another way. It has to be filled with GEOINFO objects representing the desired path and transmitted to the corresponding subsequent block (e.g. interpolator).

13.6.1 Preliminary steps

- In the declaration part an OUTQUEUE and a GEOINFO structure object, and a buffer of a desired size have to be defined:

```
QUEUE: SMC_OUTQUEUE;
BUF: ARRAY[0..49] OF SMC_GEOINFO;
GEO: SMC_GEOINFO:=(dT1:=0, dT2:=1, dToolRadius:=0, dVel:=100, dVel_End:=100,
dAccel:=200, dDecel:=500, iObj_Nr:=0);
```

- In an Init-step in the program body the OUTQUEUE structure has to be initialised:

```
QUEUE.nSize := SIZEOF(BUF);
QUEUE.pbyBuffer := ADR(BUF[0]);
```

13.6.2 Dynamic Path-Programming

In the program body the following steps have to be carried out for each GEOINFO object in the position where the path is to be created:

- Set start position (first object)

```
GEO.piStartPos.dX := 0;
```

... or copy from the previous object.

```
GEO.piStartPos := GEO.piDestPos;
```

- Define motion type, e.g.:

```
GEO.iMoveType := CCWL;
```

or

```
GEO.iMoveType := LIN;
```

- Set parameters corresponding to the motion type. If you use a circular arc (e.g. CCWL), observe that the following positions have to be set (see SMC_GEOINFO structure:

```
GEO.dP1 := 200;  
GEO.dP2 := 100;  
GEO.dP3 := 50;  
GEO.dT1 := 0;  
GEO.dT2 := 90;
```

- If required, set a start or end bit in InternMark for the path preprocessing (see SMC_GEOINFO structure).
- Calculate end position:

```
SMC_CalcEndPnt(ADR(GEO));
```

- Calculate length of the object (or have it calculated):

```
SMC_CalcLengthGeo(ADR(GEO));
```

- Save object in OUTQUEUE:

```
SMC_AppendObj(POQ:=ADR(QUEUE), PGI:=ADR(GEO));
```

- If the path has been created completely, the OUTQUEUE list has to be closed:

```
QUEUE.bEndOfList := TRUE;
```

Please note that if OUTQUEUE is full, i.e. if QUEUE.bFULL = TRUE, the program will not try to add further objects. The creation of the path then has to be interrupted until the first OUTQUEUE object has been processed. Then a further object can be added. If you want to avoid this, the selected size for the OUTQUEUE has to be so great that all GEOINFO objects of the desired path can be stored there.

The Queue object list first is transmitted to the CheckVelocities block and then to the interpolator which processes it further.

Furthermore this example shows how a kinematic transformation that is not contained in the SM_Trafo.lib library provided can be individually programmed. The blocks SMC_TRAFO and SMC_TRAFOF which are included in this project show this by taking a Cartesian X/Y system as an example.

14 Index

0-9

2,5-D interpolation process [199](#)

3-D interpolation process [199](#)

A

acit [50](#)

Additional function [68](#)

Altering the display of the curve [88](#)

Analysis blocks for SMC_CNC_REF data [215](#)

Analysis blocks for SMC_OutQueue data [215](#)

Angular tolerance 45° [74](#)

Application notes (depiction) [13](#)

Automatic code generation [26](#)

Automatic structure filling in the CNC editor [83](#)

Auxiliary functions for path rotations, translations and scalings [200](#)

Avoid Loop [81](#)

Axis group blocks [27](#)

Axis_Group_CAN [20](#)

Axis_Group_Dummy [20](#)

Axis_IO_Group [20](#)

AXIS_REF [56](#)

AXIS_REF structure [56](#)

AxisGroup [20](#)

B

bHWLimitsActive [50](#)

bOldReference [50](#)

bSWLimitEnable [25](#)

BusInterface [19](#)

BusInterface_CAN [19](#)

BusInterface_Dummy [19](#)

BusInterface_EtherCAT [19](#)

byActOpMode [50](#)

byDigitalInputs [50](#)

byDigitalOutputs [50](#)

byDriveState [50](#)

byErrAcknCounter [50](#)

bySetOpMode [50](#)

byStatusInfo [50](#)

C

Cam

Generating the general settings [88](#)

Cam (definition) [84](#)

Cam as table [88](#)

Cam data structures [103](#)

CAM editor [84](#)

Commands in the "Extras" menu [92](#)

Commands in the "Insert" menu [95](#)

CAM function blocks [254](#)

Cam tree [87](#)

Cam, name [85](#)

Cam, properties [86](#)

Cam, type [85](#)

Cams as table [88](#)

CAN communication [53](#)

CAN module parameters [21](#)

CANDrive [22](#)

CANDrive dialog [25](#)

CANDrive ID [22](#)

Changing variable values (CNC editor) [71](#)

Circle CCW Insert Mode [80](#)

Circle CW Insert Mode [79](#)

Circular interpolation [69](#)

CNC editor [62](#)

Automatic structure filling [83](#)

Changing variable values [71](#)

Commands in the "Extras" menu [79](#)

Use of variables [68](#)

CNC function blocks [250](#)

CNC graphic editor [78](#)

CNC libraries [15](#)

CNC program information [73](#)

CNC text editor [77](#)

Code generation, automatic [26](#)

Commands (travel commands) [66](#)

Commands in the "Extras" menu (CAM editor) [92](#)

Commands in the "Extras" menu (CNC editor) [79](#)

Commands in the "Insert" menu (CAM editor) [95](#)

Conditional jumps [70](#)

Configuration blocks [31](#)

Configured parameters during start-up [52](#)

Control configuration for SoftMotion [18](#)

Control mode blocks [32](#)

Conventions used [12](#)

Conversion factor [22](#)

Convert Splines/Ellipses to Lines [80](#)

Copyright [2](#)

crap [50](#)

Creating a cam [85](#)

Creating a CNC program [72](#)

Cycle time measurement [27](#)

D

D [65](#)

Define Queue Size [73](#)

Defining the angular tolerance for stopping in the CNC program [74](#)

Defining the start position of the CNC program [73](#)

Definition of a cam for SoftMotion [84](#)

Deleting a CNC program [73](#)

Diagnostic blocks [43](#)

Diagnostic function blocks [255](#)

[Direct setpoint selection](#) [33](#)
[Displaying the acceleration](#) [88](#)
[Displaying the speed](#) [88](#)
[Document history](#) [11](#)
[Drive](#) [22](#)
[Drive element Drive.lib](#) [48](#)
[Drive ID](#) [22](#)
[Drive Interface](#) [17](#)
[Drive.wControlType](#) [48](#)
[dwCycle](#) [21](#)
[dwParam1](#) [21](#)
[dwParam2](#) [21](#)
[dwParam3](#) [21](#)
[dwParam4](#) [21](#)
[dwRatioTechUnitsDenom](#) [25](#)
[dwSetPosition](#) [50](#)

E

[E](#) [65](#)
[Editing a cam](#) [88](#), [90](#)
[Editing a point / line](#) [89](#)
[Editing properties of individual cam elements](#) [89](#)
[Editor](#)
 [Cam Element Properties](#) [97](#)
[Ellipse interpolation](#) [70](#)
[Encoder](#) [45](#)
[eRampType](#) [25](#)
[Error library](#) [16](#)
[Error numbers](#) [245](#)
[Export Cam as ASCII Table](#) [94](#)

F

[F](#) [65](#)
[File service library](#) [16](#)
[Fit to Screen](#) [80](#)
[fLimitSwitchDeceleration](#) [50](#)
[fPositionPeriod](#) [25](#)
[fRampJerk](#) [25](#)
[fSWLimitNegative](#) [25](#)
[fSWLimitPositive](#) [25](#)
[fSWMaxAcceleration](#) [25](#)
[fSWMaxDeceleration](#) [25](#)
[fSWMaxVelocity](#) [25](#)
[Function blocks for path rotations, translations and scalings](#) [200](#)
[Function blocks for spatial transformation processes](#) [194](#)

G

[G](#) [65](#)
[g_udiTimeDistanceSyncSend](#) [49](#)
[G00](#) [66](#)
[G01](#) [66](#)
[G02](#) [66](#)

[G03](#) [66](#)
[G05](#) [66](#)
[G06](#) [66](#)
[G08](#) [66](#)
[G09](#) [66](#)
[G20](#) [66](#)
[G36](#) [66](#)
[G37](#) [66](#)
[G40](#) [66](#)
[G41](#) [66](#)
[G42](#) [66](#)
[G50](#) [66](#)
[G51](#) [66](#)
[G52](#) [66](#)
[G60](#) [66](#)
[G61](#) [66](#)
[G90](#) [66](#)
[G91](#) [66](#)
[G92](#) [66](#)
[G98](#) [66](#)
[G99](#) [66](#)

[Gantry systems](#) [216](#)
[Gantry systems with tool offset](#) [222](#)
[GantryCutter](#) [221](#)
[General cam editor settings](#) [88](#)

H

[H](#) [65](#)
[H gantry system with stationary drives](#) [228](#)
[Hardware limit switch \(ECSxM\)](#) [51](#)
[H-function](#) [67](#)
[Homing via digital hardware inputs](#) [41](#)
[How POU parameters work](#) [97](#)

I

[I](#) [65](#)
[iMovementType](#) [25](#)
[Import Cam from ASCII Table](#) [94](#)
[Import DXF File](#) [75](#)
[Insert cam](#) [88](#), [95](#)
[Insert line](#) [88](#), [95](#)
[Insert point](#) [88](#), [95](#)
[Inserting and managing CNC programs](#) [72](#)
[Interpolation process](#) [199](#)
[Invert Direction](#) [74](#)
[iRatioTechUnitsNum](#) [25](#)

J

[J](#) [65](#)
[Jerk](#) [23](#)

K

[K](#) [65](#)

L

[L](#) [65](#)
[L_SMC_AbortTrigger](#) [264](#)
[L_SMC_DriveUtil.lib](#) [257](#)
[L_SMC_DriveUtil.lib library](#) [257](#)
[L_SMC_ReadParameter](#) [265](#)
[L_SMC_SetTorqueLimit](#) [262](#)
[L_SMC_TouchProbe](#) [257](#)
[L_SMC_Trigger_REF](#) [258](#)
[L_SMC_WriteParameter](#) [266](#)
[Lenze specifications](#) [257](#)
[LenzeCANdrive.lib](#) [48](#)
[LenzeECS_AXIS_REF](#) [50](#)
[Liability](#) [11](#)
[LinDrive](#) [47](#)
[LinDrive_V](#) [47](#)
[Line Insert Mode](#) [79](#)

M

[M](#) [65](#)
[Manually generated cam \(example\)](#) [106](#)
[Mathematical auxiliary blocks](#) [26](#)
[MC_AbortTrigger](#) [145](#)
[MC_AccelerationProfile](#) [140](#)
[MC_CAM_REF](#) [103](#)
[MC_CamIn](#) [98](#), [148](#)
[MC_CamOut](#) [150](#)
[MC_CamTableSelect](#) [97](#), [146](#)
[MC_DigitalCamSwitch](#) [164](#)
[MC_GearIn](#) [150](#)
[MC_GearInPos](#) [152](#)
[MC_GearOut](#) [151](#)
[MC_Halt](#) [116](#)
[MC_Home](#) [114](#)
[MC_MoveAbsolute](#) [118](#)
[MC_MoveAdditive](#) [123](#)
[MC_MoveRelative](#) [126](#)
[MC_MoveSuperImposed](#) [130](#)
[MC_MoveVelocity](#) [134](#)
[MC_Phasing](#) [156](#)
[MC_PositionProfile](#) [136](#)
[MC_Power](#) [113](#)
[MC_ReadActualPosition](#) [112](#)
[MC_ReadActualTorque](#) [112](#)
[MC_ReadActualVelocity](#) [112](#)
[MC_ReadAxisError](#) [110](#)
[MC_ReadBoolParameter](#) [111](#)
[MC_ReadParameter](#) [111](#)
[MC_ReadStatus](#) [110](#)
[MC_Reset](#) [110](#)
[MC_SetPosition](#) [141](#)
[MC_Stop](#) [115](#)
[MC_TouchProbe](#) [142](#)

[MC_VelocityProfile](#) [139](#)
[MC_WriteBoolParameter](#) [111](#)
[MC_WriteParameter](#) [111](#)
[mcstopLS](#) [50](#)
[M-function](#) [68](#)
[Module parameters" dialog](#) [25](#)
[Module parameters, dialog](#) [25](#)

N

[Name of cam](#) [85](#)

O

[O](#) [65](#)
[Online mode of the cam editor](#) [96](#)

P

[P](#) [65](#)
[Parabolic speed profile](#) [23](#)
[Parallel kinematics](#) [235](#)
[Parameterisation of the drive](#) [60](#)
[Parameters in the PLC configuration](#) [19](#), [21](#)
[Path cams with SMC_XInterpolator](#) [212](#)
[PLCopen library](#) [16](#)
[PLCopen specification](#) [108](#)
[Portability](#) [16](#)
[POUs for the motion control of individual axes](#) [110](#)
[POUs for the synchronised motion control \(multi-axis\)](#) [146](#)
[pParameterlist](#) [50](#)
[Properties of the cam](#) [86](#)

Q

[Q](#) [65](#)

R

[R](#) [65](#)
[Ramp modes](#) [23](#)
[Read Cam from File](#) [93](#)
[Read CNC Program from File](#) [75](#)
[References used](#) [13](#)
[Rename CNC Program](#) [72](#)
[Renumber Program](#) [80](#)
[Rotating the CNC program](#) [74](#)
[RotDrive](#) [47](#)
[RotDrive_V](#) [47](#)
[Round Path](#) [81](#)
[rp](#) [50](#)
[rxList](#) [50](#)
[rxPDOs](#) [50](#)

S

[S](#) [65](#)
[Safety instructions \(depiction\)](#) [13](#)
[Scaling master axis](#) [86](#)

- Scaling slave axis [86](#)
- Select element [88](#), [95](#)
- Select Mode [79](#)
- Selecting and shifting in the CAM editor window [91](#)
- Set epsilon values for zero [82](#)
- Setpoint selection, direct [33](#)
- Settings [92](#)
- Settings via global variables [201](#)
- Shifting a CNC program [74](#)
- Show Bounds [92](#)
- Show Complete Cam [92](#)
- Show Interpolation Points [82](#)
- Sigmoidal speed profile [23](#)
- SM_CAN.lib [54](#)
- SM_CNC.lib [168](#)
- SM_CNC.lib library [168](#)
- SM_CNC.lib structures [202](#)
- SM_CNCDiagnostic.lib [215](#)
- SM_CNCDiagnostic.lib library [215](#)
- SM_DriveBasic.lib [26](#)
- SM_Error.lib [244](#)
- SM_Error.lib library [244](#)
- SM_FileFBs.lib [250](#)
- SM_FileFBs.lib library [250](#)
- SM_PLCopen.lib [107](#)
- SM_PLCopen.lib library [107](#)
- SM_Trafo.lib [216](#)
- SM_Trafo.lib library [216](#)
- SMC_atan2 [26](#)
- SMC_AvoidLoop [176](#)
- SMC_AxisDiagnosticLog [255](#)
- SMC_CalcDirectionFromVector [223](#)
- SMC_CAMBounds [158](#)
- SMC_CAMEditor [159](#)
- SMC_CAMRegister [161](#)
- SMC_CAMTable__1 [105](#)
- SMC_CAMTable__2 [105](#)
- SMC_CAMVisu [159](#)
- SMC_CAMXYVA [105](#)
- SMC_ChangeGearingRatio [31](#)
- SMC_CheckLimits [37](#)
- SMC_CheckVelocities [182](#)
- SMC_CNC_REF data, analysis blocks [215](#)
- SMC_ControlAxisByPos [38](#)
- SMC_ControlAxisByPosVel [39](#)
- SMC_ControlAxisByVel [39](#), [45](#), [257](#), [264](#), [265](#), [266](#)
- SMC_CoordinateTransformation3D [194](#), [239](#)
- SMC_DetermineCuboidBearin [242](#)
- SMC_DetermineCuboidBearing [197](#)
- SMC_DisableAllDrives [30](#)
- SMC_EnableAllDrives [30](#)
- SMC_ENCODER [45](#)
- SMC_Encoder [45](#)
- SMC_Error [245](#)
- SMC_Error enumeration [245](#)
- SMC_ErrorString [244](#)
- SMC_fmod [26](#)
- SMC_FollowPosition [34](#)
- SMC_FollowPositionVelocity [35](#)
- SMC_FollowVelocity [36](#)
- SMC_GCodeViewer [171](#)
- SMC_GEOINFO [203](#)
- SMC_GetAxisGroupState [27](#)
- SMC_GetCamSlaveSetPosition [157](#)
- SMC_GetMaxSetAccDec [43](#)
- SMC_GetMaxSetVelocity [43](#)
- SMC_GetMPParameters [191](#)
- SMC_GetTappetValue [163](#)
- SMC_GetTrackingError [44](#)
- SMC_Homing [41](#)
- SMC_Interpolator [185](#)
- SMC_Interpolator2Dir [192](#)
- SMC_IsAxisGroupReady [27](#)
- SMC_LimitCircularVelocities [183](#)
- SMC_MeasureDistance [44](#)
- SMC_MoveContinuousAbsolute [121](#)
- SMC_MoveContinuousRelative [128](#)
- SMC_NCDecoder [169](#)
- SMC_OUTQUEUE [207](#)
- SMC_OutQueue data, analysis blocks [215](#)
- SMC_POSINFO [202](#)
- SMC_ReadCAM [254](#)
- SMC_ReadCANParameter [54](#)
- SMC_ReadNCFile [251](#)
- SMC_ReadNCQueue [250](#)
- SMC_ReadSetPosition [167](#)
- SMC_ResetAxisGroup [29](#)
- SMC_ROTATEQUEUE2D [200](#)
- SMC_RoundPath [180](#)
- SMC_SCALEQUEUE3D [200](#)
- SMC_SetControllerMode [32](#)
- SMC_SetTorque [167](#)
- SMC_sgn [26](#)
- SMC_ShowCNCREF [215](#)
- SMC_ShowQueue [215](#)
- SMC_SmoothPath [178](#)
- SMC_TeachCoordinateSystem [240](#)
- SMC_TeachCoordinateTransformation [195](#)
- SMC_TimeAxisFB [40](#)
- SMC_ToolCorr [173](#)
- SMC_TRAFO_Gantry2 [217](#)
- SMC_TRAFO_Gantry2Tool1 [224](#)
- SMC_TRAFO_Gantry2Tool2 [226](#)
- SMC_TRAFO_Gantry3 [219](#)
- SMC_TRAFO_GantryH2 [229](#)
- SMC_TRAFO_Scara2 [231](#)

SMC_TRAFO_Scara3 [233](#)
 SMC_TRAFO_Tripod [235](#)
 SMC_TRAFOF_Gantry2 [218](#)
 SMC_TRAFOF_Gantry2Tool1 [225](#)
 SMC_TRAFOF_Gantry2Tool2 [227](#)
 SMC_TRAFOF_Gantry3 [220](#)
 SMC_TRAFOF_GantryH2 [230](#)
 SMC_TRAFOF_Scara2 [232](#)
 SMC_TRAFOF_Scara3 [234](#)
 SMC_TRAFOF_Tripod [236](#)
 SMC_TRAFOV_Gantry [221](#)
 SMC_TRANSLATEQUEUE3D [200](#)
 SMC_UnitVectorToRPY [196](#), [241](#)
 SMC_VARLIST structure [253](#)
 SMC_VECTOR3D [206](#)
 SMC_VECTOR6D [206](#)
 SMC_WriteCAM [254](#)
 SMC_WriteCANParameter [54](#)
 SMC_WriteDriveParamsToFile [61](#)
 SMC_XInterpolator [212](#)
 Smooth Path [81](#)
 SoftMotion concept and overview of the components [14](#)
 SoftMotion Drive Interface [17](#)
 SoftMotion programming examples [267](#)
 Spatial transformation processes (function blocks) [194](#)
 Spatial transformations [238](#)
 Speed profile, parabolic [23](#)
 Speed profile, sigmoidal [23](#)
 Speed profile, trapezoidal [23](#)
 Speed ramp type [23](#)
 Spline Insert Mode [80](#)
 Spline interpolation [70](#)
 Split Object [75](#)
 Starting the CAM editor [85](#)
 Starting the CNC editor [72](#)
 sTask [21](#)
 strConfigFile [50](#)
 Stretch the CNC program [74](#)
 Structure of a CNC program [63](#)
 Structure of LenzeECS_AXIS_REF [50](#)
 Structures of SM_CNC.lib [202](#)
 Suppress Record [81](#)
 Switching between cams [100](#)
 Switching points [67](#)

T

Target group [10](#)
 Three-jointed Scara systems [233](#)
 Tool Radius Correction [81](#)
 Trademarks [11](#)
 Transformation function blocks [216](#)
 Transmission options [92](#)

Trapezoidal speed profile [23](#)
 Travel commands [66](#)
 Two-jointed Scara systems [231](#)
 txList [50](#)
 txPDOs [50](#)

U

U [65](#)
 Use of variables (CNC editor) [68](#)
 Using cams [97](#)

V

V [65](#)
 Validity of the documentation [10](#)
 Variables of the AXIS_REF structure [56](#)
 Virtual time axis [40](#)
 Visualisation templates [47](#)

W

W [65](#)
 wControlType [25](#)
 wControlWord [50](#)
 wCyclicDataR1 [25](#)
 wCyclicDataR2 [25](#)
 wCyclicDataR3 [25](#)
 wCyclicDataS1 [25](#)
 wCyclicDataS2 [25](#)
 wCyclicDataS3 [25](#)
 wDriveId [25](#)
 Word identifiers [65](#)
 wp [50](#)
 wParam1 [21](#)
 wParam2 [21](#)
 wParam3 [21](#)
 wParam4 [21](#)
 Write Cam to File [93](#)
 Write CNC Program to File [75](#)
 Write OutQueue to File [76](#)
 wStatusWord [50](#)
 wStatusWord2 [50](#)
 wStatusWordOld [50](#)

X

X [65](#)

Y

Y [65](#)

Z

Z [65](#)



© 03/2011

Lenze Automation GmbH
Hans-Lenze-Str. 1
D-31855 Aerzen
Germany



+49 (0)51 54 / 82-0



+49 (0)51 54 / 82-28 00



Lenze@Lenze.de



www.Lenze.com

Service Lenze Service GmbH
Breslauer Straße 3
D-32699 Extertal
Germany



00 80 00 / 24 4 68 77 (24 h helpline)



+49 (0)51 54 / 82-11 12



Service@Lenze.de

SHPLCDSM ■ 13373834 ■ EN 5.2 ■ TD17

10 9 8 7 6 5 4 3 2 1