The Technology Objects (TO) of SIMATIC S7-1500(T)

TIA Portal V14 / SIMATIC S7-1500 / SIMATIC S7-1500T

https://support.industry.siemens.com/cs/ww/en/view/109743134

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG (“Siemens”). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens’ products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

Siemens’ products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer’s exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

Table of Contents

1 SIMATIC’s Technology Objects

1.1 Motivation

In order to be able to facilitate the use of technological functions that can be used with a SIMATIC controller, what is known as technology objects have been introduced in the programming environment of SIMATIC. Within these technology objects, the available functions are encapsulated and provided to the creator of the user program for easy access and the easy use in the programming environment.

Particularly in the "motion control” area are these technology objects used to simplify the control and handling of axes and additional motion control functionalities and to support the user in the creation of a user program with motion control functionalities. This is why the emphasis of this document is on displaying technology objects from the area of "motion control”.

1.2 Properties of a technology object (TO)

A technology object (TO) for motion control in the SIMATIC has the following properties:

- The technology object represents a software object in the controller.
- The technology object represents the mechanical components.
- The technology object encapsulates the technological functionality.
- The technology object allows a uniform setting and configuration.
- The technology object ensures a simple connection of the drives and encoders as well as the distributed I/O.
- The technology object encapsulates the mechanical configuration, the monitoring and limitations of the drive and the mechanic that is connected to it.
- The technology object is addressed via PLCopen motion control instructions from the user program.

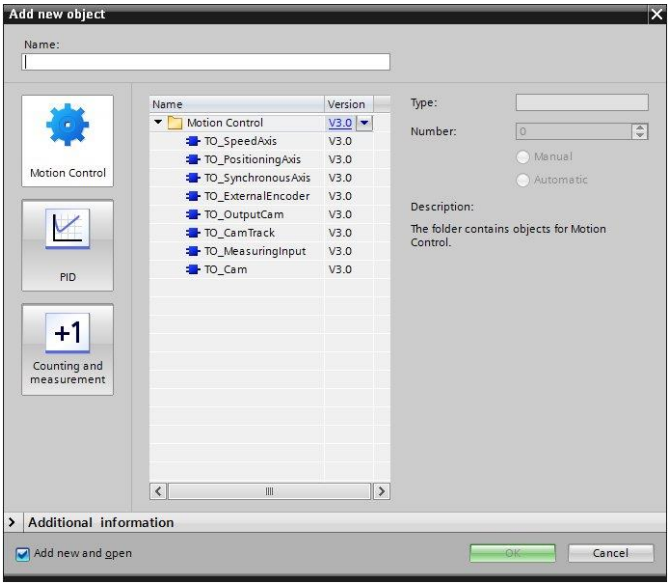
This guarantees a simple and standardized use of the motion control functionalities in the SIMATIC.

1.3 SIMATIC S7-1500 and SIMATIC S7-1500T

In the following chapters the technology objects available in the SIMATIC S7-1500 and in the SIMATIC S7-1500T are briefly introduced and explained.

1.3.1 Technology objects for motion control

Figure 1-1 Technology objects of the S7-1500 – motion control



- **Speed axis (TO_SpeedAxis)**
  The "speed axis" (TO_SpeedAxis) technology objects presents the mechanical drive component in the controller. Jobs for specifying the speed of the drive can be placed using PLCopen motion control instructions.
- **Positioning axis (TO_PositioningAxis)**
  The "positioning axis" (TO_PositioningAxis) technology object displays the mechanical drive component in the controller. Through the user program positioning jobs can be placed using PLCopen motion control instructions.
- **Synchronous axis (TO_SynchronousAxis)**
  The "synchronous axis" (TO_SynchronousAxis) technology object includes all functions of the "positioning axis" technology object. In addition, the "synchronous axis" technology object can follow the motions of a master axis. Positioning axes as well as synchronous axes can be used as master axis.
- **External encoder (TO_ExternalEncoder)**
  The "external encoder" (TO_ExternalEncoder) technology object records the actual position of an externally controlled drive. The determined actual positions can be evaluated in the user program.
- **Cam (TO_OutputCam)**
  The "cam" (TO_OutputCam) technology object generates switching signals depending on the setpoint or actual position of an axis or external encoder. The switching signals can be evaluated in the user program or output via the digital outputs. The switching signals of several cams can be interconnected with each other via AND or OR links.
- **Cam track (TO_CamTrack)**
  The "cam track" (TO_CamTrack) technology object combines up to 32 individual cams to one cam track. The start and end position of the individual cams can be freely selected. Other parameters are valid for all individual cams.

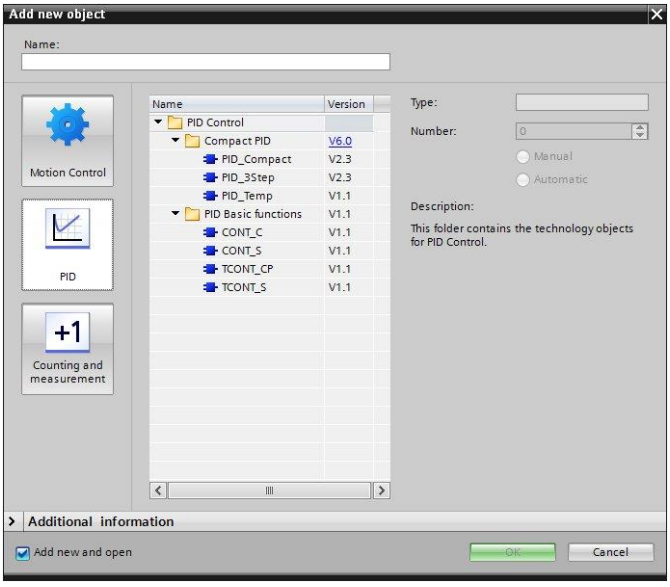
- **Measuring input (TO_MeasuringInput)**
  The "measuring input" (TO_MeasuringInput) technology object precisely detects the actual position of an axis or external encoder. The measurement is triggered with a rising and/or falling edge of a digital input. The measurement results can be evaluated in the user program.
- **Cam disk (TO_Cam) – (only S7-1500T)**
  The cam disk technology object defines a transfer function y = f(x). On a unit-neutral basis, the dependence of an output variable on an input variable is described in this transfer function. As a result, for example, the non-linear dependence of the slave axis position to a master axis in a synchronous coupling can be defined. A cam disc technology object can be used multiple times in different technology objects.

In comparison to SIMATIC S7-1500, the SIMATIC S7-1500T additionally supports the expanded synchronous functions. They offer the user the following further functions:

- Gearing can also create an absolute coupling of the slave axis to the master axis, besides the relative coupling.
- Besides gearing, the SIMATIC S7-1500T also offers support to curve synchronization, via which the coupling of two axes can be carried out with the help of a generally formulated transmission function.
- For a synchronous coupling of two axes, the actual position (actual value coupling) of another axis of an external encoder can be also be used as conductance.

1.3.2 Technology objects for PID control

Figure 1-2 Technology objects of the S7-1500 – PID control

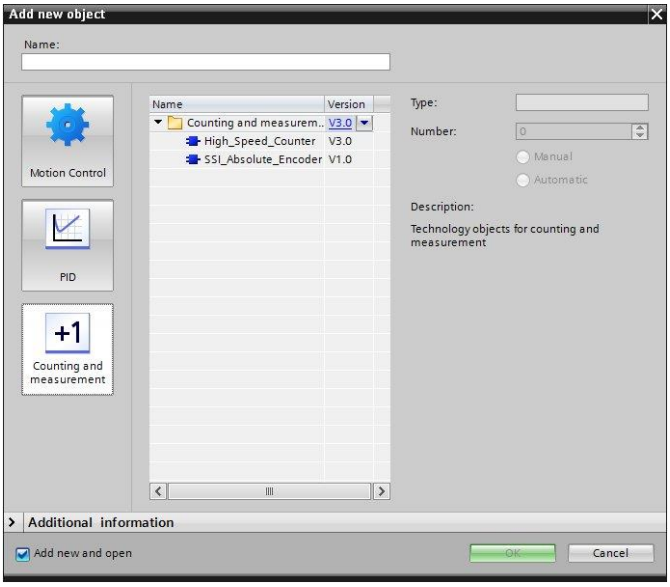


- **Universal control (PID_Compact)**
  The PID_Compact technology object provides a universal PID control with integrated optimization. It corresponds to the instance data block of the PID_Compact instruction. When calling the PID_Compact instruction this data block also has to be transferred. PID_Compact includes all settings for a specific control loop. When you open this technology object, you are supported by a special editor for the configuration of the control.
- **3 step control (PID_3Step)**
  The PID_3Step technology object provides a PID control with integrated optimization for valves. It corresponds to the instance data block of the PID_3Step instruction. When calling the PID_3Step instruction this data block also has to be transferred. PID_3Step includes all settings for a specific control loop. When you open this technology object, you are supported by a special editor for the configuration of the control.
- **Temperature control (PID_Temp)**
  The PID_Temp technology object provides a continuous PID control with integrated optimization. PID_Temp is especially designed for temperature control. Two outputs are available for this purpose, one for heating and one for cooling. The technology object corresponds to the instance data block of the PID_Temp instruction. When calling the PID_Temp instruction this data block also has to be transferred. PID_Temp includes all settings for a specific control loop. When you open this technology object, you are supported by a special editor for the configuration of the control.

- **PID basic function: Continuous control (CONT_C)**
  The CONT_C technology object provides a continuous control. It corresponds to the instance data block of the CONT_C instruction. When calling the CONT_C instruction this data block also has to be transferred. CONT_C includes all settings for a specific control loop. When you open this technology object, you are supported by a special editor for the configuration of the control.
- **PID basic function: Step control (CONT_S)**
  The CONT_S technology object provides a stepper control for integrated actuators. It corresponds to the instance data block of the CONT_S instruction. When calling the CONT_S instruction this data block also has to be transferred. CONT_S includes all settings for a specific control loop. When you open this technology object, you are supported by a special editor for the configuration of the control.
- **PID basic function: Continuous temperature control (TCONT_CP)**
  The TCONT_CP technology object provides a continuous temperature control with pulse generator. It corresponds to the instance data block of the TCONT_CP instruction. When calling the TCONT_CP instruction this data block also has to be transferred. TCONT_CP includes all settings for a specific control loop. When you open this technology object, you are supported by a special editor for the configuration of the control.
- **PID basic function: Temperature control (TCONT_S)**
  The TCONT_S technology object provides a temperature control for integrated actuators. It corresponds to the instance data block of the TCONT_S instruction. When calling the TCONT_S instruction this data block also has to be transferred. TCONT_S includes all settings for a specific control loop. When you open this technology object, you are supported by a special editor for the configuration of the control.

1.3.3 Technology objects for counting and measuring

Figure 1-3 Technology objects of the S7-1500 – counting and measuring



- **High-speed counter (High_Speed_Counter)**
  The High_Speed_Counter technology object enables the easy configuration of counter modules for the operation with incremental and pulse encoders and their use in the user program.
- **Measuring/position decoding (SSI_Absolute_Encoder)**
  The SSI_Absolute_Encoder technology object enables the easy configuration of position decoding modules for the operation with SSI absolute value encoders and their use in the user program.

2 Hardware Configuration for "Motion Control”

This chapter describes the basic hardware configuration of a motion control application with SIMATIC. In doing so, the general functions of the individual components are displayed.

2.1 Basic configuration

The basic hardware configuration of a motion control application is as follows:

Figure 2-1 Basic hardware configuration

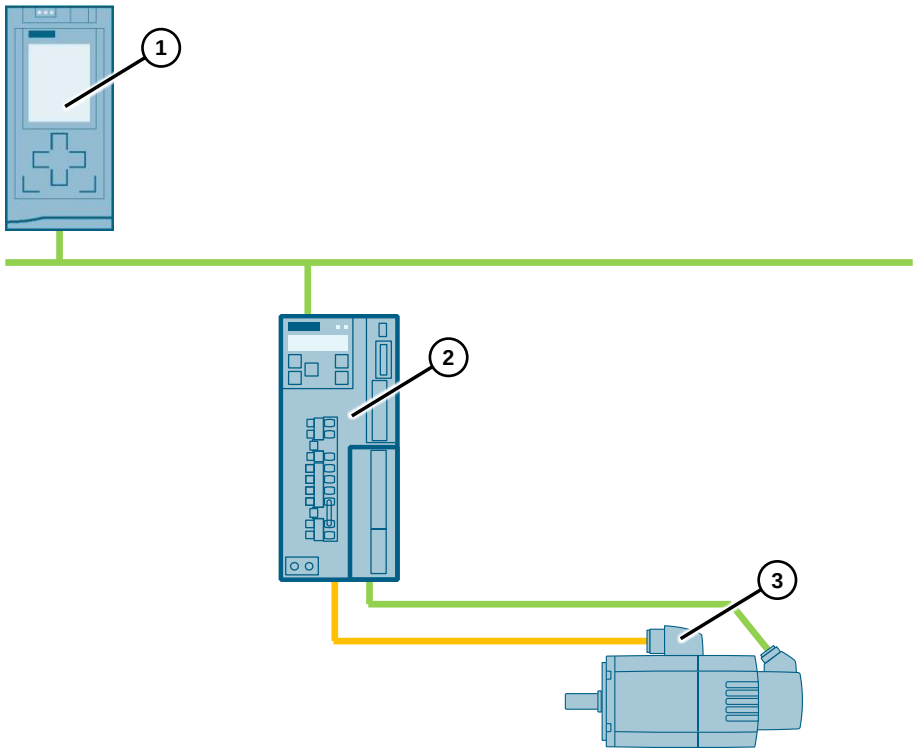


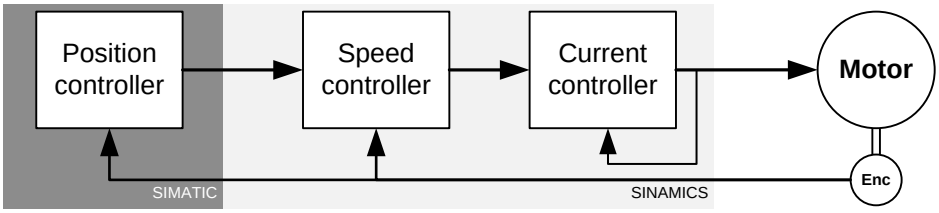
Table 2-1 Basic hardware configuration

No.	Hardware components	Function
1.	SIMATIC controller	Controller and control of the drive system and therefore of the motor of the motion control application.
2.	SINAMICS drive system	Controlling the motor. Providing the necessary motor current to move the connected motor. Position decoding of the motor axis (incremental or absolute).
3.	SIMOTICS motor	Executing the desired rotation. The motor forms the interface to the mechanical components of the motion control application.

2.2 Control loops involved

When configuring a motion control application, whilst using the technology objects of the SIMATIC controller and the SINAMICS drive technology, the following control loops are involved in cascading configuration.

Figure 2-2 Schematic diagram of the control loops involved



- **Current control**
  In dependence of the requested torque for the motor, the current control will provide the required current to control the motor.
  This functionality is part of the SINAMICS drive system.
- **Speed control**
  The speed control provides the desired torque setpoint for the current control so that the motor can reach the requested speed.
  This functionality is also part of the SINAMICS drive system.
- **Position control**
  The position control provides the speed setpoint for the speed control depending on the requested position, so that the motor or the drive axle can be moved to the requested position or the requested position can be maintained.
  The functionality of the position control is provided from the SIMATIC controller via the "axis” technology object.

In order to improve the control quality or the positioning, the following additional functions are provided in addition to the above mentioned control loop:

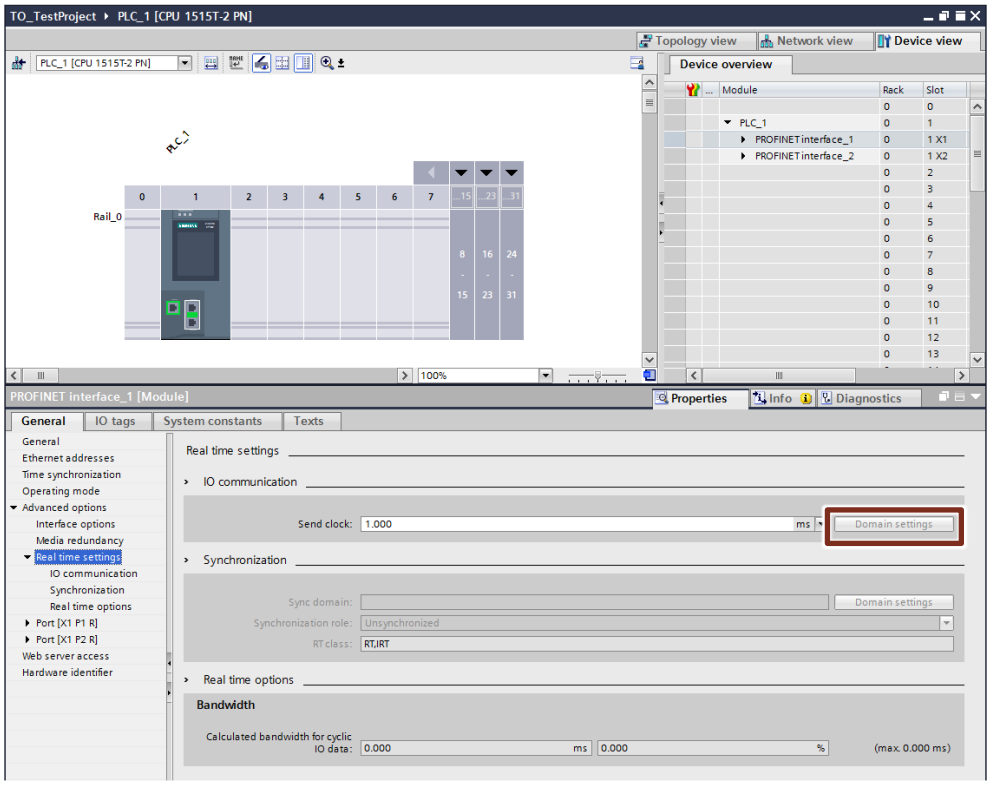
- Dynamic Servo Control (DSC):
  This moves the position control into the drive and it is usually calculated in fast speed control clock of the drive, which improves the control quality of positioning.
- Feed forward control:
  This function can enable a proportional speed feed forward control in the position control, which enables a fast forwarding of speed changes in the drive.

2.3 Isochronous data exchange

The control loop structure when using a position control is divided between the SIMATIC controller and the SINAMICS drive system. This is why the data exchange between these two components has to be designed in a way that a stable control is possible.

This is the reason why the use of the isochronous data exchange between the SINAMICS controller and the SINAMICS drive system via which the control loop spans, is highly recommended.

Figure 2-3 Isochronous data exchange



For more information about setting up the isochronous data exchange between the individual components of your application, please refer to the links and literature in chapter 10, for example, in chapter "Adding and Configuring Drives in the Device Configuration" of the S7-1500T Motion Control manual (\9\).

2.4 Motion control resources

Depending on the functionality of the motion control application, a suitable technology object is to be selected in the SIMATIC controller. Each technology object is assigned a MC resource that is recorded in the table below. Based on the resource consumption, an estimation of the CPU utilization can be determined using the technology objects deployed.

Table 2-2 Motion control resource consumption of the "axis” technology objects

Technology object	Version	MC resources	Remark
Speed axis	V3.0	40	
Positioning axis	V3.0	80	
Synchronous axis	V3.0	160	

More information about the motion control resources can be found in the "Technology objects" chapter of the manual on S7-1500T motion control (\9\).

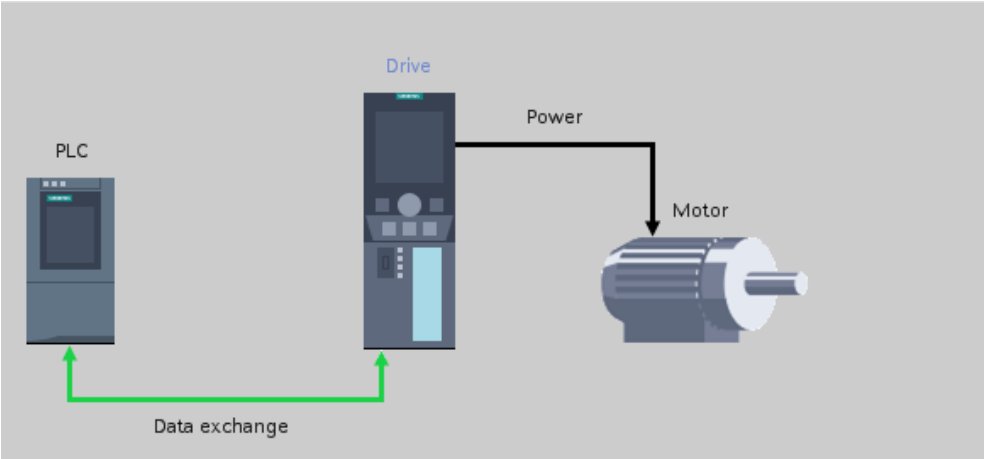
3 "Axis" Technology Object

3.1 Speed axis (TO_SpeedAxis)

3.1.1 General information

The speed axis represents the simplest technology object to control an axis from SIMATIC.

Figure 3-1 Speed axis (TO_SpeedAxis)



The technology object takes on the easy and PLCopen-compliant transfer of the desired speed setpoint to the drive, in which the actual speed control of the axis or the motor takes place. The control loop is solely responsible for maintaining the requested speed of the axis or the motor, this is why no information on the current position of the axis is provided by this technology object.

This is the reason why the speed axis is the also the technology object with the smallest resource consumption in the SIMATIC controller.

The following motion control instructions can be used together with the speed axis:

- **"MC_Power"**
  Switching on of the axis.
- **"MC_Reset"**
  Acknowledging the pending error and warning messages on the axis.
- **"MC_Halt"**
  Stopping the axis.
- **"MC_MoveVelocity"**
  Moving the axis at a preset speed.
- **"MC_MoveJog"**
  Manual moving of the axis in jog mode.
- **"MC_TorqueLimiting"**
  Enabling a force or torque limitation on the axis.

3.1.2 Fields of application

The speed axis technology object can be used for applications for which no position feedback to user program is necessary.

3.1.3 Application

This technology object can be used, for example, for the following applications:

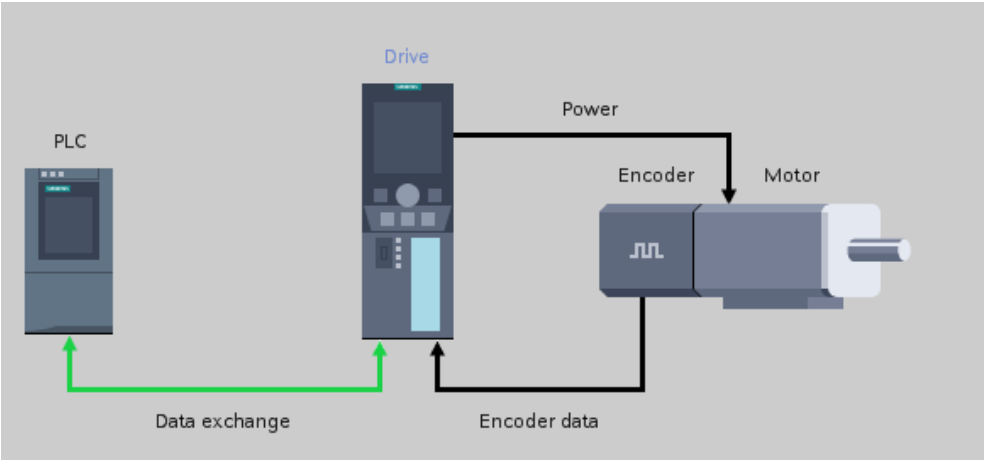
- Controlling a conveyor belt (without position detection)
- Controlling of spindles or driven tools
- Operation of pumps and fans

3.2 Positioning axis (TO_PositioningAxis)

3.2.1 General information

With the positioning axis technology object, the position control of the axis is integrated into the SIMATIC controller.

Figure 3-2 Positioning axis (TO_PositioningAxis)



The axis can be by precisely positioned by the technology object and the feedback of the current axis position is also already integrated into user program in the technology object.

The following motion control instructions can be used together with the positioning axis:

- All motion control instructions listed in the speed axis.
- **"MC_Home"**
  Actives or passive homing of the axis in the axis coordinate system.
- **"MC_MoveAbsolute"**
  Moving the axis to an absolute preset position.
- **"MC_MoveRelative"**
  Moving the axis by a specified distance.

- **"MC_MoveSuperimposed"**
  Starting an additional superimposed movement to an already running basic movement.
- **"MC_TorqueLimiting"**
  Enabling and configuring a force or torque limitation or a fixed stop detection for the position-controlled axis. This function can be enabled or disabled before or during a position-controlled movement.
- **"MC_SetSensor" (only S7-1500T)**
  Switchover of the encoder that is used for the position control of the axis. Here, it can be selected between four configured encoders on the axis.

3.2.2 Fields of application

The positioning axis technology object is necessary for all applications, in which a precise (controlled) positioning of an axis from the user program is required. Fast and defined positioning of the axis and the compensation of possible occurring faults when moving forward, reaching or stopping the target position of the axis is possible via the position control.

3.2.3 Application

This technology object can be used, for example, for the following applications:

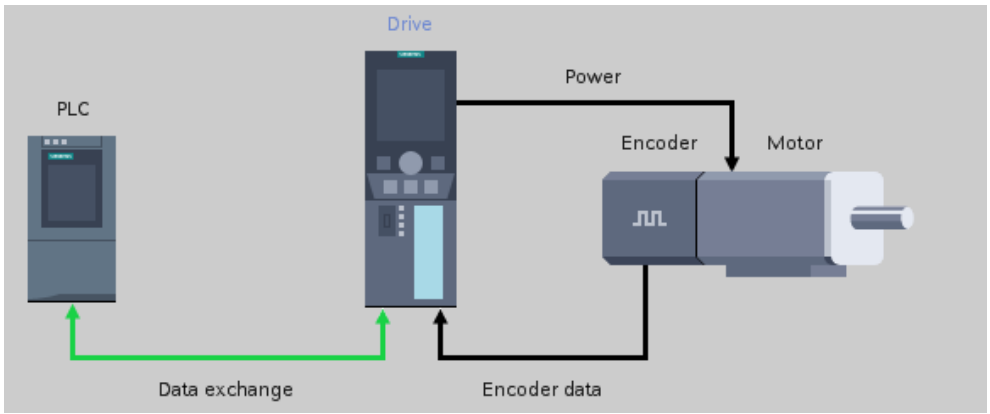
- Precise positioning of axis
- Use of suspended (gravity loaded) axes

3.3 Synchronous axis (TO_SynchronousAxis)

3.3.1 General information

The synchronous axis technology object contains all the functions of the positioning axis technology object. In addition, the axis can be interconnected using a conductance, for example, interconnecting the setpoint of another axis (master axis), so that the axis follows the position change of the master axis in synchronous mode.

Figure 3-3 Synchronous axis (TO_SynchronousAxis)

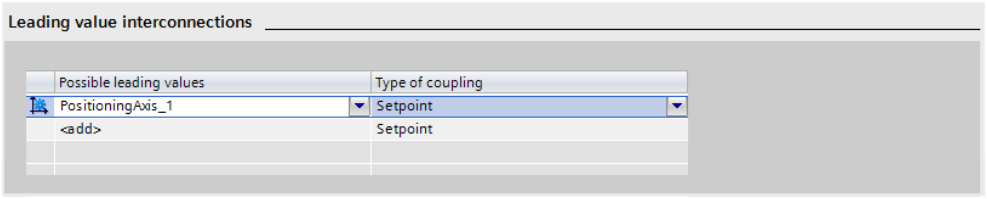


The following conductance interconnections can be used in the process.

- A linear relationship of the two axes to each other, for example, via a defined gear factor with relative or absolute position relationship between the slave and the master axis.
- A functional relationship of the two axis with each other, for example, via a predefined cam disk that represents the mapping rule between the position of the master axis and the positioning specification for the slave axis.

The slave axis has to be a synchronous axis in any of these relationships.

Figure 3-4 Definition of the conductance interconnections on the technology object



Possible conductance interconnections are:

- Setpoint or actual value of a positioning or a synchronous axis
- Actual value of an external encoder

The synchronous axis technology object can additionally be treated just as the positioning axis technology object in the user program, i.e. the axis can also be positioned to a predefined position without enabled synchronous operation from the user program.

The following motion control instructions can be used together with the synchronous axis:

- All motion control instructions listed in the positioning axis.
- **"MC_GearIn"**
  Start of relative gearing between a master axis and a slave axis.
- **"MC_GearInPos" (only S7-1500T)**
  Start of absolute gearing between a master axis and a slave axis whilst specifying a synchronous position. The synchronous position can be specified in relation to the master or the slave axis.
- **"MC_PhasingAbsolute" (only S7-1500T)**
  Absolute shifting of the conductance reference in gearing mode during an active synchronous operation.
- **"MC_PhasingRelative" (only S7-1500T)**
  Relative shifting of the conductance reference in gearing mode during an active synchronous operation.
- **"MC_CamIn" (only S7-1500T)**
  Start of absolute cam disc synchronization between a master axis and a slave axis whilst specifying a synchronous position in relation to the master axis.
- **"MC_SynchronizedMotionSimulation" (only S7-1500T)**
  Using this function sets an active synchronous operation in simulation. The synchronous relationship between master and slave axis remains active for the controller even if the slave axis is blocked via "MC_Power" or moved individually with other PLCopen functions. A renewed synchronization of the

axes is thus not necessary. However, it has to be made sure that setpoint jumps are avoided when the simulation is cancelled. At this point the slave and master axis should be standing above the position defined via the synchronous relationship.

3.3.2 Fields of application

The synchronous axis technology object is used for applications, in which there is a linear or non-linear position relationship between two axes.

The synchronous axis presents the slave axis that performs a motion depending on the current position of a slave axis.

3.3.3 Application

This technology object can be used, for example, for the following applications:

- Synchronous traversing of two non-mechanically coupled axes, for example, in extruders or in a gantry array.
- Execution of a non-linear motion profile depending on a master axis, for example, for a material feed for a press.
- Movable processing steps, such as, for example, with flying shears.
- Synchronous or position-dependent processing steps, such as, for example, with a cross cutter.

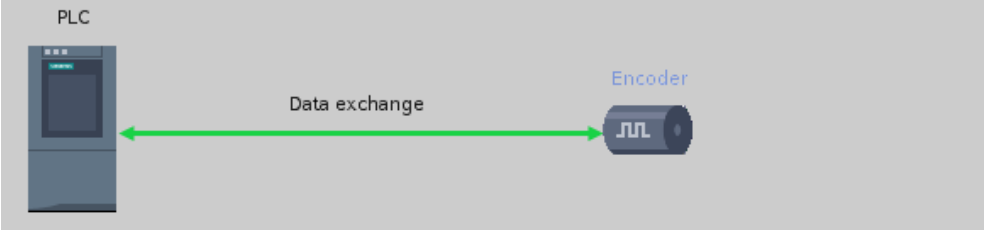
4 "External Encoder" Technology Object

4.1 General information

The external encoder technology object present an incremental or absolute value encoder for position detection, which is equipped with a PROFINET interface and can therefore directly exchange data with the SIMATIC controller.

Alternatively, a module of the ET 200SP or ET 200MP distributed I/O system for position detection, such as TM Count or TM PosInput could also be used here, which can communicate with the SIMATIC controller in accordance with the PROFIdrive standard.

Figure 4-1 "External encoder" technology object



The following motion control instructions can be used together with the external encoder:

- **"MC_Power"**
  Switching on or enabling the external encoder.
- **"MC_Reset"**
  Acknowledging the pending error and warning messages on the external encoder.
- **"MC_Home"**
  Passive or direct homing of the external encoder in the coordinate system.

4.2 Fields of application

The external encoder technology object is used for applications where motions have to be detected in the SIMATIC controller that are not directly influenced by this controller or where a slip is to be expected between drive axle and product or material.

The technology object determines the following current parameters of the recorded motion:

- Current position
- Current speed (internally calculated from the current position)
- Current acceleration (internally calculated from the current position)

4.3 Application

This technology object can be used, for example, for the following applications:

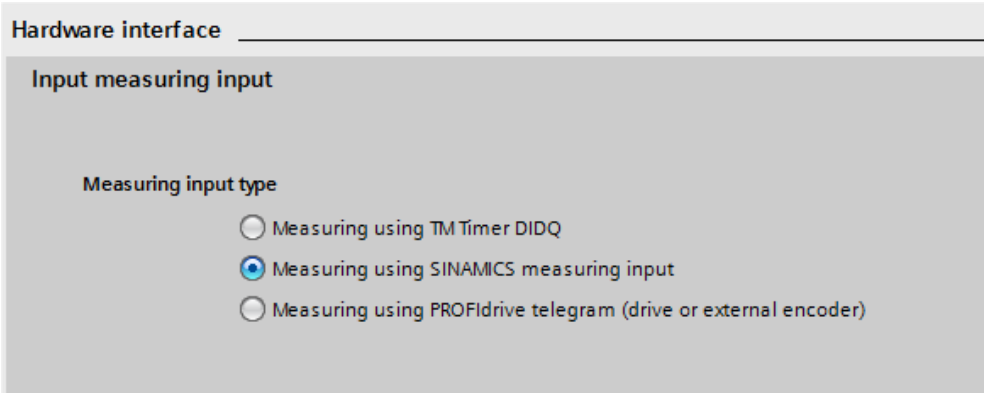
- Detection of a running material line as master axis (conductance) of a processing module in a processing line for a modular machine configuration.
- Motion detection of non-electric axes or axes that are not controlled by the SIMATIC controller. For example, the detection of a hydraulic axis that is used as conductance for a synchronous relationship in the SIMATIC S7-1500T controller.

5 "Measuring Input" Technology Object

5.1 General information

The measuring input technology object is used for the highly precise detection of axis positions based on external switching signals.

Figure 5-1 "Measuring input" technology object – selecting the detection



The detection of the external switching signal can be performed in the following ways:

- **Using TM Timer DIDQ**
  Via the TM Timer DIDQ technology module of the ET200SP or ET200MP distributed I/O system, the external switching signal is detected and provided with a time stamp. On the basis of this time stamp, the current position of the axis to be measured is then determined in the SIMATIC controller and provided via the technology object.
- **Using SINAMICS measuring inputs**
  If digital inputs of the control unit (CU) of the SINAMICS drive system are configured as measuring inputs and if these measuring inputs are not assigned to any drive connected to SINAMICS (global measurement), the switching signals detected on these inputs are also provided with a time stamp and transferred to the SIMATIC controller. From this, the SIMATIC controller can then determine the desired position value of the axis to be measured.
- **Using the PROFIdrive frame**
  If digital inputs of the control unit (CU) of the SINAMICS drive system are configured as measurement inputs, but the axis to be measured on the SINAMICS is connected to the same CU (local measurement), the time stamps of the switching signals are then processed within the CU of the SINAMICS. The determined position value is then directly transferred to the SIMATIC controller via the drive frame and provided to the user program via the measuring input technology object.

The following functions can be selected to evaluate the external switching signal:

- Measurement of rising edge of the switching signal.
  With this, for example, the beginning of a passing through object can be detected.
- Measurement of the falling edge of the switching signal.
  With this, for example, the end of a passing through object can be detected.

- Measurement of both edges of the switching signal
  For this measurement function it can be additionally be configured with what edge of the switching signal the measurement is to be started.
  This is why this measurement function can be especially used for the length or distance measurement of passing through objects.

The measurements can be influenced from the user program using the following PLCopen functions:

- **"MC_MeasuringInput"**
  A measurement is carried out once on the configured axis. As soon as the measurement has taken place, the function is terminated and the measured value is provided to the user program at the output of the PLCopen function.
- **"MC_MeasuringInputCyclic"**
  The instruction corresponds to the "MC_MeasuringInput" measuring function, with the difference that the measurement is not automatically terminated once the switching signal has been detected, but that it remains active. It has to be made sure in the user program that the measured values output are saved and evaluated after the detection, before they are overwritten again by new measured values.
- **"MC_AbortMeasuringInput"**
  Via this instruction the two above mentioned functions can be stopped again, for example, when no measured value has been acquired yet or the continuous detection is to be terminated.

Note

A measuring input can also be used on a virtual axis, for example, in order to detect the current position of a virtual axis to an external signal.

However, this functionality can only be used via the following detection functions of the switching signals:

- Using TM Timer DIDQ
- Using SINAMICS measuring inputs

However, please note that the setpoint position for virtual axes may not always match the actual position of the axis.

5.2 Fields of application

The measuring input technology object is used for applications where axis positions have to be highly-precisely detected on the basis of external signals, for example, print marks.

In doing so, the axis position of a virtual or real axis or the position of an external encoder can be exactly detected.

5.3 Application

This technology object can be used, for example, for the following applications:

- Detecting the reference position of a modular processing station on a running material line as basis for the following processing steps.
- Distance or length measurement of passing through objects.
- Detection of print marks to determine the precise position of a product in relation to the axis, which transports the product.

6 "Cam/Cam Track" Technology Object

6.1 General information

The cam or cam track technology object is used for the highly precise output of switching signals, depending on the position of an axis or an encoder.

For the output of switching signals it can be differentiated between the following two cam types:

- Distance output cams (position-based cams)
- Time-based cams

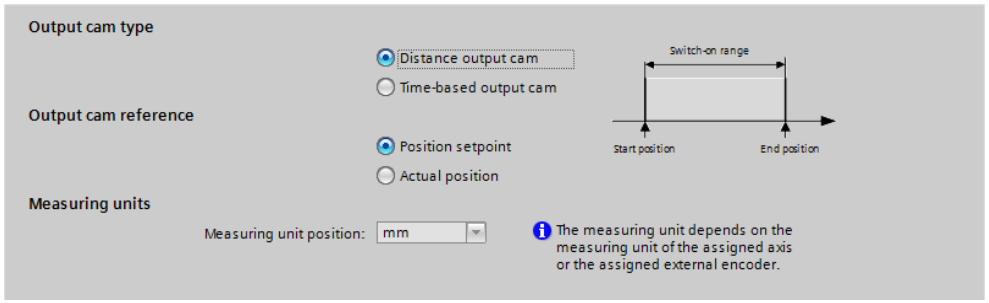
Note Within a "cam" or "cam track" technology object, only one cam type each can be used that has to be specified during the configuration of the technology object.

However, several technology objects can be interconnected on the same axis or the same encoder.

6.1.1 Distance output cams (position-based cams)

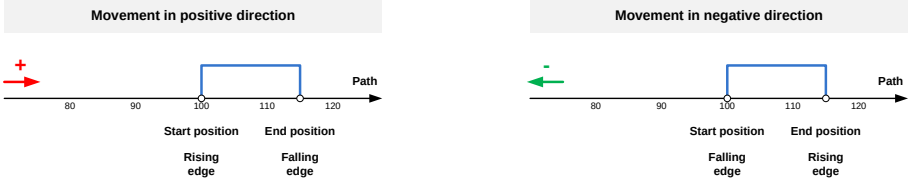
For a distance output cam, the start and the end position of the cam is specified absolute in relation to the interconnected axis. Here, it can be basically differentiated between the setpoint and actual position of the axis.

Figure 6-1 Distance output cams



A distance output (position-based) cam can be switched on depending on the direction, either only in forward direction, only in backward direction or in both directions. If the axis position is located within the start and end position of the defined cam and if the direction condition is met, the cam is switched on. Otherwise the cam is switched off.

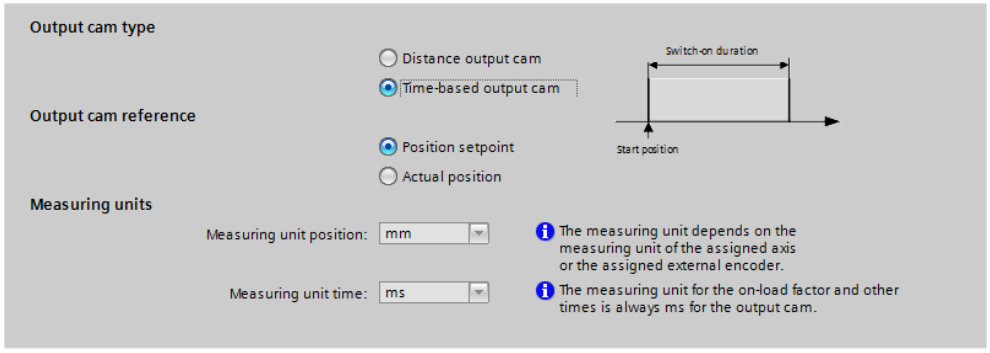
Figure 6-2 Direction-dependency of distance output (position-based) cams



6.1.2 Time-based cams

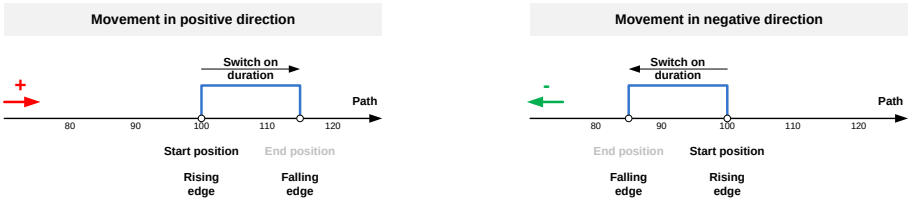
Contrary to distance output cams (position-based cams), time-based (output) cams are only defined by the absolute start position in relation to the axis. The cam length is the result of the also specified switch-on time of the cam. Here, it can also be generally differentiated between the setpoint and actual position of the axis.

Figure 6-3 Time-based (output) cam



The position range in which the time-based cam is switched on depends on the motion direction and the speed of the axis. The signal output of the cam is started by moving over the start position and lasts for the switch-on time defined.

Figure 6-4 Direction-dependency of time-based (output) cams

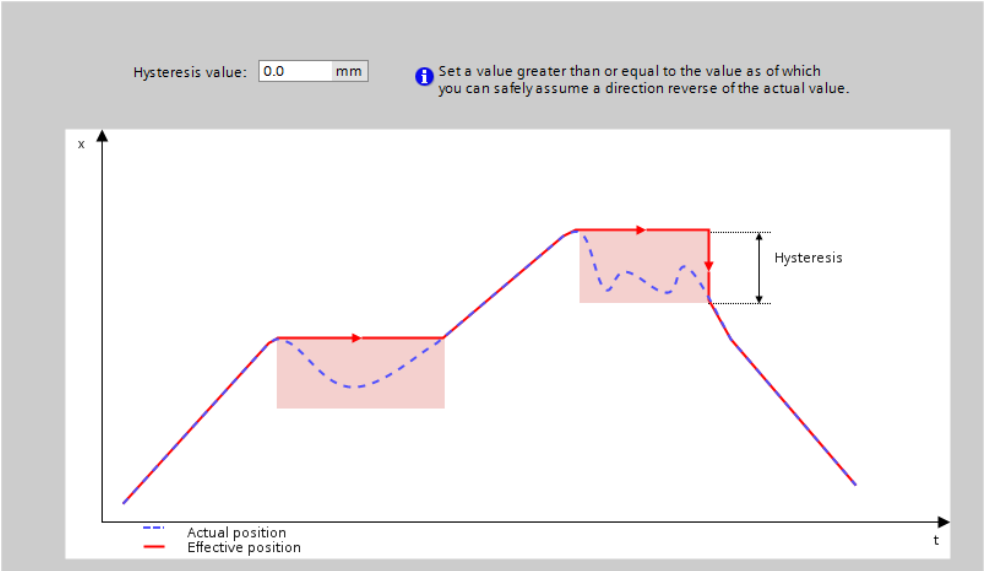


6.1.3 Hysteresis

Undesired switching actions of the cam due to fluctuating speed or small direction changes, for example, for the actual value reference of the cams, can be suppressed via the setting of the hysteresis.

The internal position of the axis or the encoder to which the responses of the cams or the cam track relates, is only changed again after a change of direction when the difference in position exceeds the set hysteresis value. However, if the axis or the encoder continuous the movement in the same direction, the internal position of the axis or the encoder is adjusted straight away again.

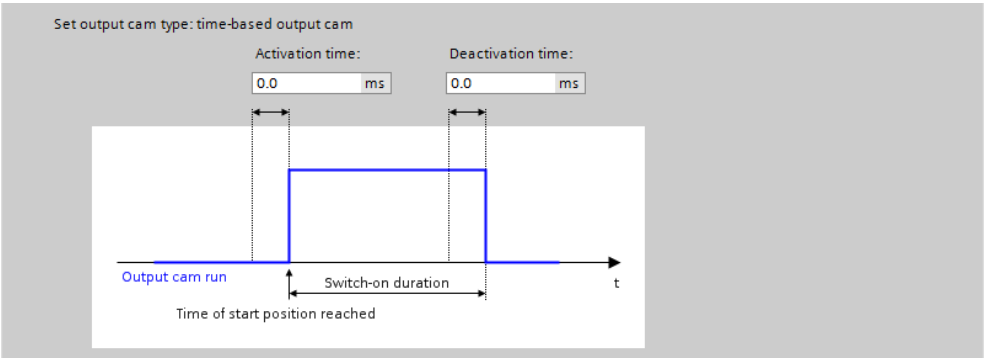
Figure 6-5 Hysteresis



6.1.4 Activation time

The activation time can define times for the two edges of a cam via which the switching delays of the actuator can be compensate for the signal output. This makes it possible to increase the precision of the cam output yet again in relation to the respective position of the axis or the encoder.

Figure 6-6 Activation time



6.2 Cams (TO_OutputCam)

6.2.1 Instruction to influence cams

The output of an individual cam can be influenced from the user program with the following instruction:

- **"MC_OutputCam"**
  Via the enable button of the function the cam output can be enabled or blocked. Changes in the switching positons become instantly effective when the cam output is enabled.

6.2.2 Effective direction of cams

The effective direction of the cam can be specified via the direction input of the "MC_OutputCam" instruction. This means that the cam…

- …is only switched in positive motion direction of the axis or the encoder and not when the axis moves in negative direction.
- …is only switched in negative motion direction of the axis or the encoder and not when the axis moves in positive direction.
- …is switched in both motion directions of the axis or the encoder.

6.2.3 Fields of application

The cam technology object is used for applications where the switching signals have to be precisely output on certain axis positions.

The axis position of a virtual or real axis or the position of an external encoder can be used as reference.

6.2.4 Application

This cam technology object, can be used, for example, for the following applications:

- Position accuracy when processing objects on a running material line.
- Position accuracy when applying materials on products on a running material line.
- Output of a chronologically precise switching signal, depending on an axis or encoder position.

6.3 Cam track (TO_CamTrack)

6.3.1 Instruction to influence a cam track

The output of up to 32 cams on a cam track can be influenced from the user program with the following instruction:

- **"MC_CamTrack"**
  Via the enable button of the function the cam output can be enabled or blocked. It can be preselected via the parameter mode whether the cam track is to be enabled straight away or whether it should only be enabled or disabled in the next track cycle. The switching output can also be switched on straight away or switched off via the block.

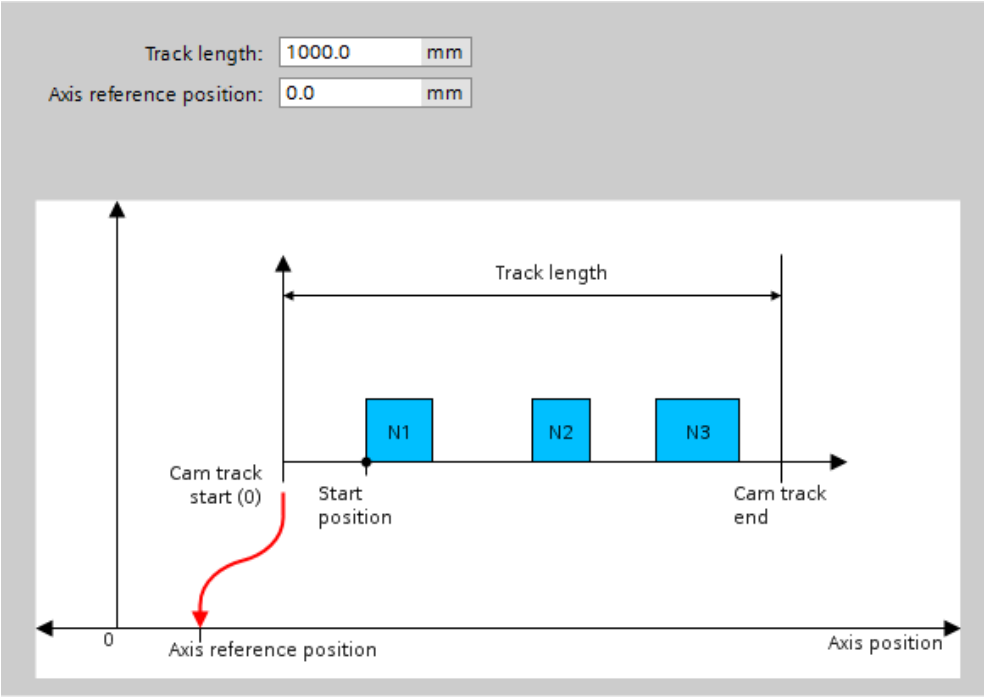
6.3.2 Axis relationship of the cam track

The cam track generally always starts at position 0.000. This is why all of the possible 32 cams of a cam track always have positive position values.

The connection of the cam track with a real axis can be made via the axis reference position. The axis reference position leads to displays the entire cam track in the coordinate system of the assigned axis or the assigned encoder.

Via the track length the end of the cam track, starting from position 0.000, can additionally be specified.

Figure 6-7 Display of the cam track on the position of an axis



6.3.3 Fields of application

The cam track technology object is used for applications where the several switching signals have to be precisely output on certain axis positions of an axis.

The axis position of a virtual or real axis or also the position of an external encoder can be used as reference.

6.3.4 Application

This cam track technology object can be used, for example, for the following applications:

- Repeated position accuracy when processing objects on a running material line.
- Position accuracy when repeatedly applying materials on products on a running material line, for example adhesive dots.
- Repeated output of a chronologically precise switching signal depending on an axis or encoder position.

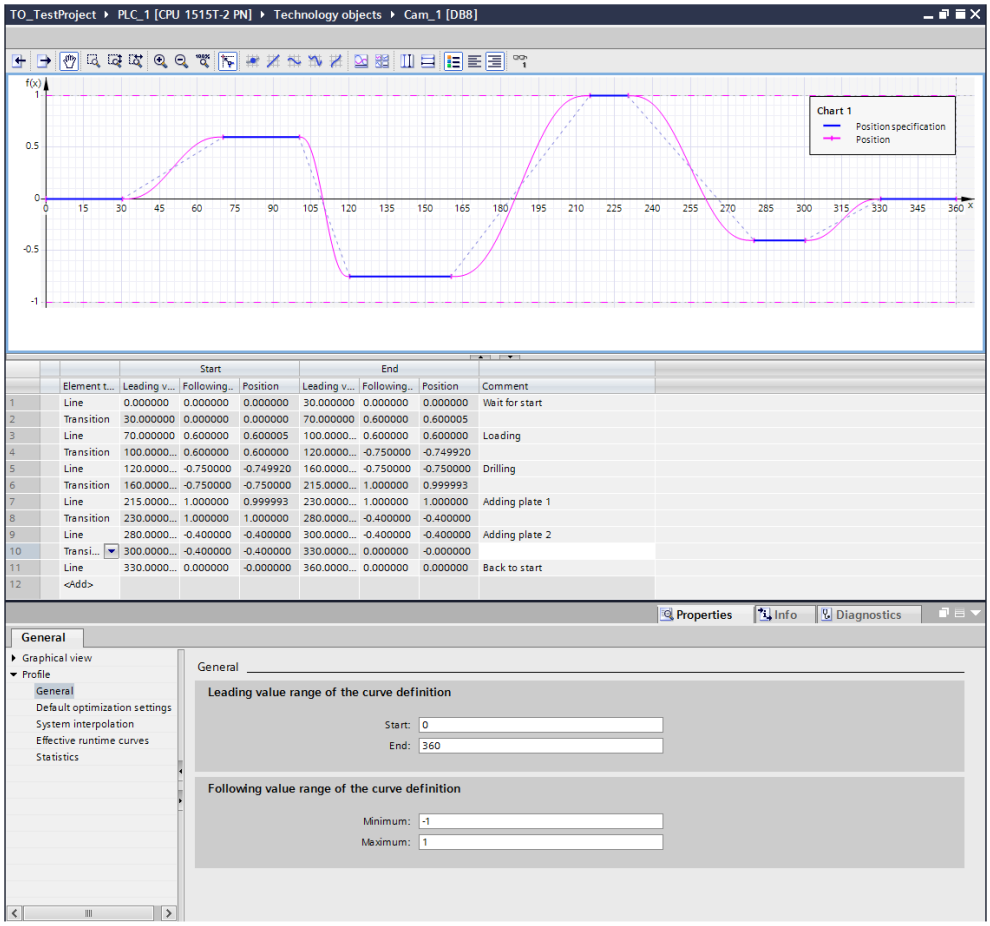
7 "Cam Disc" Technology Object

7.1 General information

The cam disc technology object defines the non-linear connection for a synchronous relationship between a slave and a master axis.

The correlation of the position between the position of the master axis and the resulting position of the slave axis can be specified via the cam disc editor in graphical form. The position of the master axis is clearly specified on the abscissa (x axis) of the cam disc coordinate system.

Figure 7-1 Cam disc in cam disc editor



The cam disc editor also provides supports for the optimization of the curve shape, for continuous and jerk-free transitions or for speed-optimal design of the cam disc.

7.2 Fields of application

A cam disc is always used when a slave axis is to move in absolute position-based synchronism to a master axis and there is no linear connection between the two axes.

Note

In gearing there is linear connection between the master and the slave axis. In a cam disc this kind of synchronism would be displayed by a straight line in the cam disc editor. The increase of the straight line would represent the gear factor.

7.3 Application

This cam disc technology object can be used for the following applications:

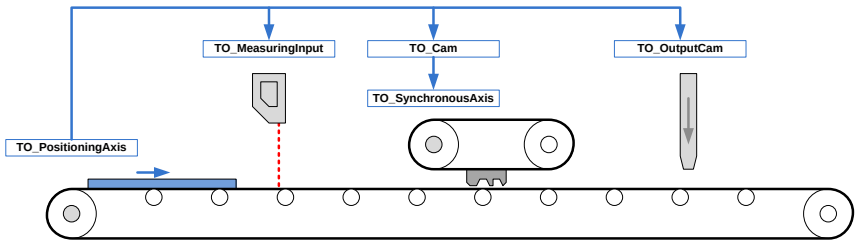
- Moving along certain position or speed profiles during flying processing in a running material line, for example, with a cross cutter.
- Moving along simple motion paths with two to three axes in plane or space. The motion rules of the individual axes are divided into independent cam discs. A master axis controls the motion of the two to three slave axes via the cam discs.
- Control of the complete product cycle of a machine via curve synchronization of all axes involved with a virtual axis as master axis.

8 Example Application

8.1 Configuration

The example application provided is structured in a way that the use of many different technology objects can be shown within a simulated production process.

Figure 8-1 Example application to display the technology objects



8.1.1 Components used

This documentation and the example project are valid for

- TIA Portal or STEP 7 as of V14 update 2 and higher
- SIMATIC S7-1500 as of firmware version V2.0

This application example has been created with the following hardware and software components:

Hardware components

Table 8-1

Component	Qty.	Article number	Note
SIMATIC S7-1500T			
SIMATIC CPU 1515T-2 PN	1	6ES7515-2TM01-0AB0	CPU 1515T-2 PN, 750kB PROG., 3MB DATA Firmware version: V2.0.3
SINAMICS V90 Demo case			
SINAMICS S7-1500 demo case with KBEI	1	A5E37844243	In the TIA Portal, the HSP 0185 "SINAMICS V90 PN" hardware support package has to be installed.

Software components

Table 8-2

Component	Qty.	Article number	Note
TIA Portal V14 Update 2	1		Always included in the individual components as an engineering framework.
STEP 7 Professional V14 Update 2	1	6ES7822-1AA04-0YC5	SIMATIC STEP 7 PROF. V14 FLOATING LICENSE; engineering software in TIA Portal.

Component	Qty.	Article number	Note
Internet Explorer 11 V11.0.9600.16428	1		Required for the display of the HTML user interface.

Example files and projects

Table 8-3

Component	Note
109743134_S7-1500T_TechnologyObjects_DOC_v10_en.pdf	This document.
109743134_S7-1500T_TechnologyObjects_PROJ_v10_en.zip	Example project with virtual axes for the use on a S7-1500T CPU without additional drives.
109743134_S7-1500T_TechnologyObjects_V90_PROJ_v10_de.zip	Example project with real axes of the SINAMICS V90 demo case for the use with a S7-1500T CPU and the axes and the periphery of the demo case. The usage of this project is described in more detail in chapter 9.

8.1.2 Functions

The example application consists of a conveyor line for passing through products (material line) and the following three processing stations:

- Product detection via measuring inputs:
  Detection of the product position on the conveyor line. The measuring input is used as reference for the two other processing stations of the product passing through.
- Sealing unit via cam disc synchronization:
  The sealing unit uses a cam disc for the realization of the following motion sequence:
  - Approach of the product processing position from the start position:
    The sealing unit is moved at fast speed from the start position up to the beginning of the parallel motion along the conveyor line.
  - Movable sealing of the product during the transport movement:
    Moving the sealing unit at the same speed as the conveyor line up to a virtual standstill of the sealing unit via the processing position on the product.
  - Fast withdrawal to the start position:
    Fast backward movement to the start position after leaving the parallel motion direction by the sealing unit as completion of the processing process.
- Material is applied using valve control via cam signal:
  The valve used for applying the material is controlled in a way that it is opened (rising edge), when the product reaches the nozzle and remains open for a specified transit length, for example, the product length.

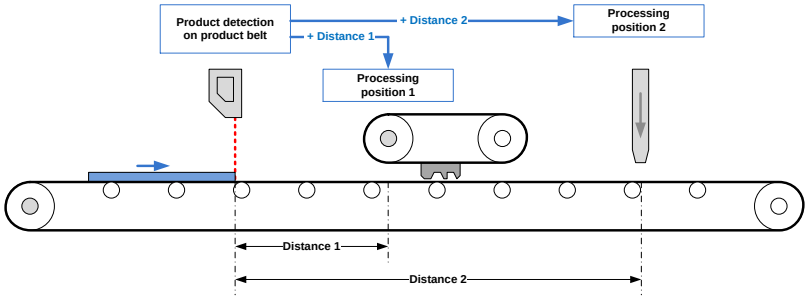
8.1.3 Basis of realization

The general realization of the example application is based on a data logger in which the product positions, detected by the measuring input, are entered. This

makes it possible to operate individual processing stations of the example application independent from each other.

The processing stations of the sealing unit and for applying the material, then only monitor the data logger. If a new value is entered there by the measuring input, a new product was detected and the start positions for the two other processing stations can be calculated internally and the processing jobs can be started on the individual stations.

Figure 8-2 Basic realization of the example application



8.2 Technology objects used

8.2.1 Positioning axis (TO_PositioningAxis)

The positioning axis technology object moves the product within the production process, by driving the conveyor belt, on which the products are located during production.

The positioning axis technology object was selected for the conveyor belt since a position value with the current position of the product is necessary for the further production process, in order to be able to carry out further production steps.

Note Although it would also be possible to specify a motion for the conveyor belt via the (TO_SpeedAxis), this is only possible via a speed value for the belt. A position value is not available for this type of technology object and can therefore also not be used, for example, for a position measurement of the tools via the measuring input.

The positioning axis technology object was configured as modulo axis, in order to prevent overflows of the position value for continuous production. The modulo length set for the modulo axis was dimensioned in a way that there is a unique position value for each product on the conveyor belt.

8.2.2 Measuring input (TO_MeasuringInput)

Via the measuring input technology object the products placed on the conveyor are measured and their position is precisely detected.

The position value detected of the product on the conveyor belt can be processed as follows in the further production process:

- Saving the conveyor belt position in which the product is located:
  The position of the conveyor belt detected via the measuring input technology object represents the belt position at which the product has reached the light barrier. The distance of the product to the light barrier can be specified in the following at any point in time whilst taking into account the current position of the conveyor belt.
- Position correction of the conveyor belt:
  The current position of the conveyor belt is corrected "on the fly" and thus highly precisely, to a defined position with the help of the measured value detected. Thus, the current position of the conveyor belt represents the current position of the tool and can therefore be directly used by the other production process of the plant.

In example application introduced here the first method is used.

Note The evaluation of the two edges of the light barrier signal as well as the product length can also be individually recorded and provided to the other production steps via the measuring input. However, for reasons of clarity of the example application this data acquisition is not used.

8.2.3 Synchronous axis (TO_SynchronousAxis)

A synchronous relationship to another axis technology object or external encoder can be created via the synchronous axis technology object. The synchronous axis presents the slave axis of the synchronous relationship.

In the example application introduced here, the synchronous axis drives the sealing unit that is moved to the positioning axis via a non-linear synchronous relationship. The synchronous axis was configured as modulo axis. The start position of the sealing unit is located on the top dead center of the left drive role, so that half the role's circumference in the cam disc can be used to adjust the speed of the sealing unit to the speed of the belt.

8.2.4 Cam disc (TO_Cam)

A non-linear absolute connection for the movement of two axis can be defined via the cam disc technology object. The cam disc represents the mathematical synchronous relationship between a master and a slave axis.

In the example application presented here, the positioning axis technology object of the conveyor belt is the master axis for the synchronous relationship with the help of the cam disc. The synchronous axis technology object is used for the slave axis. The slave axis drives the sealing unit. In parts, the sealing unit is moved synchronously to the conveyor belt via the non-linear synchronous relationship.

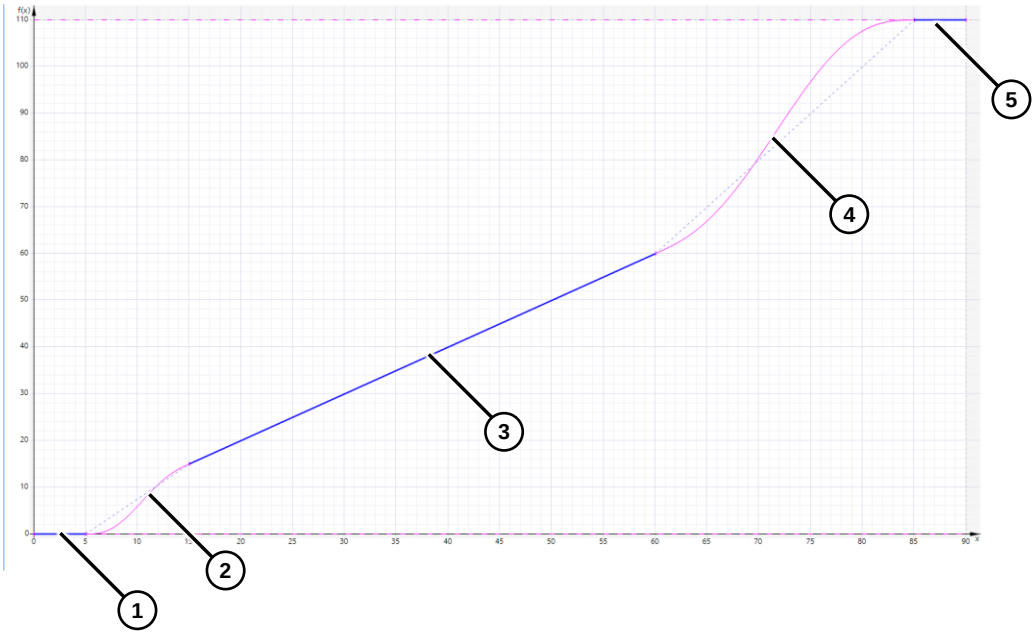
The cam disc used in the example application is displayed in the following figure and is made up of five cam disc segments. The individual segments of the cam disc are defined via the type of the cam disc element and the coordinates of the start and end point of the element in the coordinate system of the cam disc. The value x always presents the conductance position, i.e. the position of the master axis in the coordinate system of the cam disc, whilst f(x) defines the appropriate position of the slave axis.

Note

The shape of the cam disc is defined in the TIA Portal via the cam disc editor in the coordinate system of the cam disc. This coordinate system can be defined in the cam disc editor via the extension of the coordinate axis for the master axis (x) and the slave axis (f(x)).

The adjustment of the cam disc to the coordinate system of the axes is only specified when using the cam disc in the user program.

Figure 8-3 Cam disc used in the application



The individual cam disc segments have the following function in the example application, whereby the products passing through present the master axis (x) and the sealing unit presents the slave axis (f(x)):

Table 8-4 Composition of the cam disc

Segment type	Start		End		Comment
	x	f(x)	x	f(x)	
Straight line (1)	0	0	5	0	Rest of the slave axis. The slave axis remains in position f(x)=0 during the movement of the master axis. This position represents the basic position of the slave axis in the application, i.e. the sealing unit.
Transition (2)	5	0	15	15	Fast positioning of the sealing unit. With this movement the sealing unit is positioned above the passing through product.

Segment type	Start		End		Comment
	x	f(x)	x	f(x)	
Straight line (3)	15	15	60	60	Synchronism between product and sealing unit. A 1:1 synchronism between sealing unit and the product is achieved via the straight line with the slope 1 in the cam disc. During the motion, the sealing unit remains above the same position on the product.
Transition (4)	60	60	85	110	Fast prepositioning of the sealing unit to the basic position.
Straight line (5)	85	110	90	110	Standstill of the sealing unit in basic position (modulo: 110 ~ 0).

8.2.5 Cams (TO_OutputCam)

Precise position control of an output, for example, a valve, can be carried out via the cam technology object. Thus, an accurately positioned product job can be achieved on the tool via the nozzle.

In example application introduced here, the cam is assigned to the positioning axis technology object, i.e. the conveyor belt. This makes precisely positioned processing of the tool possible.

8.3 Programming of the processing stations

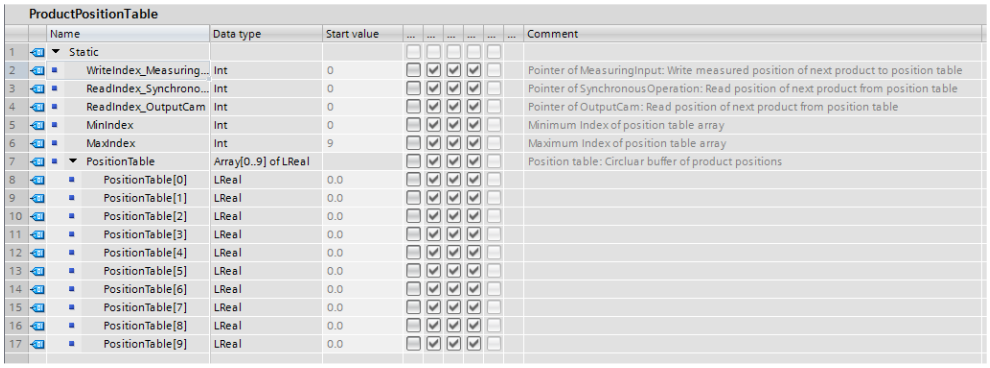
In the following chapters the function blocks of the individual processing stations are displayed in code sections and the programming realized there is explained in more detail. The blocks are already included in the example application.

8.3.1 Joint data management

Basis of the program concept implemented in this example application is the joint data management of the application. All units access the data of this example application. The data, i.e. the positions of the individual products on the conveyor belt are saved with additional management information in the form of a table in the "ProductPositionTable” data block.

The "ProductPositionTable” data block has the following configuration:

Figure 8-4 Joint data management in the "ProductPositionTable" data block



ProductPositionTable

	Name	Data type	Start value	Comment
1	Static			
2	WriteIndex_Measuring...	Int	0	Pointer of MeasuringInput: Write measured position of next product to position table
3	ReadIndex_Synchrono...	Int	0	Pointer of SynchronousOperation: Read position of next product from position table
4	ReadIndex_OutputCam	Int	0	Pointer of OutputCam: Read position of next product from position table
5	MinIndex	Int	0	Minimum Index of position table array
6	MaxIndex	Int	9	Maximum Index of position table array
7	PositionTable	Array[0..9] of LReal		Position table: Circluar buffer of product positions
8	PositionTable[0]	LReal	0.0	
9	PositionTable[1]	LReal	0.0	
10	PositionTable[2]	LReal	0.0	
11	PositionTable[3]	LReal	0.0	
12	PositionTable[4]	LReal	0.0	
13	PositionTable[5]	LReal	0.0	
14	PositionTable[6]	LReal	0.0	
15	PositionTable[7]	LReal	0.0	
16	PositionTable[8]	LReal	0.0	
17	PositionTable[9]	LReal	0.0	

Table 8-5 Configuration of the "ProductPositionTable" data block

Data element	Function
WriteIndex_MeasuringInput	Write index for the "PositionTable" array. If the position value was entered in the table through the measuring input unit, the index is increased by one or set to the next index (ring buffer function of "PositionTable").
ReadIndex_SynchronousOperation	Read index for the "PositionTable" array. The index shows the product that is currently processed by the sealing unit.
ReadIndex_OutputCam	Read index for the "PositionTable" array. The index shows the product that is currently processed by the unit for the applying the material.
MinIndex	Minimum index value as fixed value. First entry of the "PositionTable" array.
MaxIndex	Maximum index value as fixed value. Last entry of the "PositionTable" array.
PositionTable	"PositionTable" array in which the product positions are entered in form of a ring buffer. This means after "MaxIndex" the index restarts with the "MinIndex" value.

8.3.2 Functional principle of the example application

The entire example application is now controlled via the indices in the joint data management:

1. If a product is detected by the measuring input, the product position detected is entered in the table and the write index of the measuring input is set to the next index.
   Thus, the product is available for the following production process.
2. If the read index of the sealing unit differs from the write index of the measuring input unit, a product is available for processing. This starts the sealing unit.
   If the sealing procedure is finished, the read index is set to the next index.
3. The same is the case with the read index of the unit for the material job. This process is also started if there is a difference to the write index of the measuring input unit.
   If applying the material is finished, the read index is set to the next index.

The individual processing units therefore work totally independent from each other. The following principle applies:

- If there is no difference between the read index of the unit and the write index the measuring input unit, there is no product for processing and the respective process remains in the dormant phase.
- If there is a difference between the indices, the respective unit has to be started and the product has to be processed.

Note This approach even supports several products on the conveyor belt. The respective process is started until write and read index point to the same array element of the "PositionTable" again.

Note It has to be ensured that the product does not reach the processing station before the processing process was started.

8.3.3 Measuring input

Since the "MC_MeasuringInputCyclic" technology function can only be used in connection with a real measuring input, i.e. a "real hardware input", the measuring input functionality in the example application with virtual axis via the "GetMeasuringInput_Simulation" function block is simulated.

Triggering the measurement input is communicated to the block via the "GetPosition" input, which is influenced via the interface of the example application. As soon as a rising edge is detected on this input, the temporary "tempLatchProductPosition" tag is set.

```
//+++ Control machine function +++
//Check function block interface
#tempLatchProductPosition := False;
IF ((#GetPosition = True)
  AND (#statGetPositionOld = False))
THEN
  #tempLatchProductPosition := True;
END_IF;
```

If the temporary "tempLatchProductPosition" tag is set, the current axis position of the conveyor belt is entered in the table and the write index is set to the next index.

```
//+++ Realize machine function +++
IF (#tempLatchProductPosition = True)
THEN
  //Get actual write index
  #tempWriteIndex :=
                 #ProductPositionTable.WriteIndex_MeasuringInput;
  //Save actual position of product belt (product position)
  #ProductPositionTable.PositionTable[#tempWriteIndex] :=
                                  #AxisProductBelt.ActualPosition;
  //Generate next write index (write pointer)
  //Incement index pointer
  #tempWriteIndex := #tempWriteIndex + 1;
  //Check modulo area of index pointer
  IF (#tempWriteIndex > #ProductPositionTable.MaxIndex)
  THEN
    #tempWriteIndex := #ProductPositionTable.MinIndex;
  END_IF;
  //Write back index pointer
  #ProductPositionTable.WriteIndex_MeasuringInput :=
                                                  #tempWriteIndex;
END_IF;
```

Subsequently, the current status of the "GetPosition” input is also saved, in order to detect a rising edge of the input in the next cycle.

```
//+++ Internal function +++
//Edge control
#statGetPositionOld := #GetPosition;
```

Changing the program code for the use of a real measuring input

In order to be able to also use a real measuring input with current program code, the function block has to be changed as follows. The changes displayed here are already included in the "GetMeasuringInput" function block of the project for the SINAMICS V90 demo case and are used in interaction with the TM Timer DIDQ technology module.

The technology function for the measuring input functionality has to be called at the beginning of the block. To do this, the function for the one-off measuring "MC_MeasuringInput” is used in the example application. The function is connected with the "MeasuringInput” technology object to the positioning axis conveyor belt. The function is always executed via a rising edge on the "Execute” input when the function is enabled via "FunctionEnable” and no measuring value (instMeasuringInput.Done = False) has been detected. As a result, the function restarts independently after each measurement.

```
//+++ Call function block +++
#instMeasuringInput(MeasuringInput:=#MeasuringInput,
                    Execute:=(#FunctionEnable
                              AND NOT(#instMeasuringInput.Done)),
                    Mode:=0);
```

If a measured value was detected (instMeasuringInput.Done = True), virtually the same function as in the simulation will run. But now the measured value detected ("instMeasuringInput.MeasuredValue1") is entered in the "PositionTable" table.

```
//+++ Realize machine function +++
IF (#instMeasuringInput.Done = True)
THEN
  //Get actual write index
  #tempWriteIndex :=
                  #ProductPositionTable.WriteIndex_MeasuringInput;
  //Save measured position of product on product belt
  #ProductPositionTable.PositionTable[#tempWriteIndex] :=
                                #instMeasuringInput.MeasuredValue1;
  //Generate next write index (write pointer)
  //Incement index pointer
  #tempWriteIndex := #tempWriteIndex + 1;
  //Check modulo area of index pointer
  IF (#tempWriteIndex > #ProductPositionTable.MaxIndex)
  THEN
    #tempWriteIndex := #ProductPositionTable.MinIndex;
  END_IF;
  //Write back index pointer
  #ProductPositionTable.WriteIndex_MeasuringInput :=
                                                  #tempWriteIndex;
END_IF;
```

Note Instead of the "MC_MeasuringInput" function and the independent restart of the function realized in the program, this is where the "MC_MeasuringInputCyclic" technology function can also be used.

8.3.4 Sealing unit

At the beginning of the "SynchronousOperation" function block, the current parameters of the two technology objects used can be detected:

- Cam disc technology object:
  Check bit 5 of the status word of the cam disc whether the cam disc used has already been already interpolated for the use.

- Positioning axis technology object – master axis of the curve synchronization: Determination of the modulo settings of the technology object, in order to be able to correctly calculate the position for the start of the synchronism in relation to the position of the conveyor belt.

```
//+++ Get technology objects data +++
//Check interpolation of cam disc
#tempCamDiscStatusWord := #CamDisc.StatusWord;
#statCamIsInterpolated := #tempCamDiscStatusWord.%X5;

//Get modulo settings of master axis
#tempModuloAxisMaster.Enable := #AxisMaster.Modulo.Enable;
#tempModuloAxisMaster.StartValue := #AxisMaster.Modulo.StartValue;
#tempModuloAxisMaster.Length := #AxisMaster.Modulo.Length;
```

Checking the read and write index from the DB "ProductPositionTable" of the joint data management. If read and write index differ, the function is enabled, the cam disc is interpolated and if no synchronism is currently active, cam disc synchronization can be started as follows:

4. First of all the read index is determined and the position saved is read from the table.
5. Starting from the product position, the position of the processing unit (#PRODUCTBELT_SYNC_POSITION) and the synchronization length (#PRODUCTBELT_SYNC_DISTANCE), which are included in the constants, the parameters of the "MC_CamIn" technology function are determined for the synchronization.
6. If required, the "statMasterOffset" parameter may have to be corrected again, whilst taken the modulo settings of the master axis into account.
7. The synchronous operation is then started via the "statExecute" tag and the output signals of the function block are set accordingly.

```
//+++ Control machine function +++
//Check product position table
IF ((#ProductPositionTable.ReadIndex_SynchronousOperation <>
                  #ProductPositionTable.WriteIndex_MeasuringInput)
  AND (#FunctionEnable = True)
  AND (#statCamIsInterpolated = True)
  AND (#statExecute = False))
THEN
  //Get actual read index
  #tempReadIndex :=
            #ProductPositionTable.ReadIndex_SynchronousOperation;
  //Read saved product position
  #statSavedProductPosition :=
             #ProductPositionTable.PositionTable[#tempReadIndex];
```

⇨

```
  //Calculate synchronizing parameter
  #statMasterOffset := #statSavedProductPosition +
          #PRODUCTBELT_SYNC_POSITION - #PRODUCTBELT_SYNC_DISTANCE;
  #statMasterStartDistance := #PRODUCTBELT_SYNC_DISTANCE;
  #statMasterSyncPosition := #PRODUCTBELT_SYNC_DISTANCE;
  //Check modulo settings
  IF (#tempModuloAxisMaster.Enable = True)
  THEN
    #tempModuloEndPosition := #tempModuloAxisMaster.StartValue +
                                      #tempModuloAxisMaster.Length;
    IF (#statMasterOffset > #tempModuloEndPosition)
    THEN
      #statMasterOffset := #statMasterOffset -
                                      #tempModuloAxisMaster.Length;
    END_IF;
  END_IF;
  //Start synchronous operation/movement
  #statExecute := True;
  //Set output signals
  #statBusy := True;
  #statError := False;
  #statErrorId := #ERRORID_DEFAULT;
```

Based on the "EndOfProfile" output of the "MC_CamIn" technology function, it is checked whether the cam disc was fully traversed and as a result, the sealing process on the product fully completed.

After the successful completion of the function, the read index is set to the next index and the outputs of the function block are set accordingly.

```
  //Check end of synchronous operation/movement
ELSIF ((#instMcCamIn.Busy = False)
  AND (#instMcCamIn.EndOfProfile = True)
  AND (#statExecute = True))
THEN
  //Stop execution of function block
  #statExecute := False;
  //Get actual read index
  #tempReadIndex :=
            #ProductPositionTable.ReadIndex_SynchronousOperation;
  //Incement index pointer
  #tempReadIndex := #tempReadIndex + 1;
```

⇨

```
  //Check modulo area of index pointer
  IF (#tempReadIndex > #ProductPositionTable.MaxIndex)
  THEN
    #tempReadIndex := #ProductPositionTable.MinIndex;
  END_IF;
  //Write back index pointer
  #ProductPositionTable.ReadIndex_SynchronousOperation :=
                                                    #tempReadIndex;
  //Set output signals
  #statBusy := False;
  #statError := False;
  #statErrorId := #ERRORID_DEFAULT;
```

If an error occurred during the execution of the function or the technology function was replaced by another technology function, the error will be detected and passed on to the outputs of the function block.

```
  //Check error of synchronous operation/movement
ELSIF (((#instMcCamIn.Error = TRUE)
  OR (#instMcCamIn.CommandAborted = True))
  AND
  (#statExecute = True))
THEN
  //Save error id
  IF (#instMcCamIn.CommandAborted = True)
  THEN
    //No error id is set by the function block
    #statErrorId := #ERRORID_COMMAND_ABORTED;
  ELSE
    //Save error id of the function block
    #statErrorId := #instMcCamIn.ErrorId;
  END_IF;
  //Set output signals
  #statBusy := False;
  #statError := True;
  //Stop execution of function block
  #statExecute := False;
END_IF;
```

Subsequently, the following technology functions are called with preset parameters:

- "MC_InterpolateCam"
  The cam disc is only interpolated if has not yet been interpolated.

- "MC_CamIn"
  The cam disc synchronization is started with a rising edge at the "execute" input.

```
//+++ Realize machine function +++
//Function block calls
//Interpolation (if cam disc is not interpolated)
#instInterpolateCam(Cam := #CamDisc,
                    Execute := NOT (#statCamIsInterpolated));

//Synchronous operation
#instMcCamIn(Master := #AxisMaster,
             Slave := #AxisSlave,
             Cam := #CamDisc,
             Execute := #statExecute,
             MasterOffset := #statMasterOffset,
             SlaveOffset := 0.0,
             MasterScaling := 1.0,
             SlaveScaling := 1.0,
             MasterSyncPosition := #statMasterSyncPosition,
             SyncProfileReference := 1,
             MasterStartDistance := #statMasterStartDistance,
             ApplicationMode := 0,
             SyncDirection := 1,
             InSync => #InSync);
```

Subsequently, the internally saved output signals for the function block are also passed on to the output parameters of the FB.

```
//+++ Generate output data +++
#Busy := #statBusy;
#Error := #statError;
#ErrorID := #statErrorId;
```

8.3.5 Application of material

At the beginning, the current parameters of the technology object used, are also determined in the "OutputCamOperation" function block:

- Positioning axis technology object:
  Determination of the modulo settings of the technology object, in order to be able to correctly calculate the position for the start of the synchronism in relation to the position of the conveyor belt.

```
//+++ Get technology objects data +++
//Get modulo settings of master axis
#tempModuloAxisMaster.Enable := #AxisMaster.Modulo.Enable;
#tempModuloAxisMaster.StartValue := #AxisMaster.Modulo.StartValue;
#tempModuloAxisMaster.Length := #AxisMaster.Modulo.Length;
```

Checking the read and write index from the DB "ProductPositionTable" of the joint data management. If read and write index differ, the function is enabled. If no cam output is currently active, the cam output is started as follows:

1. First of all the read index is determined and the position saved is read from the table.
2. Afterwards, the start and end position of the cam output is calculated.
3. If required, the "statMasterOffset" parameter may have to be corrected again, whilst taken the modulo settings of the master axis into account.

```
//+++ Control machine function +++
//Check product position table
IF ((#ProductPositionTable.ReadIndex_OutputCam <>
                  #ProductPositionTable.WriteIndex_MeasuringInput)
  AND (#FunctionEnable = True)
  AND (#statEnable = False))
THEN
  //Get actual read index
  #tempReadIndex := #ProductPositionTable.ReadIndex_OutputCam;
  //Read saved product position
  #statSavedProductPosition :=
             #ProductPositionTable.PositionTable[#tempReadIndex];
  //Calculate position parameter
  #statOnPosition := #statSavedProductPosition +
                                                #OUTPUTCAM_POSITION;
  #statOffPosition := #statOnPosition + #OUTPUTCAM_ON_DISTANCE;
  //Check modulo settings
  IF (#tempModuloAxisMaster.Enable = True)
  THEN
    #tempModuloEndPosition := #tempModuloAxisMaster.StartValue +
                                      #tempModuloAxisMaster.Length;
    //Check OnPosition
    IF (#statOnPosition > #tempModuloEndPosition)
    THEN
      #statOnPosition := #statOnPosition -
                                      #tempModuloAxisMaster.Length;
    END_IF;
```

⇨

```
    //Check OffPosition
    IF (#statOffPosition > #tempModuloEndPosition)
    THEN
      #statOffPosition := #statOffPosition -
                                    #tempModuloAxisMaster.Length;
    END_IF;
  END_IF;
  //Start execution of function block
  #statEnable := True;
```

Based on the falling edge on the "CamOutput" output of the "MC_OutputCam" technology function it is checked whether the output of the cam signal and thus the application of the material (end position of the cam reached) has been fully completed. In this case, the read index is set to the next index.

```
  //Check end of synchronous operation/movement
ELSIF ((#instMcOutputCam.Busy = True)
  AND (#instMcOutputCam.CamOutput = False)
  AND (#statCamOutputOld = True)
  AND (#statEnable = True))
THEN
  //Stop execution of function block
  #statEnable := False;
  //Get actual read index
  #tempReadIndex := #ProductPositionTable.ReadIndex_OutputCam;
  //Incement index pointer
  #tempReadIndex := #tempReadIndex + 1;
  //Check modulo area of index pointer
  IF (#tempReadIndex > #ProductPositionTable.MaxIndex)
  THEN
    #tempReadIndex := #ProductPositionTable.MinIndex;
  END_IF;
  //Write back index pointer
  #ProductPositionTable.ReadIndex_OutputCam := #tempReadIndex;
```

If an error occurred during the execution of the function, the error will be detected and passed on to the outputs of the function block. In this case, the cam output is stopped via the "statEnable" tag.

```
  //Check error of synchronous operation/movement
ELSIF ((#instMcOutputCam.Error = TRUE)
  AND (#statEnable = True))
THEN
```

⇨

```
  //Save error id of the function block
  #statErrorId := #instMcOutputCam.ErrorId;
  //Stop execution of function block
  #statEnable := False;
END_IF;
```

Subsequently, the current status of the "CamOutput" output of the "MC_OutputCam" function is saved, in order to detect an edge from it in the next cycle.

```
//+++ Realize edge detection +++
//Save signal to next OB cycle
#statCamOutputOld := #instMcOutputCam.CamOutput;
```

Finally the "MC_OutputCam" technology function is called using the previously set parameters.

```
//+++ Realize machine function +++
//Function block calls
#instMcOutputCam(OutputCam := #OutputCam,
                 Enable := #statEnable,
                 OnPosition := #statOnPosition,
                 OffPosition := #statOffPosition,
                 Mode := 1,
                 Direction := 3,
                 CamOutput => #CamOutput,
                 Busy => #Busy,
                 Error => #Error,
                 ErrorId => #ErrorID);
```

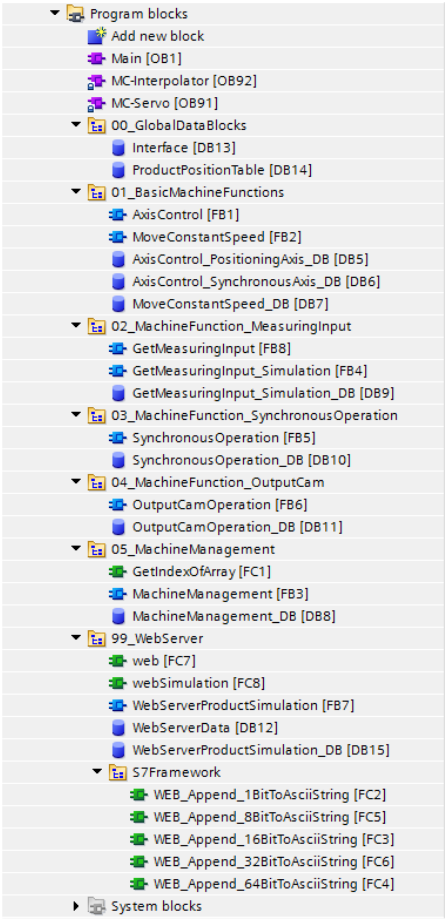
Note Calling the technology function at the end of the function block, has the advantage that based on the instance data of the technology function the parameters from the last cycle can be recalculated before the function is activated with the newly calculated parameters.

8.4 Configuration of the TIA portal project

The TIA Portal project provided, includes a runnable user program for the SIMATIC S7-1515T-2 PN including HMTL user interface, which is running on the web server of the SIMATIC CPU.

8.4.1 Program blocks

The following program blocks are included in the TIA Portal project provided:



The OB "Main" includes the calls of all blocks involved in the program.

The OB "MC-Interpolator" and "MC-Servo" are automatically creating by using the motion control functions of the SIMATIC CPU and include the interpolator and the position control for the motion functions.

The "00_GlobalDataBlocks" folder includes the cross-program data blocks.

The "01_BasicMachineFunctions" folder includes blocks for the realization of the basic functions of the axes used.

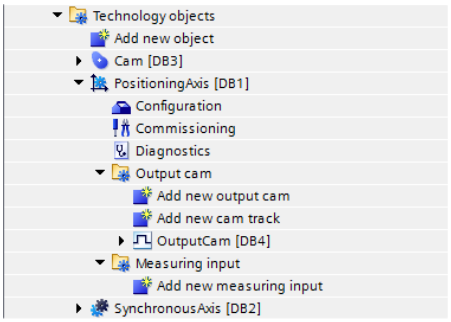
The "02_MachineFunction_..." to "04_MachineFunction_..." folders include the function blocks of the individual processing stations of the machine.

The "05_MachineManagement" folder includes the state machines for controlling the machine functions and the enabling of the individual processing stations of the machine.

The "99_WebServer" folder includes all blocks for the simulation of the machine on the web server of the SIMATIC CPU.

8.4.2 Technology objects

The technology objects are included in the TIA Portal project provided:



Cam disc: Synchronous relationship between the conveyor belt and the sealing unit.

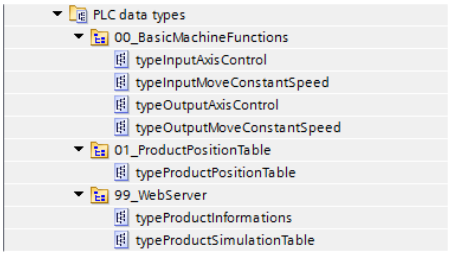
Positioning axis: Drive of the conveyor belt with subordinate technology objects:

- Cams (output cams): Material is applied based on product position or conveyor belt position.
- Measuring inputs: Determination of product position (only simulated, this is why it is not an active technology object here!)

Synchronous axis: Drive of the sealing unit

8.4.3 PLC data types

The PLC data types are included in the TIA Portal project provided:

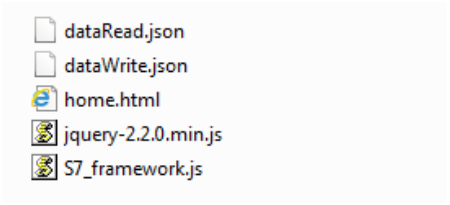


The data types are divided in the same folder structure as the program blocks and include the appropriate PLC data types.

8.4.4 Realization of the web server application for the simulation

The realization of the web server application for the simulation of the example application is not subject of this documentation. In chapter 10 the appropriate application examples are listed for this purpose. Nevertheless, the functions of the web page are briefly discussed here.

The following files are included in the TIA Portal project provided in the "HTML" subfolder for the realization of the pages of the web server of the SIMATIC CPU:



The two JSON files include the definitions for the data exchange between HTML page and the CPU.

The "home.html" file represents the actual web page.

The JS files include the JavaScript functions for the data exchange of the web page with the CPU.

Realization of the web page

The HTML file includes the web page configuration that is represented via the SIMATIC CPU web server for the simulation of the example application. The web page is configured in accordance with the HTML 5 standard whilst using various HTML objects. The definition of the appearance of the objects complies with the CSS standard, whereby this description is also included in the HTML file in the <style> section. The animation of the objects is also done with the help of JavaScript functions included in the HMTL file in the <script> section. The JavaScript functions animate the objects located on the side, depending on the process variables of the SIMATIC CPU.

Data exchange with the SIMATIC CPU

The data exchange between the web page and the SIMATIC CPU is carried out in two different ways, depending on the direction of the data:

- Sending jobs from the web page to the CPU:
  The data is transferred to the user program event-driven, for example, directly after pressing a button via the SIMATIC web server functions.
- Sending data from the CPU to the web page:
  Via the "window.setIntervall" function of the HTML page, the status data of the user program is requested every 5 milliseconds from the HTML page in the SIMATIC CPU with the help of the "S7Framework".

To be able to realize the cyclic data exchange as optimized as possible and without a high CPU load, all variables of the status data in the user program are transferred in two string tags (237 byte and 51 byte) packed and therefore only two variables are transferred to the HTML page. This is where the two strings are disassembled into individual variables again.

For this functionality the "S7Framework" is used. The framework consists of S7 functions that take on the packing of the variables into a string on the SIMATIC CPU. And of JavaScript functions in which the unpacking of the variables is realized on the HTML page.

The definition in which way the variables are packed in the string, is stored in the JSON file "dataRead.json". The length and the type of the data to be transferred is listed there in the appropriate order. The distribution of the data on the HTML page to the individual variables on the page is then done via the "UpdatePlcData" JavaScript function.

Figure 8-5 SIMATIC functions of the "S7-Framework" for packing the data

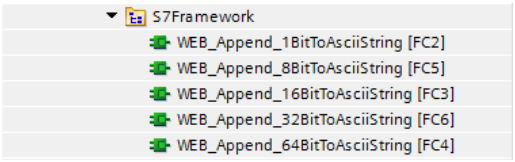


Figure 8-6 Configuration of the JSON file "dataRead.json"

```
{
"val" : ":="WebServerData".WebServerString: :="WebServerData".WebServerSimlationString:",
"len" : "16;16;16;16;4;4;4;16;16;16;16;16;16;16;16;16;1;1;1;1;4;4;4;4;4;4;4;4;4;4;4",
"typ" : "4;4;4;4;2;2;2;4;4;4;4;4;4;4;4;4;0;0;0;0;2;2;2;2;2;2;2;2;2;2;2;2",
"str" : ""
}
```

For the codes used in the JSON file, the following assignment is valid:

Table 8-6 Meaning of the code in the "len" line of the JSON file

len	Data type	Length
1	BOOL	1 bit
2	BYTE	8 bit
4	INT, WORD	16 bit
8	DINT, DWORD, REAL	32 bit
16	LINT, LWORD, LREAL	64 bit

Table 8-7 Codes in the "type" line of the JSON file for the string conversion

type	Data type
0	BOOL
1	UINT
2	INT, DINT
3	REAL
4	LREAL
5	STRING

Figure 8-7 JavaScript function on the HTML page to distribute the data

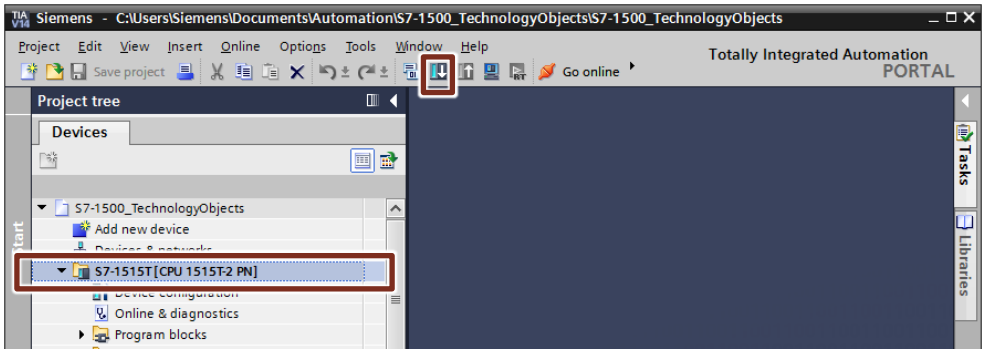
```
function UpdatePlcData(values)
{
    <!-- Set variables of actual cycle -->
    <!-- Data: web -->
    ActualPosition_PositioningAxis = values[0];
    ActualPosition_SynchronousAxis = values[1];
    ActualVelocity_PositioningAxis = values[2];
    ActualVelocity_SynchronousAxis = values[3];
    ProductPositionTable_WriteIndex_MeasuringInput = values[4];
    ProductPositionTable_ReadIndex_SynchronousOperation = values[5];
    ProductPositionTable_ReadIndex_OutputCam = values[6];
    ArrayProductPosition[0] = values[7];
    ArrayProductPosition[1] = values[8];
    ArrayProductPosition[2] = values[9];
    ArrayProductPosition[3] = values[10];
    ArrayProductPosition[4] = values[11];
    ArrayProductPosition[5] = values[12];
    ArrayProductPosition[6] = values[13];
    ArrayProductPosition[7] = values[14];
    ArrayProductPosition[8] = values[15];
    ArrayProductPosition[9] = values[16];
    CamOutput = values[17];
    <!-- Data: webSimulation -->
    indicatorMeasuring = values[18];
    indicatorSynchronousOperation = values[19];
    indicatorOutputCam = values[20];
    rotationPositioningAxis = values[21];
    rotationSynchronousAxis = values[22];
    positionProductPixel[0] = values[23];
    positionProductPixel[1] = values[24];
    positionProductPixel[2] = values[25];
    positionProductPixel[3] = values[26];
    positionProductPixel[4] = values[27];
    positionProductPixel[5] = values[28];
    positionProductPixel[6] = values[29];
    positionProductPixel[7] = values[30];
    positionProductPixel[8] = values[31];
    positionProductPixel[9] = values[32];

    <!-- Data update done -->
    busy = false;
}
```


8.5 Operating the application example

8.5.1 Downloading the project into the CPU

Open the TIA Portal project of the example application, select the SIMATIC-CPU S7-1515T-2 PN in the project tree and load the entire project into the CPU via the context menu or the download icon.

Figure 8-8 Loading project into the CPU

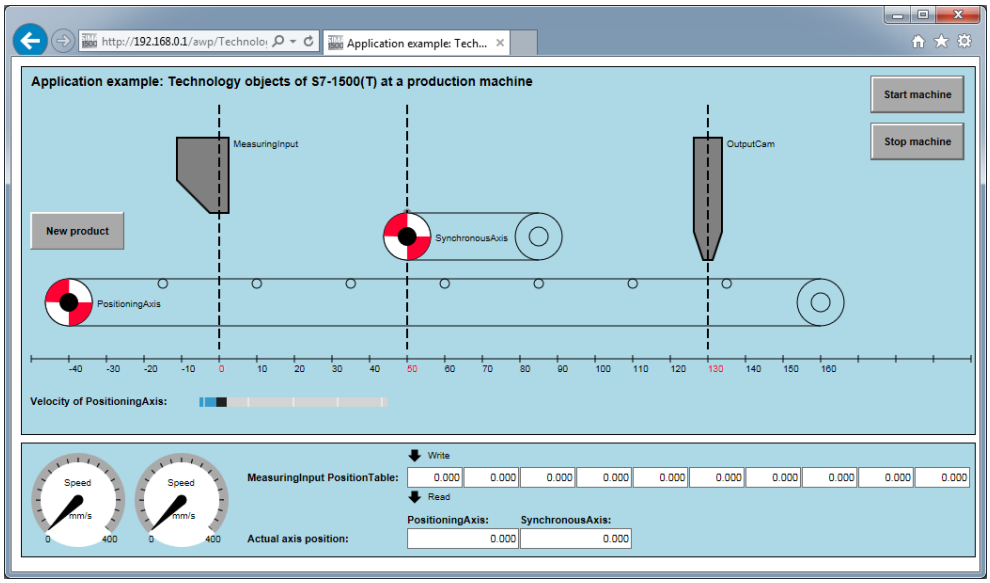


8.5.2 Calling the user interface/simulation

Connect your PC with the SIMATIC CPU and make sure that both devices are located in the same IP area (192.168.0.x).

Then open the internet explorer and call the web server of the SIMATIC CPU there via the IP address 192.168.0.1.

Figure 8-9 Opening the user interface via the web server



8.5.3 Operating the simulation

On the web page displayed, the following operating options and displays for the simulation of the example application are available.

Operation

Figure 8-10 Web page of the example application - operating options

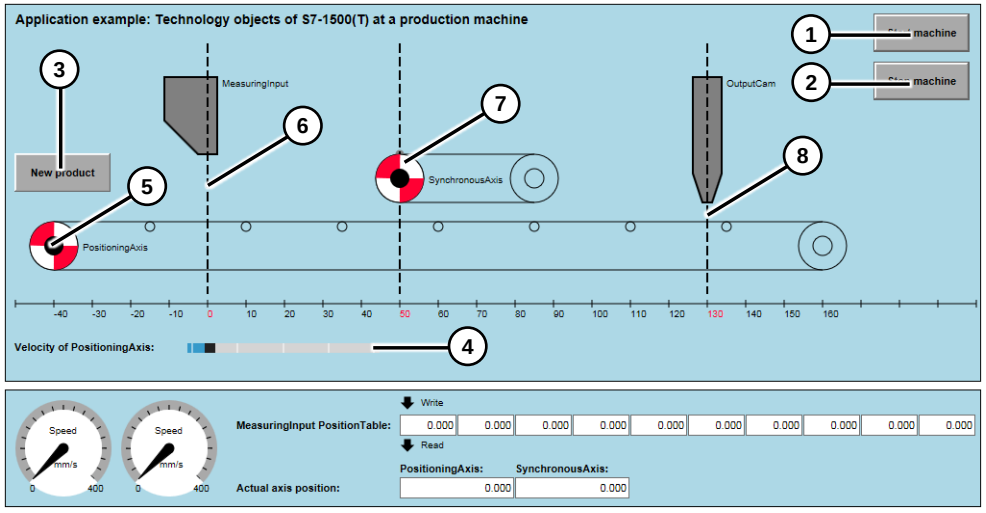


Table 8-8 Operating options and display on the web page

No.	Operating option
1.	Start of the application example The conveyor belt (5) starts at the speed (4) selected.
2.	Stop of the application example The conveyor belt (5) is stopped.
3.	Request new product With each press of the button a new product is generated on the conveyor belt (5) and moved through the individual processing stations (6-8).
4.	Speed of the conveyor belt The speed of the conveyor belt can be changed via the slide control.
5.	Drive axis conveyor belt "Positioning axis" technology object as drive of the conveyor belt.
6.	Measuring inputs If a product is detected by the measuring input, the dashed line changes color.
7.	Drive axis sealing unit "Synchronous axis" technology object as drive for the sealing unit. The motion of the sealing unit is displayed by a point on the drive belt.
8.	Application of material "Cams (output cams)" technology object to control the valve for applying material. As long as the valve is switched on, the dashed line changes color.

Display/monitoring

In this area of the web page the processing process of the example application can be monitored.

Figure 8-11 Web page of the example application - displays

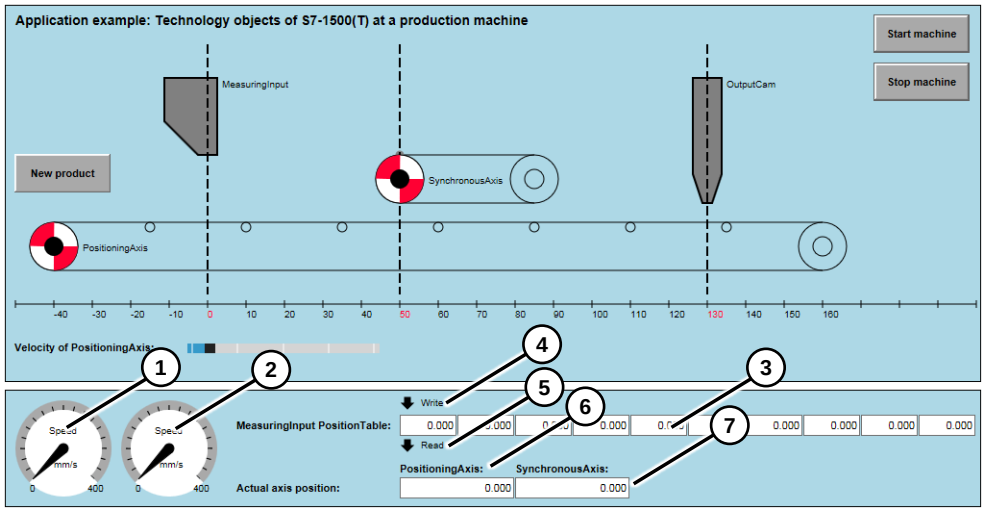


Table 8-9 Operating options and display on the web page

No.	Display
1.	Speed of the conveyor belt Display of the current speed of the conveyor belt.
2.	Speed of the sealing unit Display of the current speed of the sealing unit The synchronism of the sealing unit to the product passing through can be monitored via the pointer position that displays the conveyor speed.
3.	Data logger of the measuring input Ring buffer of the data logger in which the positions of the detected products on conveyor belt can be entered by the measuring input.
4.	Write pointer of the measuring input The display selects the memory cell of the data logger in which the next position detected by the measuring input is entered.
5.	Read pointer of the sealing unit The display selects the memory cell of the data logger from which the measured value of the currently processed product is read. When the sealing process is completed, the read pointer is moved ahead by one memory cell. For reasons of clarity there is no display of the read pointer for the application of the material.
6.	Current position of the conveyor belt This is where the current position of the conveyor belt output by the technology object is displayed. This axis is a modulo axis. For this reason the display of the axis position is within a range of 0,000 to 500,000.
7.	Current position of the sealing unit This is where the currently position of the sealing unit, output by the technology object is displayed. This axis is a modulo axis. For this reason the display of the axis position is within a range of 0,000 to 110,000.

Particularities of the simulation

In order to be able to monitor the individual process steps on the technology objects without restrictions, the simulation has no restrictions or blocks in the operation.

If new products are, for example, requested in intervals that are too short, it may happed that the sealing unit is still processing the previous product and can therefore not carry out a synchronization on the currently requested product.

The product that follows at a short distance is detected by the measuring input and entered in the data logger. However, processing the product is only possible by applying the material, because this can be done fast enough to all products, due to the distance of the measuring input.

The synchronization position of the non-processed product is calculated correctly within the program. However, the appropriate motion of the sealing unit will only take place after the modulo jump of the conveyor belt axis when the calculated position passes the sealing unit again.

This behavior of the simulation is noticeable on the pointers of the data logger, as long as no new products are requested. If both pointers do not point to the same memory cell, after the product has fully passed through the machine, this means the sealing process could not be carried out, i.e. the sealing unit could not be synchronized.

9 "SINAMICS V90 PN" Demo Case

9.1 Overview of the demo case

9.1.1 Configuration of the case

The "SINAMICS V90 PN" demo case consists of two SINAMICS V90 drives with the appropriate servo motors, one SIMATIC-CPU 1515T-2 PN and the ET 200SP distributed I/O system with the respective operating elements, such as, toggle switches, potentiometers and an emergency stop button.

Figure 9-1 "SINAMICS V90 PN" demo case

The ET 200SP distributed I/O is configured with the following modules:

- Digital inputs DI 16x24V DC
- Digital outputs DQ 16x24V DC / 0.5A
- TM Timer DIDQ 10x24V technology module
- AI 2xU analog inputs

9.1.2 Control elements

The modules of the ET 200SP distributed I/O system are connected to the demo case with different operating and display elements, which can be used for demonstrations:

- Toggle switch to operate the digital inputs
- LEDs to signal the digital outputs

- Emergency stop button for the delayed interruption of safety circuit of the two drives. The delay can be set via an "E-Stop Delay" potentiometer.
- Potentiometer P1 and P2 to specify the voltage on the analog inputs of the AI 2xU module.
- Sensors on the two drive discs of the motors to detect the zero position. The sensors to be used as measuring input are connected with the channels 0 and 1 of the TM Timer DIDQ technology module.
- Toggle switch S15 to control channel 3 of the TM Timer DIDQ technology module as further measuring sensor input.
- LED between the drive discs of the two motors. The two discs are provided with a hole through which the LED can be activated when it is covered. With the help of the "Sync." switch, the LED can either be controlled permanently (+24V) or via channel 4 of the TM Timer DIDQ technology module to display the synchronism of the two drive discs.

Note The toggle switches have a switch and lock function and can be assigned via the switches located above switch field of the ET 200SP or via the direct inputs of the SINAMICS.

9.1.3 Wiring

The diagram below shows the standard wiring of the "SINAMICS V90 PN" demo case on which most application examples are based.

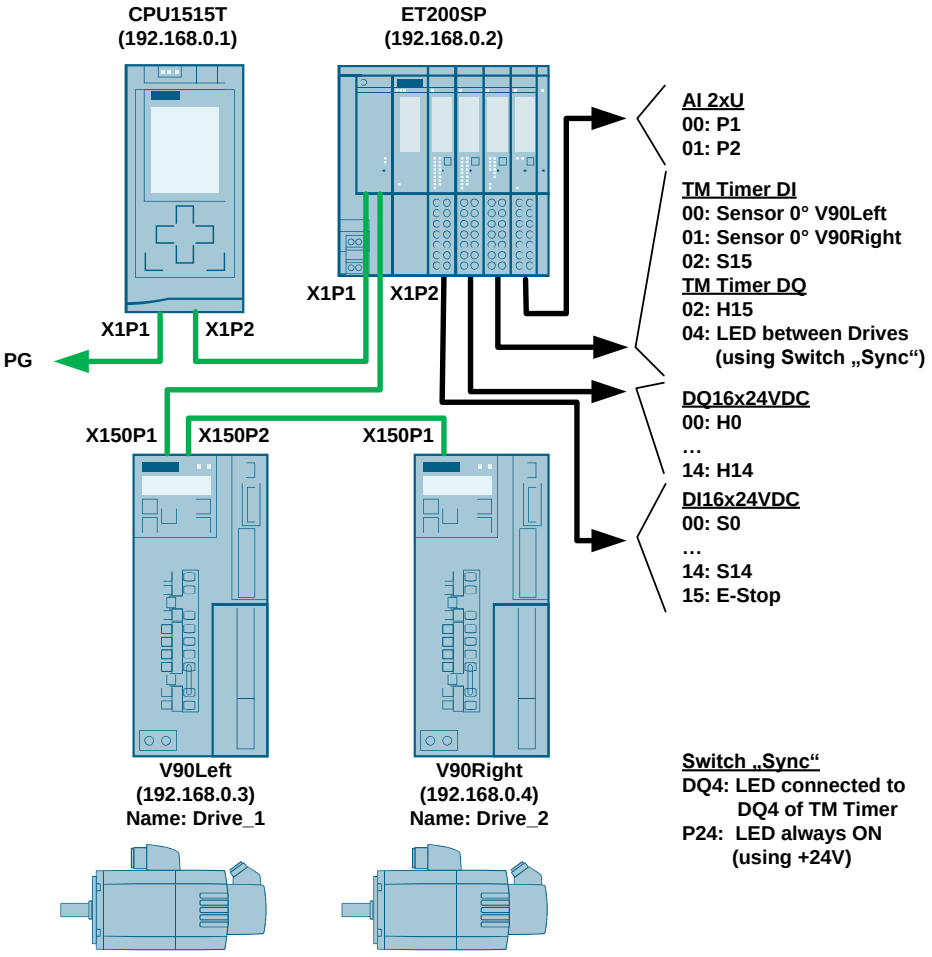
The following information regarding the creation of a user program in interaction with the demo case can be found in the diagram:

- IP addresses of the individual components of the demo case.
- PROFINET wiring of the demo case for the configuration of the topology in the TIA Portal.
- Naming of the individual components of the demo case.
- Wiring of the individual monitoring and display elements of the demo case, using the modules of the distributed I/O ET 200SP.

Note Before using the demo case with the application example, please ensure that the "SINAMICS V90 PN" demo case is wired as displayed in the diagram below.

Pay particular attention to the wiring of the PROFINET network, to avoid unnecessary errors when commissioning the application example.

Figure 9-2 Wiring of the "SINAMICS V90 PN" demo case



9.1.4 Functions

The following functions can be used via the operating elements of the demo case in interaction with the user program:

- Digital inputs via switch (S1…S15)
- Digital outputs via LED (H1…H15)
- Analog inputs 0…10V (left and right)
- Emergency stop directly wired to the drives and switched time-delayed via a settable delay element (0s...2s)
- Measuring input of the TM Timer DIDQ technology module via light barrier (axes) and toggle switch (S16)
- Cam output of the TM Timer DIDQ technology module via LED (H16) and LED between the axes

9.2 Putting the demo case into operation

9.2.1 Preparations

- Installing TIA Portal V14
- Installing HSP for SINAMICS V90 into the TIA Portal

9.2.2 Hardware configuration

Set up the hardware configuration in the TIA Portal as follows:

- Insert CPU
  - S7-1515T-2 PN
- Insert ET 200SP station
  - IM155-6PN HF head module
  - DI 16x24V DC ST input module
  - DQ 16x24V/0.5A ST output module
  - TM Timer DIDQ 10x24V ST technology module
  - AI 2xU ST analog input module
  - Server module to complete the ET 200SP station
- Add drive
  - Links: SINAMICS V90 PN, 1AC/3AC 200-240V 2,5A/1,5A, motor 100W
  - Right: SINAMICS V90 PN, 1AC/3AC 200-240V 2,5A/1,5A, motor 100W

Note Since the two drives of the demo case are identical drives, at first, only one drive can be added to the hardware configuration. This drive can then be put into operation and optimized. The second drive is then created by copying from the first drive into the hardware configuration. To do this, the copied drive only has to be assigned a new name.

9.2.3 Putting the drives into operation

Commission the added drives as follows:

- Assigning drive name: Assign PROFINET name und optionally IP address
- Establish online connection to the drive

Note If the drive cannot be commissioned as desired, it is recommended to reset the drive to factory settings and then to carry out commissioning of the drive again.

9.2.4 Optimizing drives

Optimize the speed control and the drive parameters as follows:

1. Click the tab for optimization
2. Connect online to the drive
3. Get the control priority from the drive
4. Start the automatic optimization

5. Apply the values in the drive
6. Permanently save the values in the drive (copy RAM to ROM)
7. Upload the values from the drive into the TIA Portal project.

9.3 Operating the application example

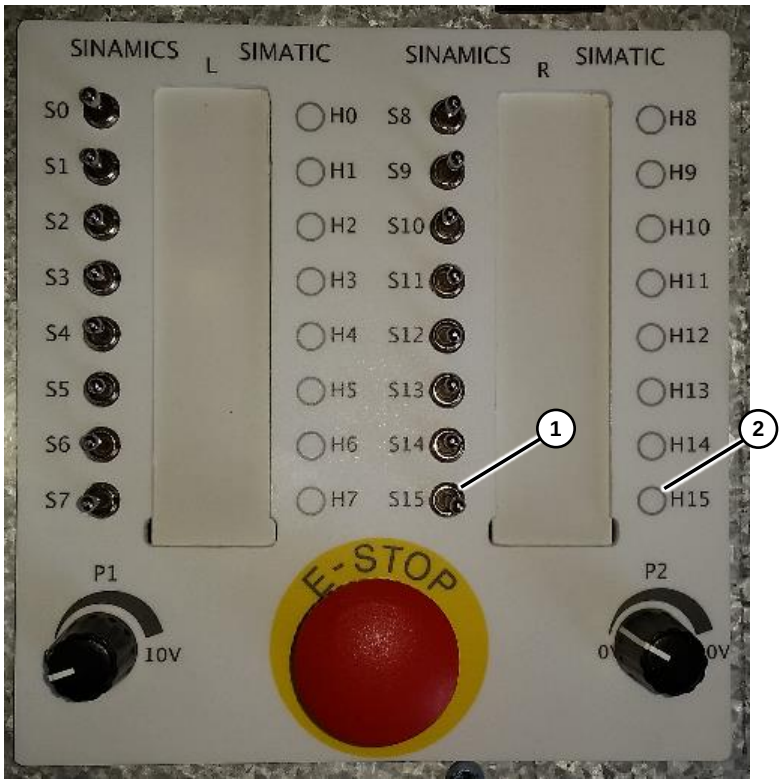
The operation of the example application corresponds to the approach introduced in chapter 8.5.

9.3.1 Differences of the simulation example application

The following differences of the example application of the "SINAMICS V90 PN" demo case exist:

- A new product is detected via the measuring input signal that is triggered via the pushbutton function of the toggle switch S15 (press gear lever to the right). This is why the "New product" button on the user interface is without function. The S15 (1) toggle switch is directly connected to channel 2 of the TM Timer DIDQ technology module of the demo case.
- The cam switch signal for controlling the value for applying the material is output via the LED H15 (2) of the demo case.

Figure 9-3 Operator panel of the "SINAMICS V90 PN" demo case



9.3.2 Functional differences

A measuring pulse (rising edge) is triggered via the S15 toggle switch on the TM Timer DIDQ technology module. This measurement is now tangibly performed on the demo case via the "MC_MeasuringInput" instruction. The currently detected

position of the positioning axis of the material line is transferred as product position to the data logger of the user program.

The following code section from the "GetMeasuringInput" function block shows the real data logging via the "MC_MeasuringInput" instruction.

```
//+++ Call function block +++
#instMeasuringInput(MeasuringInput:=#MeasuringInput,
                    Execute:=(#FunctionEnable AND
                                  NOT(#instMeasuringInput.Done)),
                    Mode:=0);

//+++ Realize machine function +++
IF (#instMeasuringInput.Done = True)
THEN
  //Get actual write index
  #tempWriteIndex :=
                 #ProductPositionTable.WriteIndex_MeasuringInput;
  //Save measured position of product on product belt
  #ProductPositionTable.PositionTable[#tempWriteIndex] :=
                              #instMeasuringInput.MeasuredValue1;
  //Generate next write index (write pointer)
  //Incement index pointer
  #tempWriteIndex := #tempWriteIndex + 1;
  //Check modulo area of index pointer
  IF (#tempWriteIndex > #ProductPositionTable.MaxIndex)
  THEN
    #tempWriteIndex := #ProductPositionTable.MinIndex;
  END_IF;
  //Write back index pointer
  #ProductPositionTable.WriteIndex_MeasuringInput :=
                                                  #tempWriteIndex;
END_IF;

//+++ Output singals +++
#NewProductDetected := #instMeasuringInput.Done;
```

Note

Due to the real measurement via the TM Timer technology module, it has to be done without a display of the products on the material line in front of the measuring input in the example application of the "SINAMICS V90 PN" demo case.

As soon as the measurement of the product position is triggered via the S15 toggle switch, can the product be displayed on the user interface.

10 Links & Literature

Table 10-1

	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Download page of the entry https://support.industry.siemens.com/cs/ww/en/view/109743134
\3\	FAQ Isochronous mode with PROFINET - an example with SIMATIC S7-1500 https://support.industry.siemens.com/cs/ww/en/view/109480489
\4\	FAQ How Can You Integrate a Drive into the TIA Portal via the Device Master File (GSD)? https://support.industry.siemens.com/cs/ww/en/view/73257075
\5\	Application example / Film Creating and using user-defined web pages on S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/en/view/68011496
\6\	Application example Migration Guide: S7-31xT to S7-1500(T) https://support.industry.siemens.com/cs/ww/en/view/109743136
\7\	Manual SIMATIC S7-1500 CPU 1516-3 PN/DP https://support.industry.siemens.com/cs/ww/en/view/59191914
\8\	Manual SIMATIC S7-1500 CPU 1515T-2 PN https://support.industry.siemens.com/cs/ww/en/view/109739189
\9\	Manual SIMATIC S7-1500 S7-1500T Motion Control V3.0 in TIA Portal V14 https://support.industry.siemens.com/cs/ww/en/view/109481326
\10\	Download Support packages for the hardware catalog in TIA Portal (HSP) https://support.industry.siemens.com/cs/ww/en/view/72341852

11 History

Table 11-1

Version	Date	Modifications
V1.0	07/2017	First version