# SoMachine

## Packaging Application Functions
## Packaging Library Guide

06/2017

EIO0000000195.09

www.schneider-electric.com

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

# Table of Contents

# Safety Information

## Important Information

### NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

**⚠ DANGER**

**DANGER** indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

**⚠ WARNING**

**WARNING** indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

**⚠ CAUTION**

**CAUTION** indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

***NOTICE***

***NOTICE*** is used to address practices not related to physical injury.

## PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

## BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

### ⚠ WARNING

**UNGUARDED EQUIPMENT**

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

**NOTE:** Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

## START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

**⚠ WARNING**

**EQUIPMENT OPERATION HAZARD**

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

**Software testing must be done in both simulated and real environments.**

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

## OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

# About the Book

## At a Glance

### Document Scope

This document describes the functions of the Packaging Library.

### Validity Note

This document has been updated with the release of SoMachine V4.3.

### Related Documents

| Title of Documentation | Reference Number |
|---|---|
| SoMachine - Miscellaneous Functions, Toolbox Library Guide | *EIO0000000096* |

You can download these technical publications and other technical information from our website at http://www.schneider-electric.com/en/download

### Product Related Information

**⚠ WARNING**

**LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

# ⚠ WARNING

**UNINTENDED EQUIPMENT OPERATION**

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

| Standard | Description |
|---|---|
| EN 61131-2:2007 | Programmable controllers, part 2: Equipment requirements and tests. |
| ISO 13849-1:2008 | Safety of machinery: Safety related parts of control systems.<br>General principles for design. |
| EN 61496-1:2013 | Safety of machinery: Electro-sensitive protective equipment.<br>Part 1: General requirements and tests. |
| ISO 12100:2010 | Safety of machinery - General principles for design - Risk assessment and risk reduction |
| EN 60204-1:2006 | Safety of machinery - Electrical equipment of machines - Part 1: General requirements |
| EN 1088:2008<br>ISO 14119:2013 | Safety of machinery - Interlocking devices associated with guards - Principles for design and selection |
| ISO 13850:2006 | Safety of machinery - Emergency stop - Principles for design |
| EN/IEC 62061:2005 | Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems |
| IEC 61508-1:2010 | Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements. |
| IEC 61508-2:2010 | Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems. |
| IEC 61508-3:2010 | Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements. |
| IEC 61784-3:2008 | Digital data communication for measurement and control: Functional safety field buses. |

| Standard | Description |
|---|---|
| 2006/42/EC | Machinery Directive |
| 2014/30/EU | Electromagnetic Compatibility Directive |
| 2014/35/EU | Low Voltage Directive |

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

| Standard | Description |
|---|---|
| IEC 60034 series | Rotating electrical machines |
| IEC 61800 series | Adjustable speed electrical power drive systems |
| IEC 61158 series | Digital data communications for measurement and control – Fieldbus for use in industrial control systems |

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

**NOTE:** The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

# Part I
## Packaging Library System Requirements

# Chapter 1
# System Requirements

## At a Glance

This chapter describes the system requirements for the Packaging Library function blocks.

## What Is in This Chapter?

This chapter contains the following topics:

# Packaging System Requirements

## Overview

This chapter describes the hardware and the software requirements, and user interface compatibility for packaging library functions.

For environments, refer to individual function block chapters.

## System Requirements

This table describes the system requirements for the following application function blocks:

| Application Function Block | Requirements |
|---|---|
| • `TemperatureControl`<br>• `TemperatureControl_Easy`<br>• `LateralPositionControl` | • Logic Controller or<br>• Motion Controller |
| • `MoveJog`<br>• `XYPickAndPlace` | • Logic Controller<br>• Lexium servo drive |
| • `DigitalTensionControlATV`<br>• `AnalogTensionControlATV` | • Logic Controller<br>• Lexium servo drive<br>• Altivar variable speed drive |
| • `DigitalTensionControlATV_Motion`<br>• `AnalogTensionControlATV_Motion` | • Motion Controller<br>• Lexium servo drive<br>• Altivar variable speed drive |
| • `RotaryKnife_Motion`<br>• `FlyingShear_Motion`<br>• `GroupingAccumulator_Motion`<br>• `GroupingStripper_Motion`<br>• `Clamping_Motion`<br>• `AnalogTensionControlILXM_Motion` | • Motion Controller<br>• Lexium servo drive |

The targeted controller for the segment Packaging are:
- M258 Logic Controller
- LMC058 Motion Controller
- LMC078 Motion Controller

The preferred drives are:
- LXM32 Lexium servo drive
- ATV32 Altivar variable speed drive
- ATV71 Altivar variable speed drive

## Communication Interface

This table describes the communication interface requirements for the following application function blocks:

| Application Function Block | Requirements |
|---|---|
| • TemperatureControl<br>• TemperatureControl_Easy<br>• LateralPositionControl | No communication interface required. |
| • MoveJog<br>• XYPickAndPlace | CANopen protocol is used for communication with servo drives. |
| • DigitalTensionControlATV<br>• AnalogTensionControlATV | CANopen protocol is used for communication with variable speed drive and servo drive. |
| • DigitalTensionControlATV_Motion<br>• AnalogTensionControlATV_Motion | CANopen protocol is used for communication with variable speed drive and CANmotion or Sercos protocol is used for communication with servo drive. |
| • Clamping_Motion | CANmotion protocol is used for communication with servo drives. |
| • RotaryKnife_Motion<br>• FlyingShear_Motion<br>• GroupingAccumulator_Motion<br>• GroupingStripper_Motion<br>• AnalogTensionControlILXM_Motion | CANmotion or Sercos protocol is used for communication with servo drives. |

**NOTE:** The LMC078 Motion Controller is using the third generation of Sercos.

## User Interface

All parameters can be supplied and modified using Magelis HMI which can be used optionally.

## Using the Library

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

- Verify the SoMachine libraries contained in your program are the correct version after updating SoMachine software.
- Verify that the library versions updated are consistent with your application specifications.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

For more detailed information, see Schneider Electric Libraries.

For IEC 61131-3 compatibility, the ability to add the EN/ENO input/output automatically to Function Blocks of certain programming languages is available to the programmer. However, for certain applications that require the complex interaction of multiple function blocks, the use of the IEC 61131-3 input to disable a function block in a series of interrelated functions affecting a process may lead to unintended operation of the system as a whole. For the functions contained in the Library that is the topic of the current document, this is especially true.

The EN/ENO inputs and outputs as defined by IEC 61131-3 are maladapted to, and therefore inappropriate for, the targeted application of these functions. Suddenly disabling one function by a falling edge on the EN input would require all outputs of the function block to immediately fall to their default states, and such an unanticipated action would cause in abrupt change to the entire process. The implication is that such an event would have deleterious results that may invoke undesirable consequences. Therefore, the EN/ENO inputs/outputs as defined by IEC 61131-3 are incompatible with the functions contained within this library.

**⚠ WARNING**

**UNINTENDED MACHINE OPERATION**

Do not use the EN/ENO functionality defined by IEC 61131-3 to control the behavior of the Application Function blocks.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**NOTE:** Verify that the EN/ENO option is disabled in the compiler options menu of SoMachine.

# Function Block Location in the Packaging Library

## Function Block Location in the Packaging Library of SoMachine

| Folder | Sub-Folder | Application Function Block |
|---|---|---|
| Clamping | | `Clamping_Motion` |
| FlyingShear | | `FlyingShear_Motion` |
| Grouping Ungrouping | | `GroupingAccumulator_Motion`<br>`GroupingStripper_Motion` |
| LateralPositionControl | | `LateralPositionControl` |
| PickAndPlace | | `MoveJog`<br>`XYPickAndPlace` |
| RotaryKnife | | `RotaryKnife_Motion` |
| TemperatureControl | | `TemperatureControl`<br>`TemperatureControl_Easy` |
| TensionControl | AnalogTensionControl | `AnalogTensionControlATV`<br>`AnalogTensionControlATV_Motion`<br>`AnalogTensionControlLXM_Motion` |
| | DigitalTensionControl | `DigitalTensionControlATV`<br>`DigitalTensionControlATV_Motion` |

# Part II
# PickAndPlace

## What Is in This Part?

This part contains the following chapters:

# Chapter 2
# MoveJog: Manual Control of a Single Axis in Different Modes

## What Is in This Chapter?

This chapter contains the following sections:

# Section 2.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Functional Description

The `MoveJog` function block controls a single axis while operating in Manual mode.

## Usage of `MoveJog` Function Block

The MoveJog function is required for moving a single axis in speed mode, position mode and to home the axis.

## Solution with the `MoveJog` Function Block

The MoveJog function provides the following modes for operating a single axis in Manual mode:

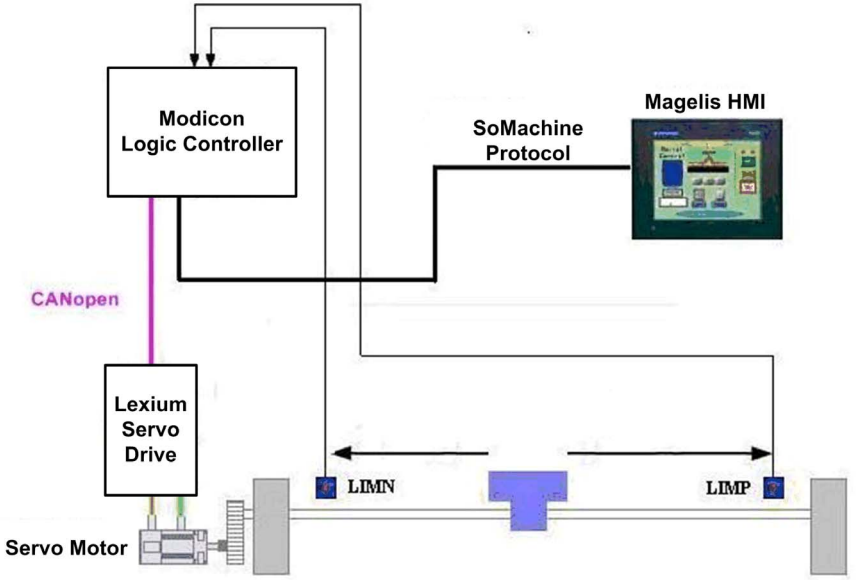
- Speed mode
- Position mode
- Homing mode

## Design & Realization Constraints and Assumptions

- Modulo Axis is run as infinite rotation with the Rotary Axis setting at the controller configuration and there is no hardware/software limit checking.
- Homing is possible when Modulo Axis is selected.

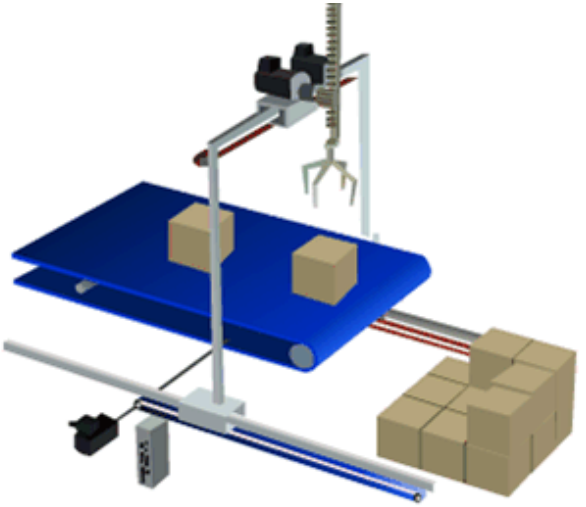
## Functional View for Linear axis

Functional view for linear servo axis on CANopen:



# Machine Overview

## Machine View



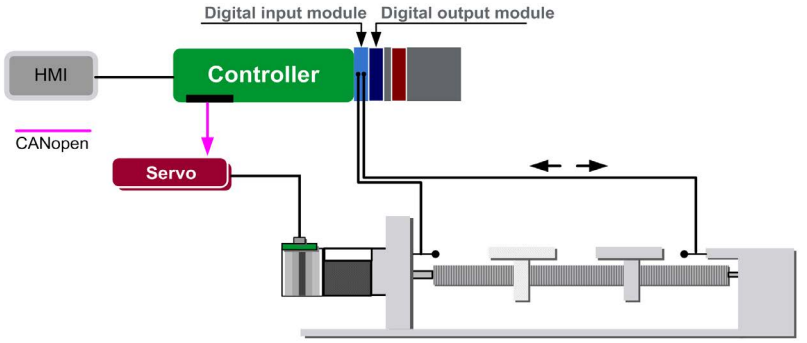
# Section 2.2
# Architecture

## What Is in This Section?

This section contains the following topics:

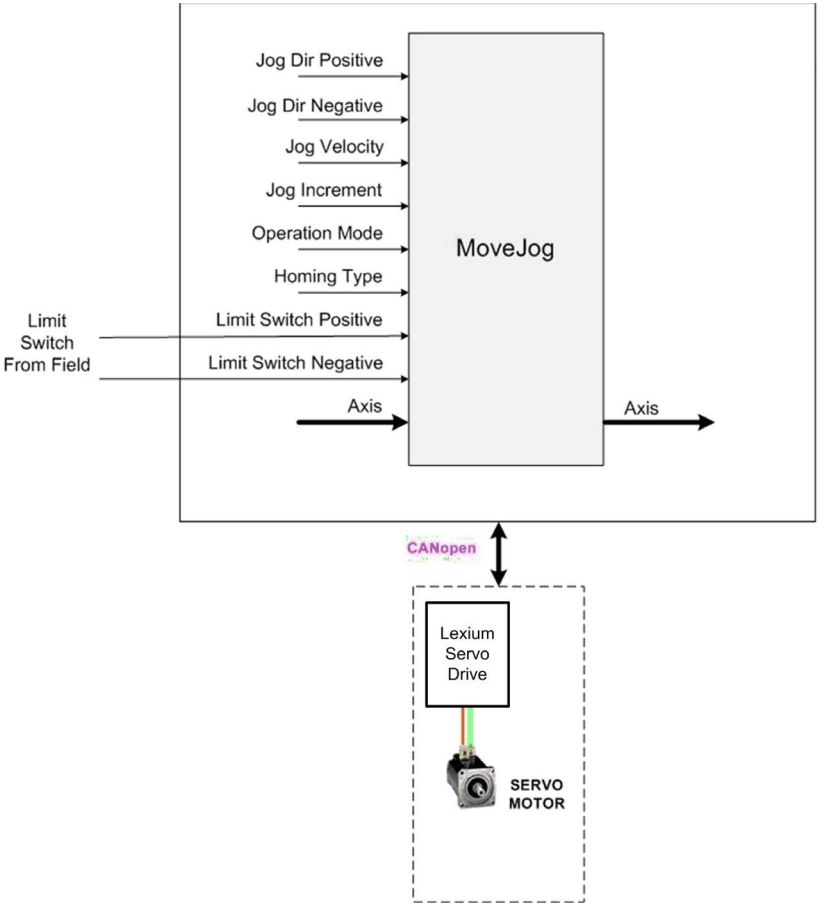
# Hardware Architecture

## Hardware Architecture Overview



# Software Architecture

## DataFlow Overview



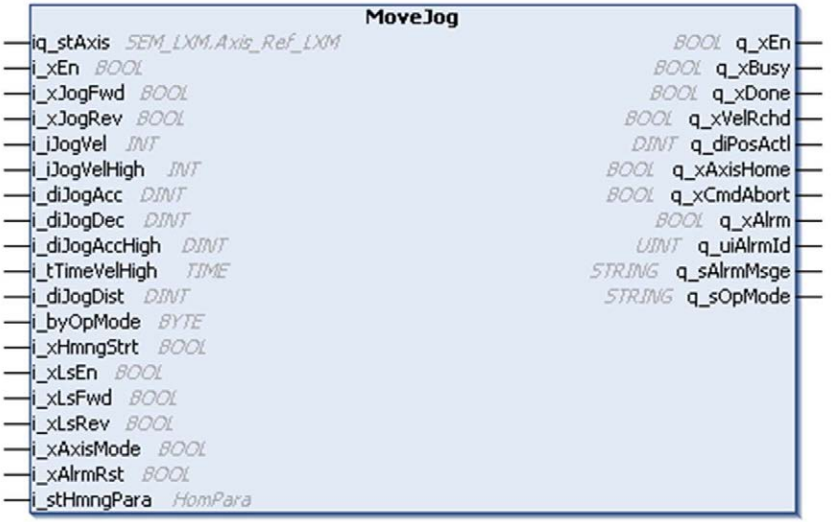
# Section 2.3
## Function Block Description

### What Is in This Section?

This section contains the following topics:

# MoveJog Function Block

## Pin Diagram




## Function Block Description

The `MoveJog` function block is used for manual control of a single Axis.

This function block includes three modes:

- Speed Mode: the direction and movement commands are given using a single Jog Button. The axis moves continuously in a specified direction as long as the Jog command is active.
- Position Mode: the direction and movement commands are given using a single Jog button. The axis moves up a specified distance with a specified direction.

  **NOTE:** Axis must be homed using Homing mode before the `MoveJog` function block can be used in Position Mode.
- Homing Mode: after a start Homing command, the axis moves according to the Homing method selected.

  **NOTE:** Select correct Homing method and parameter for the application. Improper setup or cabling may cause unintended equipment action.

| ⚠ WARNING |
| --- |
| **UNINTENDED EQUIPMENT OPERATION** |
| Verify that all configuration parameters are correct and validate cabling before commissioning. |
| **Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

## Operating Modes

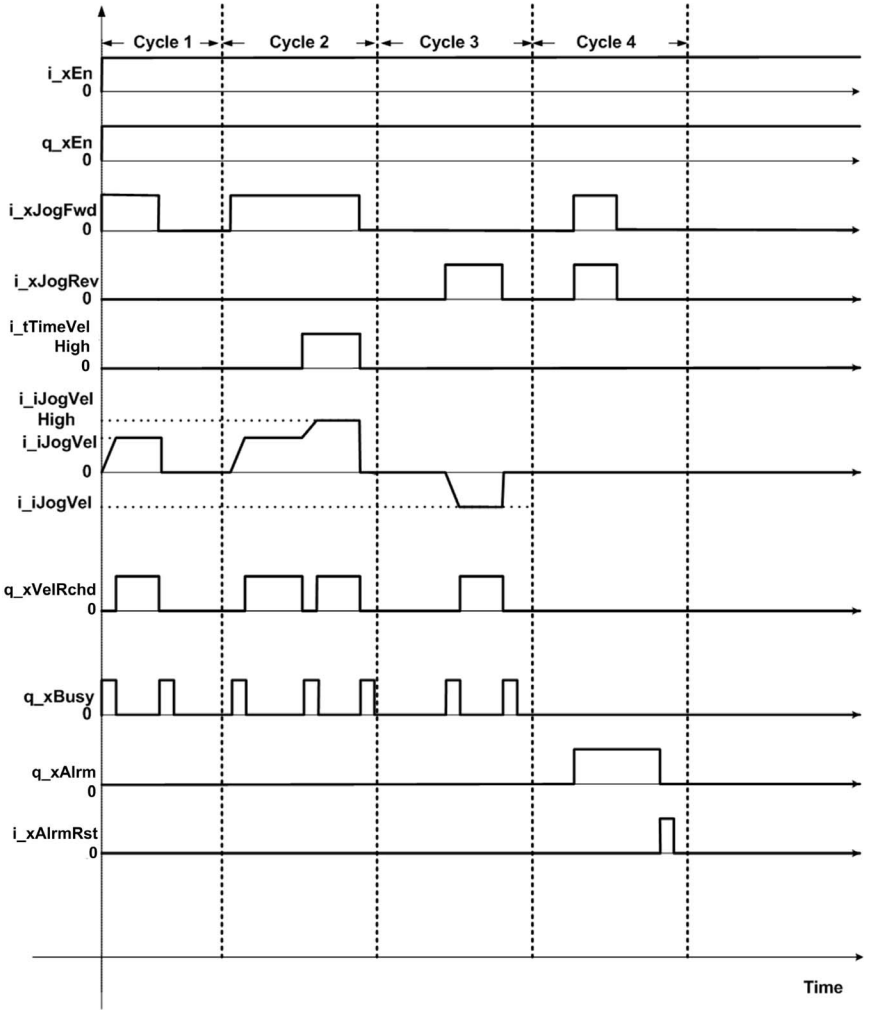
### Speed Mode: i_byOpMode=0

In Speed Mode, when the function block is enabled, enabling one of the directional inputs i_xJogFwd or i_xJogRev starts the axis of motion in the desired direction. The specified axis is then accelerated at the specified acceleration i_diJogAcc to the specified velocity i_iJogVel. As long as there are no changes in the state of the direction inputs, the axis moves in the corresponding direction. The motion is stopped by a falling edge at the respective direction input.

After i_tTimeSwcVelHigh has elapsed in the speed jog mode, the normal i_iJogVel and i_diJogAcc are increased to i_iJogVelHigh and i_diJogAccHigh as shown in cycle 2 in the Timing diagram figure below.

If both the direction inputs i_xJogFwd and i_xJogRev are TRUE at same time then the axis is stopped and displays a detected error. The detected error resets only when one or both of the direction input is FALSE and the rising edge of i_xAlrmRst is given to function block as shown in cycle 4 in the timing diagram figure below.

This figure shows the timing diagram for `MoveJog` function block in Speed Mode:



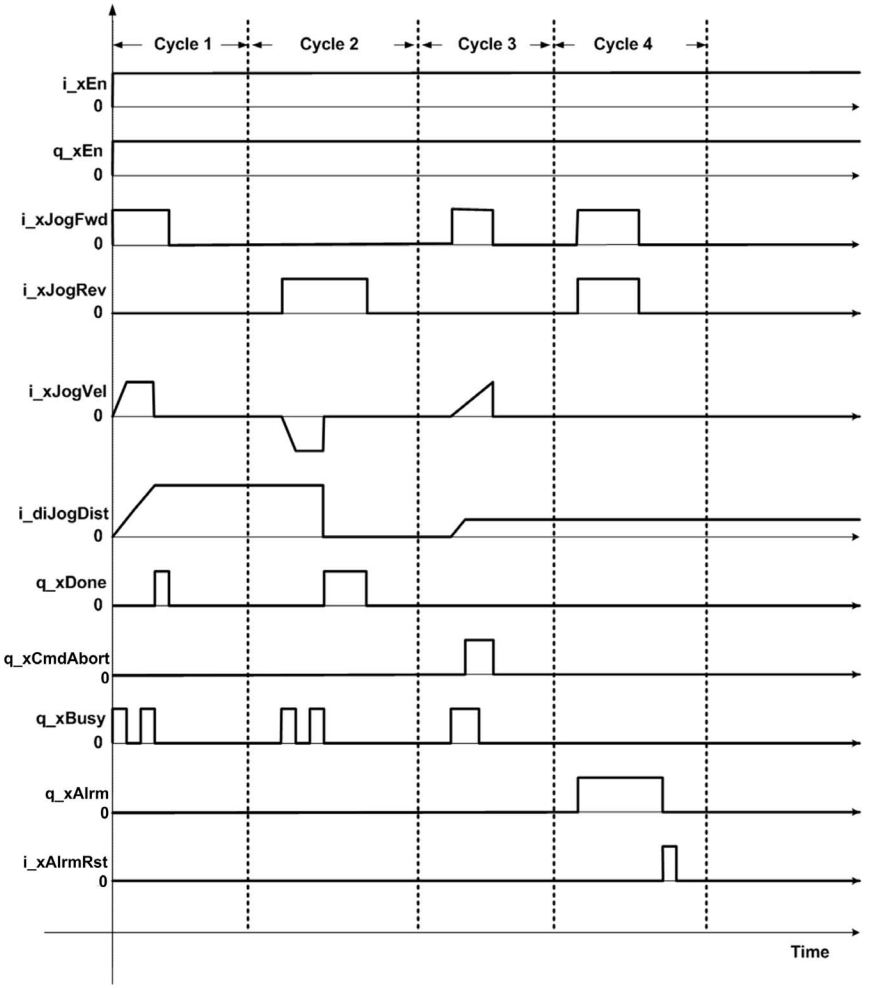

## Position Mode: `i_byOpMode=1`

In Position Mode, when the function block is enabled, an enable signal at one of the direction bits `i_xJogFwd` or `i_xJogRev` results in a single motion over a defined path `i_diJogDist` in the desired direction. The motion is accelerated at the specified acceleration `i_diJogAcc` up to the level `i_iJogVel`.

The motion is aborted by a falling edge at the respective direction bit before reaching distance as shown in cycle 3 in the timing diagram figure below.

If both the direction inputs `i_xJogFwd` and `i_xJogRev` are TRUE at same time then the axis does not move and the function block displays a detected error. The detected error is reset only when one or both of the direction inputs is FALSE and rising edge of `i_xAlrmRst` is given to function block as shown in cycle 4 in the timing diagram figure below.

This figure shows the timing diagram for the `MoveJog` function block in Position Mode:



### Homing Mode: i_byOpMode=2

In Homing Mode, i_xHmngStrt starts the Homing operation according to the Homing method. After the completion of the Homing operation, the output q_xAxisHome is set to TRUE. During Homing, additional rising edge at i_xHmngStrt results in a detected Axis error.

**NOTE:** In Homing Mode, Jog command is not applicable.

# Section 2.4
## Pin Description

### What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Enables function block and the parameters are validated.<br>FALSE: Disables function block and all outputs are set to zero. The axis movement is stopped. |
| i_xJogFwd | BOOL | TRUE: The jog motion starts in the positive/forward direction.<br>FALSE: During jog in forward direction the axis movement is stopped.<br>● i_xJogFwd must be TRUE up to the required position.<br>● This input is only applicable to Speed and Position modes.<br>● When i_xJogFwd and i_xJogRev are TRUE at the same time, the q_xAlrm is TRUE and the axis is stopped with a detected error. |
| i_xJogRev | BOOL | TRUE: Negative/reverse direction<br>FALSE: During jog in reverse direction the axis movement is stopped.<br>● i_xJogRev must be TRUE up to the desired position.<br>● This input is only applicable in Speed and Position modes.<br>● When i_xJogFwd and i_xJogRev are TRUE at the same time, the q_xAlrm is TRUE and the axis is stopped with a detected error. |
| i_iJogVel | INT | In position mode, the axis moves only with i_iJogVel velocity.<br>In Speed Mode the axis moves with i_iJogVel velocity until the i_tTimeVelHigh elapses.<br>Range: 0...32000 RPM<br>Factory setting: 20 |
| i_iJogVelHigh | INT | Jog velocity after the timer elapsed (i_tTimeVelHigh)<br>Range: 0...32000 RPM<br>Factory setting: 20<br>**NOTE:** Only applicable in Speed Mode |

| Input | Data Type | Description |
|---|---|---|
| i_diJogAcc | DINT | In position mode, the axis moves with i_diJogAcc acceleration.<br>In Speed Mode, the axis moves with i_diJogAcc acceleration until the i_tTimeVelHigh elapses.<br>Range: 30...3000000 RPM/s<br>Factory setting: 100 |
| i_diJogDec | DINT | The axis stops with deceleration time<br>Range: 750...3000000 RPM/s<br>Factory setting: 900 |
| i_diJogAccHigh | DINT | The axis moves with i_diJogAcc acceleration after the i_tTimeVelHigh elapses.<br>Range: 30...3000000 RPM/s<br>Factory setting: 300<br>**NOTE:** This is only applicable in Speed Mode |
| i_tTimeVelHigh | TIME | This is the time to switch from i_iJogVel to i_iJogVelHigh and from i_diJogAcc to i_diJogAccHigh.<br>Range: 0...4194967295 ms<br>0: disables timer function<br>Factory setting: 10 s<br>● i_tTimeVelHigh is applicable for Speed Mode.<br>● i_tTimeVelHigh timer is reset after Jog command FALSE or i_xEn is FALSE.<br>● When i_tTimeVelHigh = 0, axis moves only with i_iJogVel and i_diJogAcc. |
| i_diJogDist | DINT | Jog distance value<br>Range: 1...2147483647 user unit<br>● i_diJogDist is only applicable in position mode.<br>● Jog distance should be more than zero. If i_diJogDist is zero, then axis will not move and position output q_xDone is TRUE. |
| i_byOpMode | BYTE | 0: Speed Mode<br>1: Positioning Mode<br>2: Homing Mode |
| i_xHmngStrt | BOOL | TRUE: Starts the homing operation according to homing type selected. After the completion of homing operation, the output q_xAxisHome is set to zero.<br>● In homing mode, Jog command is not applicable.<br>● Homing is not possible with modulo axis. |
| i_xLsEn | BOOL | TRUE: Enables the hardware limit switch functionality inside the function block. |

| Input | Data Type | Description |
|---|---|---|
| i_xLsFwd | BOOL | TRUE: Forward / positive direction hardware limit switch (factory setting)<br>If enabled and forward / positive direction hardware limit switch goes low, then axis stops with positive hardware limit switch active notification.<br>**NOTE:** Limit switch is configured as normally closed. |
| i_xLsRev | BOOL | TRUE: Reverse / negative direction hardware limit switch (factory setting)<br>If enabled and reverse / negative direction hardware limit switch goes low, then axis stops with negative hardware limit switch active notification.<br>**NOTE:** Limit switch is configured as normally closed. |
| i_xAxisMode | BOOL | TRUE: Modulo axis configuration is selected and hardware/software limit switch functionality is disabled.<br>FALSE: Linear axis configuration is selected (factory setting).<br>Procedure for axis configuration:<br>● i_xEn input must be TRUE<br>● Drive must be in ready state<br>● Change the i_xAxisMode input as per the requirements<br>● Set drive to run mode and move the axis. |
| i_xAlrmRst | BOOL | TRUE: Reset the detected alarm on a rising edge. |
| i_stHmngPara | HomPara | Structure for homing parameter<br>Refer to i_stHmngPara *(see page 48)*. |

## Structured Parameter

### i_stHmngPara

This table describes the mandatory parameters for the `i_stHmngPara` inputs of the `MoveJog` function block:

| Structure Parameter | Data Type | Description |
|---|---|---|
| `diPos` | `DINT` | Absolute position when the reference signal is detected<br>In Homing mode, after successful reference movement, this position value is automatically set at the axis stop position.<br>Range: 1...2147483647 User Unit |
| `uiHmngMode` | `UINT` | Select Homing method<br>See description in table below |
| `uiHmngVel` | `UINT` | Velocity for search of reference switch<br>Range: 1...13200 RPM |
| `uiHmngVelOut` | `UINT` | Velocity for movement back to edge of reference switch<br>Range: 1...3000 RPM |
| `diHmngPosOut` | `DINT` | Maximum distance for movement back to edge of reference switch<br>Range: 0...2147483647 User Unit<br>**NOTE:** The hardware limit switch must be disabled again inside this run-off; otherwise the reference movement is aborted. |
| `diHmngDistPos` | `DINT` | Positioning from edge of reference switch<br>Range: 1...2147483647 User Unit<br>**NOTE:** After leaving the switch, the drive is still positioned in the working range for defined path and this position is defined as a reference point. |

### uiHmngMode

This table describes the methods supported by the `MoveJog` function block:

| Method Number | Description |
|---|---|
| 1 | Limit switch Negative with index pulse |
| 2 | Limit switch Positive with index pulse |
| 7 | Search Reference switch in Positive direction with index pulse, invert direction in switch, index pulse/distance outside Switch |

| Method Number | Description |
|---|---|
| 8 | Search Reference switch in Positive direction with index pulse, invert direction in switch, index pulse/distance inside Switch |
| 9 | Search Reference switch in Positive direction with index pulse, direction in switch not inverted, index pulse/distance inside Switch |
| 10 | Search Reference switch in Positive direction with index pulse, direction in switch not inverted, index pulse/distance outside Switch |
| 11 | Search Reference switch in Negative direction with index pulse, invert direction in switch, index pulse/distance outside Switch |
| 12 | Search Reference switch in Negative direction with index pulse, invert direction in switch, index pulse/distance inside Switch |
| 13 | Search Reference switch in Negative direction with index pulse, direction in switch not inverted, index pulse/distance inside Switch |
| 14 | Search Reference switch in Negative direction with index pulse, direction in switch not inverted, index pulse/distance outside Switch |
| 17 | Search Limit switch Negative |
| 18 | Search Limit switch Positive |
| 23 | Search Reference switch in Positive direction, invert direction in switch, index pulse/distance outside Switch |
| 24 | Search Reference switch in Positive direction, invert direction in switch, index pulse/distance inside Switch |
| 25 | Search Reference switch in Positive direction, direction in switch not inverted, index pulse/distance inside Switch |
| 26 | Search Reference switch in Positive direction, direction in switch not inverted, index pulse/distance outside Switch |
| 27 | Search Reference switch in Negative direction, invert direction in switch, index pulse/distance outside Switch |
| 28 | Search Reference switch in Negative direction, invert direction in switch, index pulse/distance inside Switch |
| 29 | Search Reference switch in Negative direction, direction in switch not inverted, index pulse/distance inside Switch |
| 30 | Search Reference switch in Negative direction, direction in switch not inverted, index pulse/distance outside Switch |
| 33 | Reference movement to the index pulse in Negative direction |
| 34 | Reference movement to the index pulse in Positive direction |
| 35 | Set dimensions (Direct Homing without movement and set required position to current position) |

# Input/Output Pin Description

## Input/Output Pin Description

### Input/Output Pin Description

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stAxis` | `SEM_LXM.Axis_Ref_LXM` | Servo-axis reference structure |

# Output Pin Description

## Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xEn | BOOL | TRUE: Function block enabled. If input i_xEn is TRUE, then output q_xEn is TRUE. |
| q_xBusy | BOOL | TRUE: Axis is accelerating or decelerating. |
| q_xDone | BOOL | Only in position mode.<br>TRUE: Axis moved by jog distance. |
| q_xVelRchd | BOOL | Only in speed mode.<br>TRUE: Velocity reaches jog velocity (i_iJogVel) or jog velocity high (i_iJogVelHigh). |
| q_diPosActl | DINT | Actual position of the axis. |
| q_xAxisHome | BOOL | Only in homing mode.<br>TRUE: Axis homing operation is complete and, after that, q_xAxisHome is TRUE in all modes.<br>**NOTE:** This resets after a controller power on/off or a detected axis error. |
| q_xCmdAbort | BOOL | TRUE: Command aborted<br>In the event that the motion is aborted due to external command (for example: detected communication error, Stop command issued before the completion of a movement, function block disabled while axis homing) i_xCmdAbort is set to TRUE. |
| q_xAlrm | BOOL | TRUE: Detected alarm<br>FALSE: No alarm |
| q_uiAlrmId | UINT | ID detected alarm<br>Range: 0, 1, 10, 30, 101, 102, 103, 105, 350-359.<br>Refer to Notification table *(see page 52)*. |
| q_sAlrmMsge | STRING | Refer to Notification table *(see page 52)*. |
| q_sOpMode | STRING | Output displays the function block current operating mode. |

## Notifications

| Detected Alarm ID | Notification |
|---|---|
| 0 | Okay |
| 1 | Internal function block detected alarm |
| 10 | Detected Axis alarm |
| 30 | PLCopen function block detected alarm |
| 101 | Invalid Acceleration Parameter |
| 102 | Invalid Deceleration Parameter |
| 103 | Invalid Velocity Parameter |
| 105 | Invalid Target Position Parameter |
| 350 | Invalid Operation Mode parameter |
| 351 | Invalid Absolute Position Parameter |
| 352 | Invalid Homing Mode Parameter |
| 353 | Invalid Reference Searching Velocity Parameter |
| 354 | Invalid Reference Out Velocity Parameter |
| 355 | Invalid Maximum Distance Parameter |
| 356 | Invalid Distance from Reference Parameter |
| 357 | Jog Positive and Jog Negative are Active |
| 358 | Positive Limit Switch Active |
| 359 | Negative Limit Switch Active |

### Alarm message:

`q_sAlrmMsge`: Axis not Homed! Position Mode not Allowed

# Section 2.5
## Quick Reference Guide

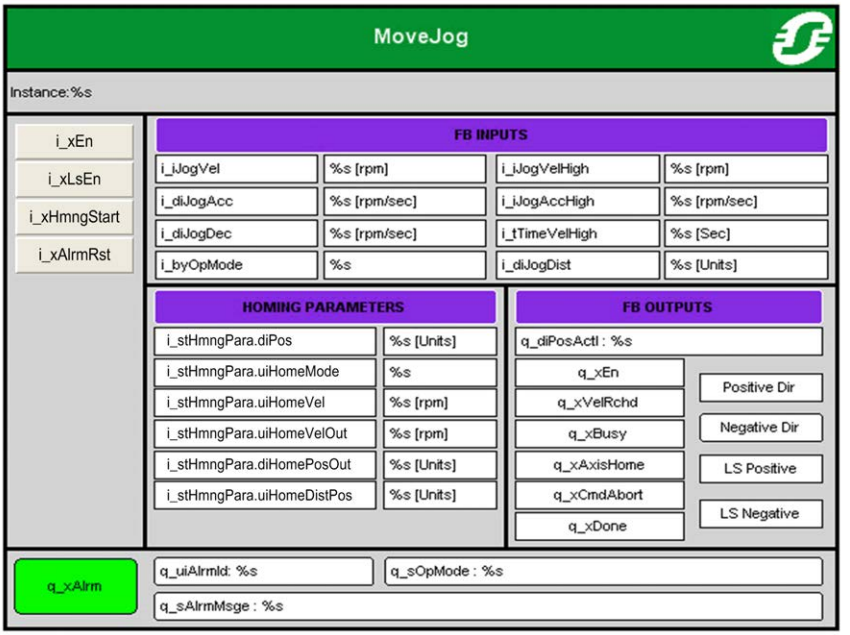
### What Is in This Section?

This section contains the following topics:

# Function Block Visualization

## Visualization

This figure shows the visualization of the `MoveJog` function block:



# Quick Commissioning Procedure

## Configuration Procedure

This table explains the procedure to configure the `MoveJog` function blocks:

| Step | Action |
|---|---|
| 1 | Add the supporting Packaging Library into the application program. |
| 2 | Configure the axis on CANopen Master. |
| 3 | Enter all parameters in controller configuration. |
| 4 | Use `MC_Power_LXM` for powering up the drive and `MC_Reset_LXM` for resetting an axis error (both included in Lexium library based on PLCopen). |
| 5 | Load the MoveJog function application on controller. |
| 6 | Power on the axis. |
| 7 | Enable `MC_Power_CAN` and `MoveJog` function blocks and enter all parameters as per requirements within range in the following tables. |
| 8 | In Speed mode, directly give the jog command for movement. |
| 9 | For Position mode, first complete the homing operation of axis in homing mode. |
| 10 | Start the FB. |

## Parameterizing in Speed Mode

This table lists the suggested parameters for speed mode:

| Parameter | Example Value |
|---|---|
| i_xEn | TRUE |
| i_xJogFwd | TRUE |
| i_xJogRev | FALSE |
| i_iJogVel | 100 |
| i_iJogVelHigh | 500 |
| i_diJogAcc | 100 |
| i_diJogDec | 1000 |
| i_diJogAccHigh | 500 |
| i_tTimeVelHigh | 1000 |
| i_diJogDist | 0 |
| i_byOpMode | 0 |
| i_xHmngStrt | FALSE |
| i_xLsEn | TRUE |
| i_xLsFwd | TRUE |
| i_xLsRev | TRUE |
| i_xAxisMode | FALSE |
| i_xAlrmRst | FALSE |

**NOTE:** The above values were tested in laboratory conditions and are for illustrative purposes only. Appropriate values should be entered as per the actual machine setup.

## Parameterizing in Position Mode

This table lists the suggested parameters for position mode:

| Parameter | Example Value |
|---|---|
| i_xEn | TRUE |
| i_xJogFwd | TRUE |
| i_xJogRev | FALSE |
| i_iJogVel | 100 |
| i_iJogVelHigh | 500 |
| i_diJogAcc | 100 |
| i_diJogDec | 1000 |
| i_diJogDist | 10000 |
| i_byOpMode | 1 |
| i_xHmngStrt | FALSE |
| i_xLsEn | TRUE |
| i_xLsFwd | TRUE |
| i_xLsRev | TRUE |
| i_xAxisMode | FALSE |
| i_xAlrmRst | FALSE |

**NOTE:** The above values were tested in laboratory conditions and are for illustrative purposes only. Appropriate values should be entered as per the actual machine setup.

## Parameterizing in Homing Mode

This table lists the suggested parameters for homing mode:

| Parameter | Example Value |
|---|---|
| i_xEn | TRUE |
| i_byOpMode | 2 |
| i_xHmngStrt | FALSE |
| i_xAxisMode | FALSE |
| i_xAlrmRst | FALSE |
| i_stHmngPara.diPos | 0 |
| i_stHmngPara.uiHmngMode | 17 |
| i_stHmngPara.uiHmngVel | 50 |
| i_stHmngPara.uiHmngVelOut | 20 |
| i_stHmngPara.diHmngPosOut | 0 |
| i_stHmngPara.diHmngDistPos | 100 |

**NOTE:** The above values were tested in laboratory conditions and are for illustrative purposes only. Appropriate values should be entered as per the actual machine setup.

# Troubleshooting

## Troubleshooting

This table shows some general issues and their solutions:

| Issue | Cause | Solution |
|---|---|---|
| Parameter invalid | Function block detected error. | 1.Remove cause of detected error and acknowledge with `i_xAlrmRst`<br>2.Restart the function block. |
| Inputs invalid | The axis movement is stopped and the detected error output is set. | 1.Remove cause of detected error and acknowledge with `i_xAlrmRst` |
| Axis error detected | The axis movement is stopped and the detected error output is set. | 1.Remove cause of detected error<br>2.Execute the axis reset using `MC_Reset` function block.<br>3.Restart the function block. |
| `q_uiAlrmId` = 1 | Internal detected error | Detected error other than the ones above.<br>Restart the function block. |
| `q_uiAlrmId` = 10 | Axis detected alarm | Verify the servo drive detected error and remove the cause of the detected error. |
| `q_uiAlrmId` = 30 | PLCopen detected alarm | Reset the function block. Give the power cycle to the controller if a detected error remains. |
| `q_uiAlrmId` = 101 | Invalid acceleration parameter | Verify that the acceleration parameter is within the specified range. |
| `q_uiAlrmId` = 102 | Invalid deceleration parameter | Verify that the deceleration parameter is within the specified range. |
| `q_uiAlrmId` = 103 | Invalid velocity parameter | Verify that the speed parameter is within the specified range. |
| `q_uiAlrmId` = 105 | Invalid target position parameter | Verify that the position parameter is within the specified range. |
| `q_uiAlrmId` = 350 | Invalid operation mode parameter | Verify that the operation parameter is within the specified range. |
| `q_uiAlrmId` = 351 | Invalid absolute position Parameter | Verify that the absolute position is within the specified range. |
| `q_uiAlrmId` = 352 | Invalid homing mode parameter | Verify that the homing mode is within the specified range. |

| Issue | Cause | Solution |
|---|---|---|
| `q_uiAlrmId` = 353 | Invalid ref. searching velocity parameter | Verify that the ref. searching velocity is within the specified range. |
| `q_uiAlrmId` = 354 | Invalid ref. out velocity parameter | Verify that the ref. out velocity is within the specified range. |
| `q_uiAlrmId` = 355 | Invalid max. distance parameter | Verify that the max. distance is within the specified range. |
| `q_uiAlrmId` = 356 | Invalid distance from reference parameter | Verify that the distance from reference is within the specified range. |
| `q_uiAlrmId` = 357 | Jog positive and Jog negative are active | Verify that only one input is ON at a time. |
| `q_uiAlrmId` = 358 | Positive limit switch active | Verify the positive limit switch position. |
| `q_uiAlrmId` = 359 | Negative limit switch active | Verify the negative limit switch position. |

# Chapter 3
# XYPickAndPlace: Moving X- and Y-Axis in a Plane Coordinate System

## What Is in This Chapter?

This chapter contains the following sections:

# Section 3.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Functional Description

XYPickAndPlace function is required in the consumer products industry for a wide variety of product transfer applications. The machine basically takes a product or products from machines such as wrappers, cartoners or fillers, and stacks them into a case, which is then ready for palletizing or shipment.

## Usage of `XYPickAndPlace` Function Block

XYPickAndPlace Application involves two functions, one is for the picking and placing of material and the other is for movement of this material in the XY plane.

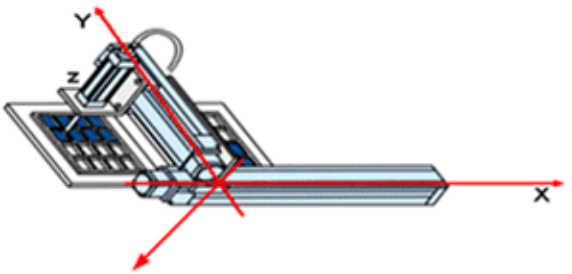
The `XYPickAndPlace` function block involves two linear axes that move the manufactured part from one assembly station to another station or from a conveyor to pallet.

## Solution with the `XYPickAndPlace` Function Block

Moving the part to the target location can be accomplished in three separate moves:

- Picking the part along the Z axis.
- Moving the part in the XY plane to the target position.
- Placing the part along the Z axis.

This figure shows the PickAndPlace assembly:



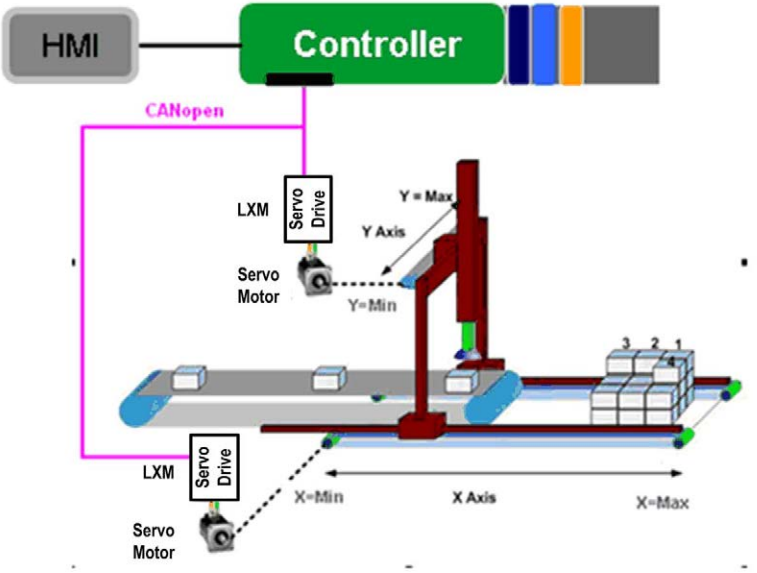

The function block is used to reduce process time, increase productivity and efficiency in packaging.

**NOTE:** Z Axis Operation is not controlled by the function block.

## Design & Realization Constraints and Assumptions

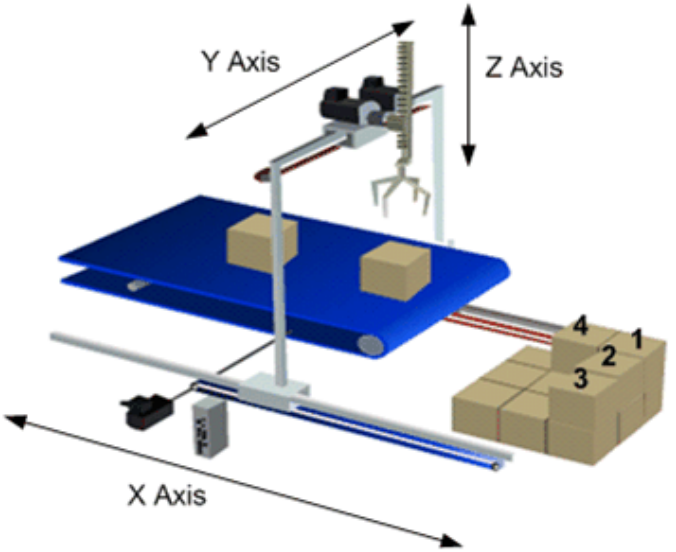
- The Shift (Skip) Position is applicable only in the Semi-Auto Mode. This option is not allowed in Auto Mode.
- In Semi Auto Mode the trigger for each cycle is done by the rising edge of the `i_xExe`, but the parameter change is done only with the `i_xEn`; this is to avoid parameter change during the cycle.
- New cycle is started once all the boxes (box shifting is counted) in the current cycle are placed as per the defined matrix. The message "Layer Completed with Skipped Box" is then displayed.
- Presently done: Skipped box can be either manually placed (by operator) or a new semi-automatic cycle can be started by resetting the Box Count `i_xBoxCntRst` input and giving respective skip number through `i_uiStepNb` to directly reach the box which needs to be placed (Box or any obstacle on the way to reach this previously skipped box position is not considered in this FB since Z axis is not controlled).
- If function block is disabled and enabled and the parameters (Direction of Placing, Number of elements in X and Y, Target, Approach and Start Position) are changed, then the box count is reset and a fresh cycle is started. If no parameter is changed, then old cycle is continued.
- Direction of Placing, Number of elements in X and Y, Target, Approach and Start Position cannot be changed when a cycle is running. New values are affected only on the Rising edge of the `i_xEn` (FB enable).

## Functional View



# Machine Overview

## Machine View



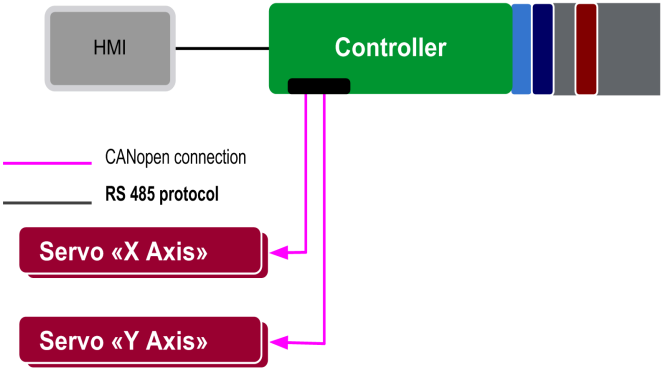
# Section 3.2
## Architecture

### What Is in This Section?

This section contains the following topics:

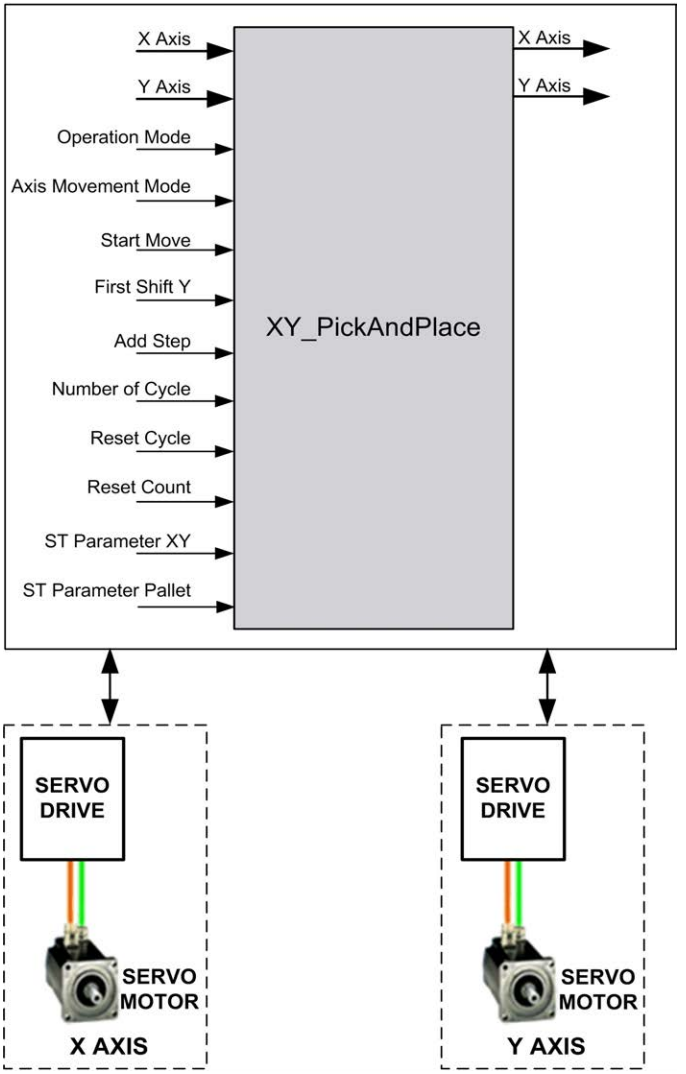
# Hardware Architecture

## Hardware Architecture Overview



# Software Architecture

## DataFlow Overview



# Section 3.3
## Function Block Description

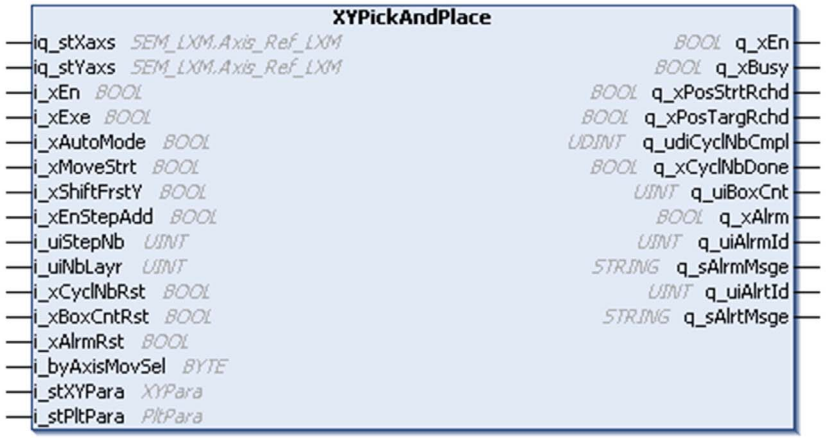
### What Is in This Section?

This section contains the following topics:

# XYPickAndPlace Function Block

## Pin Diagram

The XYPickAndPlace function block encapsulates all necessary functions for controlling the position of two axes in the XY plane.



# Axis Movement: General Description

## At a Glance

The axis are moved in a coordinated linear movement by calculating the ratio of distances travelled in the X axis and Y axis. The axis with the shorter move is multiplied by this ratio so that the axes move with the same ratio of velocity, acceleration and deceleration at any point in time.

## Approach Point Rules

Selecting the approach point permits decreasing the axes velocity when they reach the approach position.

If the selected mode is 0, 1 or 2, the resulting axis trajectory is done without an approach point, the axis moves to the target position in one move.

Following are the 5 modes of axis movement:

**0** Non-optimized mode without approach point
**1** Optimized mode without approach point
**2** Strict optimized mode without approach point
**3** Optimized mode with approach point
**4** Strict optimized mode with approach point

**NOTE:** Improper setup or cabling may cause unintended equipment action.

**⚠ WARNING**

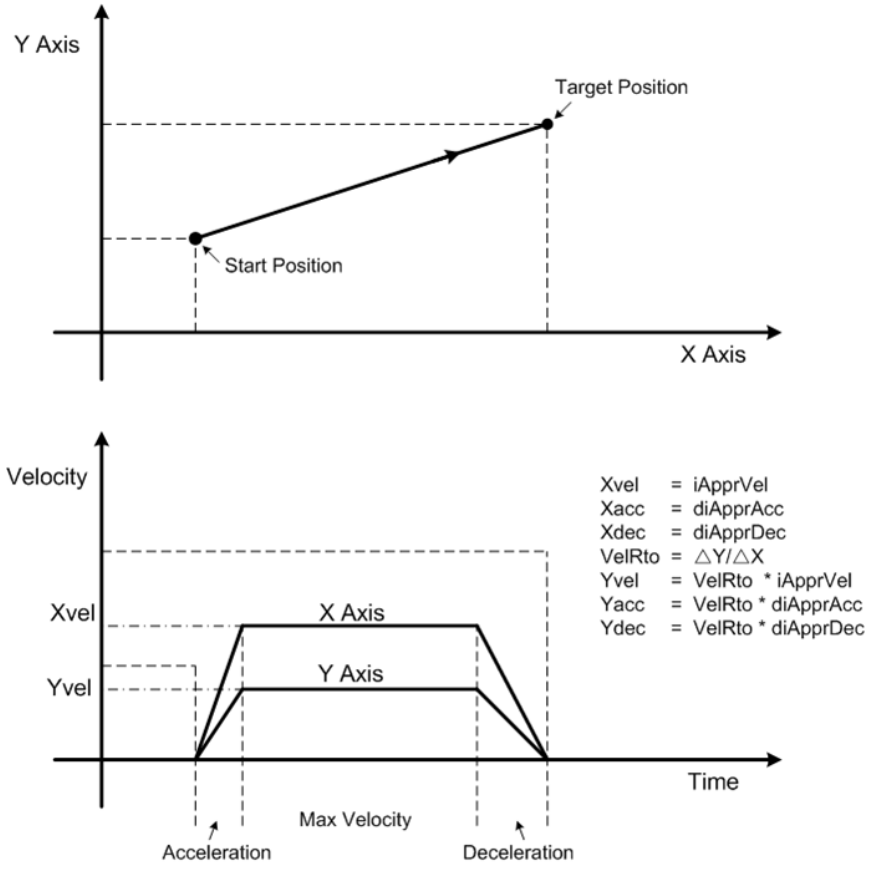
**UNINTENDED EQUIPMENT OPERATION**

Verify that all configuration parameters are correct and validate cabling before commissioning.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Linear Axis Movement

This figure shows linear axis movement by starting and stopping both axes at the same time to achieve a high level of linearity:



## Axis Movement: Non-Optimized Mode Without Approach Point

### Description

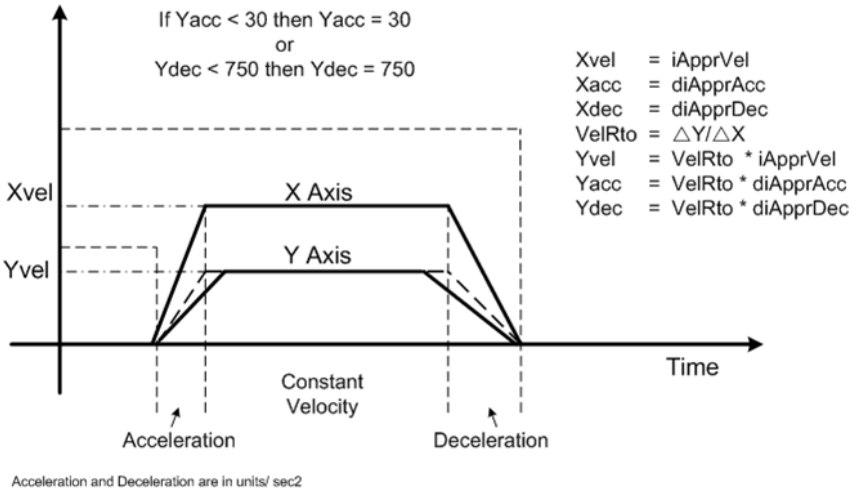
The non-optimized mode without approach point is selected when `i_byAxisMovSel` input is 0.

In this mode, if the calculated Acceleration and Deceleration of the short distance move is less than the lower limit of 30 and 750 respectively, then the Acceleration and Deceleration is set to 30 and 750 and an alarm message of "Non Optimized Mode" is ON and the straight line trajectory is not assured.

**NOTE:**

- All calculations are done before the movement of the axes. The mode selection needs to be done prior to the start of the axis movement.
- In this Mode, the user needs to select the appropriate Acceleration and Deceleration to prevent from going below the limit required to achieve straight line trajectory.

### Axis Movement




Acceleration and Deceleration are in units/ sec2

## Axis Movement: Optimized Mode Without Approach Point

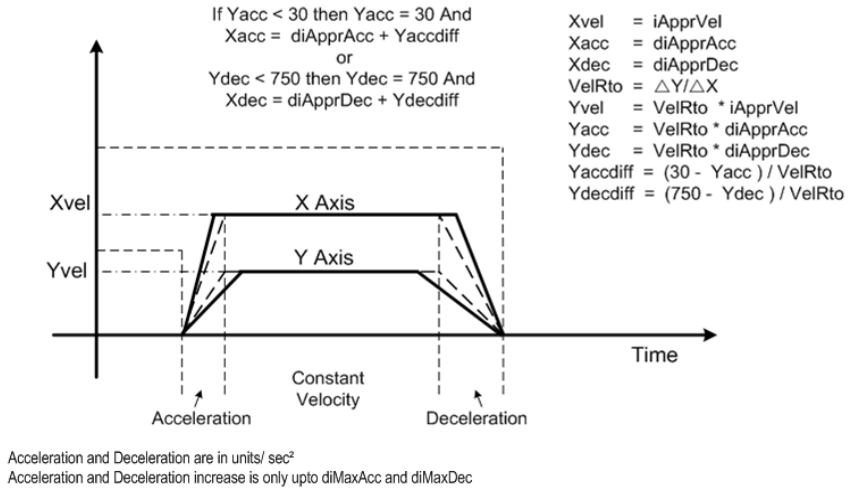
### Description

For axis movement, optimized mode without approach point is selected when `i_byAxisMovSel` input is 1.

In this mode, if the short distance movement axis calculated acceleration and deceleration is less than the lower limit of 30 and 750 respectively then acceleration and deceleration of a long distance movement axis is increased accordingly to achieve the straight line trajectory.

During this Alarm message "Auto Correction ON" is ON.

### Axis Movement




**NOTE:**

- All the calculations are done before the movement of the axes. The mode selection needs to be done prior to the start of the axis movement.
- In this mode, the user must set the Maximum Acceleration (**diAccMax**) and Maximum Deceleration (**diDecMax**) so that the calculated Acceleration and Acceleration (of the longer axis) stays within the acceptable operating limits of the machine.
- In this scenario, the alarm message " Auto Correction On with Acc / Dec Limited" is set and the straight line trajectory can not be assured.
- In this mode, there is always a movement in X- and in Y- direction and accordingly delta X and delta Y cannot be zero.

## Axis Movement

### Strict Optimized Mode without Approach Point

For axis movement, the Strict Optimized Mode without approach point is selected when `i_byAxisMovSel` is 2. In this mode, if the short distance movement axis calculated acceleration and deceleration is less than the lower limit of 30 and 750 respectively,then function block shows "Acceleration Value too Low" and "Deceleration Value too Low" respectively and there is no axis movement.

### Strict Optimized Mode with Approach Point

For axis movement, the Strict Optimized Mode with approach point is selected when `i_byAxisMovSel` input is 4. In this mode, if the short distance movement axis calculated acceleration and deceleration is less than the lower limit of 30 and 750 respectively, then function block shows "Acceleration Value too Low" and "Deceleration Value too Low" respectively and there is no axis movement.

### Optimized Mode with Approach Point

For axis movement, the Optimized Mode with approach point is selected when `i_byAxisMovSel` input is 3. In this mode, if the short distance movement axis calculated acceleration and deceleration is less than the lower limit of 30 and 750 respectively, then acceleration and deceleration of the long distance movement axis is increased accordingly to achieve a straight line trajectory. During this, the Alert message "Auto Correction ON" is ON.

**NOTE:** All calculations are done before the movement of the axis and the mode selection must be done before the axis starts to move.

## Function Block Operation: Semi-Auto Mode

### Description

In Semi-Automatic mode (`i_xAutoMode`=FALSE), after enabling `i_xEn` input, the function block is activated and the structure parameters are validated. After enabling the function block, the first rising edge of the `i_xExe` input moves the axis to `XYPara.diPosStrtX` and `XYPara.diPosStrtY`.

### Approach Distance

If the approach distance is greater or equal to the target position, then the rising edge of `i_xExe` moves the axis to the target positions with `XYPara.udiApprVel`, `XYPara.udiApprAcc` and `XYPara.udiApprDec` from the start position.

If the approach distance is less than the target position then the rising edge of `i_xExe` moves the axis to the target positions with `XYPara.udiTrgtVel`, `XYPara.udiTrgtAcc` and `XYPara.udiTrgtDec` from the start position.

When the axis is at the end position, the rising edge of `i_xExe` moves the axis to `XYPara.diPosStrtX` and `XYPara.diPosStrtY`.

**NOTE:** Movement from the start position to the end position or vice versa happens only on the rising edge of `i_xExe` input.

**NOTE:** Improper setup of acceleration, deceleration, speed parameters, or the approach point is too close to the target position may cause a position overshoot.

**⚠ WARNING**

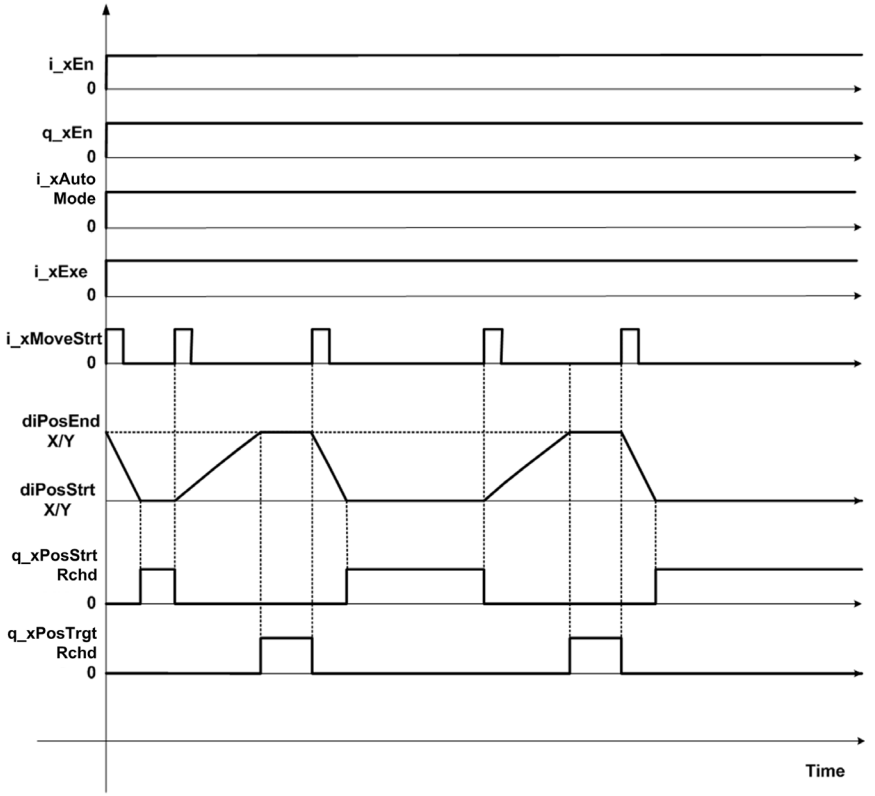
**TARGET POSITION OVERSHOOT**

Verify and validate that the parameters of acceleration, deceleration, speed and approach distance are correct before executing the function.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Timing Chart

This figure shows a timing chart in the Semi-Auto mode:



## Function Block Operation: Auto Mode

### Description

In Automatic mode (`i_xAutoMode` = TRUE), after enabling `i_xEn` input, the function block is activated and the structure parameters are validated. In Auto mode, `i_xExe` input must be TRUE. After enabling the function block, the first rising edge of the `i_xMoveStrt` input moves the axis from any position to `XYPara.diPosStrtX` and `XYPara.diPosStrtY`.

### Approach Distance

If the approach distance is greater or equal to the target position then a rising edge of `i_xMoveStrt` moves the axis to the target positions with `XYPara.udiApprVel`, `XYPara.udiApprAcc` and `XYPara.udiApprDec` from the start position.

If the approach distance is less than the target position then a rising edge of `i_xMoveStrt` moves the axis to the target positions with `XYPara.udiTrgtVel`, `XYPara.udiTrgtAcc` and `XYPara.udiTrgtDec` from the start position.

When the axis is at the end position, the rising edge of the `i_xMoveStrt` moves the axis to `XYPara.diPosStrtX` and `XYPara.diPosStrtY`.

**NOTE:** Movement from the start position to the end position or vice versa happens only on a rising edge of the `i_xMoveStrt` input. During this movement, the `i_xExe` input must be TRUE.

**NOTE:** When `i_xExe` is FALSE while the axis is in motion, the axis stops after completing the current operation.

**NOTE:** Improper setup or cabling may cause unintended equipment action.

**⚠ WARNING**

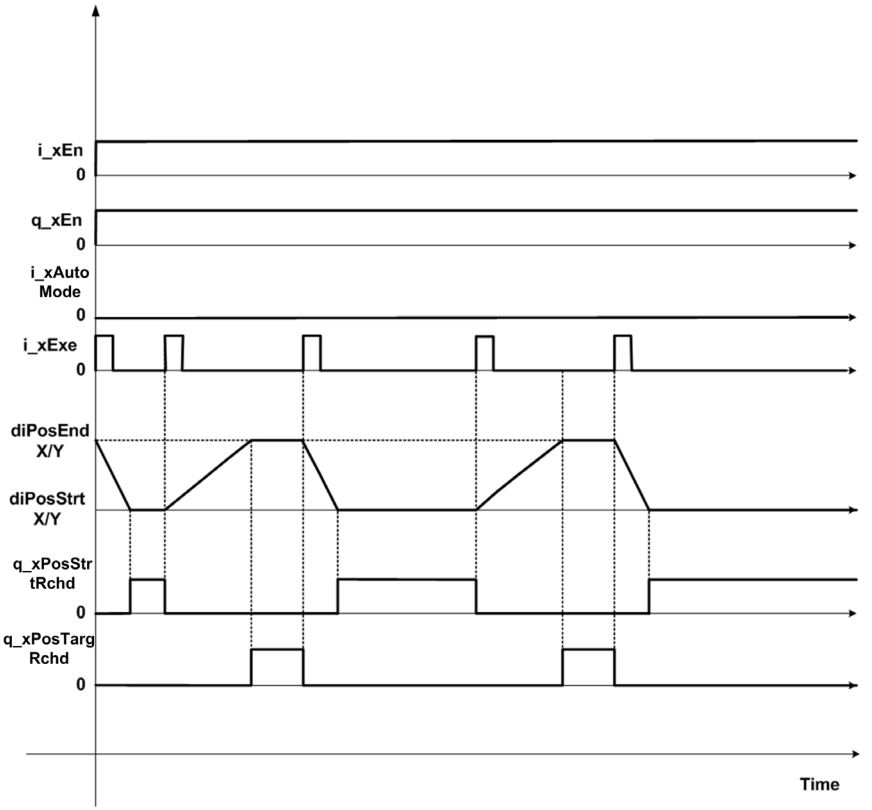
**UNINTENDED EQUIPMENT OPERATION**

Verify that all configuration parameters are correct and validate cabling before commissioning.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Timing Chart

This figure shows a timing chart in Auto mode:



# Section 3.4
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Enables function block and the parameters are validated.<br>FALSE: Disables function block and all outputs are set to zero. The axis movement is stopped. |
| i_xExe | BOOL | TRUE: Axis moves to desired position. Rising edge enables the Operating Mode selected.<br>FALSE: If in motion, axis completes the current action and then stops. |
| i_xAutoMode | BOOL | TRUE: Auto Mode, cycle executes for defined layers.<br>FALSE: Semi-Auto mode. |
| i_xMoveStrt | BOOL | TRUE: Starts the Pick and Place cycle in Auto mode. Executes on rising edge of input. |
| i_xShiftFrstY | BOOL | TRUE: Boxes are placed first in Y direction.<br>FALSE: Boxes are placed first in X direction.<br>Refer to box placement *(see page 89)*. |
| i_xEnStepAdd | BOOL (*) | TRUE: The boxes are shifted by i_iStepNb position<br>**NOTE:** The functionality is only applicable in Semi-Auto mode. |
| i_uiStepNb | UINT (*) | Number of boxes to be shifted with reference to the last placed box.<br>Range:<br>0...(i_stPltPara.uiElemNbX*i_stPltPara.uiElemNbY)<br>**NOTE:** This input is applicable only in Semi-Auto mode and when i_xEnStepAdd input is TRUE. |
| i_uiNbLayr | UINT | Number of layers to be executed<br>Range: 0...65535<br>Refer to layer description *(see page 92)*. |
| i_xCyclNbRst | BOOL | TRUE: Resets the number of layers executed.<br>Reset occurs only when axis is in start position. |
| i_xBoxCntRst | BOOL | TRUE: Resets the number of box counts.<br>Reset occurs only when axis is in start position. |
| i_xAlrmRst | BOOL (*) | TRUE: Resets the detected alarm (rising edge) |
| * Optional according to the application mode | | |

| Input | Data Type | Description |
|---|---|---|
| i_byAxisMovSel | BYTE | Mode selection for Axis movement:<br>0: Non optimized mode without approach point<br>1: Optimized mode without approach point<br>2: Strict optimized mode without approach point<br>3: Optimized mode with approach point.<br>4: Strict optimized mode with approach point.<br>All calculations are done before movement of axis and mode selection must be done before the axis starts to move. |
| i_stXYPara | XYPara | Structure for XY parameters |
| i_stPltPara | PltPara | Structure for pallet parameters |
| * Optional according to the application mode | | |

## Detailed Input Pin Description

### Input: i_xExe

Rising edge enables the selected operating mode (Auto / Semi-Auto).

In Auto mode, i_xExe must be TRUE for mode operation. If falling edge of i_xExe is detected during axis movement, the current operation will be executed and the operation will be stopped.

In Semi-Auto mode, rising edge of i_xExe is required in each cycle to move the axis from start position to target position or vice versa.

**NOTE:** After enabling the function block in Semi-Auto mode, the first i_xExe trigger will move the axis to start position.

### Input: i_xMoveStrt

In Auto mode, if i_xExe is TRUE, the function block waits for i_xMoveStrt rising edge to move the axis to i_stXYPara.diPosStrtX or i_stXYPara.diPosStrtY.

**NOTE:** i_xMoveStrt input is not applicable in Semi-Automatic mode. After enabling the function block in Auto mode, the i_xExe trigger is required to select the Auto mode and the first i_xMoveStrt trigger will move the axis to the start position

# Input: i_stXYPara

## Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| diPosStrtX | DINT | Starting point X<br>Range: 0...diPosStrtX< diPosEndX<br>Refer to axis movement figure *(see page 87)* |
| diPosStrtY | DINT | Starting point Y<br>Range: 0...diPosStrtY< diPosEndY<br>Refer to axis movement figure *(see page 87)* |
| diPosEndX | DINT | Ending point X<br>Range: 0...2147483647<br>Refer to axis movement figure *(see page 87)* |
| diPosEndY | DINT | Ending point Y<br>Range: 0...2147483647<br>Refer to axis movement figure *(see page 87)* |
| udiApprVel | UDINT | Velocity from starting point to approach point<br>Range: 0...3200 units/sec<br>Factory setting: 20<br>Refer to axis movement figure *(see page 93)* |
| udiApprAcc | UDINT | Acceleration from starting point to approach point<br>Range: 30...3000000 units/sec²<br>Factory setting: 20<br>Refer to axis movement figure *(see page 93)* |
| udiApprDec | UDINT | Deceleration from starting point to approach point<br>Range: 750...3000000 units/sec²<br>Factory setting: 30<br>Refer to axis movement figure *(see page 93)* |
| udiApprDist | UDINT | Approach distance<br>Range: 0...2147483647<br>Refer to axis movement figure *(see page 87)* |
| udiTrgtVel | UDINT | Velocity from approach point to target point<br>Range: 0...3200 units/sec<br>Factory setting: 20<br>Refer to axis movement figure *(see page 93)* |
| udiTrgtAcc | UDINT | Acceleration from approach point to target point<br>Range: 30...3000000 units/sec²<br>Factory setting: 30<br>Only if udiTrgtVel is higher than udiApprVel<br>Refer to axis movement figure *(see page 87)* |
| * Optional according to the applicable mode | | |

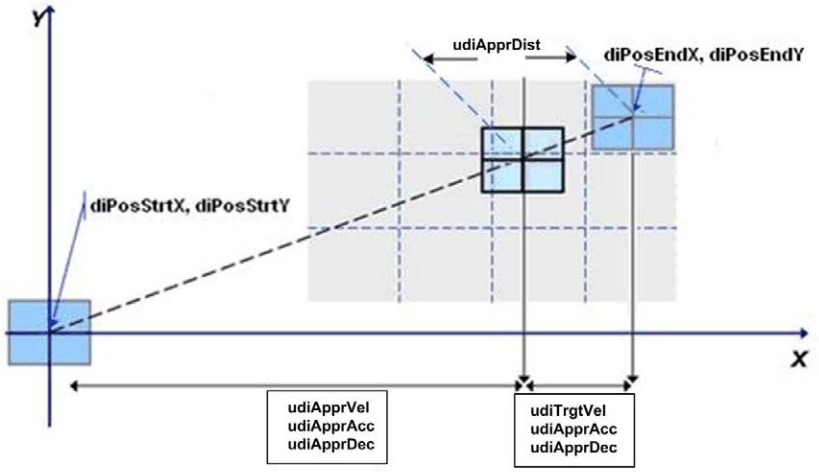
| Input | Data Type | Description |
|---|---|---|
| udiTrgtDec | UDINT | Deceleration from approach point to target point<br>Range: 750...3000000 units/sec²<br>Factory setting: 750<br>Refer to axis movement figure *(see page 87)* |
| xCyclEnd | BOOL | This pin changes the behavior of cycle count (layer count).<br>If xCyclEnd is FALSE then q_udiCycINbCmpl will be incremented with the last box of the layer on the target position.<br>If xCyclEnd is TRUE then q_udiCycINbCmpl will be incremented with the last box of the layer on the start position.<br>Factory setting: FALSE |
| * Optional according to the applicable mode | | |

**NOTE:** All values for distance, velocity and deceleration/acceleration are depending on the amount of pulses per one motor revolution. According to this the minimum/maximum values are given by the following equations:
Acceleration /*min*/= (30 * Motor pulses per revolution)/60
Acceleration /*max*/= (3000000 * Motor pulses per revolution)/60
Deceleration /*min*/= (750 * Motor pulses per revolution)/60
Deceleration /*max*/= (3000000 * Motor pulses per revolution)/60

**NOTE:** xCyclEnd input status is only latched into internal bit of function block when box count is 0 (q_uiBoxCnt = 0) and can be considered as initial instantiation of the function block. That means, internal bit is NOT changed during power cycle or disabling of the function block.

## Axis Movement

This figure shows axis movement with `udiApprDist`:




**NOTE:** Improper setup or cabling may cause unintended equipment action.

| ⚠ WARNING |
| --- |
| **UNINTENDED EQUIPMENT OPERATION** |
| Verify that all configuration parameters are correct and validate cabling before commissioning. |
| **Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

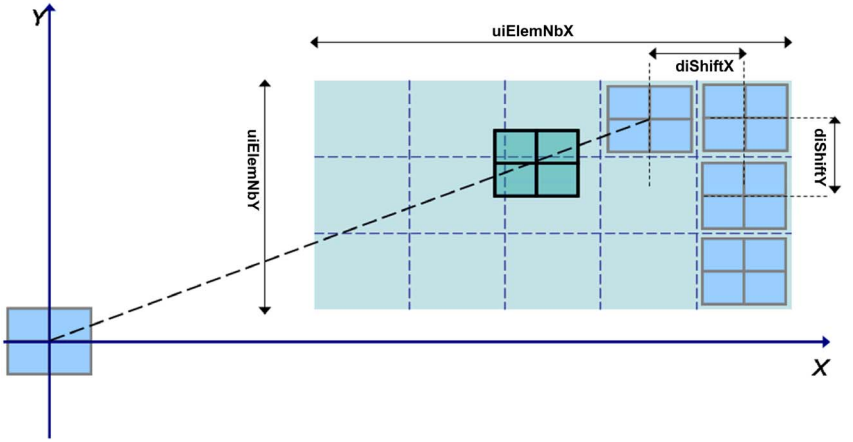
# Input: i_stPItPara

## Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| diShiftX | DINT | Distance of shifting in X direction<br>Range: 0...2147483647<br>Refer to the pallet parameters in figure below. |
| diShiftY | DINT | Distance of shifting in Y direction<br>Range: 0...2147483647<br>Refer to the pallet parameters in figure below. |
| uiElemNbX | UINT | Total number of boxes in X direction<br>Range: 0...10000<br>Refer to the pallet parameters in figure below. |
| uiElemNbY | UINT | Total number of boxes in Y direction<br>Range: 0...10000<br>Refer to the pallet parameters in figure below. |

## Pallet Parameters

This figure shows the different pallet parameters:



## Input: i_xShiftFrstY

### Description

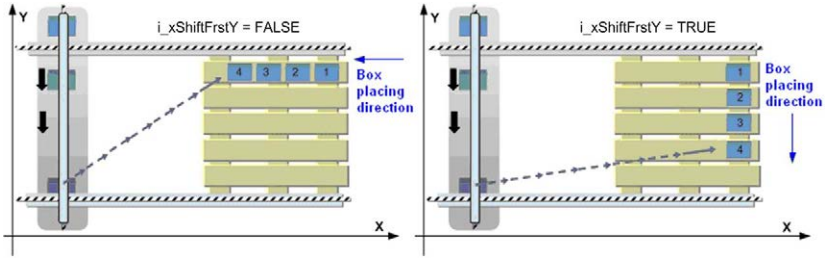
The i_xShiftFrstY input determines whether the boxes are placed first in Y axis by setting this input to TRUE.

As this priorization for the Y axis can only be accepted at the beginning of an empty pallet layer, following pre-conditions need to be fulfilled:

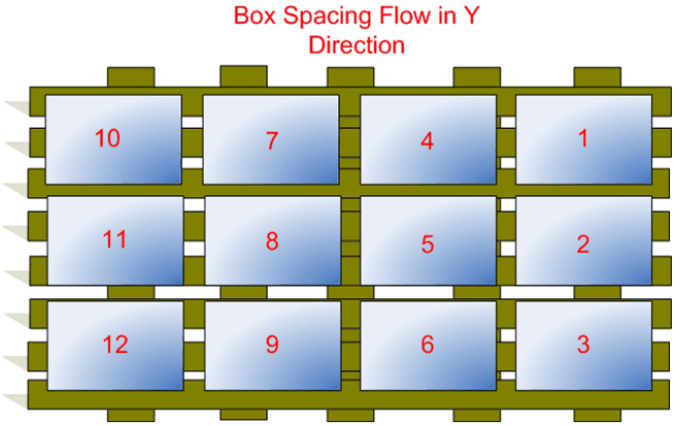
1. X and Y axis are at the start (pick) position.
2. The box count of the actual layer is zero (q_uiBoxCnt = 0)

After both conditions have been met a rising edge on i_xExe is required to enable this setting.

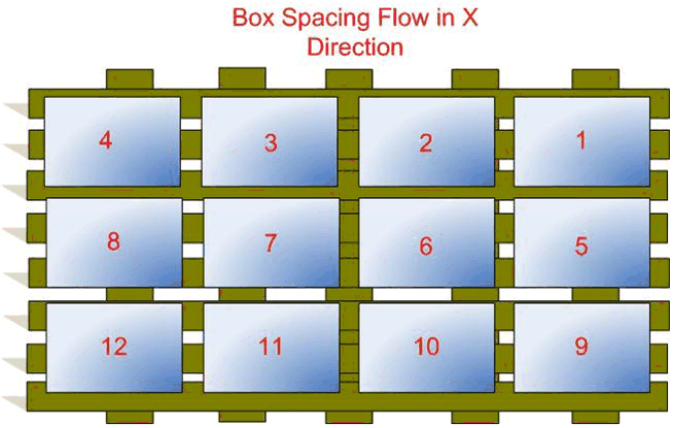
### Box placement: X or Y Axis




### Box Placement: Y Axis



## Box Placement: X Axis

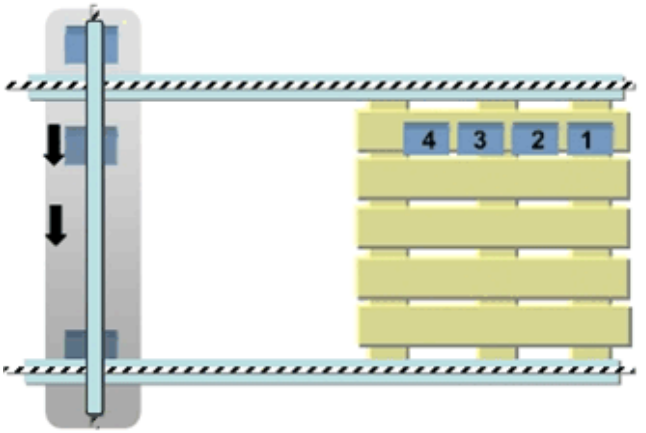


## Input: i_uiStepNb

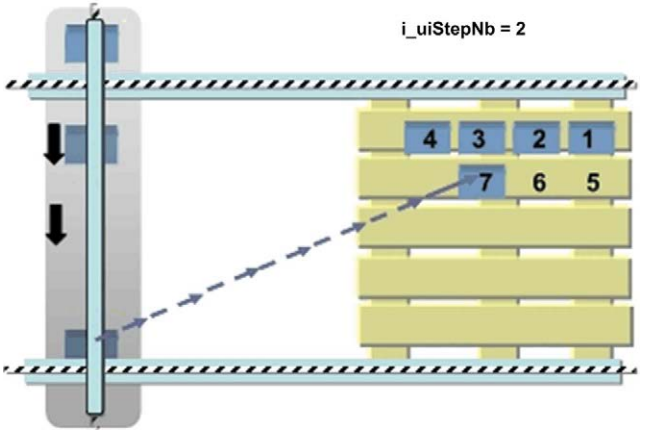
### Description

The i_uiStepNb input refers to the number of positions to be shifted with reference to the last placed box.

### Box Placement Without Shifting




### Box Placement With Shifting

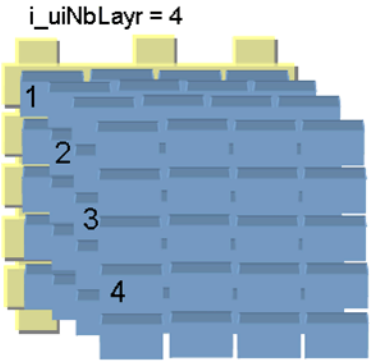


## Input: `i_uiNbLayr`

### Description

The `i_uiNbLayr` input refers to the number of layers to be executed.

This figure shows 4 layers:



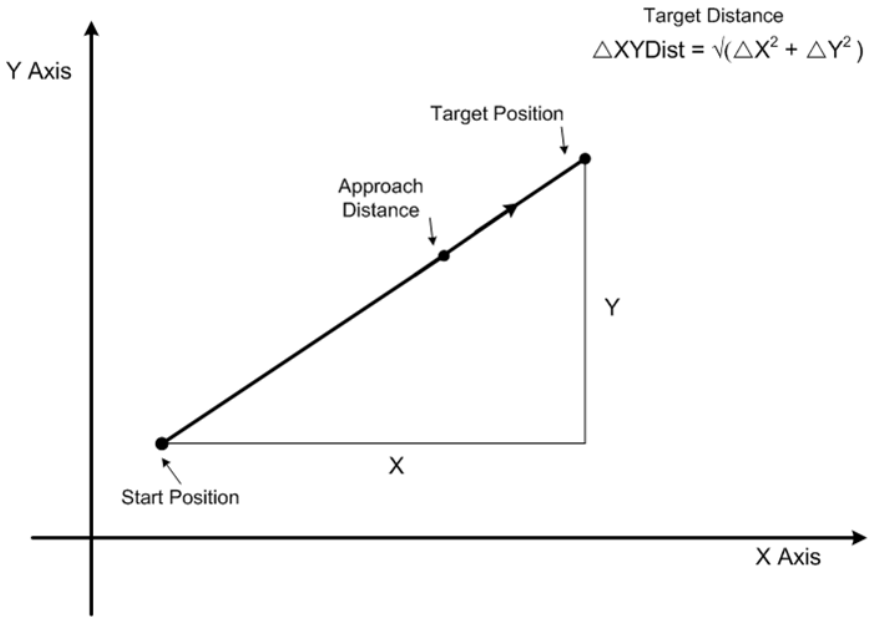

**NOTE:** When layer is completed with skipping, an alarm message "Layer completed with skipping" will be displayed. Empty space of the layer can be filled manually.

## Input: `i_stXYPara.udiApprDist`

### Description

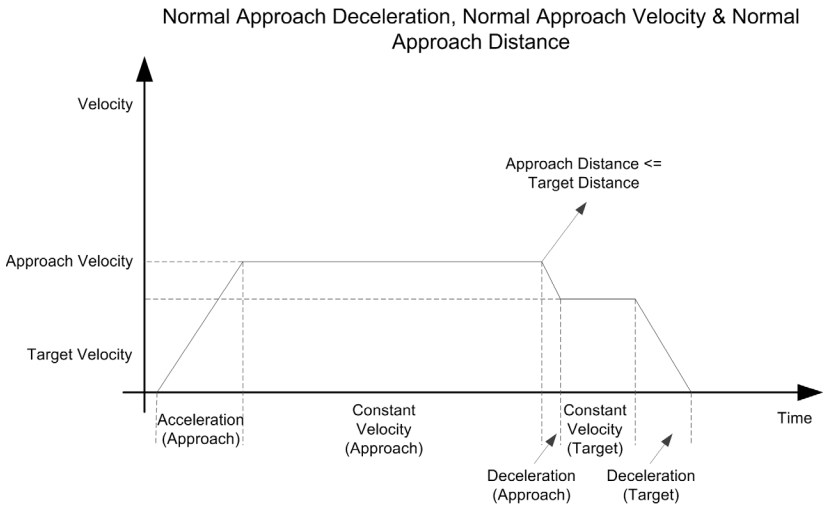
The approach distance is the distance for which the axes will travel with Target Velocity (usually less velocity than the `i_stXYPara.udiApprVel`), Target Acceleration and Target Deceleration.

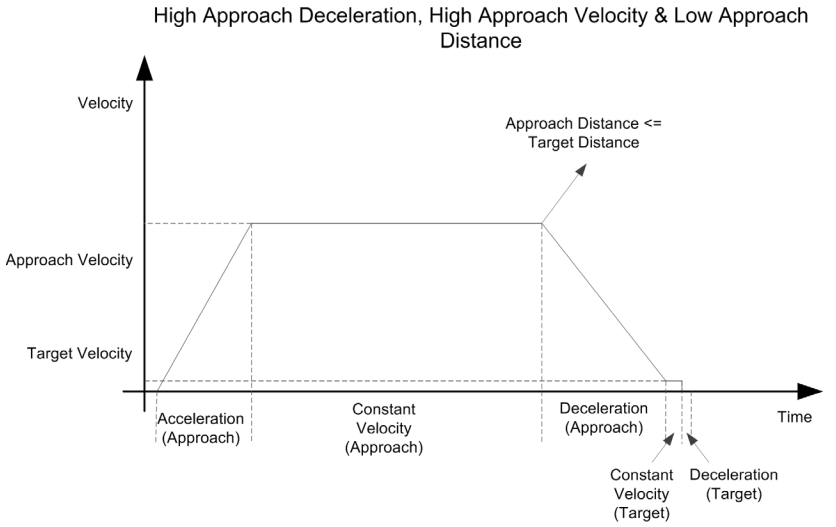
### Target Distance Calculation



## Approach With Normal Values

This figure shows normal and high value approach examples:



# Input/Output Pin Description

## Input/Output Pin Description

### Input/Output Pin Description

| Input/Output | Data Type | Description |
|---|---|---|
| iq_stXaxs | SEM_LXM.Axis_Ref_LXM | Servo axis reference structure |
| iq_stYaxs | SEM_LXM.Axis_Ref_LXM | Servo axis reference structure |

# Output Pin Description

## Output Pin Description

### Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xEn | BOOL | TRUE: function block enabled<br>If input i_xEn is TRUE, then output q_xEn is TRUE. |
| q_xBusy | BOOL | TRUE: Axis in accelerating/decelerating phase |
| q_xPosStrtRchd | BOOL | TRUE: Axis in start point<br>q_xPosStrtRchd output is TRUE when the both X and Y axis are in pick / start position |
| q_xPosTargRchd | BOOL | TRUE: Axis in target point<br>q_xPosTrgtRchd output will be TRUE when the both X and Y axis are in place / end position |
| q_udiCyclNbCmpl | UDINT | The number of cycles completed will be incremented on target position if i_stXYPara.xCyclEnd is FALSE.<br>If i_stXYPara.xCyclEnd is TRUE the number of cycles will be incremented on start position.<br>Range: 0...i_uiNbLayer |
| q_xCyclNbDone | BOOL | TRUE: Number of cycle reached<br>q_xCyclNbDone output is TRUE when all layers are completed in auto mode |
| q_uiBoxCnt | UINT | Number of boxes placed |
| q_xAlrm | BOOL | TRUE: Alarm detected<br>Refer to Notifications *(see page 97)*. |
| q_uiAlrmId | UINT | Alarm ID<br>Range: 0, 1, 11, 12, 30, 101...103, 105...107, 336, 340...349<br>Refer to Notifications *(see page 97)*. |
| q_sAlrmMsge | STRING | Notification<br>Refer to Notifications *(see page 97)*. |
| q_uiAlrtId | UINT | Alert ID<br>Range: 0...9<br>Refer to Alert messages *(see page 97)*. |
| q_sAlrtMsge | STRING | Alert message<br>Refer to Alert messages *(see page 97)*. |

## Notifications

| Detected Alarm ID | Notification |
|---|---|
| 0 | Okay |
| 1 | Internal alarm detected |
| 11 | Axis X alarm detected |
| 12 | Axis Y alarm detected |
| 30 | PLCopen alarm detected |
| 101 | Invalid acceleration parameter |
| 102 | Invalid deceleration parameter |
| 103 | Invalid velocity parameter |
| 107 | Invalid approach distance parameter |
| 336 | Starting point lies further outside the end point |
| 340 | Invalid step number parameter |
| 343 | Invalid number of elements in XY |
| 345 | Invalid axis movement parameter |
| 348 | usr/RPM scaling for both the axes is not same |
| 349 | Starting point lies inside the XY pallet |

## Alert Messages

| Alert ID | Alert Message |
|---|---|
| 0 | No alert |
| 2 | Layer completed with skipping |
| 3 | Skip out of bounds |
| 5 | Disable and enable function block |
| 6 | Verify layer to be stack |
| 7 | No optimized mode |
| 8 | Auto correct on |
| 9 | Auto correction on with acceleration / deceleration limited |

# Section 3.5
## Quick Reference Guide

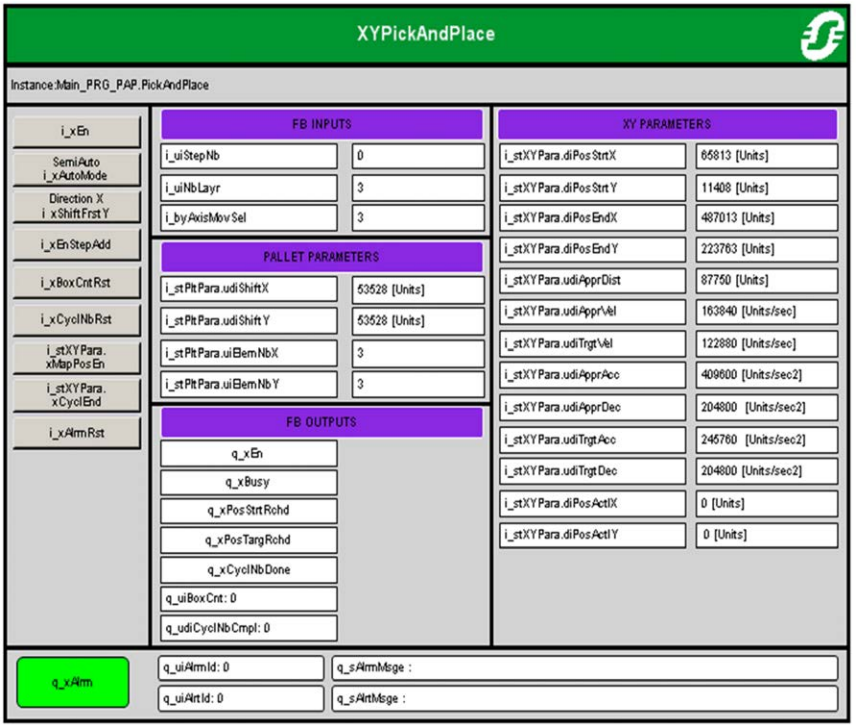
### What Is in This Section?

This section contains the following topics:

# Function Block Visualization

## Visualization

This figure shows a visualization of the `XYPickAndPlace` function block:



# Quick Commissioning Procedure

## Process Description

This table describes the quick commissioning procedure for the `XYPickAndPlace` function block:

| Step | Action |
|---|---|
| 1 | Add following supporting libraries into the application program:<br>• Packaging Library |
| 2 | Configure the axis on CANopen Master. |
| 3 | Enter all parameters in controller configuration. |
| 4 | Use `MC_Power_CAN` for powering up the drive.<br>For homing the axis, one `MoveJog` FB per axis can be used also. |
| 5 | Load the XYPickAndPlace function application on the controller. |
| 6 | Power on axis. |
| 7 | Enter desired parameter and enable the function block.<br>Refer to parametrizing table. |
| 8 | Select application mode. |
| 9 | Start the application. |

## Parameterizing

This table lists the input parameters to be entered:

| Input | Auto Mode | Semi-Auto Mode |
|---|---|---|
| `i_xEn` | TRUE | TRUE |
| `i_xExe` | TRUE | TRUE (Rising Edge) |
| `i_uiNbLayer` | 2 | 2 |
| `i_xAutoMode` | TRUE | FALSE |
| `i_xEnStepAdd` | FALSE | FALSE |
| `i_iStepNb` | 0 | 0 |
| `i_xShiftFrstY` | TRUE | FALSE |
| `i_xMoveStrt` | TRUE (Rising Edge) | FALSE |
| `i_stXYPara.diPosStrtX` | 16380 | 16380 |
| `i_stXYPara.diPosStrtY` | 86100 | 86100 |
| `i_stXYPara.diPosEndX` | 32760 | 32760 |
| `i_stXYPara.diPosEndY` | 270600 | 270600 |
| `i_stXYPara.udiApprDist` | 10000 | 10000 |
| `i_stXYPara.udiApprVel` | 1000 | 1000 |

| Input | Auto Mode | Semi-Auto Mode |
|---|---|---|
| i_stXYPara.udiApprAcc | 3000 | 3000 |
| i_stXYPara.udiApprDec | 4000 | 4000 |
| i_stXYPara.udiTrgtVel | 88 | 88 |
| i_stXYPara.udiTrgtAcc | 2000 | 2000 |
| i_stXYPara.udiTrgtDec | 3000 | 3000 |
| i_stPltPara.uiElemNbX | 4 | 4 |
| i_stPltPara.uiElemNbY | 4 | 4 |
| i_stPltPara.udiShiftX | 8190 | 8190 |
| i_stPltPara.udiShiftY | 24600 | 24600 |

**NOTE:** The above values were tested in laboratory conditions and are for sample only. Appropriate values should be entered as per the actual machine setup.

# Troubleshooting

## Troubleshooting

This table describes some general issue and their solutions:

| Issue | Cause | Solution |
|---|---|---|
| Parameter invalid | Function block error detected | 1. Remove cause of detected error and acknowledge with input `i_xAlrmRst`.<br>2. Reset the function block with `i_xEn`. |
| Inputs invalid | The axis movement is stopped and the detected error output is set. | Remove cause of detected error and acknowledge with `i_xAlrmRst`. |
| Axis error detected | The axis movement is stopped and the detected error output is set. | 1. Remove cause of detected error.<br>2. Execute the axis reset using `MC_Reset` function block.<br>3. Restart the function block. |
| Although the input `i_xShiftFrstY` is set to TRUE the boxes continue getting placed in X axis first. | Preconditions for enabling of box placement in Y axis first are not met. | Verify that X and Y axis are at the start (pick) position and the box count is zero (`q_uiBoxCnt` = 0) and give a rising edge on input `i_xExe`. |
| `q_uiAlrmID` = 1 | Internal alarm detected | Restart the function block. |
| `q_uiAlrmID` = 11 | Alarm detected in X Axis | Verify axis power. |
| `q_uiAlrmID` = 12 | Alarm detected in Y Axis | Verify axis power. |
| `q_uiAlrmID` = 30 | PLCopen alarm detected | Reset the function block. Give the power cycle to the controller if a detected error remains. |
| `q_uiAlrmID` = 101 | Invalid acceleration parameter | Verify that the acceleration parameter is within the specified range. |
| `q_uiAlrmID` = 102 | Invalid deceleration parameter | Verify that the deceleration parameter is within the specified range. |
| `q_uiAlrmID` = 103 | Invalid approach distance parameter | Verify that the approach distance parameter is not greater than the distance between pick position and place position. |
| `q_uiAlrmID` = 105 | Invalid target position parameter | Verify that the end parameter is within the specified range. |
| `q_uiAlrmID` = 106 | Invalid start position parameter | Verify that the start parameter is within the specified range. |

| Issue | Cause | Solution |
|---|---|---|
| q_uiAlrmID = 107 | Invalid approach distance parameter | Verify that the approach distance parameter is not greater than the distance between pick position and place position. |
| q_uiAlrmID = 336 | Invalid starting point | Verify that the starting point is below the target position. |
| q_uiAlrmID = 340 | Invalid step number parameter | Verify that the step number is within the specified range. |
| q_uiAlrmID = 341 | Invalid approach parameter | Verify that the approach parameter is within the specified range. |
| q_uiAlrmID = 342 | Invalid shift boxes parameter | Verify that the shift boxes parameter is within the specified range. |
| q_uiAlrmID = 343 | Invalid number of elements in XY | Verify that the number of elements in XY is within the specified range |
| q_uiAlrmID = 345 | Invalid axis movement parameter | Verify that the axis movement selection parameter is within the specified range. |
| q_uiAlrmID = 346 | Acceleration value too low | Verify the acceleration value and modified as per the axis movement mode selection. |
| q_uiAlrmID = 347 | Deceleration value too low | Verify the deceleration value and modified as per the axis movement mode selection. |
| q_uiAlrmID = 348 | Axis scaling not the same for both axes | Verify that the scaling in terms of drive user units or RPM scaling is the same for both axes. |
| q_uiAlrmID = 349 | Invalid starting point | Verify that the starting point is not placed inside the pallet. |

**NOTE:** The function block is based on the standard PLCopen library, relying mainly on SDO communications of CANopen.

# Part III
# TensionControl

## Purpose of This Part

This part explains the functionality and implementation of the `DigitalTensionControl` and the `AnalogTensionControl` function block in the packaging industry.

## What Is in This Part?

This part contains the following chapters:

# Chapter 4
# DigitalTensionControl: Controlling Consistent Film Tension Between Two Digital Sensors

## Overview

This chapter describes the following function blocks:

- `DigitalTensionControlATV`
- `DigitalTensionControlATV_Motion`

## What Is in This Chapter?

This chapter contains the following sections:

# Section 4.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Why Use the `DigitalTensionControl` Function Block?

When the film reel is unwound, there is a possibility of inconsistent speed or tension variation. This would result in variation in tension, shifting of film and film cut.

## Solution with the `DigitalTensionControl` Function Block

The purpose of the function block is to control the consistent film tension by the Altivar slave speed between two digital sensors. The position of the tension arm is controlled and kept constant to maintain constant film tension.

The position of tension arm is known with the help of two digital sensors:

- SensorArmDown
- SensorArmUp

The function block can be used for both modes, namely slave driven through surface roller and slave driven axially.

## Applications

The function block is applicable for:

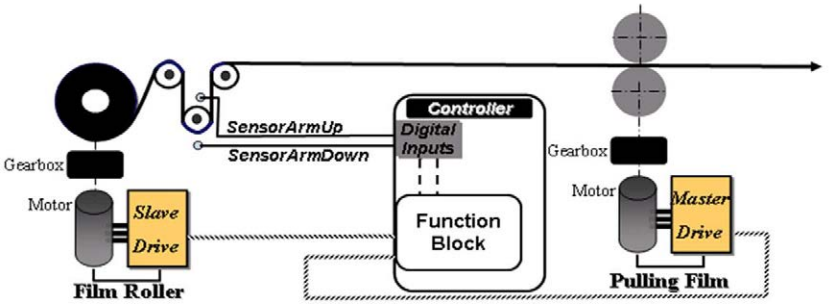
- Horizontal bagging machine
- Vertical bagging machine
- Labeling machine

## Design & Realization Constraints and Assumptions

- Normally Diameter calculator requires filter for line speed and roll speed needs to be externally added.
- Speed gradient is in range of 1 to 5000 RPM.
- Diameter calculation is done by the relation Dact (%) = Dmin (%) *(Line Speed (%) / Motor Speed (%))
- Hardware (Sensors) Faults are not considered in the Program
- If the Task cycle time less than 5 ms, the velocity of the Slave axis can not be updated after every scan. In this case, the velocity command to the Slave axis is updated after 3 scans.

**NOTE:** The tension control function is suitable for a continuous film feeding machine.

## Functional View



# Machine Overview

## Purpose

During the film unwinding operation, the arm tends to go towards the SensorArmDown sensor located at down position (see the machine view figure). The goal of this application function block template is to maintain the tension of the film between two limits, and this is achieved by controlling the position of the tension arm.

## Sychronization Process

This application function block provides the synchronization between a slave axis (typically the film roller "machine view figure: motor M1") and a master axis (the pulling film or feeder "machine view figure: motor M2) with the help of digital sensors. Based on digital sensor inputs, the angular velocity will be varied to maintaining the position of the tension arm between the two sensors.

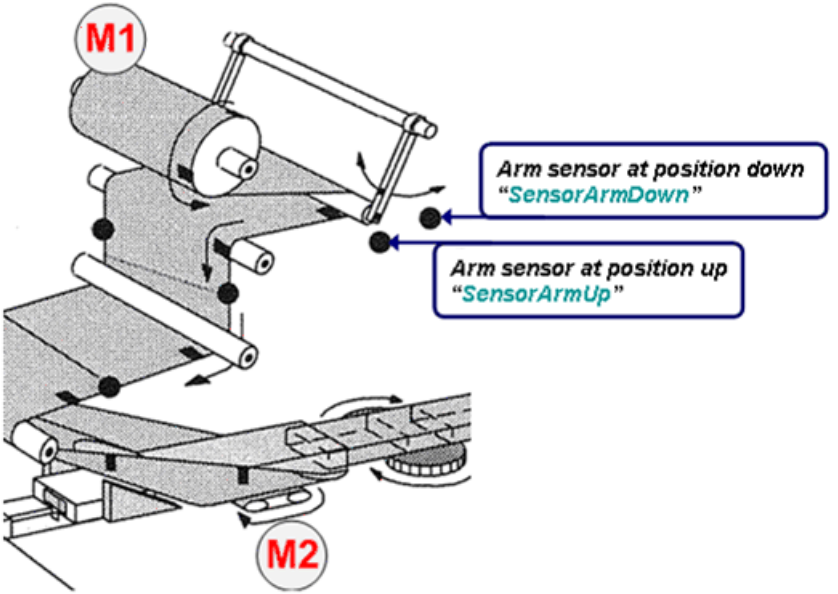
This application function block commands the speed of an axis as slave coupled to a master axis. The master axis can be a physical or virtual axis.

## Roll Diameter Control

Since the roll diameter decreases as material unwinds, the linear speed of the film will decrease. Hence, the slave axis needs to speed up to maintain the constant linear speed.

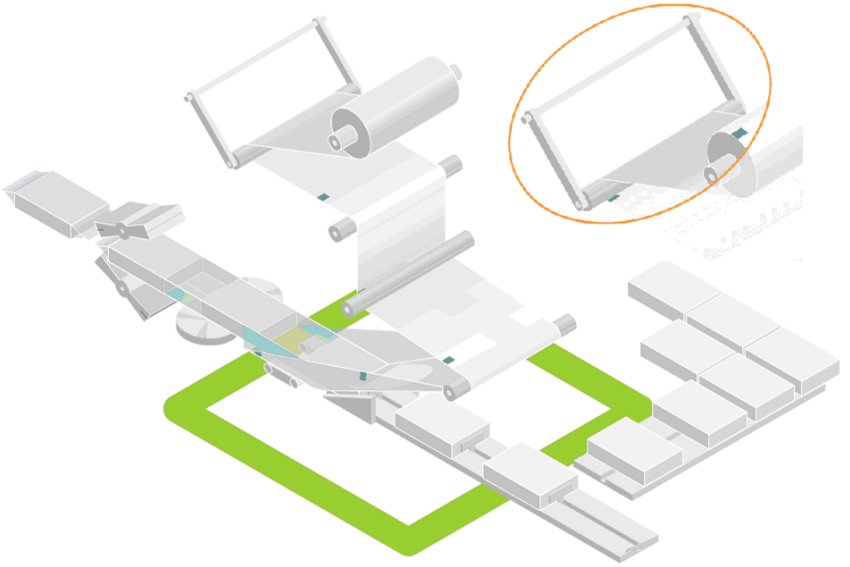
When the SensorArmDown or SensorArmUp is activated, the function block adjusts the angular velocity of the roller to maintain the position of tension arm between the sensors which in turn maintain the film tension.

## Machine View



## Horizontal Bagging Machine

This figure shows a horizontal bagging machine in which the function block can be used for tension control:



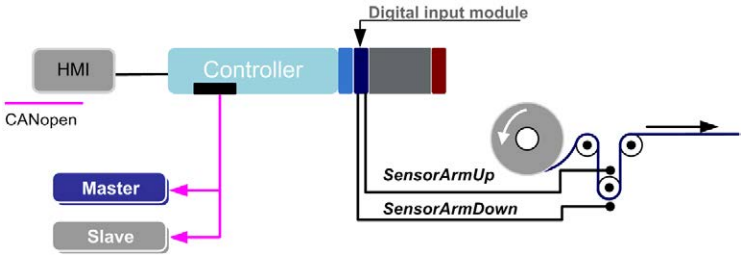
# Section 4.2
## Architecture

### What Is in This Section?

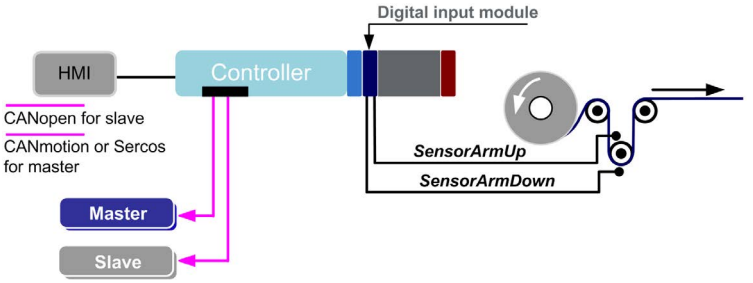
This section contains the following topics:

# Hardware Architecture

## Hardware Architecture Overview




This figure shows a hardware architecture:




## Environment

Based on the axes configuration the `DigitalTensionControlATV` has the following types of architecture:

| Master | Slave | Sensor Wiring |
|---|---|---|
| Lexium (LXM) on CANopen | Altivar (ATV) on CANopen | to the Logic Controller through a digital input module (embedded, expansion or external) |

Based on the axes configuration the `DigitalTensionControlATV_Motion` has the following types of architecture:

| Master | Slave | Sensor Wiring |
|---|---|---|
| Lexium (LXM) on<br>• CANmotion or<br>• Sercos | Altivar (ATV) on CANopen | to the Motion Controller through a digital input module (embedded, expansion or external) |

### Axial Driven

The slave axis is driven directly from the axis.

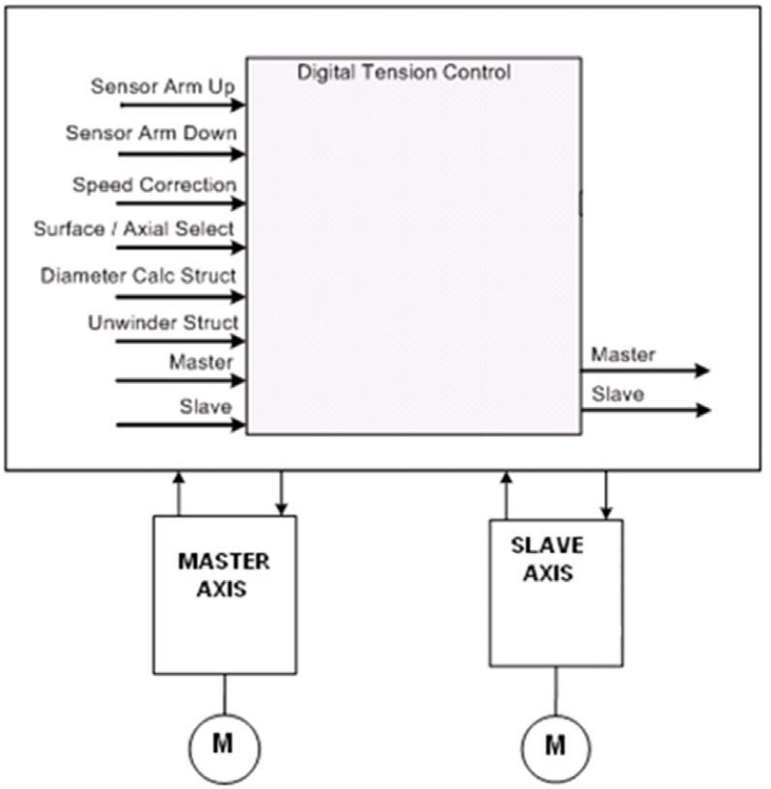
### Surface Driven

The slave axis is driven directly from the surface.

**NOTE:** The function block is applicable only for Altivar drives as slave.

# Software Architecture

## DataFlow Overview

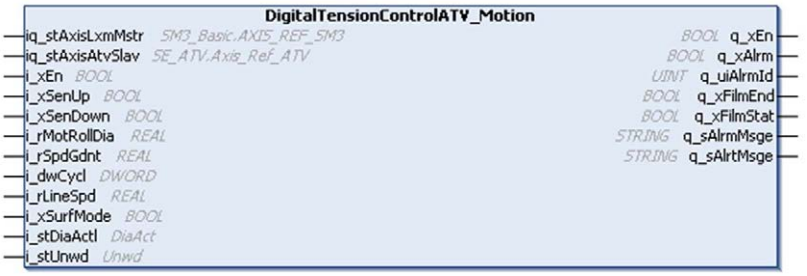


# Section 4.3
## Function Block Description

## `DigitalTensionControl` Function Blocks

### Pin Diagram of `DigitalTensionControlATV`


### Pin Diagram of `DigitalTensionControlATV_Motion`




### Function Block Description

In a packaging process, the packaging system must be equipped with an efficient tension control system to ensure that there is a smooth unwinding of the film reel. There must be constant tension on the film reel when it is unwound. The function block controls the speed of the slave axis, relative to the master axis to maintain the constant tension on the web.

## Operating Modes

The function block works in two different modes:

- Slave driven through surface roller
- Slave driven axially

# Section 4.4
# Pin Description

## What Is in This Section?

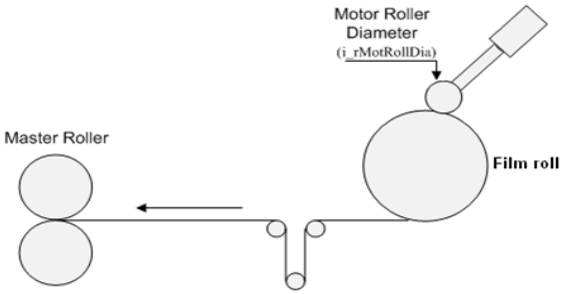
This section contains the following topics:

# Input Pin Description

## Input Pin Description

| Input | DataType | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Enables function block and parameters are validated<br>FALSE: Disables function block and all outputs are set to zero |
| i_xSenUp | BOOL | TRUE: Arm up sensor sensed.<br>If the input is active, the arm is at up position.<br>It means that the slave speed is less than the master speed, the slave speed will be increased by the correction factor.<br>Refer to i_dwCycl *(see page 122)* |
| i_xSenDown | BOOL | TRUE: Arm down sensor sensed.<br>If the input is active, the arm is at down position. It means that slave speed is more than master speed, slave speed will be decreased by the correction factor.<br>Refer to i_dwCycl *(see page 122)* |
| i_rMotRollDia | REAL | Motor roll diameter (surface roller)<br>Range: 1.0...2000.0 mm<br>This parameter will be used only when the surface driven (i_xSurfMode) mode is selected.<br>Refer to film roll setup *(see page 122)* figure |
| i_rSpdGdnt | REAL | Speed gradient<br>Range: 1.0...5000.0 RPM<br>Speed Gradient input parameter with which the slave speed will be corrected to maintain the constant tension on the film. |
| i_dwCycl | DWORD | Task execution time for speed correction<br>Range: 1...1000 ms |
| i_rLineSpd | REAL | Actual line speed<br>Range: 0.0...DiaAct.i_rLineSpdMax in m/min |
| i_xSurfMode | BOOL | TRUE: Surface driven<br>FALSE: Axial driven<br>Driven mode must be selected only while function block is disabled.<br>Mode change is not allowed. |
| i_stDiaActl | DiaAct | Structure for diameter calculation<br>Refer to i_stDiaActl *(see page 123)* description. |
| i_stUnwd | Unwd | Structure for unwinder details<br>Refer to i_stUnwd *(see page 125)* description. |

## Film Roll Setup




### Input: `i_rSpdGdnt`

Slave Axis Speed increase or decrease are calculated based on this value. The Speed correction is done only if the sensor is active for 60 seconds.

Example:

1000 RPM needs to be decreased or increased according the sensor arm position. For details, refer below for `i_dwCycl` description.

### Input: `i_dwCycl`

For effective speed correction, this input should be same as the Target cycle time. This parameter decides the speed correction per scan.

Example:

`i_rSpdGdnt` = 1000 RPM and `i_dwCycl` = 10 ms, then Speed Correction = (1000/60000)*10 = 0.166 revolutions will be added or subtracted from the slave speed.

## Input: i_stDiaActl

### Description

This structure is used for the diameter calculation.

When the machine is started with a new film roll, its actual diameter needs to be entered in i_stDiaActl.rDia and i_stDiaActl.xDiaSet needs to be set TRUE.

Then initially, the slave will start with the speed as per the entered diameter.

Speed ratio (i_stDiaActl.rSpdRto) is the gearing factor for the slave motor. If gear is not used, set the i_stDiaActl.rSpdRto to 1.0.

**NOTE:** Whenever the slave speed (actual speed of the drive) is below i_stDiaActl.rSpdMin and actual Line speed is below i_stDiaActl.rLineSpdMin, then the actual diameter calculation will stop.

### Mathematical Background

$$\text{Diameter Actual (\%)} = \frac{\text{(Line Speed (\%) * Minimum Diameter (\%))}}{\text{Un-winder Speed (\%)}}$$

$$\text{Un-winder Speed (rpm)} = \frac{\text{(Speed ratio * Line Speed (m/min) * 1000)}}{\text{(3.14 * Diameter Actual (mm))}}$$

### Example

If the line speed is 50 % (250 m/min) of the maximum speed (500 m/min), the minimum diameter is 10 % (100 mm of the maximum diameter (1000 mm) and the un-winder speed is 40 % (636.94 RPM) of the maximum un-winder speed (1592.35 RPM))

Un-winder speed (Max) = (1* 500*1000) / (3.14* 100) = 1592.35 RPM.

Where Speed Ratio = 1, Line Speed (Max) = 500 m/min, Minimum Diameter = (100 mm).

Then the actual diameter value can be calculated as:

Diameter Actual (%) = (50*10)/40= 12.5% of the maximum diameter.

Diameter Actual (mm) = (250)/(3.14*636.94) = 0.125 m = 125 mm

**NOTE:** i_rLineSpdMax structure parameter is common for surface and axial mode, other structure parameters of i_stDiaActl are used only for the axial driven mode. The reset of the diameter calculation in axial driven mode is mandatory in case the film reel is changed. This will be executed by resetting the internal state machine of the function block by disabling and enabling the function block.

## Structured Parameters

| Structured Parameter | Data Type | Description |
|---|---|---|
| i_rDiaMin | REAL | Minimum diameter value<br>Range: 1.0...i_rDiaMax in mm |
| i_rDiaMax | REAL | Maximum diameter value<br>Range: 1.0...10000.0 mm |
| i_xDiaSet | BOOL | TRUE: Set diameter |
| i_rDia | REAL | Set diameter value<br>Range: i_rDiaMin...i_rDiaMax in mm |
| i_rSpdMin | REAL | Minimum slave speed to calculate the diameter<br>Range: 1.0...100.0 RPM |
| i_rLineSpdMin | REAL | Minimum line speed to calculate diameter<br>Range: 1.0...100.0 m/min |
| i_rLineSpdMax | REAL | Maximum line speed<br>Range: 1.0...1000.0 m/min |

**NOTE:** The axis status of the drives can be read via using PLCopen FB (MC_ReadStatus_ATV & MC_ReadStatus_LXM or MC_ReadStatus). Before starting the machine in axial driven mode, actual diameter value of unwinder roll (i_stDiaActl.rDia) must be passed to the function block and this value will be accepted only on the rising edge of i_stDiaActl.xDiaSet.
If the **real diameter** of the roll is **much greater** than the parameter setting **more** material might be unwound after first cycles and film tension may be too low.
If the **real diameter** of the roll is **much smaller** than the parameter setting **less** material might be unwound after first cycles and film tension may be too high.
The film might tear or the tension arm might be pulled by the tightened film into its mechanical end position.

| *NOTICE* |
|---|
| **INCORRECT DIAMETER SETTINGS**<br>Be sure that the actual diameter value of the un-winder roll is established (i_stDiaAct1.rDia) before starting the machine in axial driven mode.<br>**Failure to follow these instructions can result in equipment damage.** |

## Input: i_stUnwd

### Description

This structure is used for setting the following unwinding parameters:

- acceleration time (i_stUnwd.diAcc)
- deceleration time (i_stUnwd.diDec)
- unwinder minimum diameter (i_stUnwd.rUnwdDiaMin) to initiate notification
- time for detecting the film end (i_stUnwd.dwFilmTimeEnd) when the sensor arm is down

The speed ratio (i_rSpdRto) is the ratio to calculate the slave speed in surface driven mode.

### Mathematical Background

$$\text{Un- winder Speed (RPM)} = \frac{\text{Speed Ratio} * \text{Line Speed (m/min)} * 1000}{3.14 * \text{Motor Roll Diameter (mm)}}$$

### Structured Parameters

| Structured Parameter | Data Type | Description |
|---|---|---|
| rSpdRto | REAL | Ratio to calculate to slave speed<br>Range: 0.01...100.0 |
| diAcc | DINT | Acceleration time for the slave<br>Range: 1...6000<br>Unit: x X Sec<br>Remarks: X scaling factor |
| diDec | DINT | Deceleration time for the slave<br>Range: 1...6000<br>Unit: x X Sec<br>Remarks: X scaling factor |
| rUnwdDiaMin | REAL | Minimum diameter to initiate notification<br>Range: 1.0...300.0 mm |
| dwFilmTimeEnd | DWORD | Wait time for film end notification<br>Range: 1...10000 ms |

**NOTE:** i_stUnwd.diAcc and i_stUnwd.diDec scaling factor (X) is in the drive parameter setting. Factory setting is 0.1. If i_stUnwd.diAcc is 100 s, then drive interpreted time is 10 s.

# Input/Output Pin Description

## Input/Output Pins of `DigitalTensionControlATV`

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stAxisLxmMstr` | `SEM_LXM.AXIS_REF_LXM` | Servo axis reference structure |
| `iq_stAxisAtvSlav` | `SE_ATV.Axis_Ref_ATV` | ATV axis reference structure |

## Input/Output Pins of `DigitalTensionControlATV_Motion`

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stAxisLxmMstr` | `SM3_Basic.AXIS_REF_SM3` | Servo axis reference structure |
| `iq_stAxisAtvSlav` | `SE_ATV.Axis_Ref_ATV` | ATV axis reference structure |

**NOTE:** The Master's line speed is always interpreted as absolute value. The direction of rotation for the Slave will not be changed by inverting the line speed of the Master.

# Output Pin Description

## Output Pin Description

### Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xEn | BOOL | TRUE: Function block enabled. If input i_xEn is TRUE, then output q_xEn is TRUE. |
| q_xAlrm | BOOL | TRUE: Alarm detected<br>FALSE: No alarm<br>Remark: If true, after resetting detected alarm, rising edge of i_xEn is required to restart the function block. |
| q_uiAlrmId | UINT | 0: Okay<br>>0: Alarm detected<br>Range: 0, 1, 11, 12, 20, 30, 40, 101, 102, 116...127<br>Refer to detailed Notifications *(see page 128)* description. |
| q_xFilmEnd | BOOL | TRUE: Film end<br>Remarks: Film end will trigger a notification.<br>Refer to detailed q_xFlmEnd *(see page 127)* description. |
| q_xFilmStat | BOOL | TRUE: Min diameter reminder<br>FALSE: No reminder |
| q_sAlrmMsge | STRING | Status of function block display<br>Refer to detailed Notifications *(see page 128)* description. |
| q_sAlrtMsge | STRING | Alert message display |

### Output: q_xFilmEnd

If the line speed is above dead band and if down sensor remains sensed for a predefined time (i_stUnwd.dwFlmTimeEnd), then the output (q_xFlmEnd) is set TRUE.

**NOTE:** DeadBand has a constant value of 1.3 m/min in the function block. When the i_xSenDown is FALSE, the q_sErrMsge will be reset.

## Notifications

| Detected Alarm ID | Notification |
|---|---|
| 0 | Okay |
| 1 | Internal alarm detected |
| 11 | Master axis alarm detected |
| 12 | Slave axis alarm detected |
| 20 | Invalid cycle time |
| 30 | PLCopen internal alarm detected |
| 40 | Incorrect sensor state |
| 101 | Invalid acceleration parameter |
| 102 | Invalid deceleration parameter |
| 116 | Invalid motor roll diameter limit parameter |
| 117 | Invalid speed gradient limit parameter |
| 118 | Invalid speed ratio value parameter |
| 119 | Invalid minimum unwinder diameter parameter |
| 120 | Invalid film end time parameter |
| 121 | Invalid minimum diameter value parameter |
| 122 | Invalid maximum diameter value parameter |
| 123 | Invalid set diameter value parameter |
| 124 | Invalid minimum unwinder speed value parameter |
| 125 | Invalid line speed value parameter |
| 126 | Invalid maximum line speed value parameter |
| 127 | Invalid minimum line speed value parameter |

## Alarm Messages

The following alarm message will be displayed as and when required:

"SLAVE POSITIVE FREQ LIMIT REACHED" - When the slave frequency reaches the maximum frequency set in the drive.

"SLAVE MINIMUM FREQ LIMIT REACHED" - When the slave frequency reaches the minimum frequency set in the drive.

"DIAMETER SETTING REQUIRED OR MINIMUM DIAMETER REACHED" - When internal diameter is less than minimum diameter or diameter setting is not done.

# Section 4.5
## Quick Reference Guide

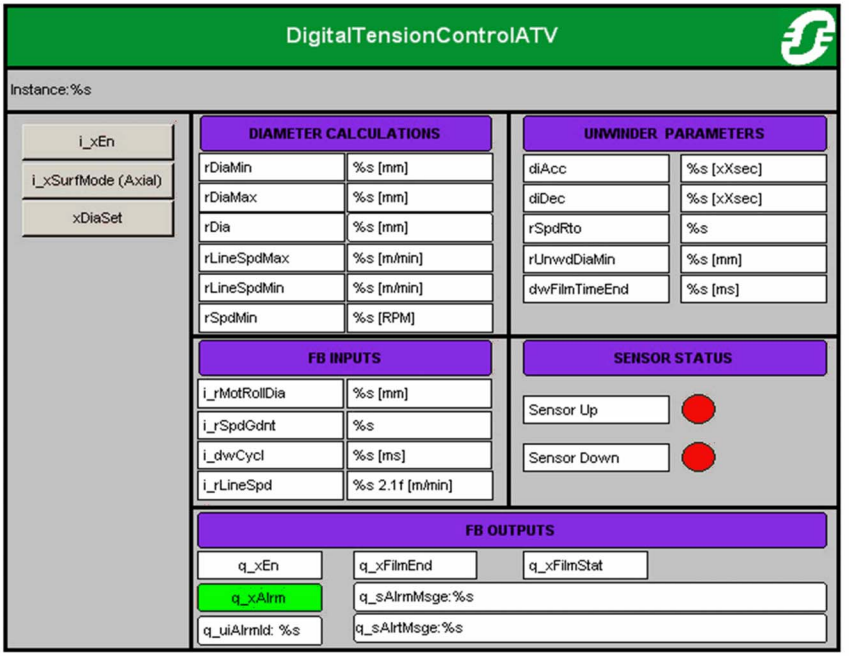
### What Is in This Section?

This section contains the following topics:

# Function Block Visualization

## Visualization

This figure shows the HMI visualization for `DigitalTensionControl` function block:




**NOTE:** The calculation of the remaining, actual diameter of the unwinding film reel works sensorless and based therefore on a mathematical model.
Due to this the following factors describe the limiting effects on reliable results:

- The calculation requires a constant line speed of the film. If the digital tension control function block is operated in intermittent mode no accurate values can be evaluated.
- In addition to a constant line speed a continuous movement of the unwinding Variable Speed Drive (VSD) is necessary. As long as the VSD goes into standstill or a setpoint speed close to 0 Hz (high motor slip on asynchronous machines) during operation is caused to regulate the tension, no appropriate value can be calculated.
- The first calculation of diameter results will be provided after the speed of the VSD has been monitored and averaged over a certain period. How long this period will last depends on the mechanical parameters (film reel diameter plus motor/gearbox configuration) and cannot be given in an absolute time frame in general.

# Quick Commissioning Procedure

## Commissioning Procedure

This table describes the commissioning procedure for the `DigitalTensionControl` function block:

| Step | Action |
|---|---|
| 1 | Add the supporting libraries into the application program:<br>● Load the Packaging Library for using the `DigitalTensionControl` function block in your application. |
| 2 | Select the unwinding type:<br>● Surface driven (Surface Drv)<br>● Axial driven (Axial Drv) |
| 3 | Define the different input parameters for the `DigitalTensionControl` function block from the visualization screen.<br>Refer to:<br>● mandatory inputs *(see page 132)* and parameter example *(see page 133)* for surface-driven mode and<br>● mandatory inputs *(see page 132)* and parameter example *(see page 134)* for axial-driven mode. |
| 4 | Define the speed of the master and power on the master. |
| 5 | Power on the slave drive and enable the `DigitalTensionControl` block. |
| 6 | If axial-driven mode is selected, set the initial diameter by pressing Set Diameter once. |
| 7 | Start the master in velocity mode. |

## Mandatory Inputs in Surface-Driven Mode

The following inputs are mandatory in surface-driven mode:

| Input | Description |
|---|---|
| i_xEn | Enable or disable the function block |
| i_xSenUp | Sensor arm up sensor is to be connected here. |
| i_xSenDown | Sensor arm down sensor is to be connected here. |
| i_rMotRollDia | The motor roll diameter in terms of millimeter |
| i_rSpdGdnt | The parameter with which the slave speed will be corrected to maintain the constant tension on the Film. Example: i_rSpdGdnt is 1000 RPM and i_dwCycl is 10 ms, then Speed Correction = (1000 /60000)*10 => 0.166 revolutions will be added to or subtracted from the slave speed as per SensorArmUp or SensorArmDown signal. |
| i_dwCycl | Target execution time for speed correction |
| i_rLineSpd | Actual line speed of machine in meters per minute |
| i_xSurfMode | If TRUE then Surface Driven mode is selected |
| i_stUnwd.rSpdRto | Ratio to calculate slave speed |
| i_stUnwd.diAcc | Acceleration time for the slave |
| i_stUnwd.diDec | Deceleration time for the slave |
| i_stUnwd.rUnwdDiaMin | Minimum diameter to raise notification |
| i_stUnwd.dwFilmTimeEnd | Wait time for film end notification |
| i_stDiaActl.rLineSpdMax | Maximum line speed limit |

## Mandatory Inputs in Axial-Driven Mode

The following inputs are mandatory in axial-driven mode:

| Input | Description |
|---|---|
| i_xEn | Enable or disable the function block |
| i_xSenUp | Sensor arm up sensor is to be connected here. |
| i_xSenDown | Sensor arm down sensor is to be connected here. |
| i_rSpdGdnt | The parameter with which the slave speed will be corrected to maintain the constant tension on the film |
| i_dwCycl | Target execution time for speed correction |
| i_rLineSpd | Actual line speed of machine in meters per minute |
| i_xSurfMode | If FALSE then Axial Driven mode is selected. |

| Input | Description |
|---|---|
| i_stUnwdr.rSpdRto | Ratio to calculate slave speed |
| i_stUnwd.diAcc | Acceleration time for the slave |
| i_stUnwd.diDec | Deceleration time for the slave |
| i_stUnwd.rUnwdDiaMin | Minimum diameter to raise notification |
| i_stUnwd.dwFilmTimeEnd | Wait time for film end notification |
| i_stDiaActl.rDiaMin | Minimum diameter value |
| i_stDiaActl.rDiaMax | Maximum diameter value |
| i_stDiaActl.xDiaSet | To set the diameter initially |
| i_stDiaActl.rDia | Set diameter value |
| i_stDiaActl.rSpdMin | Minimum slave speed to calculate diameter |
| i_stDiaActl.rLineSpdMin | Minimum line speed to calculate diameter |
| i_stDiaActl.rLineSpdMax | Maximum line speed limit |

## Parameter Example for Surface-Driven Mode

This table shows a parameterization example of the function block in surface-driven mode:

| Input | Description |
|---|---|
| i_xEn | TRUE |
| i_xSenUp | TRUE |
| i_xSenDown | FALSE |
| i_rMotRollDia | 200.00 mm |
| i_rSpdGdnt | 100.00 RPM |
| i_dwCycl | 10 ms |
| i_xSurfMode | TRUE |
| i_stUnwd.rSpdRto | 5 |
| i_stUnwd.diAcc | 100 s |
| i_stUnwd.diDec | 100 s |
| i_stUnwd.rUnwdDiaMin | 100.00 mm |
| i_stUnwd.dwFilmTimeEnd | 10000 ms |
| i_stDiaActl.rLineSpdMax | 300.00 m/min |
| i_stDiaActl.rLineSpdMin | 10.00 m/min |

**NOTE:** The above values were tested in laboratory conditions and are for sample only. Appropriate values should be entered as per the actual machine setup.

## Parameter Example for Axial-Driven Mode

This table shows a parameterization example of the function block in axial-driven mode:

| Input | Description |
|---|---|
| i_xEn | TRUE |
| i_xSenUp | TRUE |
| i_xSenDown | FALSE |
| i_rMotRollDia | 200.00 mm |
| i_rSpdGdnt | 100.00 RPM |
| i_dwCycl | 5 ms |
| i_xSurfMode | FALSE |
| i_stUnwd.rSpdRto | 5 |
| i_stUnwd.diAcc | 100 s |
| i_stUnwd.diDec | 100 s |
| i_stUnwd.rUnwdDiaMin | 100.00 mm |
| i_stUnwd.dwFilmTimeEnd | 10000 ms |
| i_stDiaActl.rLineSpdMax | 300.00 m/min |
| i_stDiaActl.rLineSpdMin | 10.00 m/min |
| i_stDiaActl.rDiaMin | 50.00 mm |
| i_stDiaActl.rDiaMax | 1000.00 mm |
| i_stDiaActl.rDiaSet | TRUE |
| i_stDiaActl.rDia | 900.00 mm |

**NOTE:** The above values were tested in laboratory conditions and are for sample only. Appropriate values should be entered as per the actual machine setup.

# Troubleshooting

## Troubleshooting

This table describes some general issues and their solutions:

| Issue | Cause | Solution |
|---|---|---|
| Error detected at output | Invalid parameter or drive error detected | 1. Verify the detected error ID and read the corresponding notification.<br>2. Restart the function block. |
| System not responding | Invalid value | Restart the function block. |
| Slave not responding | Due to CANopen bus traffic / PLCopen function block error detected | Disable the function block, power off master, slave and restart the system. |
| Slave axis unwinds too much material. | LSP setting is above set point speed | Reduce LSP settings or adjust value to 0.0 Hz. |
| Slave axis doesn't stop, even if set point frequency is 0.0 Hz. | LSP setting is above set point speed | Reduce LSP settings or adjust value to 0.0 Hz. |
| | Altivar VSD parameters are not properly set | Verify the Altivar `CHCF` parameter for the correct setting: command and reference must not be separated. |
| Slave frequency doesn't follow the master. | Changing of axial/surface driven mode on the fly. | On the fly mode change is not allowed. For accepting the new drive mode, restart the function block again. |
| `q_uiAlrmId` = 1 | Internal alarm detected | Restart the function block. |
| `q_uiAlrmId` = 11 | Master axis alarm detected | Verify axis power. |
| `q_uiAlrmId` = 12 | Slave axis alarm detected | Verify axis power. |
| `q_uiAlrmId` = 20 | Invalid cycle time | Verify that the `i_dwCycl` parameter is within the specified range. |
| `q_uiAlrmId` = 30 | PLCopen alarm detected | Reset the function block. Give the power cycle to the controller if a detected error remains. |
| `q_uiAlrmId` = 40 | Incorrect sensor state | Verify that the sensor setup parameter is done according to the physical configuration on the machine. |
| `q_uiAlrmId` = 101 | Invalid acceleration parameter | Verify that the acceleration parameter is within the specified range. |
| `q_uiAlrmId` = 102 | Invalid deceleration parameter | Verify that the deceleration parameter is within the specified range. |
| `q_uiAlrmId` = 116 | Invalid motor roll diameter limit parameter | Verify that the `i_rMotRollDia` parameter is within the specified range. |

| Issue | Cause | Solution |
|---|---|---|
| q_uiAlrmId = 117 | Invalid speed gradient limit parameter | Verify that the i_rSpdGdnt parameter is within the specified range. |
| q_uiAlrmId = 118 | Invalid speed ratio value parameter | Verify that the i_stUnWd.rSpdRto parameter is within the specified range. |
| q_uiAlrmId = 119 | Invalid minimum unwinder diameter parameter | Verify that the i_stUnWd.rUnwdDiaMin parameter is within the specified range. |
| q_uiAlrmId = 120 | Invalid film end time parameter | Verify that the i_stUnwd.dwTimeFilmEnd parameter is within the specified range. |
| q_uiAlrmId = 121 | Invalid minimum diameter value parameter | Verify that the i_stDiaActl.rDiaMin parameter is within the specified range. |
| q_uiAlrmId = 122 | Invalid maximum diameter value parameter | Verify that the i_stDiaActl.rDiaMax parameter is within the specified range. |
| q_uiAlrmId = 123 | Invalid Set Diameter value parameter | Verify that the i_stDiaActl.rDia parameter is within the specified range. |
| q_uiAlrmId = 124 | Invalid minimum unwinder speed value parameter | Verify that the i_stDiaActl.rSpdMin parameter is within the specified range. |
| q_uiAlrmId = 125 | Invalid line speed value parameter | Verify that the i_rLineSpd parameter is within the specified range. |
| q_uiAlrmId = 126 | Invalid maximum line speed value limit parameter | Verify that the i_stUnwd.rLineSpdMax parameter is within the specified range. |
| q_uiAlrmId = 127 | Invalid minimum line speed value limit parameter | Verify that the i_stUnwd.rLineSpdMin parameter is within the specified range. |

# Chapter 5
# AnalogTensionControl: Controlling Consistent Film Tension Depending on Analog Sensor Feedback

## Overview

This chapter describes the following function blocks:

- `AnalogTensionControlATV`
- `AnalogTensionControlLXM_Motion`
- `AnalogTensionControlATV_Motion`

## What Is in This Chapter?

This chapter contains the following sections:

# Section 5.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Functional Description

Analog tension control makes use of an analog sensor to provide tension feedback to the control system. The linear speed of the film is determined by the Master axis. Analog feedback from a film tension sensor is used to adjust the slave axis speed to maintain the desired film tension.

## Compatibility

The Function Block is applicable to the following types of packaging machines:

- Vertical bagging machine
- Horizontal bagging machine
- Labeling machine

**NOTE:** FB_PID is necessary to use with analog tension control applications.

**NOTE:** Master axis is not controlled by this function block

## Solution With the `AnalogTensionControl` Function Block

The function block controls the velocity of unwinder.

The function block works as a variable gear box between the master and the unwinder. The unwinder axis unwinds the film roll in packaging machines.

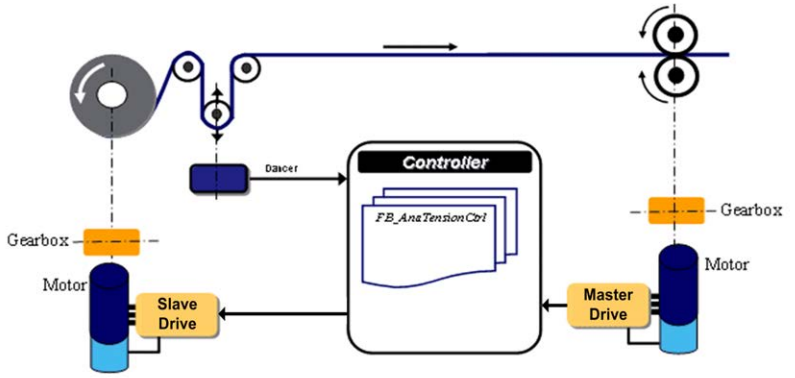
FB_PID can be used to take tension sensor feedback and generate the variable gear ratio for this function block.

## Design & Realization Constraints and Assumption

1. The configuration of OTB on CANopen is not part of these function blocks. It is assumed that application developer will do the required configuration and read the analog value of tension sensor.

2. It is assumed that application developer will do the scaling of analog value of tension sensor and set point if required.

## Functional View



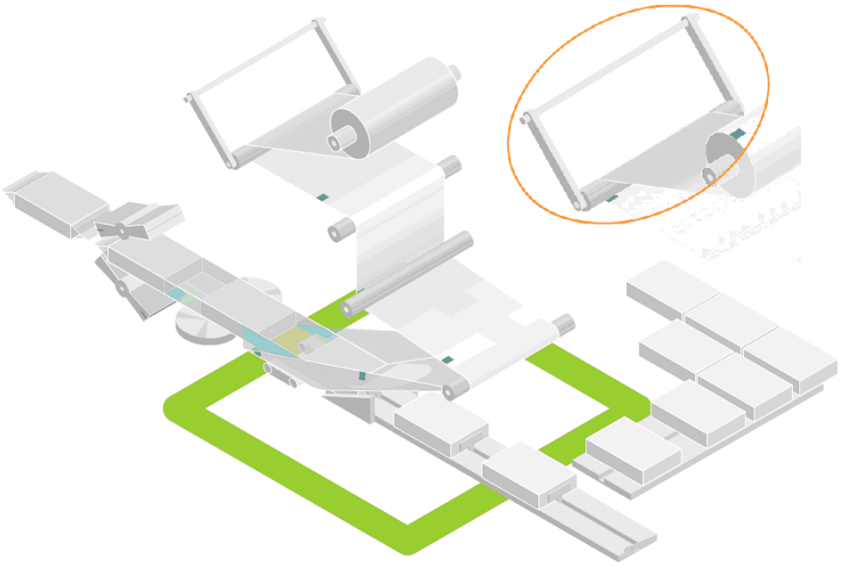

The function block calculates the velocity of a feeder according to the film velocity and the dancer position of the unwinder axis.

Depending on the machine, the motor can be located directly on the shaft of the feeder. This means a large diameter of the roller has a low velocity, whereas a small diameter has a high velocity.

If the diameter of the roller is not measured by using a dedicated sensor in this the average diameter of the roller can be specified. The velocity must be completely regulated using the dancer, from the largest diameter to the smallest diameter.

# Machine Overview

## Horizontal Bagging Machine



# Section 5.2
## Architecture

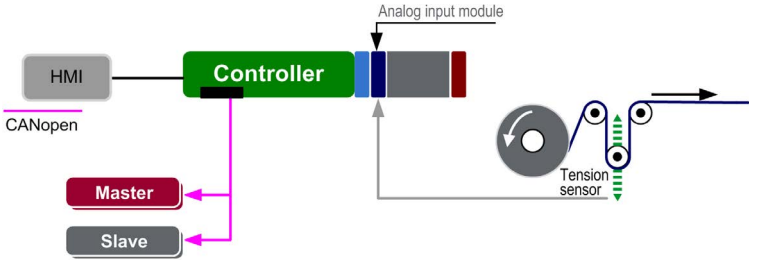
### What Is in This Section?

This section contains the following topics:

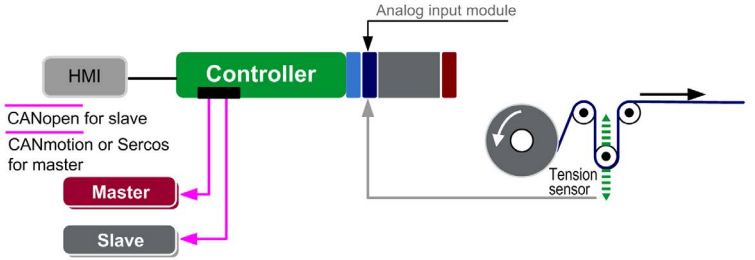
# Hardware Architecture

## Hardware Architecture Overview

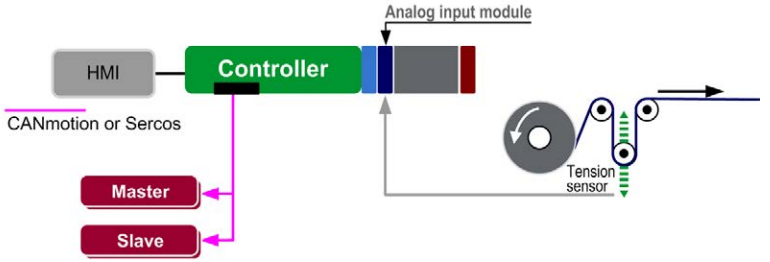
The figure shows the hardware architecture with a Logic Controller, servo drive and a variable speed drive on CANopen.




The figure shows the hardware architecture with a Motion Controller, servo drive on CANmotion or Sercos and a variable speed drive on CANopen.




The figure shows the hardware architecture with a Motion Controller and servor drives on CANmotion or Sercos.



## Hardware Environments

This table describes the architecture types based on the axes configuration of the `AnalogTensionControlATV` function block:

| Master | Slave | Sensor Wiring |
|---|---|---|
| Lexium (LXM) on CANopen | Altivar (ATV) on CANopen | to the Logic Controller through an<br>● analog input module (embedded, expansion or external)<br>or<br>● analog input on drives |

This table describes the architecture types based on the axis configuration of the `AnalogTensionControlATV_Motion` function block:

| Master | Slave | Sensor Wiring |
|---|---|---|
| Lexium (LXM) on<br>● CANmotion or<br>● Sercos | Altivar (ATV) on CANopen | to the Motion Controller through an<br>● analog input module (embedded, expansion or external)<br>or<br>● analog input on drives |

This table describes the architecture types based on the axis configuration of the `AnalogTensionControlLXM_Motion` function block:

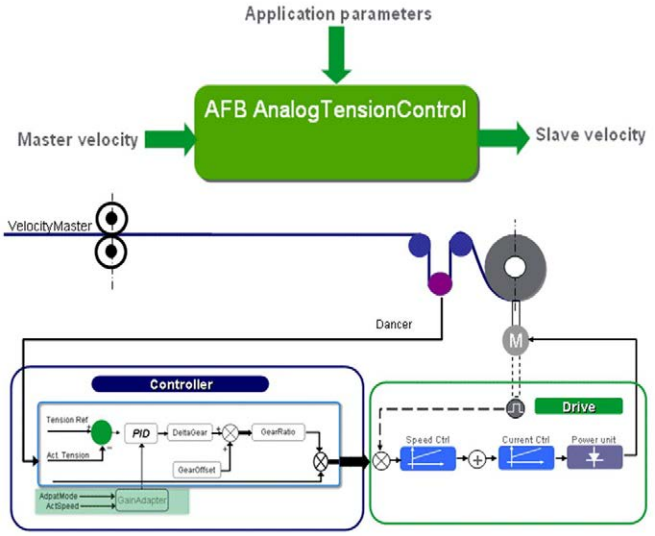
| Master | Slave | Sensor Wiring |
|---|---|---|
| Lexium (LXM) on<br>● CANmotion or<br>● Sercos | Lexium (LXM) on<br>● CANmotion or<br>● Sercos | to the Motion Controller through an<br>● analog input module (embedded, expansion or external)<br>or<br>● analog input on drives |

# Software Architecture

## DataFlow Overview

In analog tension control, actual tension is measured by a tension sensor which is an analog input in the form of voltage (0-10 V) or current (0-20 mA, 4-20 mA).

The function block acts as a variable gear box between master and slave axis.



# Section 5.3
## Function Block Description

## AnalogTensionControl Function Blocks

### Pin Diagram of AnalogTensionControlATV


### Pin Diagram of AnalogTensionControlATV_Motion


### Pin Diagram of AnalogTensionControlLXM_Motion

## Function Block Description

During operation, the Slave axis follows the Master according to the calculated gear ratio. If there is any detected parameter error or drive error, then the function block decelerates the Slave axis to stop and indicates the detected error status and error message.

**NOTE:** If the Gear Ratio is high, there can be unwanted movement in the Slave Axis even if the Master Axis appears to be at standstill. This is due to the nonzero instantaneous feedback values from the Master Axis.
Slave velocity = (master velocity * gear ratio).

# Section 5.4
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

The following table describes the input pin description for AnalogTensionControlATV:

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Enables function block and parameters are validated.<br>FALSE: Disables function block and all outputs are set to zero. |
| i_xCnstGrMode | BOOL | TRUE: Gear ratio = i_rCnstGr<br>FALSE: Gear ratio= (i_rDeltGr * i_rIputCtrl) + i_rGearOfSt |
| i_rIputCtrl | REAL | This pin accepts variable gear ratio.<br>Range: <=0<br>PID output will be connected to this input. |
| i_rCnstGr | REAL (*) | Gear ratio between master axis and slave axis when i_xCnstGrMode is TRUE. |
| i_rDeltGr | REAL | Scaling factor applicable when i_xCnstGrMode is FALSE.<br>-1 <= i_rDeltGr < 0 and 0 < i_rDeltGr <= 1.0<br>Factory setting: 0.1<br>0.0 is not allowed. |
| i_rGearOfSt | REAL (*) | Offset factor applicable when i_xCnstGrMode is FALSE.<br>Factory setting: 0.0<br>Example: If i_rIputCtrl = 1.5, i_rDeltGr = 0.8 and i_rGearOfst = 2.0, then Gear ratio = (1.5*0.8) + 2 = 3.2 |
| i_diAccLim (AnalogTensionControlATV) | DINT | The acceleration limit is used to limit the acceleration when the control input (i_rIputCtrl) changes according to step function. This will help to reduce stress on the mechanical system of the machine. This parameter is applicable to slave axis.<br>Range: 0...6000 (multiplied by X)<br>Factory setting: 10<br>X depends on the drive parameter (Inr) of ramp increment (x0.01, x0.1 or x1)<br>Unit: s |
| i_diDecLim (AnalogTensionControlATV) | DINT | A deceleration ramp or braking ramp can be set independently of one another using ramp-down time. This allows a controlled movement of axis when the control input (i_rIputCtrl) is changed.<br>Range: 0...6000 (multiplied by X)<br>Factory setting: 10<br>X depends on the drive parameter (Inr) of ramp increment (x0.01, x0.1 or x1)<br>Unit: s<br>0 < i_rDecLim <= 6000 x X s ; Factory setting: 100<br>Unit: RPM/s |
| * Optional according to applicable mode | | |

The following table describes the input pin description for motion blocks:

| Input | Data Type | Description |
|---|---|---|
| i_rAccLim (AnalogTensionControlLXM_Motion) | REAL | The acceleration limit is used to limit the acceleration when the control input (i_rIputCtrl) changes according to step function. This will help to reduce stress on the mechanical system of the machine. This parameter is applicable to slave axis.<br>0 < i_rAccLim; Factory setting: 100<br>Unit: RPM/s |
| i_rDecLim (AnalogTensionControlLXM_Motion) | REAL | A deceleration ramp or braking ramp can be set independently of one another using ramp-down time. This allows a controlled movement of axis when the control input (i_rIputCtrl) is changed.<br>0 < i_rDecLim; Factory setting: 100<br>Unit: RPM/s |
| i_diAccLim (AnalogTensionControlATV_Motion) | DINT | The acceleration limit is used to limit the acceleration when the control input (i_rIputCtrl) changes according to step function. This will help to reduce stress on the mechanical system of the machine. This parameter is applicable to slave axis.<br>Range: 0...6000 (multiplied by X)<br>Factory setting: 10<br>X depends on the drive parameter (Inr) of ramp increment (x0.01, x0.1 or x1)<br>Unit: s |
| i_diDecLim (AnalogTensionControlATV_Motion) | DINT | A deceleration ramp or braking ramp can be set independently of one another using ramp-down time. This allows a controlled movement of axis when the control input (i_rIputCtrl) is changed.<br>Range: 0...6000 (multiplied by X)<br>Factory setting: 10<br>X depends on the drive parameter (Inr) of ramp increment (x0.01, x0.1 or x1)<br>Unit: s |

**NOTE:** i_rIputCtrl, i_rDeltGr and i_rGearOfSt is applicable only when i_xCnstGrMode = FALSE. To reset a detected error, a rising edge of i_xEn is required.

**NOTE:** i_diAccLim and i_diDecLim scaling factor (X) is in the drive parameter setting. Factory setting is 0.1. Incase i_diAccLim is a value of 100, then drive interpreted time is 10.0 s.

**NOTE:** To change from Auto Gear mode to Fixed Gear mode and vice versa is possible by using the input i_xCnstGrMode.

## Input/Output Pin Description

### Input/Output Pins of `AnalogTensionControlATV`

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stAxisLxmMstr` | `SEM_LXM.AXIS_REF_LXM` | Servo axis reference structure |
| `iq_stAxisAtvSlav` | `SE_ATV.Axis_Ref_ATV` | ATV axis reference structure |

**NOTE:** Both described In/Outputs are mandatory because these inputs link the function block to the controlled axis.

### Input/Output Pins of `AnalogTensionControlLXM_Motion`

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stAxisLxmMstr` | `SM3_Basic.AXIS_REF_SM3` | Servo axis reference structure |
| `iq_stAxisLxmSlav` | `SM3_Basic.AXIS_REF_SM3` | Servo axis reference structure |

### Input/Output Pins of `AnalogTensionControlATV_Motion`

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stAxisLxmMstr` | `SM3_Basic.AXIS_REF_SM3` | Servo axis reference structure |
| `iq_stAxisAtvSlav` | `SE_ATV.Axis_Ref_ATV` | Servo axis reference structure |

**NOTE:** The Master's line speed is always interpreted as absolute value. The direction of rotation for the Slave will not be changed by inverting the line speed of the Master.

# Output Pin Description

## Output Pin Description

This table describes the output pins of the AnalogTensionControlATV function block:

| Output | Data Type | Description |
|---|---|---|
| q_xEn | BOOL | TRUE: function block enabled. If input i_xEn is TRUE, then output q_xEn is TRUE. |
| q_xBusy | BOOL | TRUE: Function block active and no alarm detected<br>FALSE: Function block disabled or alarm detected |
| q_xAlrm | BOOL | TRUE: Function block detects alarm<br>FALSE: No alarm detected |
| q_uiAlrmId | UINT | Notification when detected alarm output is set.<br>Refer to Notification *(see page 152)*. |
| q_sAlrmMsge | STRING | Notification |
| q_sAlrtMsge | STRING | Alert message<br>This shows the alert message pertaining to slave axis but does not stop the axis.<br>Alert message 'Slave frequency limit reached'. |

**NOTE:** "Slave Maximum Frequency Reached" alarm is generated as soon as the command frequency goes beyond the nominal frequency of the slave axis.

**NOTE:** The LSP and HSP of the slave axis should be taken into consideration and modified as per the requirement of the application. It is recommended to set LSP value to 0.0 Hz.

## Notifications

This table describes the detected alarm IDs and notifications of the function block:

| Detected Alarm ID | Notification |
|---|---|
| 0 | Okay |
| 1 | Internal alarm detected |
| 11 | Master axis alarm detected |
| 12 | Slave axis alarm detected |
| 30 | PLCopen internal alarm detected |
| 31 | PLCopen command aborted |
| 108 | Invalid gear ratio parameter |
| 110 | Invalid acceleration and deceleration parameter |
| 111 | Invalid gear ratio calculation parameter<br>(Only AnalogTensionControlLMX_Motion) |
| 112 | Invalid delta gear ratio parameter |

# Section 5.5
## Quick Reference Guide

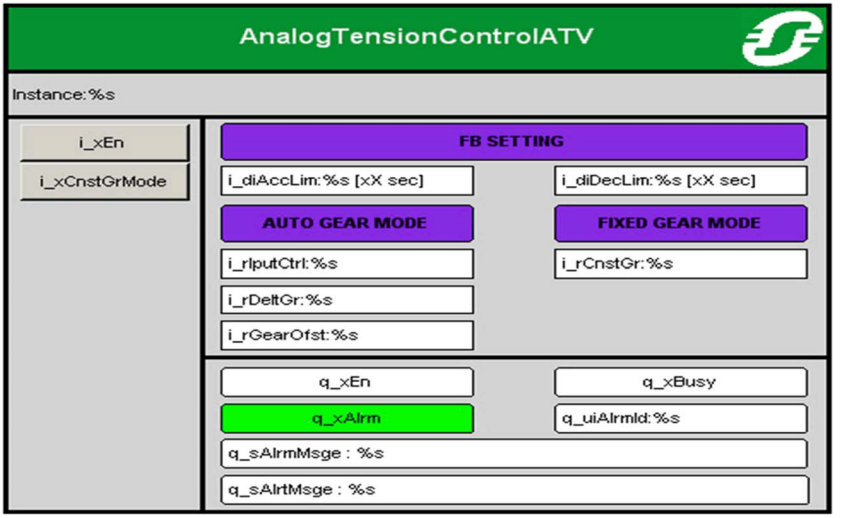
### What Is in This Section?

This section contains the following topics:

# Function Block Visualization

## Visualization

This figure shows a visualization for the `AnalogTensionControlATV` function block:



# Quick Commissioning Procedure

## Quick Commissioning Procedure

This table describes the steps for configuring the `AnalogTensionControlATV` function block:

| Step | Action |
|---|---|
| 1 | Add the Toolbox library into the application program. |
| 2 | Load the FB_PID from Toolbox library and `AnalogTensionControlATV` function block in your application. |
| 3 | Verify that each CANopen Manager, Master and Slave axis have different node numbers and same baud rate. |
| 4 | Define the different input parameters for the `AnalogTensionControlATV` function block from SoMachine Visualization screen or from HMI.<br>Refer to:<br>● parameterizing in fixed gear mode<br>● parameterizing in auto gear mode |
| 5 | Power on the Master axis and enter its speed. |
| 6 | Power on the Slave axis and enable the `AnalogTensionControlATV` function block. |
| 7 | Start the master in velocity mode. |

## Parameterizing in Fixed Gear Mode

This table describes the applicable parameters in fixed gear mode:

| Input | Description | Example |
|---|---|---|
| `i_xEn` | Enables function block | TRUE |
| `i_xCnstGrMode` | Enables fixed gear mode | TRUE |
| `i_rCnstGr` | Value of fix gear ratio | 2.0 |
| `i_diAccLim` | Value of acceleration | 10<br>Unit: xX sec |
| `i_diDecLim` | Value of deceleration | 10<br>Unit: xX sec |
| `iq_stAxisLxmMstr` | Specify Master axis in function block | |
| `iq_stAxisAtvSlav` | Specify Slave axis in function block | |

**NOTE:** The above values were tested in laboratory conditions and are for illustrative purposes only. Appropriate values should be entered as per the actual machine setup.

## Parameterizing in Auto Gear Mode

This table describes the applicable parameter values in fixed gear mode:

| Input | Description | Example |
|---|---|---|
| i_xEn | Enables function block | TRUE |
| i_xCnstGrMode | Enables fixed gear mode | FALSE |
| i_rIputCtrl | Tension sensor feedback | 1.0 |
| i_rDeltGr | Value of delta gear | 0.2 |
| i_rGearOfst | Value of offset gear ratio | 1.00 |
| i_diAccLim | Value of acceleration | 10<br>Unit: xX sec |
| i_diDecLim | Value of deceleration | 10<br>Unit: xX sec |
| iq_stAxisLxmMstr | Specify Master axis in function block | |
| iq_stAxisAtvSlav | Specify Slave axis in function block | |

**NOTE:** The above values were tested in laboratory conditions and are for illustrative purposes only. Appropriate values should be entered as per the actual machine setup.

# Troubleshooting

## Troubleshooting

This table describes some general issues and their solutions:

| Issue | Cause | Solution |
|---|---|---|
| Alarm is raised | Invalid parameter or drive alarm detected | Please verify the detected alarm ID and read the corresponding notification. |
| Slave not responding | Due to CANopen bus traffic / PLCopen function block detected alarm | Disable the function block, switch power off Master, Slave and restart the system. |
| System not responding | Possible invalid value | Disable and enable the function block. |
| Slave axis unwinds too much material. | LSP setting is above set point speed | Reduce LSP settings or adjust value to 0.0 Hz. |
| Slave axis unwinds less material. | HSP setting is below set point speed | Increase HSP setting |
| Slave axis doesn't stops, even if set point frequency is 0.0 Hz | LSP setting is above set point speed | Reduce LSP settings or adjust value to 0.0 Hz. |
| | Altivar parameters are not properly set | Verify the Altivar `CHCF` parameter for the correct setting: command and reference must not be separated. |
| `q_uiAlrmId` = 1 | Internal alarm detected | Reset the function block. |
| `q_uiAlrmId` = 11 | Master axis alarm detected | Verify for detected alarm in Master axis. |
| `q_uiAlrmId` = 12 | Slave axis alarm detected | Verify for alarms in Slave axis. |
| `q_uiAlrmId` = 30 | PLCopen internal alarm detected | Reset the function block. Give the power cycle to the controller if a detected alarm remains. |
| `q_uiAlrmId` = 31 | PLCopen command aborted | Reset the axis. |
| `q_uiAlrmId` = 108 | Invalid variable gear ratio parameter | Verify the parameter `i_rIputCtrl` is in the specified range. |
| `q_uiAlrmId` = 110 | Invalid acceleration and deceleration parameter | Verify the parameter `i_diAccLim` / `i_rAccLim` and `i_diDecLim` / `i_rDecLim` are in the specified range. |
| `q_uiAlrmId` = 112 | Invalid delta gear ratio parameter. | Verify the parameter `i_rDeltGr` is in the specified range. |

# Part IV
# TemperatureControl

## Purpose of This Part

This part explains the functionality and implementation of the `TemperatureControl` function block in the packaging industry.

## What Is in This Part?

This part contains the following chapters:

| Chapter | Chapter Name | Page |
|---|---|---|
| 6 | TemperatureControl: Monitoring and Controlling of Temperature-Dependent Packaging Processes | 161 |
| 7 | TemperatureControl_Easy: Monitoring and Simplified Controlling of Temperature-Dependent Packaging Processes | 193 |

# Chapter 6
# TemperatureControl: Monitoring and Controlling of Temperature-Dependent Packaging Processes

## Overview

This chapter explains the functionality and implementation of the `TemperatureControl` function block in the packaging industry.

## What Is in This Chapter?

This chapter contains the following sections:

# Section 6.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Why Use the `TemperatureControl` Function Block?

The Packaging machines require precise temperature control. For more accurate control of temperature, Proportional, Integral and Derivative functions (PID) should be applied.

The `TemperatureControl` function block incorporates a PID control algorithm for more accurate temperature control in packaging machines.

## Solution with the `TemperatureControl` Function Block

The `TemperatureControl` function block is equipped with an `AutoTuning` algorithm to find the initial parameters for PID loop. The temperature can be effectively controlled by the PID control algorithm when the `AutoTuning` fixes the parameters.

`TemperatureControl` uses FB_PID to generate the control value. FB_PID is equipped with an anti reset windup which holds the integral component when PID output reaches the maximum limit.

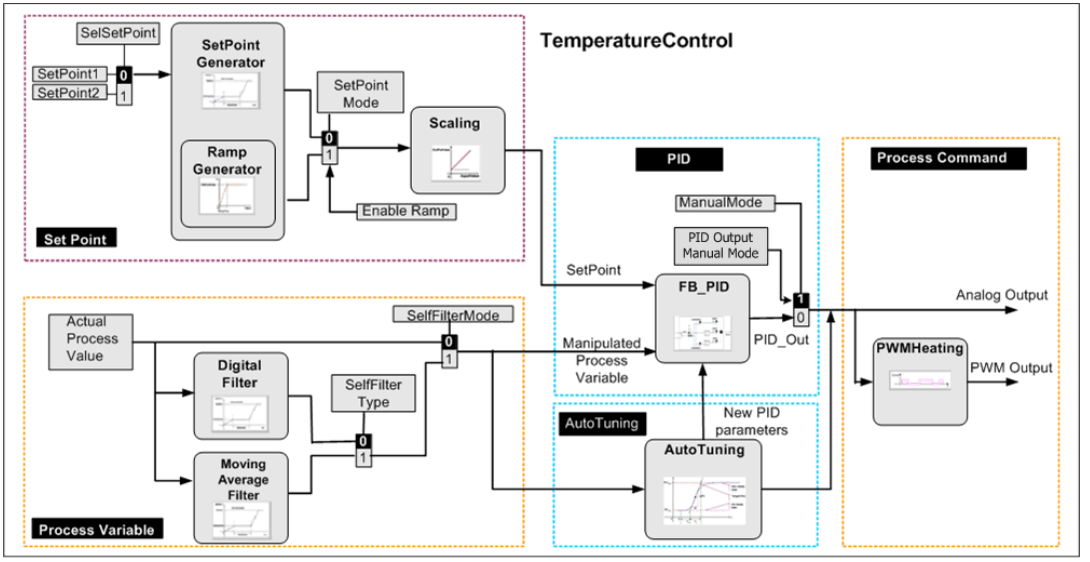
**NOTE:** The `TemperatureControl` function block was developed for heating purpose only.

## Application

The `TemperatureControl` function block is applicable to the following machines:

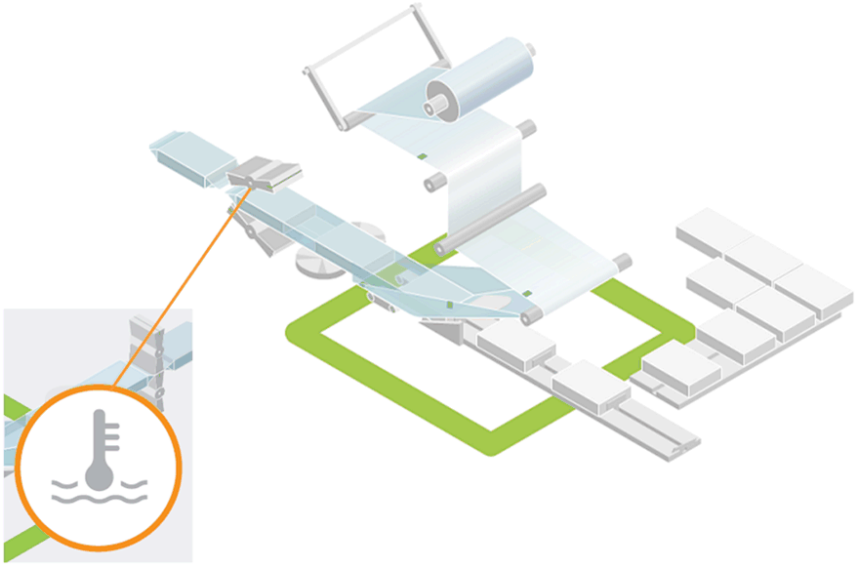
- Horizontal bagging machine
- Vertical bagging machine
- Shrinking machine
- Thermoforming machine

## Functional View



# Machine Overview

## Horizontal Bagging Machine



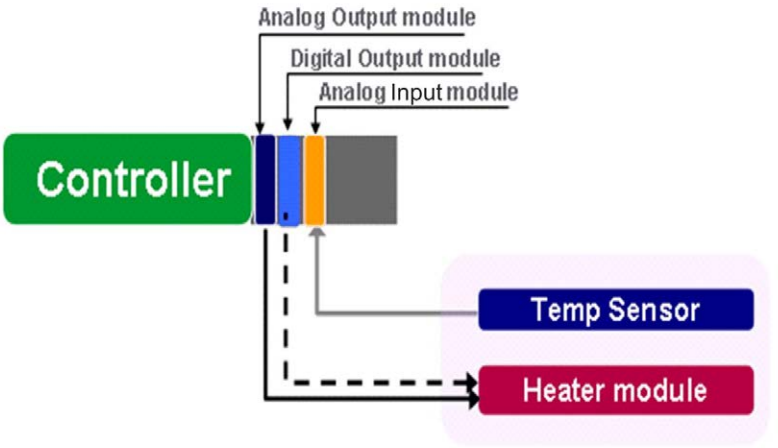
# Section 6.2
## Architecture

### What Is in This Section?

This section contains the following topics:

# Hardware Architecture

## Hardware Architecture Overview



# Software Architecture

## Mathematical Background

The FB_PID function block operates according to the transfer function:

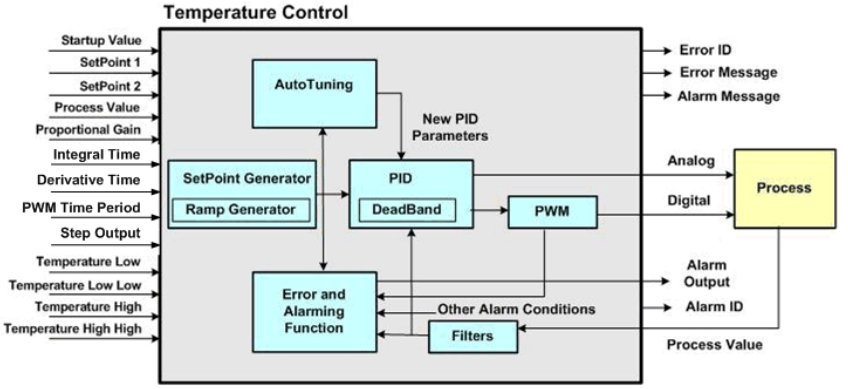
$$G(s) = e(s) \times \left( Kp \left( 1 + \frac{1}{Tn \times s} + \frac{Tv \times s}{1 + Td \times s} \right) \right)$$

The SetPointGenerator generates "SetPoint1" (temperature setting point) and "SetPoint2" (standby set value). Internally it uses a RampGenerator to ramp the setpoint.

The PWM function block transforms FB_PID output into a pulse train with a constant period by modulating the pulse width. Optionally, digital filters are used to reduce the noise in the process value.

The AutoTuning function block operates on the Ziegler-Nichols inflectional tangent method. This function block evaluates the process and provides calculated parameters to the FB_PID function block.

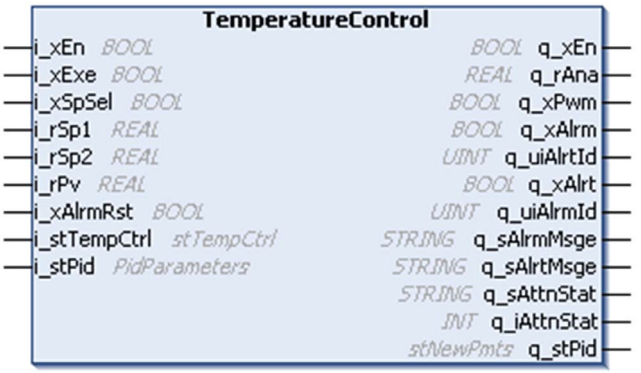
## DataFlow Overview



# Section 6.3
## Function Block Description

## `TemperatureControl` Function

### Pin Diagram




### Function Block Description

The `TemperatureControl` function block is designed for monitoring and controlling a wide variety of temperature dependant processes. It includes the functionality of AutoTuning and the `FB_PID` function block with anti-reset windup.

The function block has three modes of operation:

- Automatic mode (closed loop with PID control or OnOff control)
- Manual mode (open loop)
- Auto tuning mode

In Automatic mode, the control output is generated by `FB_PID`.

In Manual mode, the control output is the desired value entered by you.

In Autotuning mode, only the autotuning function is executed and all other function except the filters are disabled.

## Temperature Measurement

A thermocouple or a resistance thermometer (for example, Pt100) is used to measure the process temperature. The set value for the `FB_PID` function block is limited by SetPoint high and low value, and it can be ramped optionally. The control output is available in the form of an analog output and a PWM output.

## AutoTuning Algorithm

The AutoTuning algorithm is based on the Ziegler-Nichols inflectional tangent method. With this method, the maximum response rate and dead time of the process is calculated. After calculating the time constant, the PID parameters are calculated. There are different monitoring parameters which conditions can be monitored as follows:

- Temperature Low Low
- Temperature Low
- Temperature High High
- Temperature High

## Integrated Sub Functions

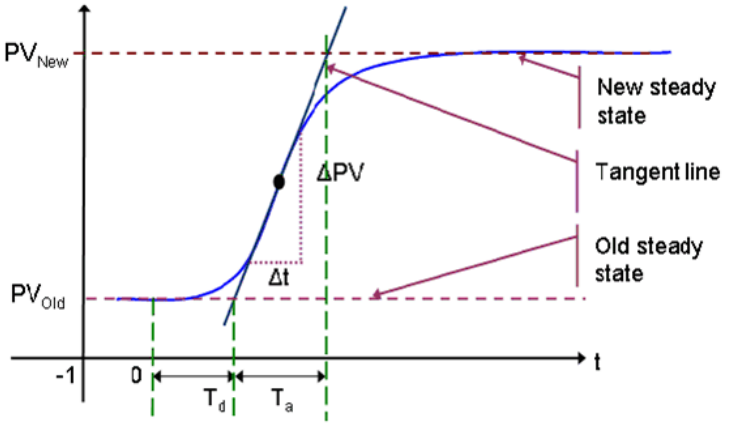
The TemperatureControl function has the following integrated sub functions:

- **AutoTuning**
- **Digital Filter**
- **PWM**

## Auto Tuning

- **Functional Description**
  AutoTuning once enabled disables the PID action or output from the Process.
  A step signal is applied through AutoTuning the process value will be sampled and checked for the steady state by comparing the process value stability within tolerance limits.
  Once the process value reaches the steady state, then the dead time, delay time and the response rate are calculated.
  Based on these values the PID parameters are calculated. These PID values calculated through AutoTuning can be accepted / rejected for executing the PID.
- **Time Constant Function**

This figure shows AutoTuning calculating the time constant:




- **PID Parameter Calculation**

The PID parameters Kp, Tn and Tv will be calculated by the conditions measured during autotuning phase.

| PID Parameters | Formula |
|---|---|
| Kp | 0.60 [TimeConstant/(Dead Time*Ks)] |
| Tn | 1.00 * TimeConstant |
| Tv | 0.50 * TimeConstant |

## Digital Filter

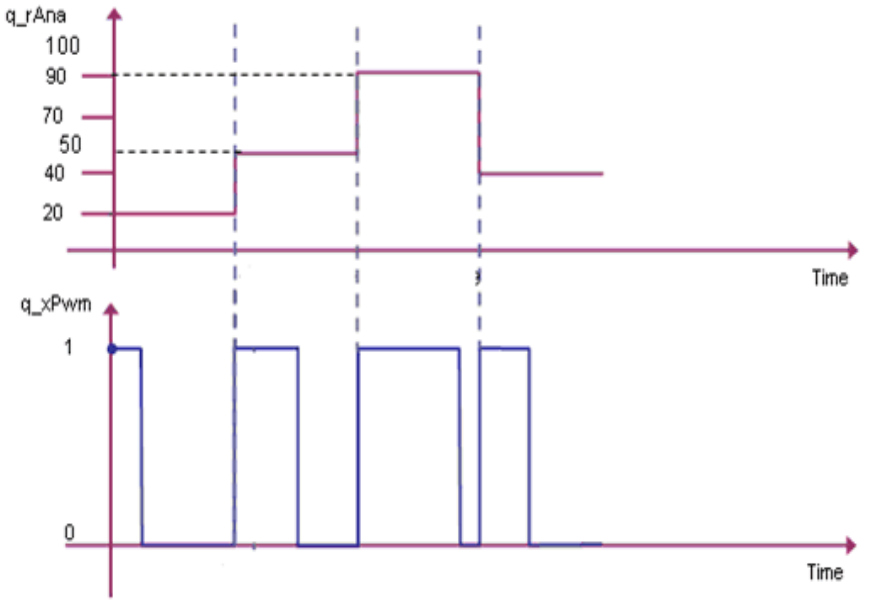
This function incorporates the functionality of digital filter to reduce the noise at the input.

## PWM

The internal PWM Heating function generates pulse width modulation output (q_xPwm) based on the FB_PID output and FB_PID maximum limit.

$$ONTime = \left(\left(\frac{i\_rPidMv}{i\_rPidValMax}\right) \times i\_tTimePerd\right)$$

This figure shows the timing diagram for the `PWM Heating` function block:




**NOTE:** If time period = 5 s and PID output is 20 % (PID high limit = 100) then the `q_xPwm` is high for 1 s and low for 4 s.

# Section 6.4
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

### Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL<br>(All modes) | TRUE: Enables function block and parameters are validated.<br>FALSE: Disables function block and all outputs are set to zero |
| i_xExe | BOOL | TRUE: Accepts new input parameters at the rising edge.<br>FALSE: Does not accept new input parameters. |
| i_xSpSel | BOOL(*)<br>(Auto and Manual modes) | TRUE: Selects i_rSp2<br>FALSE: Selects i_rSp1 |
| i_rSp1 | REAL<br>(Auto and Manual modes) | Value of the actual SetPoint1. The actual setpoint is the temperature that needs to be maintained in the system.<br>Range: i_rLowSp...i_rHighSp<br>Factory setting: 0.0 user temperature unit |
| i_rSp2 | REAL(*)<br>(Auto and Manual modes) | Value of SetPoint 2 in standby mode. The standby setpoint is used to reduce the temperature of the system when the actual temperature control operation is not running.<br>Range: i_rLowSp...i_rSp1<br>Factory setting: 0.0 user temperature unit |
| i_rPv | REAL<br>(Auto and Manual modes) | Temperature value from the process.<br>Range: 0.0 .. 3.4 $e^{+38}$ |
| i_xAlrmRst | BOOL<br>(Auto and Manual modes) | This pin is used to reset the detected errors. To perform the reset you have to change the parameter value which caused the detected error and then initiate a rising edge at this pin. After reset, the rising edge of i_xExe is not required to accept the parameter change.<br>TRUE: Detected error is reset on rising edge<br>FALSE: Detected error is not reset |
| i_stTempCtrl | stTempCtrl | Includes various parameters needed for temperature control<br>Refer to i_stTempCtrl *(see page 179)* table. |
| i_stPid | PidParameters | Includes all the PID parameters<br>Refer to i_stPid *(see page 175)* table. |
| (*) Optional, according to the applicable mode. | | |

# Input: i_stPid

## Input: i_stPid

This structured input consists of PID and AutoTuning parameters. This table describes the input details:

| Structure Parameter | Data Type | Description |
|---|---|---|
| xHold | BOOL(*)<br>(Auto and Manual modes) | TRUE: Holds the FB_PID action.<br>FALSE: Resumes the FB_PID action. |
| rKp | REAL<br>(Auto mode) | Proportional Gain for heating or hysteresis value if On Off Control is active.<br>Range: 0.0...$3.4e^{+38}$<br>Factory setting: 8.0 |
| rTn | REAL<br>(Auto mode) | Integral time for heating<br>Range: 0.0...$3.4e^{+38}$ s<br>Factory setting:152 |
| rTv | REAL<br>(Auto mode) | Derivative time for heating<br>Range: 0.0...$3.4e^{+38}$ s<br>Factory setting: 32 |
| rTd | REAL<br>(Auto mode) | Damping time for heating<br>Range: CycleTime...$3.4e^{+38}$ s<br>Factory setting: 3 |
| xManMode | BOOL(*)<br>(Manual mode) | TRUE: Enables Manual mode<br>FALSE: Disables Manual mode |
| rManVal | REAL(*)<br>(Manual mode) | Manual mode PID output<br>Range: 0...rHighLim<br>Factory setting: 0.0 |
| rTargCyclTime | REAL<br>(All modes) | Cycle time of controller<br>Range: 0...60000 ms<br>Factory setting: 20 ms |
| rLowLim | REAL<br>(Auto and Manual modes) | Low limit of FB_PID output. This value is generally configured to the analog output module specifications.<br>Range: 0.0...less than rHighLim |
| rHighLim | REAL<br>(Auto and Manual modes) | High limit of FB_PID output. This value is generally configured to the analog output module specifications.<br>Range: Greater than rLowLim...$3.4e^{+38}$ |
| rInerWdo | REAL<br>(Auto and Manual modes) | Inner window for reduced I-part<br>Range: 0.0...less than rOterwdo |
| rOterWdo | REAL<br>(Auto and Manual modes) | Outer window for disabling I-part<br>Range: Greater than rInerWdo...$3.4e^{+38}$ |

| Structure Parameter | Data Type | Description |
|---|---|---|
| rDbnd | REAL(*)<br>(Auto mode) | Deadband for detected error and is generally used to make the PID output to settle down.<br>Range: 0.0...100.0<br>Factory setting: 0.0<br>**NOTE:** Any deadband value higher than 0 negatively influences the precision of temperature control. |
| xAttn | BOOL(*)<br>(Auto tuning mode) | TRUE: Enables AutoTuning<br>FALSE: Disables AutoTuning<br>Range: 0...1 |
| rTnce | REAL | Tolerance value for checking the initial steady state to enter Auto tuning procedure.<br>Range: 0...3.4e$^{+38}$<br>Factory setting: 10.0 |
| uiCycl | UINT | Number of cycles to a sample the process value during Auto Tuning<br>Range: 0...65535<br>Factory setting: 200 |
| xAcptPara | BOOL(*)<br>(Auto tuning mode) | TRUE: Accepts new PID parameters given by AutoTuning.<br>FALSE: No action |
| xRjctPara | BOOL(*)<br>(Auto tuning mode) | TRUE: Rejects new PID parameters given by AutoTuning.<br>FALSE: No action |
| xOnOffCtrl | BOOL | TRUE: OnOff control is active.<br>Factory setting: FALSE<br>**NOTE:** If xOnOffCtrl is TRUE, the rKp parameter become the hystersis value. |
| tPwmTimePrd | TIME | In time period for PWM, the heater remains steadily switched on as the process temperature is below the set point temperature.<br>Range: i_stPid.rTarg...4194967295 ms<br>Factory setting: 5 s |
| rPwmTimeOnMin | REAL | If this pin has been set to at least 1 ms, the binary output will be switched on for at least this time. The controller will still attempt to keep to the defined PWM cycle time, but the minimum rPwnMinOnTime will have priority.<br>Range: 0...4194967295 s<br>Factory setting: 0.0 s |
| rPwmTimeOnMax | REAL | If this pin has been set to less than tPwmTimePeriod, the binary output will be switched on for at least this time.<br>Range: 0...4194967295 s<br>Factory setting: 5.0 s |
| (*) Optional, according to the applicable mode. | | |

- rKp: The proportional term of PID changes the output proportional to process error. The proportional output is adjusted by multiplying the process error with gain Kp. The value of gain is defined at the pin rKp.
- rTn: The integral term contribution to PID output is proportional to both the magnitude of process error and duration of process error. Integral term will sum the process error over time (Integrate the process error).The integrated value of process error is divided by integral time. The magnitude of integral output is adjusted by Proportional gain. The value of integral time is defined in variable rTn.
- rTv: Derivative term determines the rate of change of process error over time and multiplying it with derivative time is the derivative output. The derivative term output is adjusted by derivative time. The value of derivative time is defined at the pin rTv.

  **NOTE:** The value of Td should not be less than cycletime. If it is less than cycletime then Td value will be overwritten with the value of cycletime. Parameters uiCycl or rTargCyclTime have an influence on the calculated results of the PID parameters. If parameters uiCycl or rTargCyclTime will be changed afterwards a new execution of the AutoTuning process is mandatory.
- rInerWdo: If the absolute value of the process error is less than rInerWdo then integral calculations are scaled by factor [ABS (process error value) / Inner Window]. The process error value is the difference between the setpoint and the actual process value. If process error value is greater than rInerWdo and less than rOterWdo, normal integral calculations are performed. This window slows down the change in FB_PID output caused by integral part.
  Process error value = Set Point - Process value
- rOterWdo: If the absolute value of the process error is greater than rOterWdo, integral calculations are stopped and integral output is held at the last value. This prevents the integral part from winding up too high and helps to avoid overshoot and oscillation of the controlled system.
  Example: With rInerWdo=10 and rOterWdo=200.
  If Setpoint=250 and Process value=100, then process error value=150 and it is within the range mentioned by window parameters, so normal integral calculation is executed.
  If Setpoint=250 and Process value=245, then process error value=5 and it is less than rInerWdo, so integral calculation is scaled.
  If Setpoint=250 and Process value=30, then process error value=220 which is above rOterWdo, so integral calculation is hold with the previous value.

  **NOTE:** The rInerWdo and rOterWdo parameters should not be configured to have the same value. If these parameters values are configured equally, then the FB will generate a configuration error. The corresponding Error ID is 16 and the notification is INVALID I WINDOW PARAMETER.

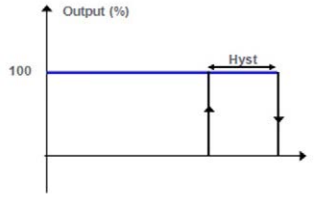
- `i_xAcptPara`,`i_xRjctPara`: These inputs are used to accept or reject the PID parameters calculated by the `AutoTuning` function block. To accept the parameters, you have to enable `i_xAcptPara` followed by a rising edge signal on the `i_xExe` pin. Then the `FB_PID` function block will generate an output based on the new PID parameters.
  To reject the parameters, enable the `i_xRjctPara` pin followed by the rising edge signal on the `i_xExe` pin.The `FB_PID` function block will generate an output based on the PID parameters provided by the user.

  **NOTE:** Once the parameters are accepted, the `PID` function block uses the new parameters calculated by `Auto Tuning` function block. After this if for any other parameter change, execute rising edge is given then PID will use parameters provided through `i_stPid.rKp`, `i_stPid.rTn`, `i_stPid.rTv`.

### `xOnOffCtrl`

If `xOnOffCtrl` is true, the PID is ignored and the control output of the heater system is based on On Off control. The `Kp` parameter became the hysteresis of On Off control.

An On Off control function acts like a thermostat. If the process is below the setpoint, it closes the output contact, and the heating runs at full power. If the setpoint is reached, the power is set to 0%, i.e switched off. If the process value now falls, when it goes below the level (setpoint minus switching differential hysteresis), the output will be switched on again.




If a On Off control is operated in a first order process, then the heating is switched on if the actual temperature of the system is less than the setpoint. Since there is only one component to store the heater energy, the temperature will instantly start to rise. If the setpoint is reached, the power is set to 0%, i.e. switched off, and the process value will not go above the setpoint.

Theoretically, the process value instantly starts to fall and after a certain time it will have fallen to the lower switching point.

The heating is switched on again, and the process value starts to rise again. In the first order process, the process value moves within the band defined by the switching differential. The switching frequency is higher and the controlled becomes faster as the hysteresis is set smaller.

# Input: i_stTempCtrl

## Input: i_stTempCtrl

This structured input consists of function block parameters other than PID and AutoTuning:

| Input | Data Type | Description |
|---|---|---|
| rSpHigh | REAL<br>(Auto mode) | High limit for setpoint<br>Range: > rSpLow2...3.4e$^{+38}$ user temperature unit<br>Factory setting: 0.0 user temperature unit |
| rSpLow | REAL<br>(Auto mode) | Low limit for setpoint<br>Range: 0.0...rSpHigh user temperature unit<br>Factory setting: 0.0 user temperature unit |
| xStup | BOOL(*)<br>(Manual mode) | TRUE: Enables startup function<br>FALSE: Disables startup function |
| rStupVal | REAL(*)<br>(Manual mode) | Percentage of output to the process when startup function is enabled<br>Range: 0.0...100.0%<br>Factory setting: 0.0 % |
| tStupTime | TIME(*)<br>(Manual mode) | Startup function active for this time<br>Range: 0...4194967295 ms<br>Factory setting: 0 ms |
| xRampEn | BOOL(*)<br>(Auto mode) | TRUE: Enables RampGenerator<br>FALSE: Disables RampGenerator |
| rRampGdntRise | REAL(*)<br>(Auto mode) | Value in temperature units/s of ramping time from a lower value to high value<br>Range: 1.0...i_rSp1 or 1.0...i_rSp2 if i_xSelSp is TRUE.<br>Factory setting: 0.0 degree per second |
| rRampGdntFall | REAL(*)<br>(Auto mode) | Value in temperature units/s of ramping time from a higher value to low value.<br>Range: 1.0...i_rSp1 or 1.0...i_rSp2 if i_xSelSp is TRUE.<br>Factory setting: 0.0 degree per second |
| rFltrTime | REAL | Filter Time for the Digital Filter (Noise reduction) in ms. If no value is given the filter is not activated.<br>Range: 0.0...3.4e+38<br>Factory setting: 0.0 |

| Input | Data Type | Description |
|---|---|---|
| rTempLow | REAL<br>(Auto and Manual modes) | Low value of temperature in inner band given in user temperature unit. This limit is related to the actual setpoint and represents the delta below the setpoint value.<br>Example: If i_rSp1=100 degree and rTempLow=10 degree then the low temperature limit will be 90 degree.<br>Range: 0.0...rTempLL<br>Factory setting: 10.0 |
| rCnst | REAL | The constant for the Digital Filter (Noise reduction) is without any unit. The filter is deactivated if no value given for rFltrTime. Increasing of rCnst will reduce noise level but will slow down the change rate of the filtered process value.<br>Range: 1.0...3.4e+38<br>Factory setting: 1.0 |
| rTempLL | REAL<br>(Auto and Manual mode) | Low (LL) value of temperature in outer band given in user temperature unit. This limit is related to the actual setpoint and represents the LL delta below the setpoint value.<br>Example: If i_rSp1=100 degree and rTempLow=20 degree then the low temperature limit will be 80 degree.<br>Range: r_TempLow...3.4e+38<br>Factory setting: 20.0 |
| rTempHigh | REAL<br>(Auto and Manual mode) | High value of temperature in outer band given in user temperature unit. This limit is related to the actual setpoint and represents the delta above the setpoint value.<br>Example: If i_rSp1=100 degree and rTempHigh=10 degree then the high (HH) temperature limit will be 110 degree.<br>Range: 0.0..rTempHH<br>Factory setting: 10.0 |
| rTempHH | REAL<br>(Auto and Manual mode) | High (HH) value of temperature in outer band given in user temperature unit. This limit is related to the actual setpoint and represents the HH delta above the setpoint value.<br>Example: If i_rSp1=100 degree and rTempHH=20 degree then the high (HH) temperature limit will be 120 degree.<br>Range: rTempHigh...3.4e+38<br>Factory setting: 20.0 |

| Input | Data Type | Description |
|---|---|---|
| xAlrtSel2 | BOOL | TRUE: The function block will confirm the value of TempAbsLow and rTempAbsHigh.<br>FALSE:The function block will confirm the value of rTempAbsLow.<br>The value of rTempAbsLow and rTempAbsHigh is Alarm 1 and Alarm 2 in the visualization. |
| rTempAbsLow | REAL<br>(Auto and Manual mode) | Low value of absolute temperature<br>Range: 0.0...3.4e$^{+38}$ user temperature unit |
| rTempAbsHigh | REAL<br>(Auto and Manual mode) | High value of absolute temperature<br>Range: 0.0...3.4e$^{+38}$ user temperature unit |
| (*) Optional, according to the applicable mode. | | |

**NOTE:** User temperature units can be in Celsius, Fahrenheit, or Kelvin.

**rSpHigh, rSpLow**

These values decide the range of i_rSp1. These values will be decided with the minimum and maximum temperature which can be applied to the process, so that it acts as a safeguard to stop the temperature from increasing or decreasing than the desired limits.

**xStup, tStupTime**

xStup enables the startup function. When this is enabled the percentage of output mentioned in rStupVal is applied to the process till tStupTime. This functionality is used at the initial stage to maintain the temperature at lower level before the actual setpoint is effective. During the startup function FB_PID is in manual mode.

**rRampGdntRise, rRampGdntFall**

These pins are used for RampGenerator function. RampGenerator changes its internal setpoint from any present value at the input of i_rPv until the selected setpoint, given by i_rSp1 or i_rSp2 is reached. The value is incremented/decremented once per second by rRampGdntRise for increasing ramp or rRampGdntFall for decreasing ramp.

Ramping functionality is used to increase or decrease the set point gradually so that there will not be sudden increase or decrease in the control value.

# Output Pin Description

## Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xEn | BOOL | TRUE: Function block enabled, if input i_xEn is TRUE, then output q_xEn is TRUE. |
| q_rAna | REAL | Analog form of FB_PID output<br>Range: i_rPidHighLim...i_rPidLowLim<br>Factory setting: 0.0 |
| q_xPwm | BOOL | This is the digital form of FB_PID output<br>TRUE: PWMHeating output high<br>FALSE: PWMHeating output low |
| q_xAlrm | BOOL | TRUE: Alarm detected in temperature control<br>FALSE: No alarm detected (factory setting) |
| q_uiAlrtId | UINT | This gives the alert status of TemperatureControl function block and the unique number for alert.<br>Range: [1...10]<br>Factory setting: 0 |
| q_xAlrt | BOOL | TRUE: Presence of alert in temperature control<br>FALSE: No alerts (factory setting) |
| q_uiAlrmId | UINT | Gives unique notification for detected alarms.<br>Range: [0...340]<br>Factory setting: 0 |
| q_sAlrmMsge | STRING | Displays the notification according to the detected alarm ID. |
| q_sAlrtMsge | STRING | Displays the alert message according to the alert ID. |
| q_sAttnStat | STRING | This is the string output which gives the status message of the Auto Tuning process.<br>Refer to structured output q_sAttnStat *(see page 185)* table.<br>factory setting string: Inactive |
| q_iAttnStat | INT | This gives the AutoTuning status.<br>Range: 0...8<br>Factory setting: 0 |
| q_stPid | stNewPmts | This structured input includes the PID parameters calculated by Auto Tuning.<br>Refer to structured output q_stPid *(see page 186)* table. |

## Notifications

This structured output provides the detected alarm status of `TemperatureControl` function block:

| Detected Alarm ID | Notification |
|---|---|
| 0 | Okay |
| 1 | INTERNAL ALARM |
| 20 | INVALID CYCLETIME |
| 115 | INVALID DEADBAND VALUE |
| 200 | INVALID PID PARAMETER |
| 202 | INVALID I WINDOW PARAMETER |
| 203 | INVALID PID LIMIT PARAM |
| 204 | ALL PID PARAMETERS ARE ZERO |
| 300 | PWM TIME PERIOD INVALID |
| 301 | PWM MINIMUM ONTIME INVALID |
| 302 | PWM MAXIMUM ONTIME INVALID |
| 310 | RAMP GDNT RISE GREATER THAN SP1 OR SP2 OR EQUAL TO ZERO |
| 311 | RAMP GDNT FALL GREATER THAN SP1 OR SP2 OR EQUAL TO ZERO |
| 320 | SETPOINT HIGH LIMIT VALUE LESS THAN LOW SETPOINT |
| 322 | STARTUP TIME INVALID |
| 323 | STARTUP VALUE INVALID |
| 331 | TEMP LOWLOW VALUE INVALID |
| 332 | TEMP HIGH VALUE INVALID |

## Alert Messages

This structured output provides the alert status of `TemperatureControl` function block:

| Alert ID | Alert Message |
|---|---|
| 1 | SETPOINT1 GREATER THAN HIGH SETPOINT |
| 2 | SETPOINT1 LESS THAN LOW SETPOINT |
| 3 | SETPOINT2 GREATER THAN SETPOINT1 OR LESS THAN LOW SETPOINT |
| 4 | TEMP ABS LOW AND HIGH VALUES INVALID |
| 5 | ABSOLUTE TEMPERATURE LOW |
| 6 | ABSOLUTE TEMPERATURE HIGH |
| 7 | TEMPERATURE LOW LOW |
| 8 | TEMPERATURE LOW |
| 9 | TEMPERATURE HIGH |
| 10 | TEMPERATURE HIGH HIGH |
| 11 | STARTUP FUNCTION CANNOT BE ENABLED AT THIS TIME. |

# Output: q_sAttnStat

## Output: q_sAttnStat

This is the string output which gives the status message of the AutoTuning process:

| q_iAttnStat | q_sAttnStat | Description |
|---|---|---|
| 0 | NO ERROR DETECTED | AutoTuning is checking the steady |
| 1 | FB ERROR DETECTED AUTOTUNING COULD NOT PROCEED | AutoTuning is not running due to detected FB error |
| 2 | CHECKING THE STEADY STATE | AutoTuning is checking the steady state of process before going to apply step output to the process |
| 3 | TIME OUT | Process is not in steady state |
| 4 | WAIT | AutoTuning is calculating the PID parameters |
| 5 | CANCELLED | New steady state not reached |
| 6 | READY | AutoTuning has calculated the PID parameters |
| 7 | OK | The new PID parameters are accepted by the user |
| 8 | PARAMETERS ARE NOT ACCEPTED | The new PID parameters are rejected by the user |

**NOTE:** If the status message is displayed as "CANCELLED" and "TIME OUT", then to restart AutoTuning, you have to disable and enable the FB.

# Output: q_stPid

## Output: q_stPid

This structured output includes the PID parameters calculated by AutoTuning. The structure details are explained in the given table:

| Output | Data Type | Description |
|---|---|---|
| rKp | REAL | Value of proportional gain calculated by AutoTuning<br>Range: 0.0...3.4e$^{+38}$<br>Factory setting: 0.0 |
| rTn | REAL | Value of integral time calculated by AutoTuning<br>Range: 0.0...3.4e$^{+38}$<br>Scaling/unit: s<br>Factory setting: 0.0 |
| rTv | REAL | Value of derivative time calculated by AutoTuning<br>Range: 0.0...3.4e$^{+38}$<br>Scaling/unit: s<br>Factory setting: 0.0 |

**NOTE:** In the event of a detected error, the function block will be initialized. To execute the function block again, correct the detected error and enable i_xRst.
Structure elements i_stPid.uiCycl or i_stPid.rTargCyclTime have an influence on the calculated results of the PID parameters. If parameters uiCycl or rTargCyclTime will be changed afterwards a new execution of the AutoTuning process is mandatory.

# Section 6.5
## Quick Reference Guide

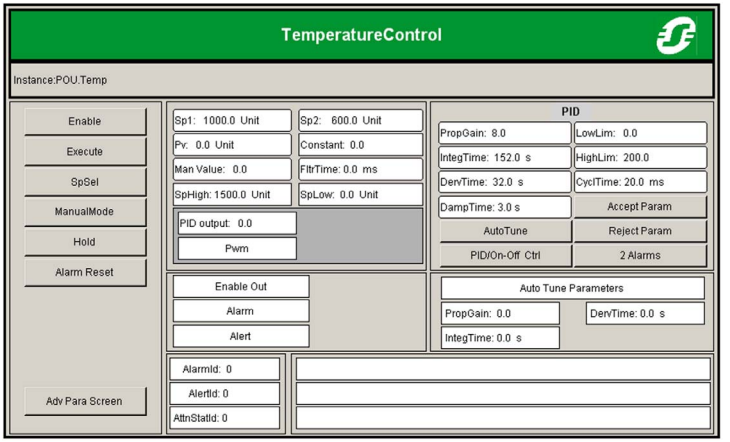
### What Is in This Section?

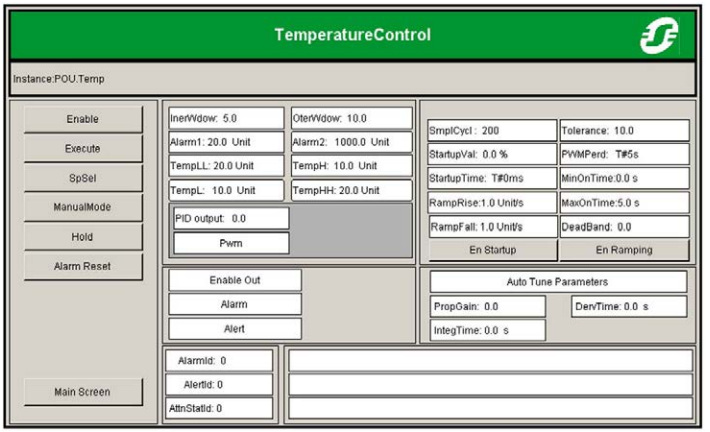
This section contains the following topics:

# Function Block Visualization

## Introduction

In the following figures, you will see the visualization of the function block in the software:



# Quick Commissioning Procedure

## Commissioning Procedure

This table describes the quick commissioning procedure for the `TemperatureControl` function block:

| Step | Action |
|---|---|
| 1 | Add the Packaging Library into the application program. |
| 2 | Load the `TemperatureControl` function block in your application. |
| 3 | Enter all parameters in the controller configuration. |
| 4 | Connect to the function block input `i_rPv` the variable assigned to the temperature sensor. |
| 5 | Connect the function block output `q_xPwm` to the variable assigned to the heater module. |
| 6 | Enable the function block with `i_xEn`. |
| 7 | Enable AutoTuning with parameter `i_stPid.xAttn`. |
| 8 | Accept all new inputs with a rising edge signal on `i_xExe`. |
| 9 | AutoTuning is running. The status must be verified at the output `q_iAttnStat` and `q_sAttnStat`. |
| 10 | After AutoTuning has calculated the PID parameter (state 6), you have to accept (`i_stPid.xAcptPara`) or reject (`i_stPid.xRjctPara`) them, followed by a rising edge signal on the `i_xExe` pin. |
| 11 | Disable the parameter `i_stPid.xAttn` and `i_stPid.xAcptPara`. |
| 12 | After verification of the Auto Tuning result, use the calculated PID parameter at `q_stPid` as the input parameter for `i_stPid`. |

**NOTE:** For commissioning, it is helpful to use the function block visualization and also to create a trace with the parameter `i_rSp1`, `i_rPv`, `q_rAna` and `q_xPwm`.

## Parameterizing

This table lists the input parameters to be entered:

| Parameter | Example Value |
|---|---|
| i_xSpSel | FALSE |
| i_rSp1 | 145 |
| i_rSp2 | 120 |
| i_stTempCtrl.xAlrtSel2 | TRUE |
| i_stTempCtrl.rSpHigh | 200 |
| i_stTempCtrl.rSpLow | 20 |
| i_stTempCtrl.xStup | FALSE |
| i_stTempCtrl.rStupVal | 0 |
| i_stTempCtrl.tStupTime | 0 |
| i_stTempCtrl.xRampEn | FALSE |
| i_stTempCtrl.rRampGdntRise | 1 |
| i_stTempCtrl.rRampGdntFall | 1 |
| i_stTempCtrl.rFltrTime | 0 |
| i_stTempCtrl.rCnst | 1 |
| i_stTempCtrl.rTempLow | 5 |
| i_stTempCtrl.rTempLL | 10 |
| i_stTempCtrl.rTempHigh | 5 |
| i_stTempCtrl.rTempHH | 10 |
| i_stTempCtrl.rTempAbsLow | 10 |
| i_stTempCtrl.rTempAbsHigh | 180 |
| i_stPid.xHold | FALSE |
| i_stPid.xOnOffCtrl | FALSE |
| i_stPid.rKp | 8 |
| i_stPid.rTn | 152 |
| i_stPid.rTd | 3 |
| i_stPid.xManMode | FALSE |
| i_stPid.rManVal | 0 |
| i_stPid.rTargCyclTime | 20 |
| i_stPid.rLowLim | 0 |
| i_stPid.rHighLim | 100 |
| i_stPid.rInerWdow | 0 |
| i_stPid.rOterWdow | 0 |

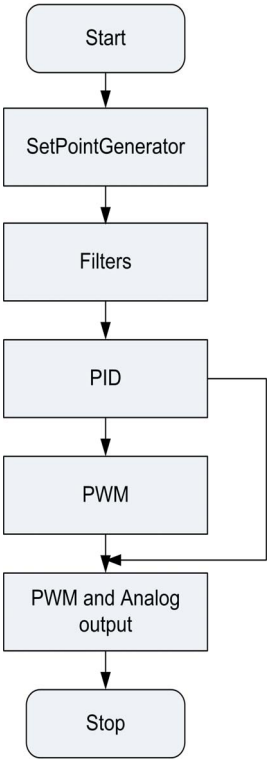
| Parameter | Example Value |
|---|---|
| i_stPid.rDbnd | 0 |
| i_stPid.uiCycl | 200 |
| i_stPid.rTnce | 10 |
| i_stPid.xAcptPara | FALSE |
| i_stPid.xRjctPara | FALSE |
| i_stPid.tPwmTimePrd | 10 000 |
| i_stPid.rPwmTimeOnMin | 0 |
| i_stPid.rPwmTimeOnMax | 10 |

**NOTE:** The values above were tested under laboratory conditions and are for illustrative purposes only. Appropriate values should be entered as per the actual machine setup.
It is important that the task cycletime is given to `i_stPid.rTargCyclTime`. Otherwise, the calculation of the PID output will be imprecise (I-part).

## Internal Process Diagram

### Execution Order of TemperatureControl Function Block

This figure shows the execution order of internal functions in `TemperatureControl` function block:



# Chapter 7
# TemperatureControl_Easy: Monitoring and Simplified Controlling of Temperature-Dependent Packaging Processes

## Overview

This chapter explains the functionality and implementation of the `TemperatureControl_Easy` function block in the packaging industry.

## What Is in This Chapter?

This chapter contains the following sections:

# Section 7.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Why Use the TemperatureControl_Easy Function Block?

The packaging machines require precise temperature control. For more accurate control of temperature, proportional, integral, and derivative functions (PID) should be applied.

The TemperatureControl_Easy function block incorporates a PID control algorithm for more accurate temperature control in packaging machines.

## Solution with the TemperatureControl_Easy Function Block

TemperatureControl_Easy uses FB_PID to generate the control value. FB_PID is equipped with an anti reset windup which holds the integral component when PID output reaches the maximum limit.

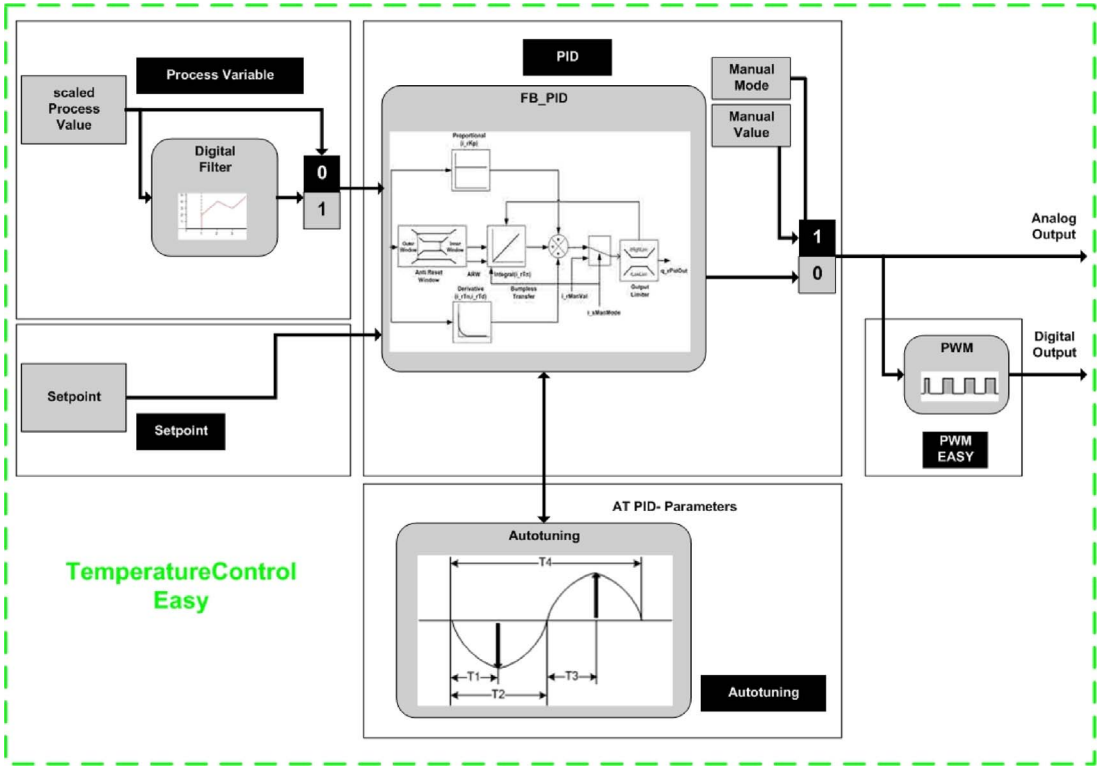
**NOTE:** The TemperatureControl_Easy function block was developed for heating purpose only.

## Application

The TemperatureControl_Easy function block is applicable to the following machines:

- Horizontal bagging machine
- Vertical bagging machine
- Shrinking machine
- Thermoforming machine

## Functional View

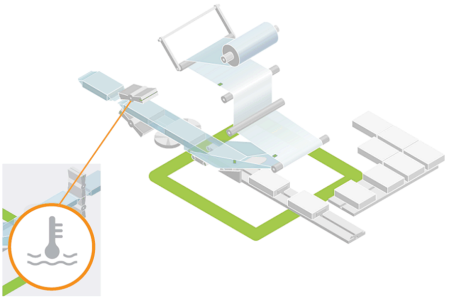


# Machine Overview

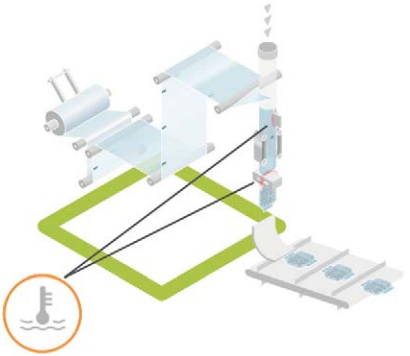
## Introduction

The `TemperatureControl_Easy` function block is applicable to the following machines:

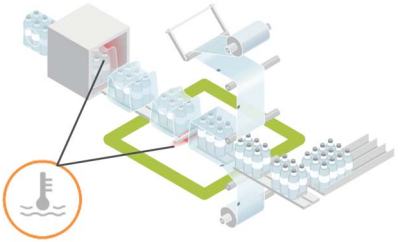
Horizontal Bagging Machine



Vertical Bagging Machine



Shrinking Machine



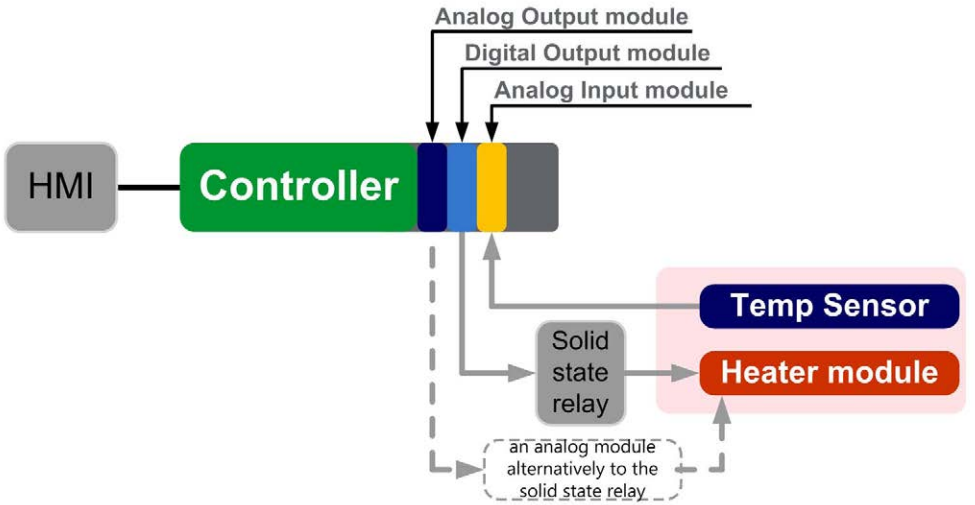
# Section 7.2
## Architecture

### What Is in This Section?

This section contains the following topics:

# Hardware Architecture

## Hardware Architecture Overview



# Software Architecture

## Mathematical Background

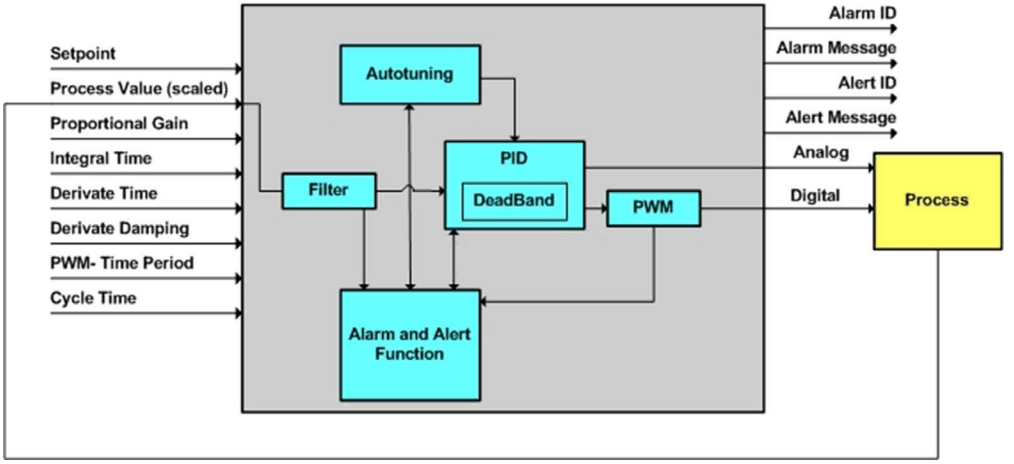
The FB_PID function block, from the Toolbox library *(see SoMachine, Miscellaneous Functions, Toolbox Library Guide)*, operates according to the transfer function:

$$G(s) = e(s) \bullet (Kp) + e(s) \bullet (Kp)\left(1 + \frac{1}{Tn \bullet s}\right) + Kp \bullet e(s) \bullet \left(\frac{Tv \bullet s}{1 + Td \bullet s}\right)$$

The PWM function block transforms FB_PID output into a pulse train with a constant period by modulating the pulse width. Optionally, digital filters are used to reduce the noise in the process value.

The AutoTuning function block operates on the relay method. This function block evaluates the process and provides calculated parameters to the FB_PID function block.

## DataFlow Overview



# Section 7.3
# Function Block Description

## TemperatureControl_Easy Function

### Pin Diagram


### Function Block Description

The `TemperatureControl_Easy` function block is designed for monitoring and controlling a wide variety of temperature dependant processes. It includes the functionality of `AutoTuning` and the `FB_PID` function block with anti-reset windup.

The function block has three modes of operation:

- Automatic mode (closed loop with `PID` control)
- Manual mode (open loop)
- AutoTuning mode

In Automatic mode, the control output is generated by `FB_PID`.

In manual mode, the control output is the desired value entered by you.

In AutoTuning mode, only the `AutoTuning` function is executed and all other function except the filters are disabled.

### Temperature Measurement

A thermocouple or a resistance thermometer (for example, Pt100) is used to measure the process temperature. The set value for the FB_PID function block is limited by SetPoint high and low value. The control output is available in the form of an analog output and a PWM output.

### AutoTuning Algorithm

The AutoTuning algorithm is based on the Relay method. In this method the process is induced by oscillations. After the completion of three oscillations the AutoTuning calculates a set of PID parameters.

### Integrated Sub Functions

The TemperatureControl_Easy function has the following integrated sub functions:

- AutoTuning
- Digital Filter
- PWM

# Section 7.4
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

| Input | Data type | Description |
|---|---|---|
| i_xEn | BOOL<br>(All modes) | TRUE: Enables function block and parameters are validated.<br>FALSE: Disables function block and all outputs are set to zero |
| i_rSp | REAL<br>(Auto and AutoTuning modes) | The SetPoint is the temperature that needs to be maintained by the system.<br>Range: rSpLow ... rSpHigh<br>Factory setting: 0.0 user temperature unit(1) |
| i_rPv | REAL<br>(Auto and AutoTuning modes) | Temperature value from the process.<br>The process temperature value has to be scaled outside of the function block (for example, by FB_Scaling)(1).<br>Range: 0.0.. 3.4 $e^{+38}$ |
| i_xManMode | BOOL<br>(Manual mode) | TRUE: Enables Manual mode<br>FALSE: Disables Manual mode |
| i_xManVal | REAL<br>(Manual mode) | Manual mode PID output<br>Range: rLowLim ... rHighLim<br>Factory setting: 0.0 |
| i_xRst | BOOL<br>(All modes) | FALSE: Alarm and Alerts were not reset<br>TRUE: Alarm and Alerts are reset on rising edge<br>To perform the reset, modify the parameter which caused the alarm, and then give a rising edge on pin i_xRst. |
| i_stTempCtrlPara | stTempCtrl_Easy | Includes various parameters needed for temperature control<br>Refer to i_stTempCtrlPara *(see page 205)* table. |
| (1) You must use the same unit for the set point (i_rSp) and process value (i_rPv). | | |

### i_stTempCtrlPara

This structured input (using structure stTempCtrl_Easy) consists of the following parameters.

| Input | Data type | Description |
|---|---|---|
| xHold | BOOL | TRUE: Holds the FB_PID action.<br>FALSE: Resumes the FB_PID action. |
| rHighLim | REAL | High limit of FB_PID output.<br>Range: rLowLim...100.0%<br>Factory setting: 0.0% |
| rLowLim | REAL | Low limit of FB_PID output.<br>Range: 0.0%... rHighLim<br>Factory setting: 0.0% |
| rDbnd | REAL<br>(Auto mode) | Dead band for detected error and is used to make the PID output to settle down.<br>Range: 0.0...100.0 user temperature unit<br>Factory setting: 0.0 user temperature unit<br>**NOTE:** Any dead band value greater than 0 negatively influences the precision of temperature control. |
| xAttn | BOOL<br>(AutoTuning mode) | Enables AutoTuning by a rising edge.<br>Going back to FALSE does not stop AutoTuning.<br>Range: Pulse 0...1<br>Factory setting: FALSE |
| rPwmTimePeriod | REAL<br>(All modes) | Time period for PWM<br>The Time Period scales the q_rAna to PWM-Output-ON-Time (q_xPwm)<br>Range: 5.0...60.0 s<br>Factory setting: 10.0 s |
| rSpHigh | REAL<br>(All mode) | High limit for setpoint<br>Range: > rSpLow...3.4e$^{+38}$ user temperature unit<br>Factory setting: 100.0 user temperature unit |
| rSpLow | REAL<br>(All mode) | Low limit for setpoint<br>Range: 0.0...rSpHigh user temperature unit<br>Factory setting: 0.0 user temperature unit |
| rFltrTime | REAL<br>(All modes) | Filter time for the digital filter (Noise reduction) in ms. If no value is given, the filter is not activated.<br>Range: 0.0...3.4e+38 ms<br>Factory setting: 0.0 ms |

**NOTE:** The cycle time is calculated internal by the function SysTimeGetMs.

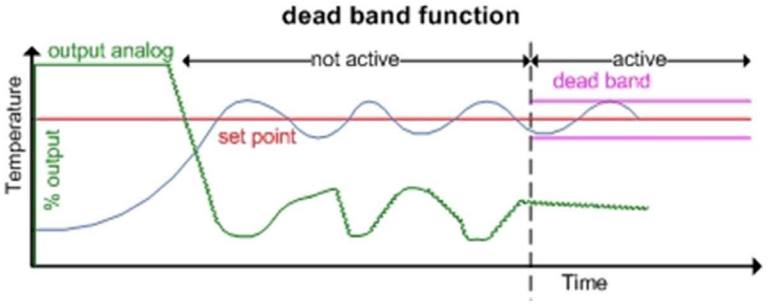
**NOTE:** User temperature units can be in Celsius, Fahrenheit, or Kelvin.

### rLowLim, rHighLim

The values at these pins define the range of PID output. These values are configured as per the analog output module specification.The range is 0.0 to 100.0 percent.

### rDbnd

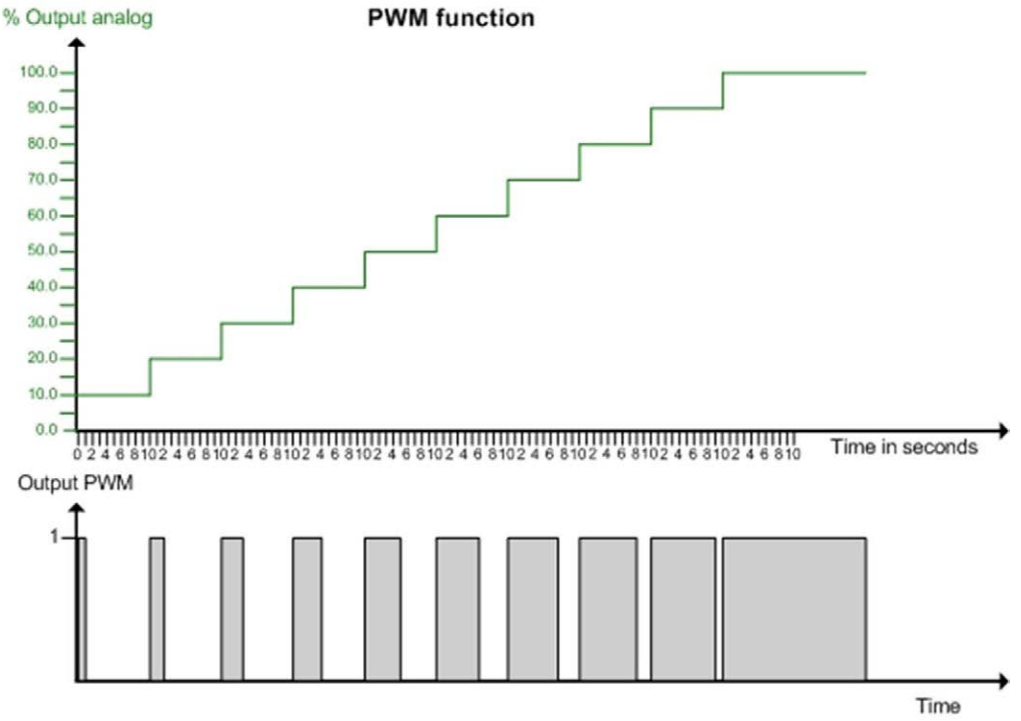
The value at this pin gives the limit of the dead band for detected error. This dead band function is used to settle down the PID output.




### rPwmTimePrd

**Example**: rPwmTimePrd:=10.0 s, rHighLim:=100.0%, rLowLim:=0.0%, q_rAna:=10.95% (actually)

- PWM Time On: = 1,095 seconds.
- PWM Time Off: = 8,905 seconds.

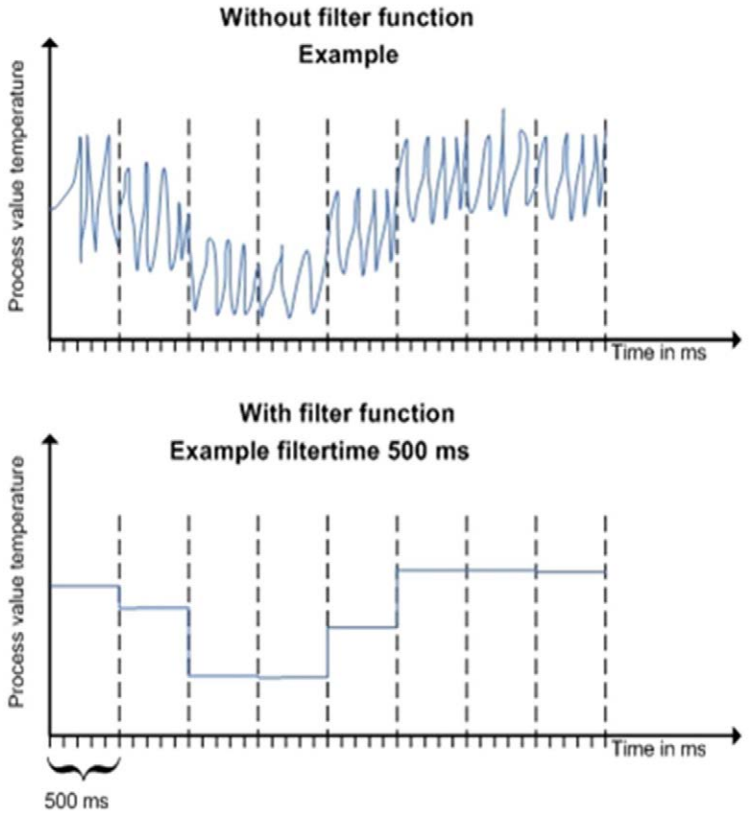



**rSpHigh, rSpLow**

These values determine the range of `i_rSp`. These values are decided with the minimum and maximum temperature which can be applied to the process, so that it stops the temperature from increasing or decreasing then the desired limits.

**rFltrTime**

If you get a noisy signal of the process value, you have to set the filter time. The filter is active when the filter time is greater than 0.0 ms. When the filter time is set, you get the medium process value for the entered filter time.


Without filter function
Example
Process value temperature
Time in ms
With filter function
Example filtertime 500 ms
Process value temperature
Time in ms
500 ms

# Input/Output Pin Description

## Input/Output Pin Description

The following table includes the input/output of the function block along with the description.

| Structure parameter | Data type | Description |
|---|---|---|
| iq_stPid | stPmtsNew_Easy | Includes the PID parameters<br>**NOTE:** Refer the stPmtsNew_Easy following table. |
| iq_iCtrlMode | INT | Change the PID parameters:<br>0= Default Para<br>1= Manual Para<br>2= ATR-Para (slow)<br>3= ATR-Para (medium)<br>4= ATR-Para (aggressive) |

**NOTE:** ATR= AutoTuning Results (PID-Parameter).

### iq_iCtrlMode

Five different control modes can be selected.

Each control mode includes a set of PID parameter (rKp, rTn, rTv, rTd)

| Value | Control mode | Description |
|---|---|---|
| 0 | Default parameters | rKP:=2.0, rTn:= 200.0, rTv:=0.0, rTd:=0.0 |
| 1 | Manual mode | You can set parameters, default is the same as control mode 0. |
| 2 | Slow control | Conservative variant and reduce the overshoot. |
| 3 | Medium control | Medium variant |
| 4 | Aggressive control | Aggressive variant reaches faster the set point temperature, but a greater overshot must be taken into account. |

**NOTE:** After AutoTuning, the calculated parameters are available and control mode 3 is selected automatically.

### iq_stPid

This structured input/output (using structure stPmtsNew_Easy) consists of PID parameters and is described in the following table:

| Structure parameter | Data type | Description |
|---|---|---|
| rKp | REAL | Value of proportional gain<br>Range: 0.0...3.4e$^{+38}$<br>Factory setting: 2.0 |
| rTn | REAL | Value of integral time<br>Range: 0.0...3.4e$^{+38}$ s<br>Factory setting: 200 s |
| rTv | REAL | Value of derivative time<br>Range: 0.0...3.4e$^{+38}$ s<br>Factory setting: 0.0 s |
| rTd | REAL | Value of damping time for derivative action<br>Range: CycleTime...3.4e$^{+38}$ s<br>Factory setting: 0.0 s |

**NOTE:** In runtime, this structure shows the actual set of PID parameter.

### rKp

The proportional term of PID changes the output proportional to process error. The proportional output is adjusted by multiplying the process error (difference between setpoint and actual process value) with gain Kp. The value of gain is defined at the pin rKp.

### rTn

The integral term contribution to PID output is proportional to both the magnitude of process error and duration of process error. Integral term will sum the process error over time (Integrate the process error).The integrated value of process error is divided by integral time. The magnitude of integral output is adjusted by proportional gain. The value of integral time is defined in variable rTn.

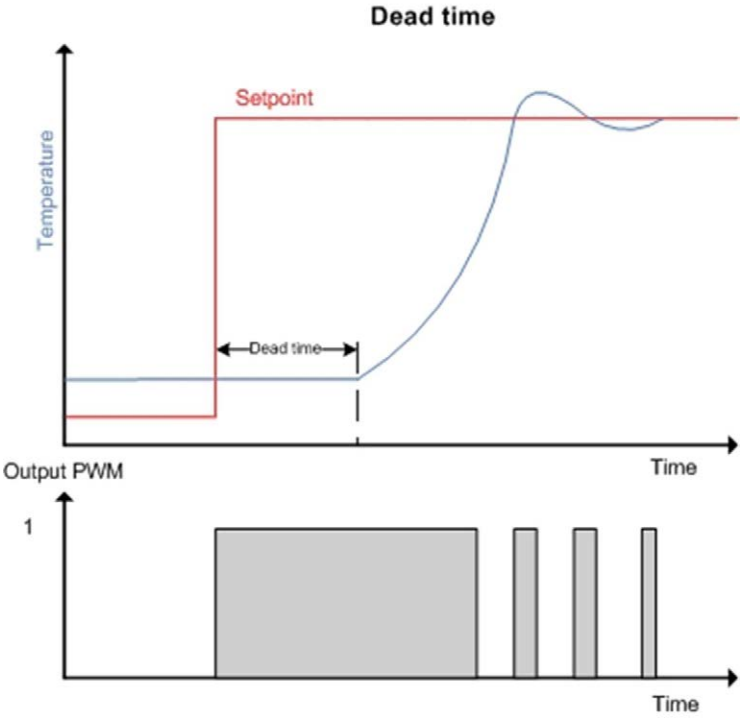
### rTv

Derivative term determines the rate of change of process error over time and multiplying it with derivative time is the derivative output. The derivative term output is adjusted by derivative time. The value of derivative time is defined at the pin rTv.

**rTd**

Damping time reduce the influence of the derivate term proportionally.

**NOTE:** In systems with dead time (see figure below), `rTv` and `rTd` should be 0.0. The value of `Td` should not be less than cycletime. If it is less than cycletime, then `Td` value is overwritten with the value of cycletime. If all of these input parameters are configured as zero and if `AutoTuning` has not calculated the values, then the default parameters are set after a reset of the alarm.



# Output Pin Description

## Output Pin Description

| Output | Data type | Description |
|---|---|---|
| q_xEn | BOOL | TRUE: Function block enabled, if input i_xEn is TRUE, then output q_xEn is TRUE.<br>FALSE: Function block disabled |
| q_rAna | REAL | Analog form of FB_PID output or manual value (in manual mode)<br>Range: rLowLim ... rHighLim<br>Factory setting: 0.0 |
| q_xPwm | BOOL | PWM (ON/OFF) signal according to q_rAna<br>TRUE: PWMHeating on<br>FALSE: PWMHeating off |
| q_xAttnEn | BOOL | FALSE: AutoTuning inactive<br>TRUE: AutoTuning active |
| q_iAttnStat | INT | This gives the AutoTuning status.<br>Range: 0...4<br>Factory setting: 0 |
| q_sAttnStat | STRING | This is the string output which gives the status message of the AutoTuning process.<br>Refer to table AutoTuning state. *(see page 213)*<br>Factory setting string: Tuning inactive |
| q_xAlrm | BOOL | TRUE: Alarm<br>FALSE: No alarm |
| q_uiAlrmId | UINT | Gives unique notification for alarms.<br>Range: Refer to table Notifications *(see page 213)*.<br>Factory setting: 0 |
| q_sAlrmMsge | STRING | Displays the notification according to the alarm ID. |
| q_xAlrt | BOOL | TRUE: Alert<br>FALSE: No alerts |
| q_uiAlrtId | UINT | Gives unique alert ID for alerts.<br>Range: Refer to table Alert messages *(see page 214)*.<br>Factory setting: 0 |
| q_sAlrtMsge | STRING | Displays the alert message according to the alert ID. |
| q_stPidPara | ARRAY [0...4] stPmtsNew_Easy | This array of structured PID parameters includes the Default, Manual and by AutoTuning calculated values.<br>Refer to Array of PID Parameters *(see page 214)* table. |

## AutoTuning State

This table list the AutoTuning state number and the corresponding message.

| State number<br>q_iAttnStat | State message<br>q_sAttnStat | Description |
|---|---|---|
| 0 | Tuning inactive | Only default |
| 1 | Tuning in progress | – |
| 2 | Tuning completed system OK | AutoTuning is finished and the system comparison detected a balanced system. |
| 3 | Tuning completed system is oversized | AutoTuning is finished and the system comparison detected an oversized system.<br>In this case the calculated PID parameter could be not applicable. Then enter manual parameter or do AutoTuning again. |
| 4 | Tuning completed system is undersized | AutoTuning is finished and the system comparison detected an undersized system.<br>In this case the calculated PID parameter could be not applicable. Then enter manual parameter or do AutoTuning again. |

For further information about system comparison, refer to System comparison after AutoTuning *(see page 219)*.

## Notifications

This table list the possible alarm IDs with the corresponding alarm messages.

| Alarm ID<br>q_uiAlrmId | Alarm message<br>q_sAlrmMsge |
|---|---|
| 0 | Okay (no alarm) |
| 1 | INTERNAL ALARM |
| 20 | INVALID CYCLETIME |
| 115 | INVALID DEAD BAND VALUE |
| 200 | INVALID PID PARAMETER |
| 205 | GENERAL ALARM FROM FB_PID |
| 206 | PID LOW LIMIT IS EQUAL TO HIGH LIMIT |
| 300 | PWM TIME PERIOD INVALID |
| 303 | INVALID OUTPUT RANGE |
| 304 | OUTPUT LOW LIMIT GREATER THAN HIGH LIMIT |
| 305 | INVALID FILTER TIME |
| 320 | SETPOINT HIGH LIMIT VALUE LESS THAN LOW SETPOINT |

**NOTE:** In case of alarm, `q_xAlrm` is TRUE. Under this condition, the function block is initialized. To execute the function block again, you have to rectify the alarm and enable `i_xRst`. A new AutoTuning is not required.

## Alert Messages

This table list the possible alert IDs and the corresponding alert messages.

| Alert ID<br>`q_uiAlrtId` | Alert Message<br>`q_sAlrtMsge` |
|---|---|
| 0 | NO ALERT |
| 1 | SETPOINT GREATER THAN HIGH SETPOINT |
| 2 | SETPOINT LESS THAN LOW SETPOINT |
| 12 | PARAMETER MODE HIGH OR LOW--> ONLY 0, 1, 2, 3, 4 |
| 15 | INVALID MANUAL VALUE: out of range `rLowLim` ... `rHighLim` |

## `q_astPidPara` - Array of PID Parameters

This output array consists of five `PID` parameter sets (using structure `stPmtsNew_Easy`), including the default parameter [0], manual/user parameter [1] and the calculated parameter (slow [2], medium [3] and aggressive [4] *(see page 220)*) by `AutoTuning`.

The structure details are explained in the given table:

| Structure parameter | Data type | Description |
|---|---|---|
| `rKp` | REAL | Value of proportional gain<br>Range: 0.0...3.4e$^{+38}$<br>Factory setting: 2.0 |
| `rTn` | REAL | Value of integral time<br>Range: 0.0...3.4e$^{+38}$ s<br>Factory setting: 200 s |
| `rTv` | REAL | Value of derivative time<br>Range: 0.0...3.4e$^{+38}$ s<br>Factory setting: 0.0 s |
| `rTd` | REAL | Value of damping time for derivative action<br>Range: CycleTime...3.4e$^{+38}$ s<br>Factory setting: 0.0 s |

**NOTE:** The necessary retrained memory you need for `AutoTuning` parameters are 80 bytes for each `TemperatureControl_Easy` in your program.

# Section 7.5
## Quick Reference Guide

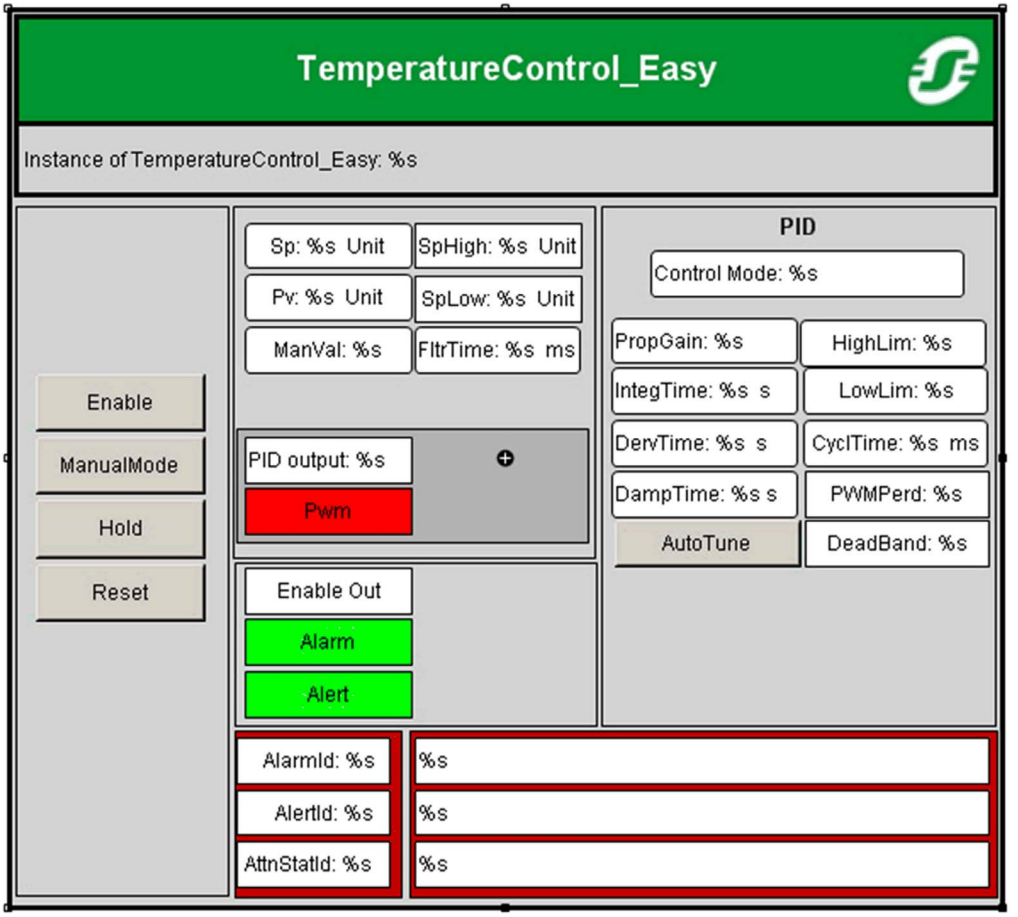
### What Is in This Section?

This section contains the following topics:

# Function Block Visualization

## Introduction

In the following figure, you see the visualization of the function block in the software:



# AutoTuning and Starting the TemperatureControl_Easy

## Introduction

The relay method of AutoTuning is implemented. When AutoTuning is enabled, it induces oscillations in the process around the set point. After the completion of three oscillations, AutoTuning calculates a set of PI parameters. Enter Tv and Td if necessary.

## Before Starting AutoTuning or Enable the TemperatureControl_Easy

| Step | Action |
|---|---|
| 1 | Scale the process value by FB_Scaling (toolbox library *(see SoMachine, Miscellaneous Functions, Toolbox Library Guide)*). Refer to Program Example *(see page 226)*. |
| 2 | Verify the plausibility of the process value (i_rPv). You have to ensure, outside of the TemperatureControl_Easy function block, that the process value is plausible.<br>**Result**: If the i_rPv value is not plausible, deactivate the TemperatureControl_Easy. |
| 3 | Verify the set point high and low limit (default 0.0...100.0 user temperature unit). |
| 4 | Enable the TemperatureControl_Easy. |

If you use the library visualization and the TemperatureControl_Easy is enabled, it shows at the process value the value of i_rPv or if filter time is greater than 0.0, it shows the filtered i_rPv value.

## Start AutoTuning

The set point is the only parameter which is required for AutoTuning. The set point has to be given to input i_rSp. Start AutoTuning through the pin i_stCtrl.xAttn, The oscillation starts and at the end of the third period, the three sets of PI parameters are calculated. Then AutoTuning is completed.

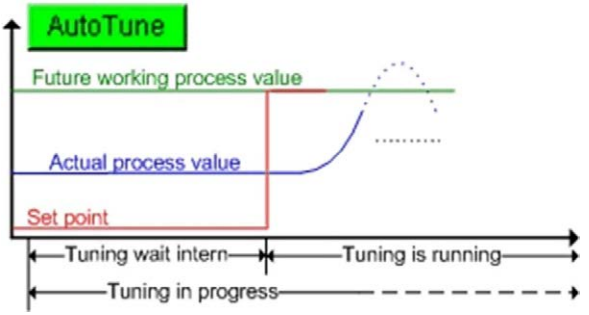
The PI parameters are calculated based on the different measured times and amplitude of the last period.

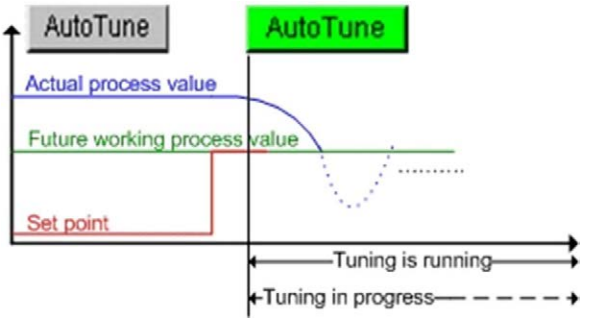
The calculated PI parameters were given to the array structure element q_astPidPara.

**NOTE:** The set point for AutoTuning should be close to the future process (working) temperature. You have to take into account that during AutoTuning the process temperature will overshoot to the given set point temperature. The value of the overshoot depends on the system used (heating energy vs. capacity) and you have to evaluate before AutoTuning.
After AutoTuning, the control mode is set to iq_iCtrlMode=3 (medium) automatically.

Status messages are provided to function block output to inform about AutoTuning. For further information, refer to section Output Pin Description *(see page 213)*.

The following two figures show when AutoTuning is running and in progress. AutoTuning will be running/active after the start command and setpoint value, independent if the setpoint value is above or below the actual process value.




## Hysteresis During AutoTuning

To avoid that one or more oscillations are skipped due to bounces of the process value, there is a hysteresis value used during AutoTuning. The hysteresis is 0.2% of the setpoint value to switch on the heating.

Example:

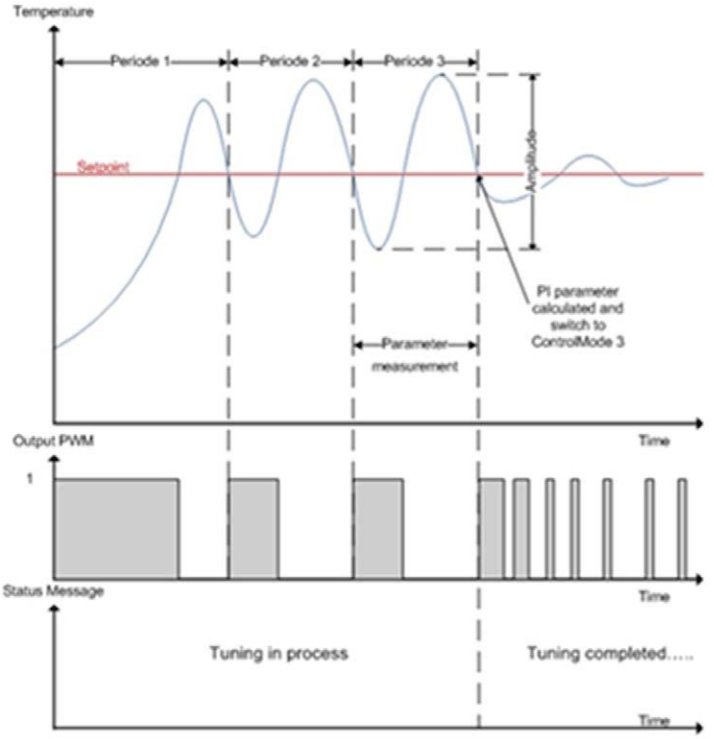
At a setpoint of 200.0°C, the hysteresis is 0.4°C. The temperature distance must be greater, like 0.5°C (for example, temperature resolution is 0.1°C).

Heating switched on at 199.5°C or less

Heating switched off at 200.0°C or greater

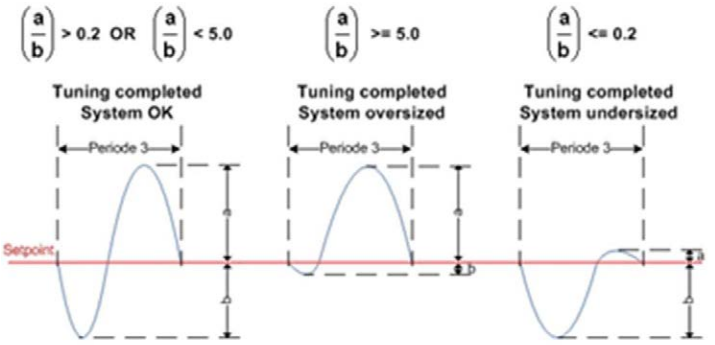
## Exemplary Sequence of AutoTuning

After the third oscillation, the three PI parameter sets are calculated and control mode 3 (medium) is set automatically.




## System Comparison After AutoTuning

In parallel to the PI parameter calculation the system is compared. This means that, from the last oscillation, the undershoot value is compared with the overshoot value. Due to the result the system is classified as system OK (balanced), oversized or undersized.



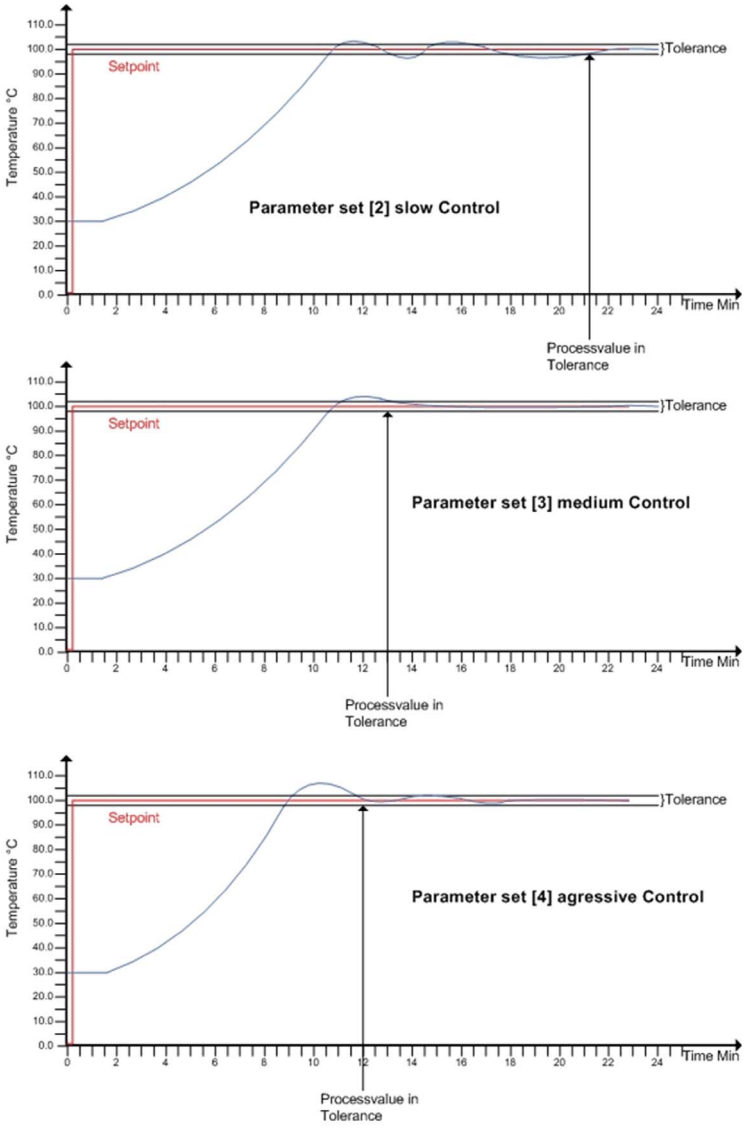

## Tuning Completed

You can select between five parameter sets by changing `iq_CtrlMode`. For further information, refer to Input/Output Pin Description *(see page 209)*.

In q_astPidPara, all PID parameter sets (rKp, rTn, rTv and rTd) are in one array:

| | | | |
|---|---|---|---|
| q_astPidPara | ARRAY [0..4] OF stN... | | |
| q_astPidPara[0] | stNewPmts | | |
| rKp | REAL | 2 | |
| rTn | REAL | 200 | Default |
| rTv | REAL | 0 | |
| rTd | REAL | 0 | |
| q_astPidPara[1] | stNewPmts | | |
| rKp | REAL | 3.75 | |
| rTn | REAL | 68 | Manual |
| rTv | REAL | 0 | |
| rTd | REAL | 0 | |
| q_astPidPara[2] | stNewPmts | | |
| rKp | REAL | 7.990552 | |
| rTn | REAL | 189.3105 | Auto tuning slow |
| rTv | REAL | 0 | |
| rTd | REAL | 0 | |
| q_astPidPara[3] | stNewPmts | | |
| rKp | REAL | 4.429833 | |
| rTn | REAL | 55.9107 | Auto tuning medium |
| rTv | REAL | 0 | |
| rTd | REAL | 0 | |
| q_astPidPara[4] | stNewPmts | | |
| rKp | REAL | 5.906444 | |
| rTn | REAL | 55.9107 | Auto tuning aggressive |
| rTv | REAL | 0 | |
| rTd | REAL | 0 | |

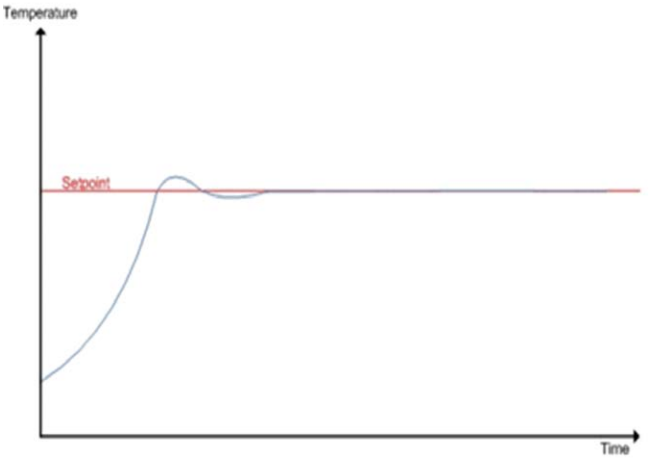
The three `AutoTuning` parameter [2], [3], [4], have following effects. This is only an example.



# Manual Tuning

## Manual Tuning

This procedure describes how to get manual PID parameter.

| Step | Action |
|---|---|
| 1 | Set iqCtrlMode = 1 (Manual Para). |
| 2 | Set rTn, rTv, and rTd to 0.0. |
| 3 | Increase rKp until the system begins to oscillate. |
| 4 | Decrease rKp until the system does not oscillate. |
| 5 | Decrease rKp by another few points. |
| 6 | Set rTn, beginning with a large value for time, for example 300.0 s (system dependent) and make sure that the system does not oscillate. |
| 7 | Decrease rTn until the system is stable and without oscillation.<br>**Result**: In the figure below a typically correct regulation is shown.<br>  |

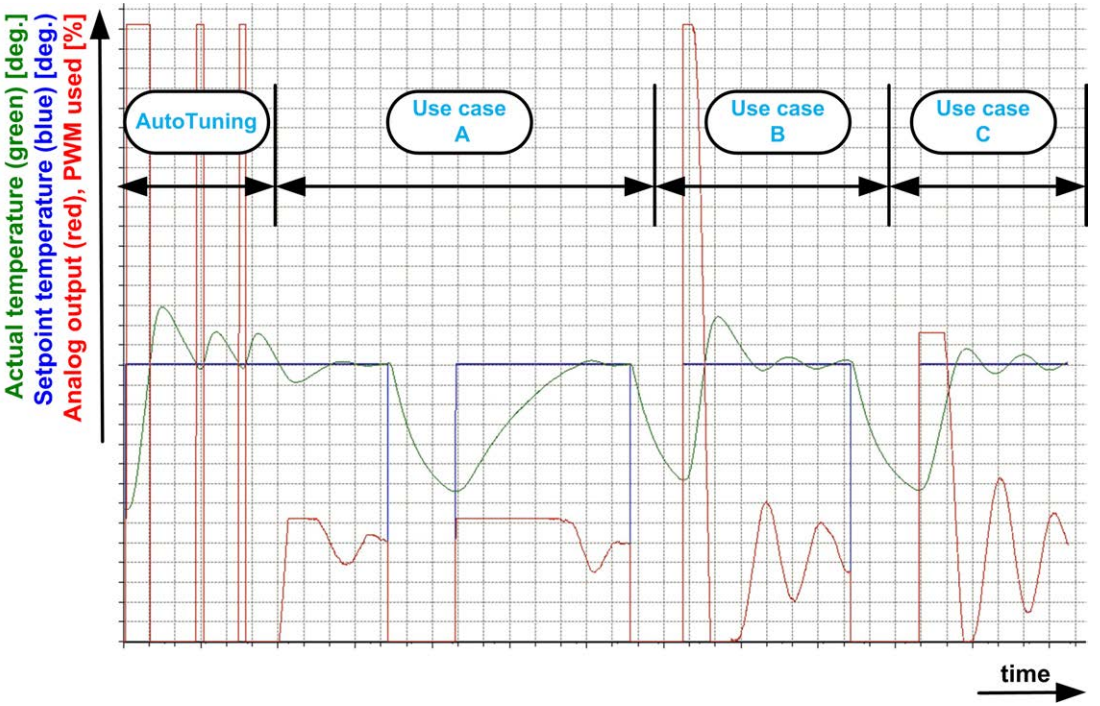
## Options If You Get Great Overshoot

### Reducing the Size of Overshoot

If after `AutoTuning` the message **Tuning completed System oversized** is displayed, the heating element is oversized.

To reduce the size of an overshoot (if needed), the energy to the heating element can be limited with the input `rHighLim` (part of `i_stTempCtrlPara`; structure `stTempCtrl_Easy`).

The input `rHighLim` limits the output `q_rAna`, which is, in turn, a basis for the `PWM` output `q_xPwm`.




**NOTE:** The limitation of the output must fit to the system. The `rHighLim` value must be above the steady state value (in this example ~20%).

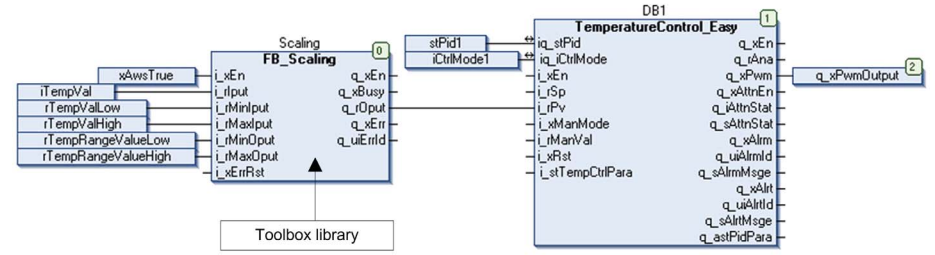
| Position | Description |
|---|---|
| AutoTuning | `AutoTuning` with `rHighLim` = 25<br>`AutoTuning` is done with a value of 100 at output `q_rAna` (red line). |
| Use case A | Limitation with `rHighLim` = 25<br>After `AutoTuning`, the output (red line) is limited and the overshoot of the temperature (green line) is small.<br>The overshoot is also small if the start temperature is low. However, to reach the setpoint it takes the greatest amount of time (~4.8 time units) in our example. |
| Use case B | No limitation with `rHighLim` = 100<br>Start with low temperature. The PID control is not limited and as a result the overshoot is the highest, but the time to reach the setpoint is the least amount of time (~0.8 time units) in our example. |
| Use case C | Limitation with `rHighLim` = 50<br>Start with low temperature. The overshoot and the time (~1.4 time units) to reach the setpoint is somewhere between the use cases A and B. |

# Program Example `TemperaturControl_Easy` Function Block

## Program Example with the `TemperaturControl_Easy` Function Block

PROGRAM Main_Programm

| | Scope | Name | Address | Data type | Initialization | Comment |
|---|---|---|---|---|---|---|
| 1 | VAR | **DB1** | | TemperatureControl_Easy | | Instance of TemeratureControl_Easy |
| 2 | VAR | **stPid1** | | stPmtsNew | | In/Out- Structure of TemeratureControl_Easy |
| 3 | VAR | **iCtrlMode1** | | INT | | In/Out-integer variable of TemeratureControl_Easy |
| 4 | VAR | **Scaling** | | FB_Scaling | | To scale the Processvalue |
| 5 | VAR | **xAwsTrue** | | BOOL | TRUE | Always True |
| 6 | VAR | **iTempVal** | %IW2 | INT | | Aktual temperature value of the Analog/Digital input modul |
| 7 | VAR | **rTempValHigh** | | REAL | 4095.0 | Analog/Digital resolution of the input modu (Here 0 to 4095) |
| 8 | VAR | **rTempValLow** | | REAL | 0.0 | Analog/Digital resolution of the input modu (Here 0 to 4095) |
| 9 | VAR | **rTempRangeValueHigh** | | REAL | 600.0 | Temperature Range of PT100 (Here -200.0 to +600.0°C) |
| 10 | VAR | **rTempRangeValueLow** | | REAL | -200.0 | Temperature Range of PT100 (Here -200.0 to +600.0°C) |
| 11 | VAR | **q_xPwmOutput** | %QX2.0 | BOOL | | Digital Output PWM |




In addition, you can use the trace function of SoMachine (an example is shown at Options If You Get Great Overshoot *(see page 224)*), with the important values, like actual temperature value `i_rPv`, setpoint `i_rSp` and the control output `q_rAna`.

# Troubleshooting

## Troubleshooting

This table shows some general issues and their solutions:

| Issue | Cause | Possible solution |
|---|---|---|
| `q_uiAlrmId` = 1 | The controller is not supported. | Replace the controller. |
| `q_uiAlrmId` = 20 | Cycle time out of the valid range | The function block verifies in which interval it is called. Use a valid cycle time (0.0...10000 ms). |
| `q_uiAlrmId` = 115 | Invalid dead band value | Enter a valid value (0.0...100.0 unit). |
| `q_uiAlrmId` = 200 | Invalid `PID` parameter (the parameters are 0.0) | Reset for default values. |
| `q_uiAlrmId` = 205 | General alarm from `FB_PID` | Restart the (`i_xEn`) `TemperatureControl_Easy`. |
| `q_uiAlrmId` = 206 | `PID` low limit is equal to high limit | Enter a valid value (high limit have to be greater than low limit). |
| `q_uiAlrmId` = 300 | Invalid value or cycle time is greater than PWM time period | Enter a valid value (5.0 ... 60.0 s). |
| `q_uiAlrmId` = 303 | Invalid output range. | Enter a valid value (0.0 ... 100.0%). |
| `q_uiAlrmId` = 304 | Output low limit greater than high limit. | Enter a valid value. |
| `q_uiAlrmId` = 305 | Filter time is active (>0.0) and less than cycle time or filter time is less than 0.0 ms. | Enter a valid value. |
| `q_uiAlrmId` = 320 | Set point high limit value less than low set point. | Enter a valid value. |
| `q_uiAlrtId` = 1 | The set point is greater than the set point high limit. | Enter a valid value. |
| `q_uiAlrtId` = 2 | The set point is less than the set point low limit. | Enter a valid value. |
| `q_uiAlrtId` = 12 | The value is less than 0 or greater than 4. | Enter a valid value. |
| `q_uiAlrtId` = 15 | Manual value is outside range | Enter a valid value (`rLowLim ... rHighLim`). |
| `q_iAttnStat` = 3 | `AutoTuning` was finished and the system comparison *(see page 219)* detected an inhomogeneous system. The heater could be oversized for the working temperature. | If possible, replace the heater. Or, refer to Options if you get great overshoot *(see page 224)*. Or, enter manual PID parameters *(see page 223)* control mode 1 |

| Issue | Cause | Possible solution |
|---|---|---|
| q_iAttnStat = 4 | AutoTuning was finished and the system comparison *(see page 219)* detected an inhomogeneous system. The heater could be undersized. | Verify if the regulation with the calculated PID parameter is sufficient to the customer needs, or enter manual parameters *(see page 223)* and use control mode 1. |
| After AutoTuning the calculated PID parameter sets are not plausible (for example, rKP=300.0, rTn=0.4 s). | Noisy signal of the actual process value. | Set the filter (for example, 1000 ms) and start AutoTuning again. |
| Noisy process value | Issues between sensor and controller input | Verify the installation.<br>Verify the sensor.<br>Set the digital filter (for example, 1000 ms). |
| You cannot enter manual PID parameter | An invalid control mode is selected | Enter the control mode 1 |
| Several AutoTuning loops on the same system and set point provides not the same calculated PID parameter. | You have started AutoTuning in different ways (for example, first from a low temperature, second from a high temperature). | Verify if the environment temperature and the system are still the same.<br>**NOTE:** Parameters could have differences up to 15%. |
| AutoTuning is running for a long time (>1 h). | The system is slow or undersized. | Monitor with a trace.<br>Stop AutoTuning.<br>Set manual PID parameters. |
| TemperatureControl_Easy regulation do not fulfill the requirement | Unsuitable setting or disturbance from around the heating system. | Verify the process value if it is noisy.<br>Test the different control modes (2..4).<br>Set manual (control mode 1) or default parameters (control mode 0).<br>Verify that the PWM time period (refer to rPwmTimePrd *(see page 206)*) fit to system. |

# Part V
## LateralPositionControl

# Chapter 8
# LateralPositionControl: Controlling Correct Lateral Film Position with Two Digital Sensors

## Overview

This chapter describes the `LateralPositionControl` function block.

## What Is in This Chapter?

This chapter contains the following sections:

# Section 8.1
# Functional and Machine Overview

## Functional Overview

### Why Use the `LateralPositionControl` Function Block ?

When the film reel is unwound, there is a possibility of a lateral shift in position due to uneven thickness of the film or inconsistent speed and tension variation.

The function block checks the lateral shift and corrects the film position.

Lateral film position correction is required to:
- Avoid a film break-up,
- Have a constant product quality,
- Help increase the cutting accuracy by keeping the print mark detection under the photo cell.

### Solution with the `LateralPositionControl` Function Block

The function block reads film sensor inputs from the field. It checks the sensors' status and generates control outputs (digital or analog) to command the movement of film to its correct position.

### Design & Realization Constraints and Assumptions

#### Constraints

1. The function block does not consider the inoperable state of sensors; it is assumed that sensors are always in healthy state.

2. The function block allows any change in input parameters, as well as the operating mode on the fly. However any errors caused due to this will not be handled by the function block.

#### Assumptions

1. Sensor inputs are active high when the sensor is detecting the film directly above/below it..

2. A positive value of `q_rOputAna` output will move the film leftwards and likewise a negative value of `q_rOputAna` will move the film rightwards.

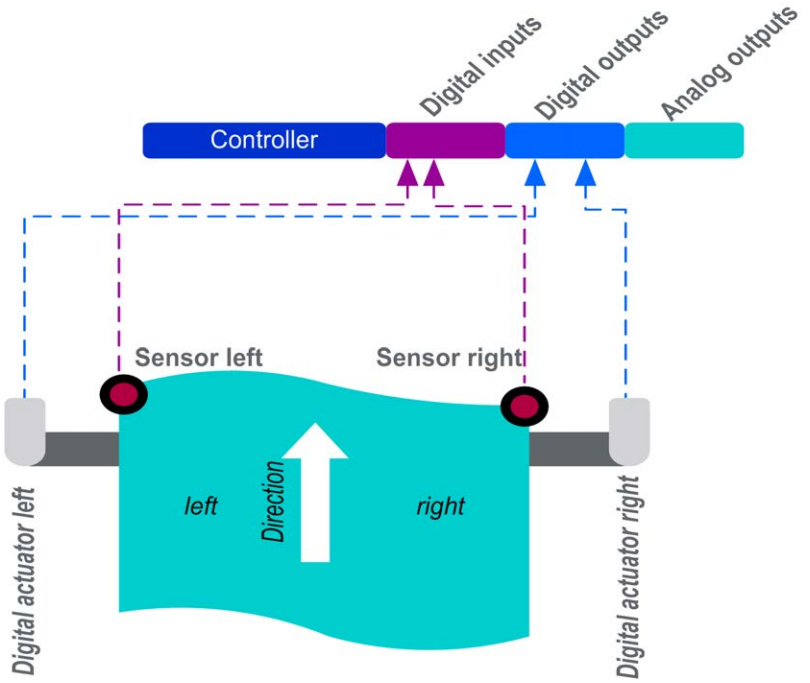
### Application

In packaging machines `LateralPositionControl` function block verifies the correct lateral position of the film.

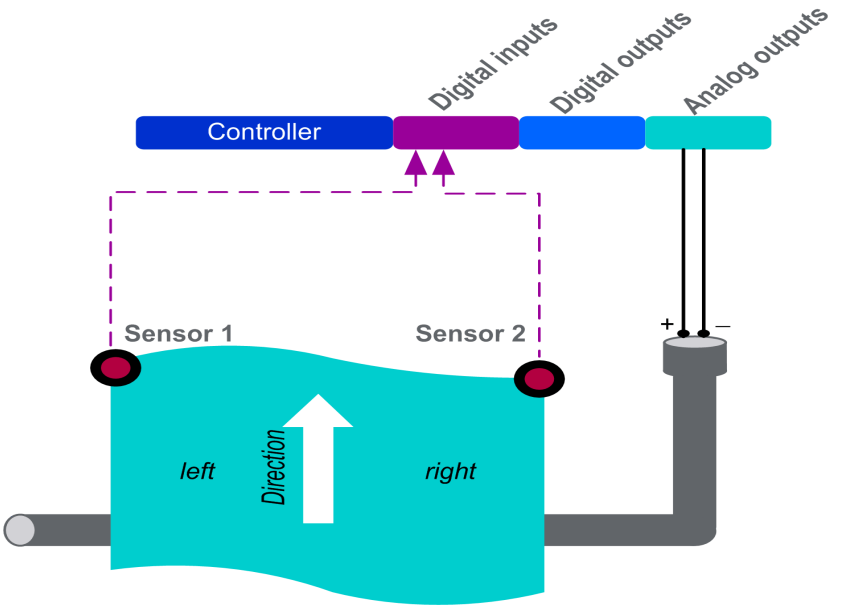
The function block is applicable to the following types of packaging machines:
- Vertical bagging machine
- Horizontal bagging machine

## Functional View

This figure shows the `LateralPositionControl` function block generating digital outputs:



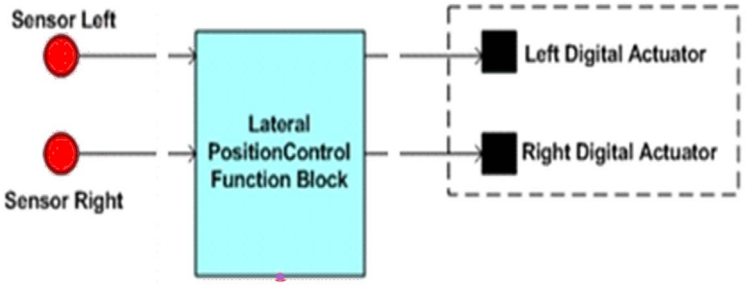
This figure shows the `LateralPositionControl` function block generating analog outputs:



# Section 8.2
## Architecture

## Software Architecture

### DataFlow Overview



# Section 8.3
## Function Block Description

### What Is in This Section?

This section contains the following topics:

# LateralPositionControl Function Block

## Pin Diagram


## Function Block Description

`LateralPositionControl` system consists of film position sensors and digital or analog actuators (variable speed drives) to control movement of the film reel.

The function block has two operating modes:

- Automatic
- Manual

The output of the function block can be selected to:

- Digital or
- Analog

**NOTE:** The operating modes as well as the output modes can be changed on-the-fly.

## Manual Inputs

In Manual mode with no detected error, manual input `i_xManLeftMove` is used to control the left movement of the film. Similarly, manual input `i_xManRighMove` is used to control right movement of the film.

**NOTE:** In Manual mode if both `i_xManRighMove` and `i_xManLeftMove` are TRUE, then `i_xManRighMove` is given priority. In Automatic mode, the manual inputs are ignored and only sensor inputs are taken into account.

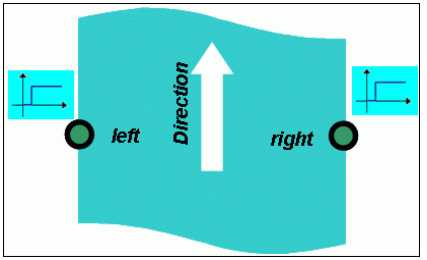
## Lateral Position of Film

Position sensors feed in the current position of the film reel. Based upon this feedback, the function block activates the digital actuators via digital outputs or the analog actuator (example: Altivar variable speed drive) via analog output through CANopen to correct the lateral position of film.

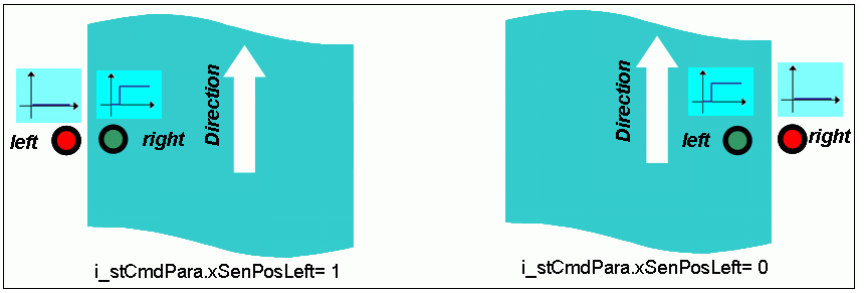
## Position Configuration

The function block support 3 sensor position configurations.

This figure shows symmetrically placed sensors:




This figure shows asymmetrically places sensors:

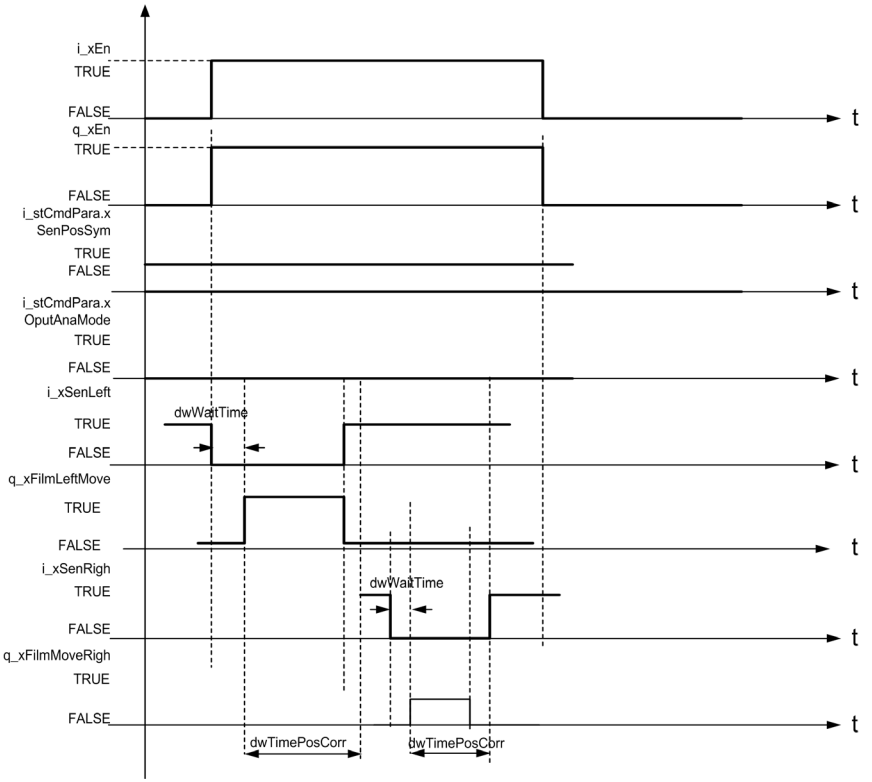


# Automatic Mode

## De-bouncing

In Automatic mode of operation, to avoid the sensor signals bouncing due to film vibration, the signals are continuously monitored for inactivity for a time greater than `i_stCmdPara.dwTimeWait` before the position correction outputs (`q_xFilmLeftMove`, `q_xFilmRighMove` or `q_rOputAna`) are activated.

This diagram shows the timing diagram for digital output:



### Digital Output

While operating in Automatic mode with digital outputs, film position has to be corrected within `i_stCmdPara.dwTimePosCorr` time. If correction is not achieved, the function block displays timeout alarm. The value of `i_stCmdPara.dwPosCorrTime` must be set greater than zero in Automatic mode.

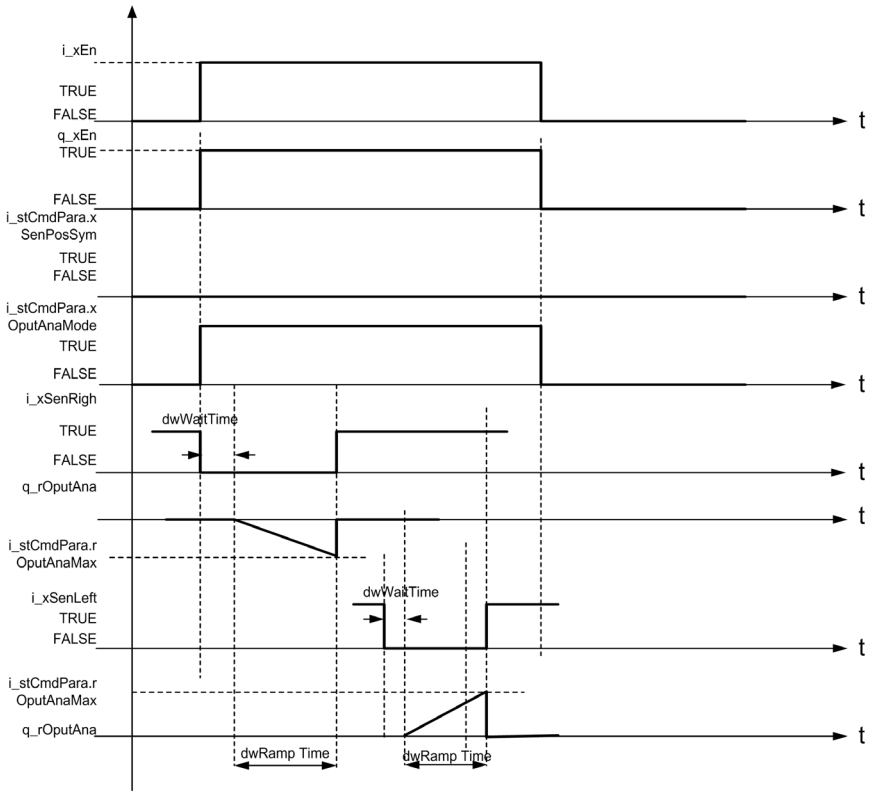
In Manual mode, the `i_stCmdPara.dwPosCorrTime` value can be set to zero to disable the functionality.

**NOTE:** The value of the `i_stCmdPara.dwRampTime` must be set greater than zero in Automatic mode. If it is not configured greater than zero, then function block detects an error.

### Analog Output

The analog output (`q_rOputAna`) from the function block follows a ramp with `i_stCmdPara.dwRampTime` as the ramp time and `i_stCmdPara.rOputAnaMax` as maximum attainable value. To command left movement, `q_rOputAna` is positive and to command right movement, it is negative.

This timing diagram is for analog output:



# Managing Detected Errors

## Description

Errors detected inside the function block are displayed at the output pin `q_xAlrm`.

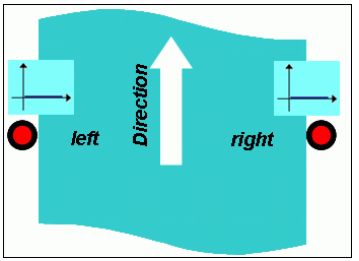
Additionally, the notification ID will be set at the output pin `q_uiAlrmId` to indicate the type of error detected.

Detected errors are acknowledged with the falling edge of `i_xEn` input.

An unacknowledged detected error resets all outputs.

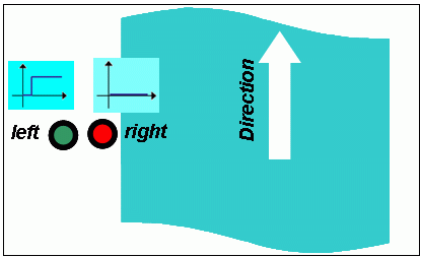
## Symmetric Sensor External Error Detected

Both sensors are inactive as shown in the figure below.

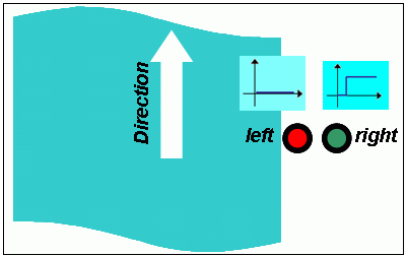



## Asymmetric Sensor External Error Detected

If both sensors are inactive, so there will be left movement of the film. When the film is moving leftwards and the left sensor signal becomes active without right sensor becoming active first.




If both sensors are inactive, so there will be right movement of the film. When the film is moving rightwards and the right sensor signal becomes active without left sensor becoming active first.




**NOTE:** The `LateralPositionControl` function block allows changes in control parameters while it is enabled. But it is recommended not to explicitly do so on-the-fly this may result in incorrect operation.

| *NOTICE* |
| --- |
| **MISFEEDS CAUSING MATERIAL DAMAGE**<br>You must ensure that the sensor placement configuration matches the mechanical system.<br>**Failure to follow these instructions can result in equipment damage.** |

# Section 8.4
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Enables function block and parameters are validated.<br>FALSE: Disables function block and all outputs are set to zero(factory setting) |
| i_xAutoMode | BOOL | Selects between Automatic and Manual modes.<br>TRUE: Corrects automatically the lateral position of the film depending upon the feedback from the 2 sensors.<br>FALSE: The adjustment of the film position is achieved through input pins i_xManLeftMove and i_xManRighMove.(factory setting) |
| i_xSenLeft | BOOL<br>(*)Auto mode | Left sensor Input<br>FALSE: No input<br>TRUE: Active high input from sensor |
| i_xSenRigh | BOOL<br>(*)Auto mode | Right sensor input<br>TRUE: Active high input from sensor<br>FALSE: No input |
| i_xManLeftMove | BOOL<br>(*)Manual mode | Manual move command to move the film left without checking the sensor feedback.<br>TRUE: Move left command<br>FALSE: No command |
| i_xManRighMove | BOOL<br>(*)Manual mode | Manual move command to move the film right without checking the sensor feedback.<br>TRUE: Move right command<br>FALSE: No command |
| i_stCmdPara | CmdParameterFilmPos | Includes various control parameters for sensor position, output selction and position correction. Refer to the structured input CmdParameterFilmPos *(see page 246)*. |
| (*)Optional according to the application mode | | |

### Input: CmdParameterFilmPos

This table describes the CmdParameterFilmPos structured input:

| Structure Element | Data Type | Description |
|---|---|---|
| xOputAnaMode | BOOL | Selects between analog and digital output.<br>TRUE: The function block corrects the film position via an analog output q_rOputAna.<br>FALSE: The function block corrects the film position via digital outputs q_xFilmLeftMove and q_xFilmRighMove. |
| xSenPosSym | BOOL<br>(*)Auto mode | Selects between the two types of sensor position configuration.<br>TRUE: Sensors placed at both sides of the film.<br>FALSE: Both sensors placed at the same side of the film. |
| xSenPosLeft | BOOL<br>(*)Auto mode | If xSenPosSym = FALSE, this parameter will select which side of the film sensors are placed.<br>TRUE: Sensors located on left.<br>FALSE: Sensors located on right. |
| rOputAnaMax | REAL<br>(*)Auto mode | Maximum generated value of the analog output at the function block output q_rOputAna<br>Range: 0.0...1e$^{32}$. |
| dwTimePosCorr | DWORD<br>(*)Auto mode | Time to correct the position of film.<br>Range: 0...4294967295 ms.<br>0: Functionality is disabled. Not allowed in Auto mode with digital outputs. |
| dwRampTime | DWORD<br>(*)Auto mode | Represents the time in which q_rOputAna should achieve the maximum analog output.<br>Range: 0...4294967295 ms.<br>0: Functionality is disabled. Not allowed in Auto mode with analog output. |
| dwTimeWait | DWORD<br>(*)Auto mode | Represents the time in which the sensor signal is inactive to start the position correction.<br>Range: 0...4294967295 ms. |
| (*)Optional according to the application mode | | |

# Output Pin Description

## Output Pin Description

### Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xEn | BOOL | TRUE: Function block enabled if input i_xEn is TRUE, then output q_xEn is TRUE. |
| q_xFilmLeftMove | BOOL | Output to the left actuator<br>TRUE: ON<br>FALSE: OFF |
| q_xFilmRighMove | BOOL | Output to the right actuator<br>TRUE: ON<br>FALSE: OFF |
| q_rOputAna | REAL | Value of analog output.<br>Range: - i_stCmdPara.rOputAnaMax...<br>+ i_stCmdPara.rOputAnaMax |
| q_xTmot | BOOL | This pin will be set when operating in Automatic mode with:<br>• digital outputs and film position is not corrected within the specified time, i_stCmdPara.dwPosCorrTime<br>• analog output and the film position is not corrected within the specified ramp time i_stCmdPara.dwRampTime.<br>TRUE: Error detected<br>FALSE: Ok |
| q_xAlrm | BOOL | Indicates that a detected alarm has occurred inside the block<br>TRUE: Alarm detected<br>FALSE: Ok |
| q_uiAlrmId | UINT | Displays the notification number when the detected alarm output is set, see table q_uiAlrmId and q_sAlrmMsge. |
| q_sAlrmMsge | STRING(40) | Displays the function block notification message |

### Notifications

| Detected Alarm ID | Notification | Description |
|---|---|---|
| 0 | Okay | Function block is in healthy state |
| 1 | Internal alarm detected | Function block in unknown state |
| 40 | External alarm detected | Incorrect state or positioning of sensors |
| 70 | Time-out alarm detected | Indicates a timeout when film position is not corrected within the specified time in Automatic mode. |
| 100 | Invalid parameter | For example: input parameter values to the function block are out of range. |

# Section 8.5
## Quick Reference Guide

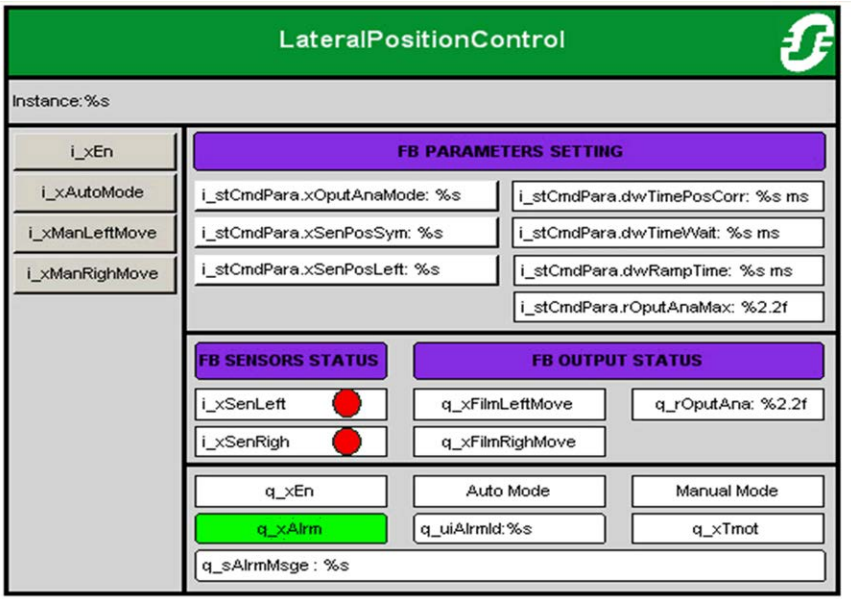
### What Is in This Section?

This section contains the following topics:

# Function Block Visualization

## Visualization

This figure shows the visualization of the `LateralPositionControl` function block:



# Quick Commissioning Procedure

## Commissioning Procedure Overview

This table explains the procedure on how to configure the `LateralPositionControl` function block:

| Step | Action |
|---|---|
| 1 | Add the Packaging.library in the library manager. |
| 2 | Load the `LateralPositionControl` function block from Packaging library included in your application. |
| 3 | Determine the type of sensor position configuration. |
| 4 | Connect the outputs to the corresponding output fields. |
| 5 | Perform a sensor input test. |
| 6 | Start the system. |

## Commissioning Procedure in Manual Mode

### Running in Manual Mode

To run the `LateralPositionControl` function block in Manual mode, configure the following parameters:

- `i_xEn` = TRUE,
- `i_xAutoMode` = FALSE.

### Performing a Right Move

To perform a right move operation make

`i_xManRighMove` = TRUE.

The lateral position of film starts moving to the right side while `i_xManRighMove` is TRUE.

When `i_xManRighMove` = FALSE, the movement of lateral position of the film is stopped.

### Performing a Left Move

To perform a left move operation make

`i_xManLeftMove` = TRUE.

The lateral position of film starts moving to the left side while `i_xManLeftMove` is TRUE.

When `i_xManLeftMove` = FALSE, the movement of lateral position of the film is stopped.

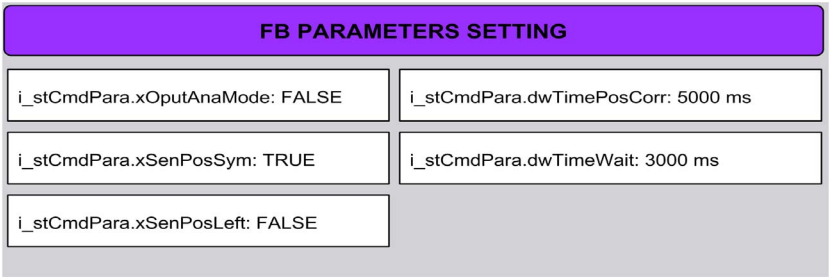
## Commissioning Procedure in Auto Mode With Symmetric Sensors

### Symmetric sensors configuration with digital output

To execute the function block in symmetric sensors configuration, configure the following parameters:

- `i_stCmdPara.xSenPosSym` = TRUE,
- `i_stCmdPara.xOputAnaMode` = FALSE,
- `i_stCmdPara.dwTimePosCorr` = 5000ms,
- `i_stCmdPara.dwTimeWait` = 3000ms.

This figure shows the visualization for the above configuration:




Then enable the function block.

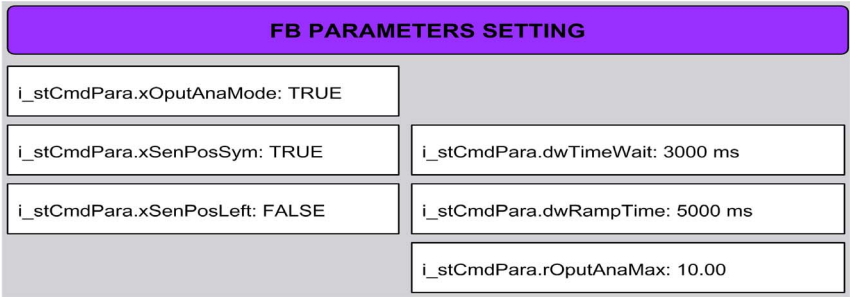
In this configuration, both `i_xSenRigh` and `i_xSenLeft` should be always TRUE . If any of the sensor `i_xSenRigh` or `i_xSenLeft` is FALSE, then after `i_stCmdPara.dwTimeWait` the lateral position of the film is moved to the corresponding side by enabling `q_xFilmLeftMove` or `q_xFilmRighMove`.

### Symmetric sensors configuration with analog output

To execute the function block in symmetric sensors configuration, configure the following parameters:

- `i_stCmdPara.xSenPosSym` = TRUE,
- `i_stCmdPara.xOputAnaMode` = TRUE,
- `i_stCmdPara.dwTimeWait` = 3000ms,
- `i_stCmdPara.dwRampTime` = 5000ms,
- `i_stCmdPara.rOputAnaMax` = 10.0.

This figure shows the visualization for the configuration:




Then enable the function block.

In this configuration both `i_xSenRigh` and `i_xSenLeft` should be always TRUE. If any of the sensor `i_xSenRigh` or `i_xSenLeft` is FALSE, then after `i_stCmdPara.dwTimeWait` the lateral position of the film is moved to the corresponding side by providing analog output through `q_rOputAna`.

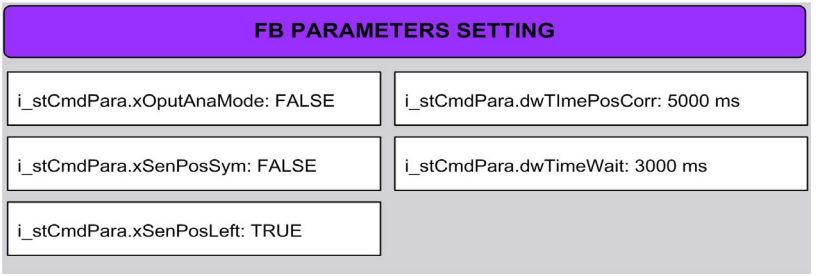
## Commissioning Procedure in Auto Mode and Left Placed Sensors

### Asymmetric left placed sensors configuration with digital output

To execute the function block in asymmetric left sensors configuration, configure the following parameters:

- `i_stCmdPara.xSenPosSym` = FALSE,
- `i_stCmdPara.xSenPosLeft` = TRUE,
- `i_stCmdPara.dwTimePosCorr` = 5000ms,
- `i_stCmdPara.dwTimeWait` = 3000ms.

This figure shows the visualization for the configuration:




Then enable the function block.

In this configuration `i_xSenRigh` should be TRUE and `i_xSenLeft` should be FALSE.

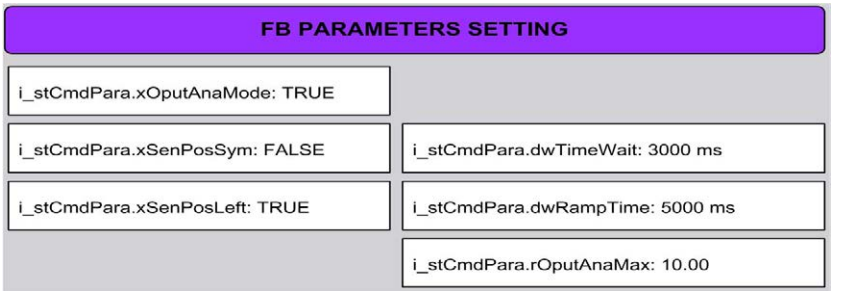
If `i_xSenRigh` is FALSE, then after `i_stCmdPara.dwTimeWait` function block will enable `q_xFilmLeftMove`. If both the sensors are TRUE, then after `i_stCmdPara.dwTimeWait` function block will enable `q_xFilmRighMove`.

### Asymmetric left placed sensors configuration with analog output

To execute the function block in asymmetric left placed sensors configuration, configure the following parameters:

- `i_stCmdPara.xSenPosSym` = FALSE,
- `i_stCmdPara.xSenPosLeft` = TRUE,
- `i_stCmdPara.xOputAnaMode` = TRUE,
- `i_stCmdPara.dwTimeWait` = 3000ms.
- `i_stCmdPara.dwRampTime` = 5000ms.
- `i_stCmdPara.rOputAnaMax` = 10.0.

This figure shows the visualization for the configuration:




Then enable the function block.

In this configuration `i_xSenRigh` should be TRUE and `i_xSenLeft` should be FALSE.

If `i_xSenRigh` is FALSE, then after `i_stCmdPara.dwTimeWait` function block will provide analog output through `q_rOputAna`. In this case, the value of `q_rOputAna` is positive to move the lateral position of the film in left side.

If both the sensors are TRUE, then function block will provide analog output through `q_rOputAna`.

In this case, the value of `q_rOputAna` is negative to move the lateral position of the film in right side.

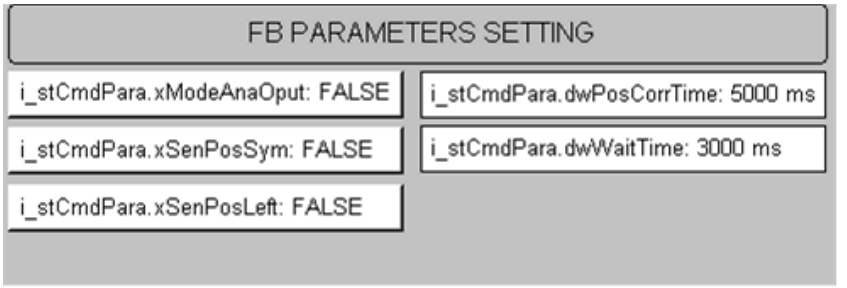
## Commissioning Procedure in Auto Mode and Right Placed Sensors

### Asymmetric right placed sensors configuration with digital output

To execute the function block in asymmetric right placed sensors configuration, configure the following parameters:

- `i_stCmdPara.xSenPosSym` = FALSE,
- `i_stCmdPara.xSenPosLeft` = FALSE,
- `i_stCmdPara.xOputAnaMode` = FALSE,
- `i_stCmdPara.dwTimePosCorr` = 5000ms,
- `i_stCmdPara.dwTimeWait` = 3000ms.

This figure shows the visualization for the configuration:




Then enable the function block.

In this configuration `i_xSenRigh` should be TRUE and `i_xSenLeft` should be FALSE.

If `i_xSenRigh` is FALSE, then after `i_stCmdPara.dwTimeWait` function block will provide analog output through `q_rOputAna`. In this case, the value of `q_rOputAna` is positive to move the lateral position of the film in left side.

If both the sensors are TRUE, then function block will provide analog output through `q_rOputAna`.

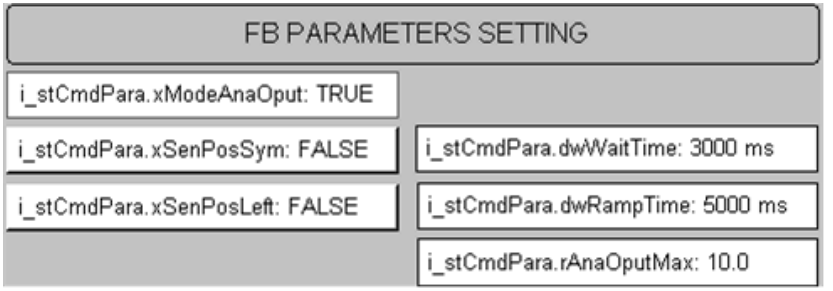
In this case, the value of `q_rOputAna` is negative to move the lateral position of the film in right side.

### Asymmetric right placed sensors configuration with analog output

To execute the function block in asymmetric right placed sensors configuration, configure the following parameters:

- `i_stCmdPara.xSenPosSym` = FALSE,
- `i_stCmdPara.xSenPosLeft` = FALSE,
- `i_stCmdPara.xOputAnaMode` = TRUE,
- `i_stCmdPara.dwTimeWait` = 3000ms,
- `i_stCmdPara.dwRampTime` = 5000ms,
- `i_stCmdPara.rOputAnaMax` = 10.0.

This figure shows the visualization for the configuration:




Then enable the function block.

In this configuration `i_xSenRigh` should be FALSE and `i_xSenLeft` should be TRUE.

If `i_xSenLeft` is FALSE, then after `i_stCmdPara.dwTimeWait` function block will provide analog output through `q_rOputAna`. In this case, the value of `q_rOputAna` is negative to move the lateral position of the film in right side.

If both the sensors are TRUE, then function block will provide analog output through `q_rOputAna`.

In this case, the value of `q_rOputAna` is positive to move the lateral position of the film in left side.

# Troubleshooting

## Troubleshooting

This table shows some general issues and their solutions:

| Issue | Cause | Solution |
|---|---|---|
| No sensor feedback | Inoperative sensors | Inoperative sensors will not be detected by the function block. The user must verify that the sensors are always in good condition. |
| Unpredictable behavior leading to improper position correction | Parameter change on-the-fly | You should avoid in this case to explicitly change the control parameters of function block without disabling the function block. To do so, disable the function block, feed new parameter values and then re-enable the function block. |
| `q_uiAlrmId` = 1 | Internal alarm detected | Restart the function block. |
| `q_uiAlrmId` = 70 | Alarm detected due to `q_xTmot` | 1. Change the input parameters (`i_stCmdPara.dwPosCorrTime` for digital output mode or `i_stCmdPara.dwRampTime` for analog output mode) to suitable values.<br>2. Restart the function block. |
| `q_uiAlrmId` = 100 | Parameters such as `i_stCmdPara.dwPosCorrTime`, `i_stCmdPara.dwRampTime`, `i_stCmdPara.dwWaitTime`, `i_stCmdPara.rAnaOputMax` is configured less than zero or equal to zero | Configure the value of parameters greater than zero. |

# Part VI
## RotaryKnife

# Chapter 9
# RotaryKnife_Motion: Synchronization of a Linear Axis and a Rotary Axis to Perform On The Fly Operations

## What Is in This Chapter?

This chapter contains the following sections:

# Section 9.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Why Use the `RotaryKnife_Motion` Function Block?

The `RotaryKnife_Motion` function block controls a machine that performs operations on a moving part.

Typical operations can include:

- Cutting
- Sealing
- Marking

## Solution with the `RotaryKnife_Motion` Function Block?

The `RotaryKnife_Motion` function is required for moving the operational axis to synchronize it with the forward motion of the part.

This introduces the concept of master and slave axis.

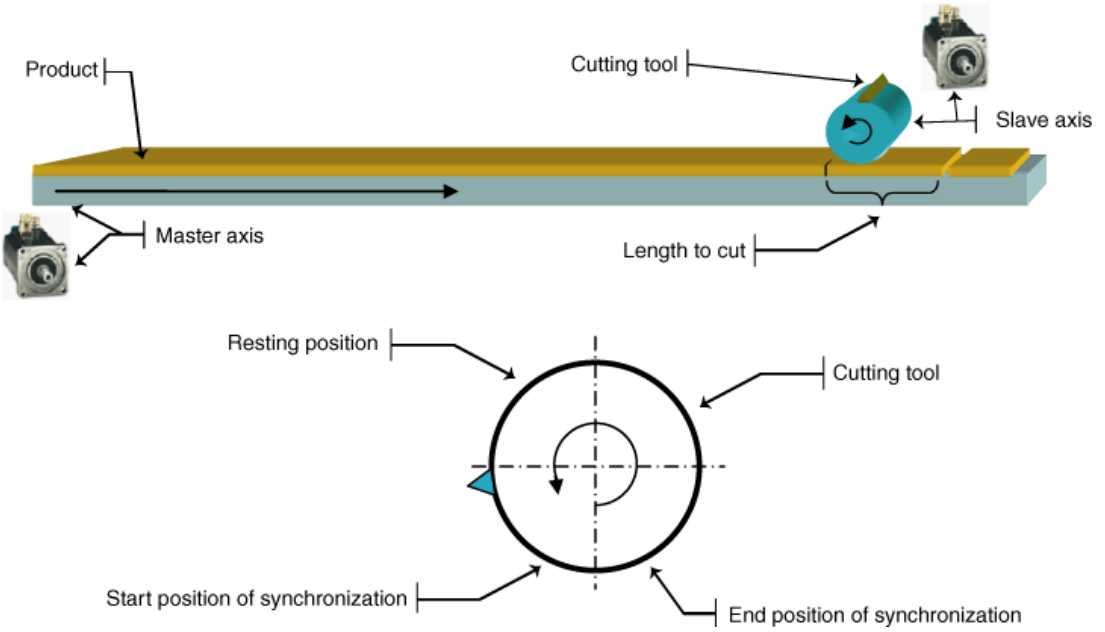
- Master: moves the part forward
- Slave: performs the operation

**NOTE:** Although the Rotary Knife function can be used for many different applications, the examples shown in this document will refer to a typical cutting application.

# Machine Overview

## Machine View

The figure below gives an example of a Rotary Knife application. Master axis is a linear axis type whereas the axis slave is a rotary axis.




## Rotary Knife

With the application function blocks it is possible to handle Multi Blade Rotary Knife. The condition is to have equidistant knifes.

In this case the parameter `lrPeri` is defined as the distance between the blades.

## Note

Please also refer to:

- Master Axis *(see page 266)*
- Slave Axis *(see page 267)*
- Cutting Tool *(see page 267)*
- Capture Position of the Product *(see page 267)*

# Section 9.2
## Architecture

### What Is in This Section?

This section contains the following topics:

# Hardware Architecture

## Master Axis

**NOTE:** This architecture is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

The master axis moves a product forward continuously. It is an infinite axis.

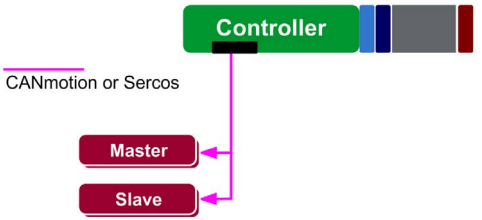
The master axis can be of the following type:
- servo drive (Lexium) on CANmotion or Sercos
- any type of axis (Altivar or Lexium) coupled with a master encoder
- virtual axis

Example: Tube cutting

The forward motion of the tube is managed by the rolling mill and thus by the device which controls it. Therefore the movement of this axis is not controlled by the RotaryKnife/FlyingShear functions.

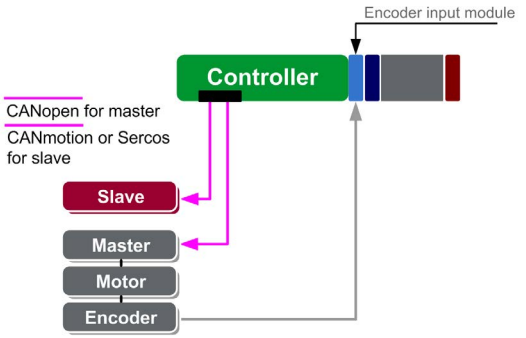
The master and the slave axis can both be on CANmotion or Sercos.

Master and slave on CANmotion or Sercos




If the master axis is not a servo drive on the same CANmotion or Sercos as the slave, then the position of the master is determined by an encoder signal connected to the input of the Motion Controller:

Example of a master axis controlled with an Altivar via CANopen communication protocol



## Slave Axis

The slave axis has to be a servo drive on CANmotion/Sercos or stepper drive on CANmotion.

According to the length to cut (input `i_lrLentoCut`), the slave axis synchronizes itself with the master, remains synchronized during the cut (or another operation) and finally returns to its resting position (parameter `lrPosRest`) and/or continues with a new cut.

Synchronization may correspond to:

- A positioning of the slave in relation to the master according to the cutting point. This position is defined as position 0 of the slave.
- A speed of the slave proportional to the speed of the master.

## Cutting Tool

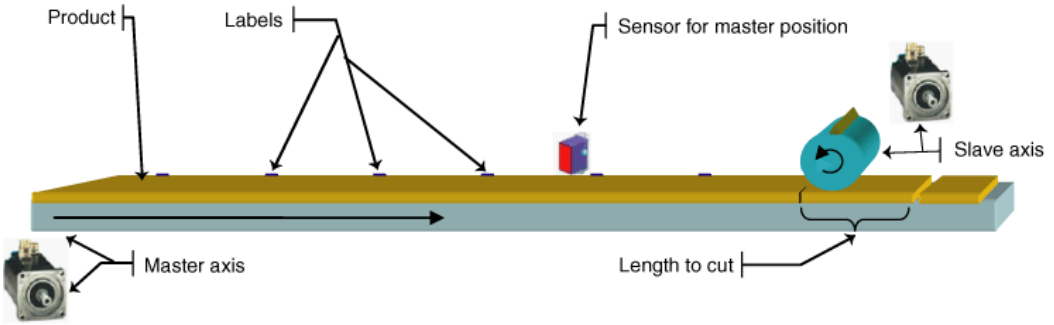
If needed, the knife starts its movement after the beginning of the synchronization phase (of the master and the slave). It must have completed its movement before the end of synchronization phase and return to its original position before a new cut is begun.

The movement of the cutting tool is managed by the application, rather than the `RotaryKnife_Motion`/`FlyingShear_Motion` application function blocks. However, the functions provide a "Beginning of Synchronization" signal (`q_xInSync`) which can be used by the application to control the start of the cutting tool.

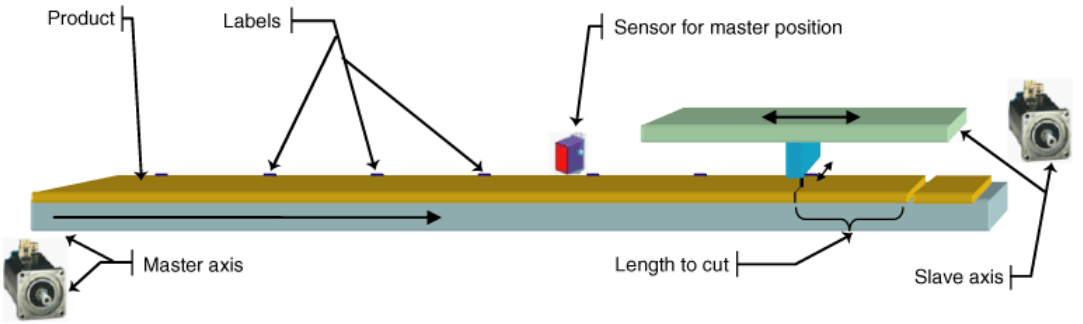
## Capture Position of the Product

In certain applications the product is not continuous (e.g. individual parts to be labeled), or may not be rigid which presents a possibility of stretching (e.g. plastic film). In these cases, a sensor is placed on the master axis to capture the position of labels on the product.

Rotary Knife: Principle of operation sensor wiring



Flying Shear: Principle of operation sensor wiring




## Sensor Wiring

| If... | Then... |
|---|---|
| the master is the encoder input | the sensor may be wired to a Touch Probe input (TP1, TP2…) of the Motion Controller. The TP inputs have an acquisition time of 30µs. |
| the master is a servo drive on CANmotion or Sercos bus | the sensor may be wired to one of the capture inputs of the drive. (See configuration of capture inputs in documentation of the drive.) |

The standard inputs of the Motion Controller can also be used. In this case however, the acquisition time is related to the cycle time of the Motion Controller.

# Software Architecture

## Requirements for Configuration

**NOTE:** This architecture is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

Configuration of the axes with the Motion Controller.

Master:

- Even if the master axis is considered to be a linear axis (e.g. a conveyor) being `infinite`, it must nonetheless be configured as a modulo axis.

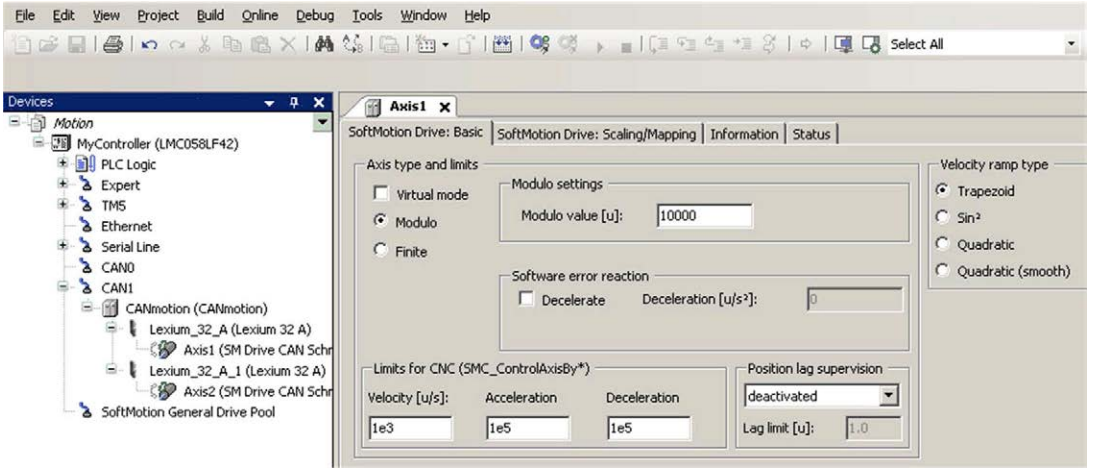
| *NOTICE* |
| --- |
| **UNSATISFACTORY EQUIPMENT OPERATION**<br>Set the `modulo` value higher than the maximum possible length to cut plus Touch Probe distance to the synchronization point (greater than the sum of `max i_lrLentoCut` and `lrTpDistToSypt`).<br>**Failure to follow these instructions can result in equipment damage.** |

Slave:

- For Rotary Knife:
  The slave axis has to be configured as a rotary axis.
- For Flying Shear:
  The slave axis has to be configured as a linear axis.

| *NOTICE* |
| --- |
| **UNSATISFACTORY EQUIPMENT OPERATION**<br>Set the `modulo` value of the slave greater than or equal to than 2 times the distance between the knife blades (perimeter) (greater than or equal to 2*`lrPeri`).<br>**Failure to follow these instructions can result in equipment damage.** |

The axis type and the modulo is configured in the device configuration.




## Configuration of the Slave Axis Servo Drive.

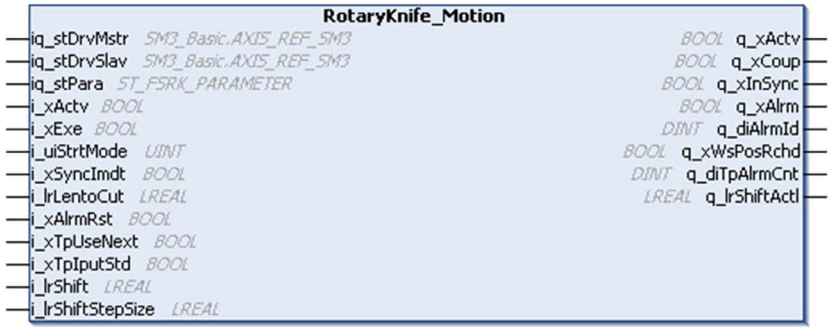
In addition to the communication parameters (addresses on the bus, speed transmission), the following parameter adjustments should be made to the drive in order to optimize the response of the servo drive that controls the slave axis:

- increase the maximum slopes of acceleration and deceleration
- set the parameter `N_ref_Filter` to OFF
- various loop parameters. They include proportional gain speed loop, integral time speed loop and the proportional gain position.

# Section 9.3
## Function Block Description

## `RotaryKnife_Motion` Function Block

### Pin Diagram




### Profiles of Master/Slave Positions

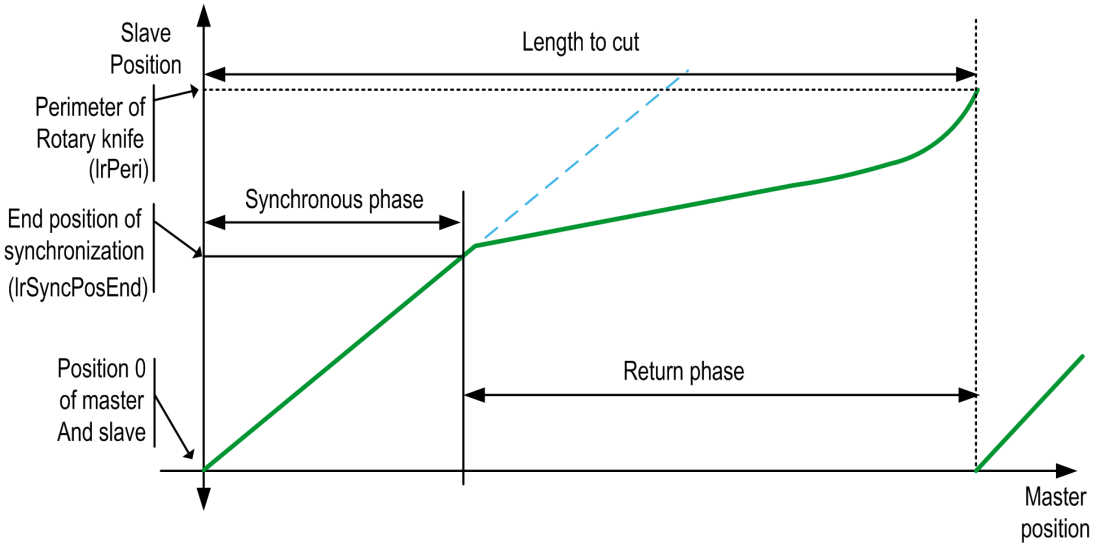
The curves (see case 1 and 2) show the position profile of the slave axis in relation to the master during the execution of the Rotary Knife function.

A Rotary Knife cycle consists of 3 phases:

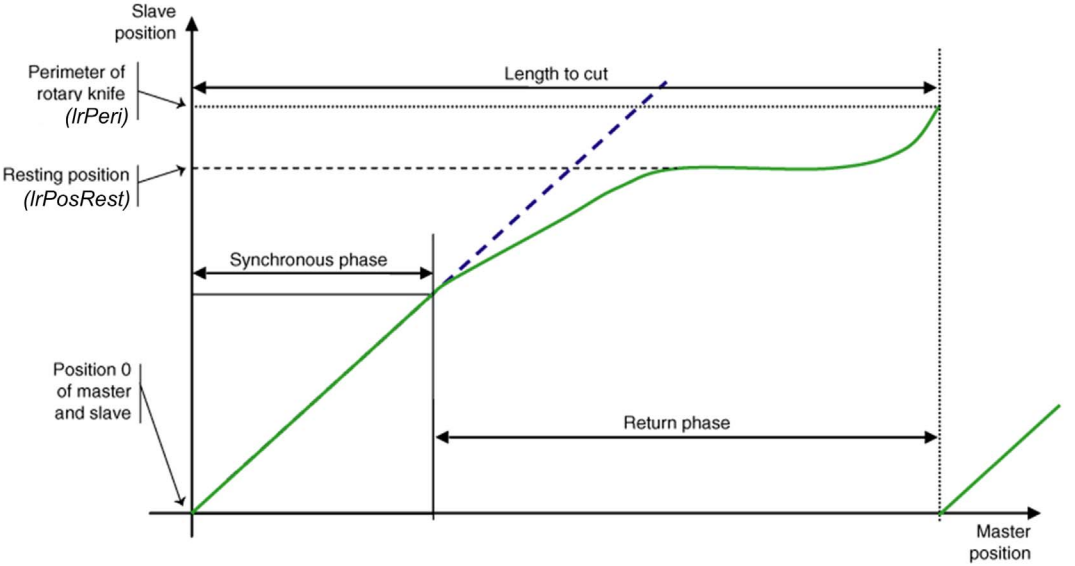
| Phase | Description |
|---|---|
| Start Phase | The slave accelerates up to the master velocity to be synchronized at the beginning of the synchronous phase.<br>● The start phase distance on the slave is equal to: Slave Start Phase Distance = ABS (`lrPeri -lrPosRest`)<br>● The start phase distance on the master can be calculated with the following formula: Master Start Phase Distance = (ABS (`lrPeri - lrPosRest`) / `lrM`) * 1.875 |
| Synchronous Phase | The cut (or the operation) is performed. The synchronous phase always starts with position 0 of the slave. |
| Return Phase | The slave returns, makes a half-turn before it begins a new synchronous phase. |

Depending on the length to cut and the perimeter of the Rotary Knife different cases are to be distinguished:

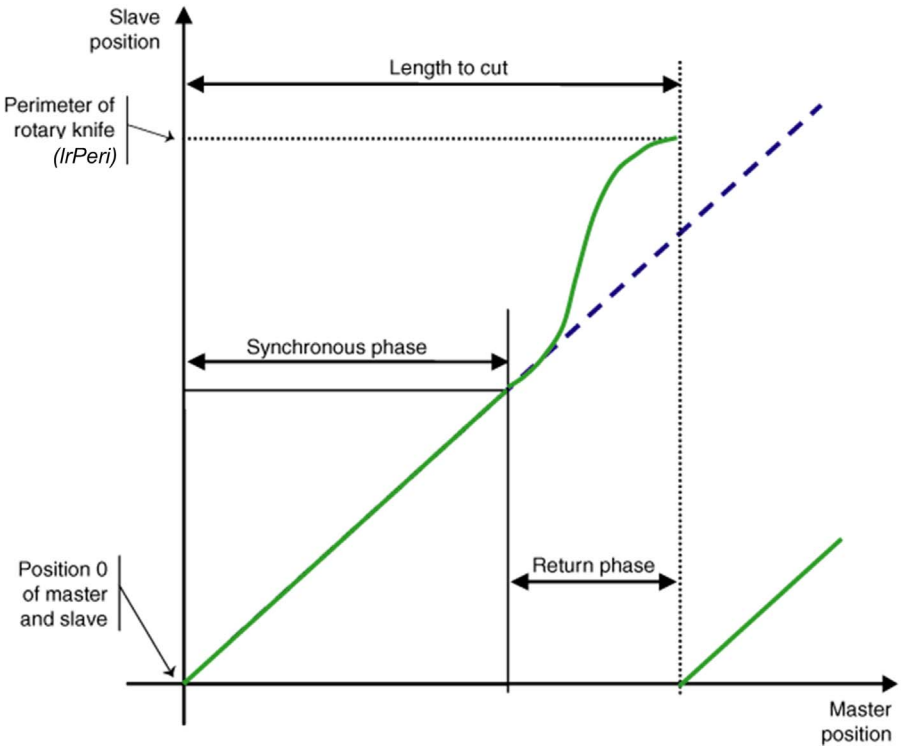
At the end of the synchronous phase, the slave decelerates and enters a new synchronous phase (see figure below).



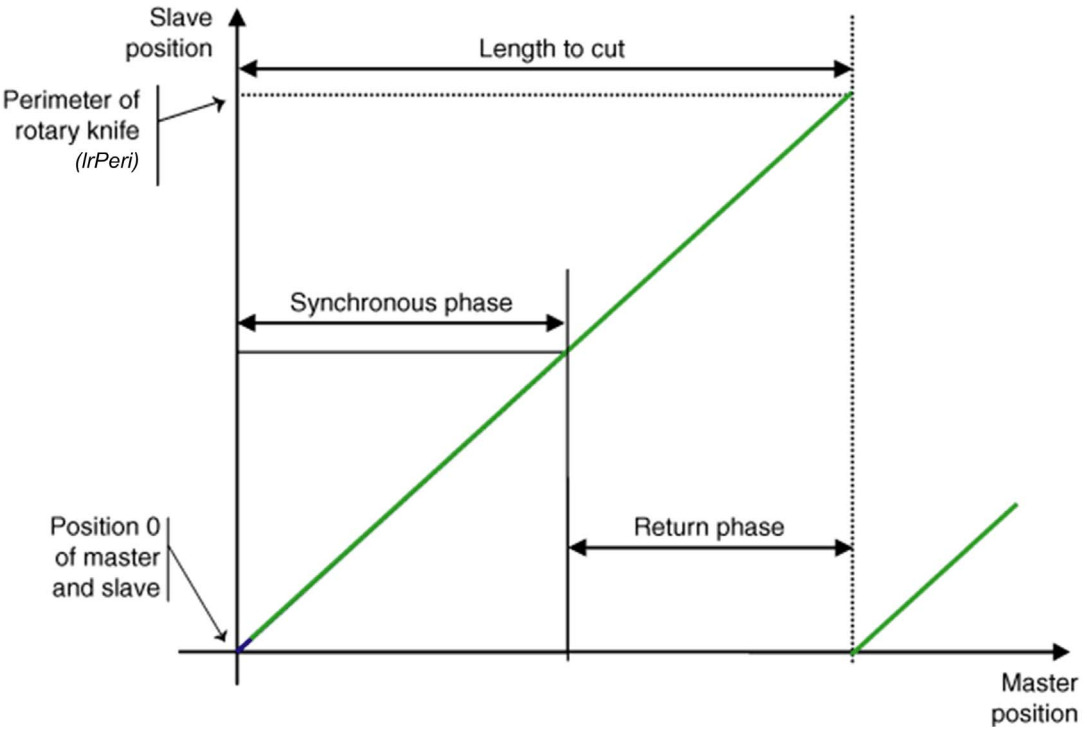
At the end of the synchronous phase, the slave decelerates, stops at the resting position and awaits a new cut position (see figure below).



At the end of the synchronous phase, the slave accelerates to enter a new synchronous phase as fast as possible (see figure below).



The length to cut is equal to the perimeter of the knife (see figure below).




The distance covered by the master during this cycle corresponds to the length to cut (`i_lrLentoCut`).

The distance covered by the slave during this cycle corresponds to the perimeter of the knife (`lrPeri`).

## Backward Movement with a Rotary Knife

Depending upon the machine, a backward movement of the slave (Rotary Knife) is prohibited during the synchronous phase therefore please pay attention to the following information:

Inside the synchronous phase, a backward movement of the slave is possible with the following situations:

- If the master moves backward. In this case the slave will follow the master backward and stop at 0. Do not move the master backward to avoid this situation.
- With warm start mode 1, 2, 4. Use warm start 3 or 5 to avoid this situation.

Outside the synchronous phase, a backward movement of the slave is possible with the following situations:

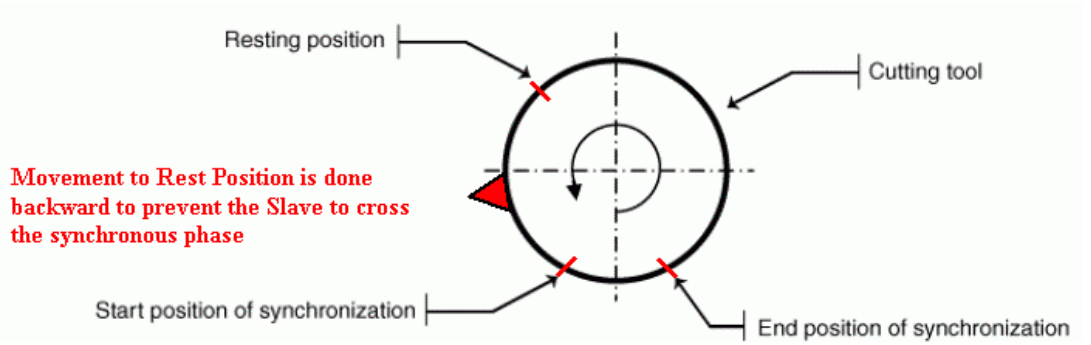
- If the master moves backward. In this case the slave will follow the master backward and stop at 0. Do not move the master backward to avoid this situation.
- With warm start 1 to 5 outside the synchronous phase
- If the slaves has to return to rest position (`lrPosRest`) when its position is included between `lrPosRest` and `lrPeri`. This is to prevent the salve to cross the synchronous phase (see figure below).

**⚠ CAUTION**

**ASYNCHRONOUS OPERATION OF CUTTING TOOL**

Use additional measures to avoid backward movement of the slave (Rotary Knife) unless the application accounts for such motion.

**Failure to follow these instructions can result in injury or equipment damage.**



# Section 9.4
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

Inputs/outputs are linked directly to the application function block interface. The main difference with the parameters is that inputs/outputs are refreshed cyclically, therefore they can be modified on the fly.

**NOTE:** These inputs are valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function block.

| Input | Data Type | Description | |
|---|---|---|---|
| `i_xActv` | `BOOL` | Activate:<br>TRUE: Application function block is activated.<br>FALSE: Application function block is de-activated.<br>Factory setting: FALSE<br>Start and Stop *(see page 300)* | |
| `i_xExe` | `BOOL` | Execute:<br>TRUE: Movement is started depending on the operating mode and the starting mode.<br>FALSE: Movement is stopped.<br>**NOTE:** If the slave is inside the synchronous phase, the movement will not be stopped immediately.<br>Factory setting: FALSE<br>Start and Stop *(see page 300)* | |
| `i_uiStrtMode` | `UINT` | Start mode: | |
| | | 0 | warm start mode 0 |
| | | 1 | warm start mode 1 |
| | | 2 | warm start mode 2 |
| | | 3 | warm start mode 3 |
| | | 4 | warm start mode 4 |
| | | 5 | warm start mode 5 |
| | | Factory setting: 0<br>Starting Modes *(see page 303)* | |
| `i_xSyncAbort`<br>(for FlyingShear only) | `BOOL` | Abort synchronization:<br>TRUE: Interruption of the synchronization phase.<br>Factory setting: FALSE<br>Interruption of the Synchronous Phase *(see page 343)* | |

| Input | Data Type | Description |
|---|---|---|
| i_xSyncImdt | BOOL | Immediate synchronization:<br>TRUE: Synchronize the slave immediately to the master.<br>This input is considered only when the slave is in resting position (lrPosRest). Only significant for operating modes 0.<br>Factory setting: FALSE |
| i_lrLentoCut | LREAL | Length to cut:<br>Can be modified on the fly. Only significant for operating modes 0 and 1.<br>Factory setting: 0<br>Range: lrLentoCutMin ≤..≤9999999 [Master User Unit] |
| i_xAlrmRst | BOOL | Alarm quit:<br>TRUE: Reset detected alarm.<br>If (and only if) i_xAlrmRst is triggered whereas the function block is OK, it will reset q_diTpAlrmCnt<br>Factory setting: FALSE<br>Troubleshooting *(see page 325)* |
| i_xTpUseNext | BOOL | Use next TP:<br>TRUE: The following position capture will be accepted as a valid capture even if it is outside of lrTpWdow.<br>Only significant for operating modes 1.<br>Factory setting: FALSE |
| i_xTpIputStd | BOOL | TP standard input:<br>Standard input used for the master position capture. Only significant if diCptrNb = 0.<br>Factory setting: FALSE<br>Selection of the Capture Mode *(see page 298)* |
| i_lrShift | LREAL | Shift:<br>Value of the shift to apply on the cutting point.<br>Factory setting: 0<br>Range: -lrPosSyncEnd ≤..≤ +lrPosSyncEnd [Master User Unit] |

<table>
<tr><th>Input</th><th>Data Type</th><th colspan="2">Description</th></tr>
<tr><td rowspan="6">i_lrShiftStepSize</td><td rowspan="6">LREAL</td><td colspan="2">Shift step size:<br>Maximum value of the shift to apply on the cutting point in 1 slave cycle.<br>0: Shift interruption. The shift of the tool stays constant at the current value even if the final shift: i_lrShift has not been reached.<br>NOTE: If i_lrShiftStepSize is greater than i_lrShift the shift is applied in one step.<br>Factory setting: 0<br>Range (according to operating mode):</td></tr>
<tr><td>Op. mode</td><td>Range</td></tr>
<tr><td>0</td><td>0 ≤..≤ (i_lrLentoCut-lrPosSyncEnd/lrM)*0.9)</td></tr>
<tr><td>1</td><td>0 ≤..≤ (i_lrLentoCut-lrPosSyncEnd/lrM-lrTpWdow/2)*0.9)</td></tr>
<tr><td>2</td><td>0 ≤..<br>[Master User Unit]</td></tr>
<tr><td></td><td></td></tr>
</table>

# Output Pin Description

## Output Pin Description

Inputs/outputs are linked directly to the application function block interface. The main difference with the parameters is that inputs/outputs are refreshed cyclically, therefore they can be modified on the fly.

**NOTE:** These outputs are valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

| Output | Data Type | Description |
|---|---|---|
| `q_xActv` | `BOOL` | Active:<br>TRUE: Application function block is initialized and activated.<br>FALSE: Application function block is inactive. |
| `q_xCoup` | `BOOL` | Coupled:<br>TRUE: Slave is coupled with the master on the Rotary Knife/FlyingShear profile curve. |
| `q_xInSync` | `BOOL` | In synchronization:<br>TRUE: Slave is in synchronous phase of the curve and synchronized to the master. |
| `q_xAlrm` | `BOOL` | Detected Alarm:<br>TRUE: An interruption occurred<br>Troubleshooting *(see page 325)* |
| `q_diAlrmId` | `DINT` | Detected Alarm ID:<br>Range: `0 ≤..≤ 99999`<br>Troubleshooting *(see page 325)* |
| `q_xWsPosRchd` | `BOOL` | Warm start position reached:<br>TRUE: Slave has reached the warm start position.<br>Starting Modes *(see page 303)* |
| `q_diTpAlrmCnt` | `DINT` | TP alarm counter:<br>Number of successive missed position captured.<br>One valid TP will reset `q_diTpAlrmCnt`<br>Only significant for operating modes 1.<br>Range: `0 ≤..≤ 99999` |
| `q_lrShiftActl` | `LREAL` | Current shift:<br>Current applied shift.<br>Range: `Master User Unit` |

# Input/Output Pin Description

## Input/Output Pin Description

**NOTE:** These inputs/outputs are valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stDrvMstr` | `SM3_Basic.AXIS_REF_SM3` | master axis reference structure |
| `iq_stDrvSlav` | `SM3_Basic.AXIS_REF_SM3` | slave axis reference structure |
| `iq_stPara` | STRUCT `ST_FSRK_PARAMETER` | block parameters structure |

## Structured Parameter

### iq_stPara

Parameters are entered to the application function block via the structure iq_stPara.

The main difference with inputs/outputs is that parameters are only refreshed after activation of the application function block (i_xActv=TRUE). Therefore parameters are only taken into account after a re-activation of the application function block.

### Inputs - iq_stPara

**NOTE:** These parameters are valid for the RotaryKnife_Motion and FlyingShear_Motion function blocks.

| Structure Parameter | Data Type | Description |
|---|---|---|
| lrPeri<br>(for the Rotary Knife only) | LREAL | Knife Perimeter<br>Perimeter of the Rotary Knife.<br>Distance between the blades for Multi Blade Rotary Knifes.<br>Factory setting: 0<br>Range: 0 ≤..≤ 999999 [Slave User Unit] |
| lrPosRest | LREAL | Rest Position:<br>Rest position of the slave<br>Factory setting: 0<br>Range: -999999 ≤..≤ 999999 [Slave User Unit] |
| lrSyncPosEnd | LREAL | Synchronization end position:<br>Final position of the synchronous phase (related to the slave axis).<br>Synchronous phase start at position 0 until position lrPosSyncEnd.<br>Factory setting: 0<br>Range: 0 ≤..≤ 999999 [Slave User Unit] |

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

- Set the modulo value of the master axis at least 25 times greater than the lrSyncPosEnd parameter.
- Verify that the modulo value of the master axis is as large as possible.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

| | | | |
|---|---|---|---|
| lrDecStop | LREAL | Stop deceleration:<br>Deceleration ramp to stop the slave in case of:<br>● detected alarm<br>● deactivation of the application function block (falling edge of i_xActv).<br>Factory setting: 1000<br>Range: 0 ≤..≤ max acc of the drive [Slave User Unit] | |
| lrWsWdow | LREAL | Warm start window:<br>Tolerance window for warm start. Only significant for warm start mode 2, 3, 4, 5.<br>Factory setting: 30<br>Range: 0 ≤..≤ 999999 [Master User Unit]<br>Starting Modes *(see page 303)* | |
| lrVel | LREAL | Velocity:<br>Slave speed used for the following movement:<br>● in warm start mode to move to the warm start position<br>● in cold start to move to rest position<br>● With a falling edge of i_xExe to move to rest position<br>Factory setting: 100<br>Range: 0 ≤..≤ max speed of the drive [Slave User Unit] | |
| lrAccDec | LREAL | Acceleration / Deceleration:<br>Acceleration and deceleration used for the following movement:<br>● in warm start mode to move to the warm start position<br>● during a cold start to move to rest position<br>● with a falling edge of i_xExe to move to rest position<br>Factory setting: 1000<br>Range: 0 ≤..≤ max acc of the drive [Slave User Unit] | |
| diCptrNb | DINT | Capture number:<br>Selection of the input for the position capture: | |
| | | 0 | position capture is made on a standard input. In this case the input has to be wired to input i_xTpIputStd |
| | | 1 | position capture is made on the Touch Probe 1 (TP1) |
| | | 2 | position capture is made on the Touch Probe 2 (TP2) |
| | | **NOTE:** The edge (rising edge or falling edge) can be changed using the configuration dialog of the encoder or by means of the drive setup tool.<br>Factory setting: 0<br>Range: 0 ≤..≤ 2<br>Selection of the Capture Mode *(see page 298)* | |

| | | |
|---|---|---|
| lrTpDistToSypt | LREAL | TP distance to the synchronization point:<br>Distance between the position capture sensor and the starting point of the synchronous phase.<br>Only significant for operating mode 1 and 2.<br>Factory setting: 500<br>Range: lrTpDistToSyptMin ≤... ≤ 999999 [Master User Unit] |
| lrTpWdow | LREAL | TP Window:<br>Tolerance window of the position capture.<br>For operating mode 1: Range within the captured position are valid.<br>For operating mode 2: Range in which, after an active TP, no other TP will be accepted.<br>Factory setting: 50<br>Range (for operating mode 1 and 2): lrTpWdow > ((lrPerim-lrPosRest)/lrm)*1,875 |
| diOpMode | DINT | Operating mode:<br>Selection of the operating mode.<br>0 – continuous product<br>1 – continuous product with mark compensation<br>2 – non continuous product (cut on the mark)<br>Factory setting: 0<br>Range: 0 ≤...≤ 2<br>Operating Modes *(see page 288)* |
| diTpMisdMax | DINT | TP missed maximum:<br>Maximum number of captures outside the tolerance window (defined with lrTpWdow).<br>An alarm is generated if q_diTpAlrmCnt > diTpMisdMax.<br>Only significant in operating mode 1<br>Factory setting: 3<br>Range: 0 ≤...≤ 999999 |
| lrM | LREAL | RM:<br>Introduce a slope coefficient in the synchronous phase.<br>Allows the definition of a different speed ratio during this phase.<br>Factory setting: 1<br>Range: 0 ≤...≤ 10 |
| xOfstMode | BOOL | Offset mode:<br>TRUE: Activate offset mode<br>FALSE: Deactivate offset mode<br>Factory setting: FALSE<br>Range: FALSE / TRUE [Slave User Unit] |

| | | |
|---|---|---|
| lrCamOfst | LREAL | CAM offset:<br>Offset added to the original slave profile.<br>Only relevant if offset.<br>Factory setting: 0<br>Range: 0 ≤...≤ lrPosSyncEnd/2 [Slave User Unit] |
| lrPosOfstStrt | LREAL | Offset start position:<br>Slave position from where the offset distance is added.<br>Only relevant if offset mode is activated (xOfstMode=TRUE)<br>Factory setting: 0<br>Range: 0 <...< lrPosOfstStrt [Slave User Unit] |
| lrPosOfstEnd | LREAL | Offset end position:<br>Slave position where the offset distance is reached.<br>Only relevant if offset mode is activated (xOfstMode=TRUE)<br>Factory setting: 0<br>Range: lrPosOfstStrt <...< lrPosSyncEnd [Slave User Unit] |

## Outputs - iq_stPara

| Structure Parameter | Data Type | Description |
|---|---|---|
| lrTpDistToSyptMin | LREAL | TP minimum distance:<br>Calculated by the application function block for information.<br>Minimum distance between the position capture sensor and the starting point of the synchronous phase (position 0 of the slave).<br>Only relevant in Operating mode 1 and 2.<br>Calculation for Rotary Knife:<br>● lrTpDistToSyptMin = ABS (lrPeri - lrPosRest / lrM) *1.875) + lrTpWdow / 2<br>Calculation for Flying Shear:<br>● lrTpDistToSyptMin = ABS (lrPosRest / lrM) *1.875) + lrTpWdow / 2<br>Range: [Master Unit] |

| Structure Parameter | Data Type | Description |
|---|---|---|
| `lrLentoCutMin` | `LREAL` | Minimum length to cut:<br>Calculated by the application function block.<br>Minimum possible value of length to cut.<br>Calculation for Rotary Knife:<br>● in operating mode 0: `lrLentoCutMin` = `lrPosSyncEnd`<br>● in operating mode 1: `lrLentoCutMin` = `lrPosSyncEnd` + `lrTpWdow` / 2<br>● in operating mode 2: `lrLentoCutMin` = `lrPosSyncEnd` + ABS ((`lrPeri - lrPosRest` / `lrM`) * 1.875)<br>Calculation for Flying Shear:<br>● in operating mode 0: `lrLentoCutMin` = `lrSyncPosEnd`<br>● in operating mode 1: `lrLentoCutMin` = `lrSyncPosEnd` + `lrTpWdow` / 2<br>● in operating mode 2: `lrLentoCutMin` = `lrSyncPosEnd` + ABS ((`lrPosRest` / `lrM`) * 1.875)<br>**NOTE:** This minimum length to cut value is theoretical and cannot be reached. Practically, reducing the length to cut increase the velocity, acceleration and deceleration constraints on the slave axis. Therefore the minimum length to cut depends on the velocity, acceleration and deceleration that the drive and the mechanic can handle.<br>If the maximum master velocity is known as well as the maximum velocity supported by the slave axis then it is possible to calculate the real minimum length to cut with the following formulas:<br>Formula for Rotary Knife:<br>$i\_lrLentoCut\min = \frac{lrSyncPosEnd}{lrM} + \frac{15\ ?\ (lrPeri - lrSyncPosEnd)}{8?\frac{V\max Slave}{V\max Master} + 7?lrM}$<br>Formula for Flying Shear:<br>$i\_lrLentoCut\min = \frac{lrSyncPosEnd}{lrM} + \frac{-15?lrSyncPosEnd}{-8?\frac{V\max Slave}{V\max Master} + 7?lrM}$<br>**NOTE:** These formulas do not take into account the acceleration.<br>Range: `[Master Unit]` |

# Section 9.5
## Operating Modes

### What Is in This Section?

This section contains the following topics:

# Operating Mode 0: Continuous Product

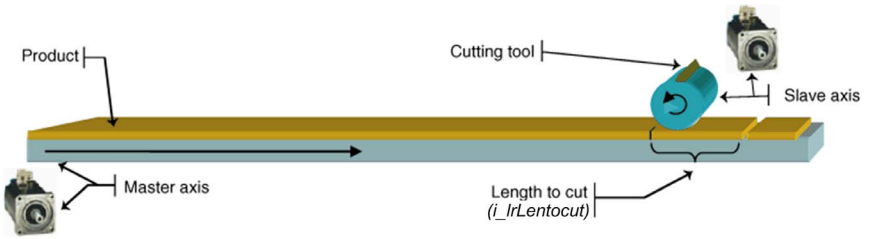
## Overview

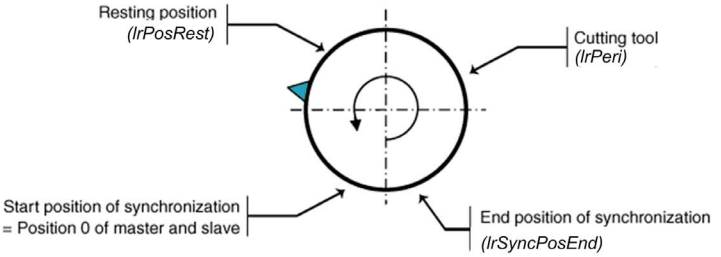
**NOTE:** This operating mode description is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

Operating mode 0 is used to process products with a constant length. This distance corresponds to the length of one process cycle and is defined via the input variable `i_lrLentoCut`.

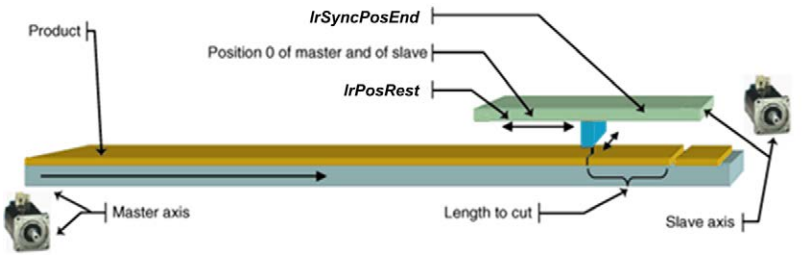
The parameter `i_lrLentoCut` can be modified during the movement. The modification is accepted for the next cycle.

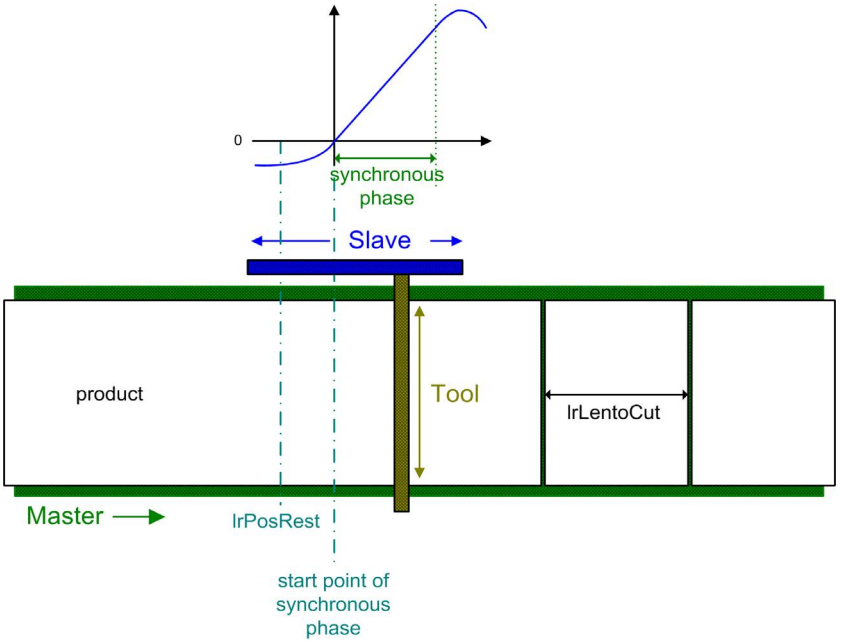
## Rotary Knife Representation



## Flying Shear Representation



## Operating Mode 1: Continuous Product with Mark Compensation

### Overview

This operating mode description is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

Operating mode 1 has basically the same functionality as operating mode 0, extended with an automatic correction of the product length, calculated from a position capture.

The product length correction is principally interesting in case of flexible product (with risk of deformation, stretching, etc.). In this case a label or a mark is needed on the product.

The cutting condition is then related to the length to cut parameter (`i_lrLentoCut`) which can be corrected according to the position captured at the time of label detection.

If a mark is detected inside the tolerance window, the measured length is used for cutting. Any mark before the tolerance window will be ignored. If the position leaves the window without seeing a mark, the default length is used and the output `q_diTpAlrmCnt` is incremented by 1.

The parameter `i_lrLentoCut` can be modified during the movement. The modification is accepted for the next the cycle.

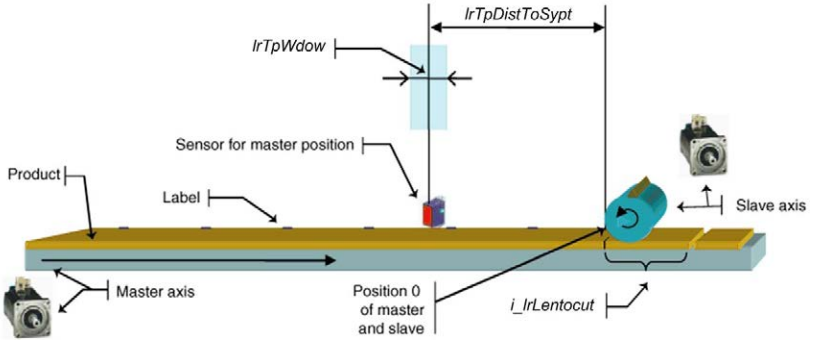
The parameter `lrTpDistToSypt` corresponds to the distance between the sensor for position capture and the beginning of the synchronous phase (position 0 of the slave).

Depending upon this distance, several labels may be located between the sensor and the beginning of the synchronous phase. Those captures are stored in a buffer. The RotaryKnife/FlyingShear functions allow the storage of up to 15 positions. Over 15 positions an alarm is detected.

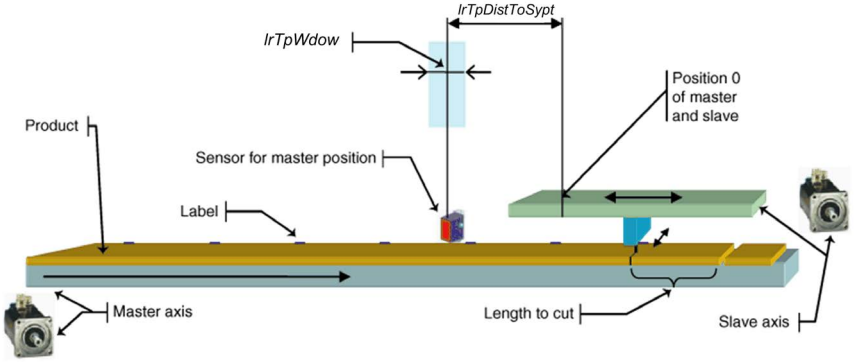
To improve the accuracy, `lrTpDistToSypt` has to be reduced to a minimum. However, depending upon the length to cut, this distance must be chosen higher than the minimum value, calculated by the block function and given by parameter output `lrTpDistToSyptMin`.

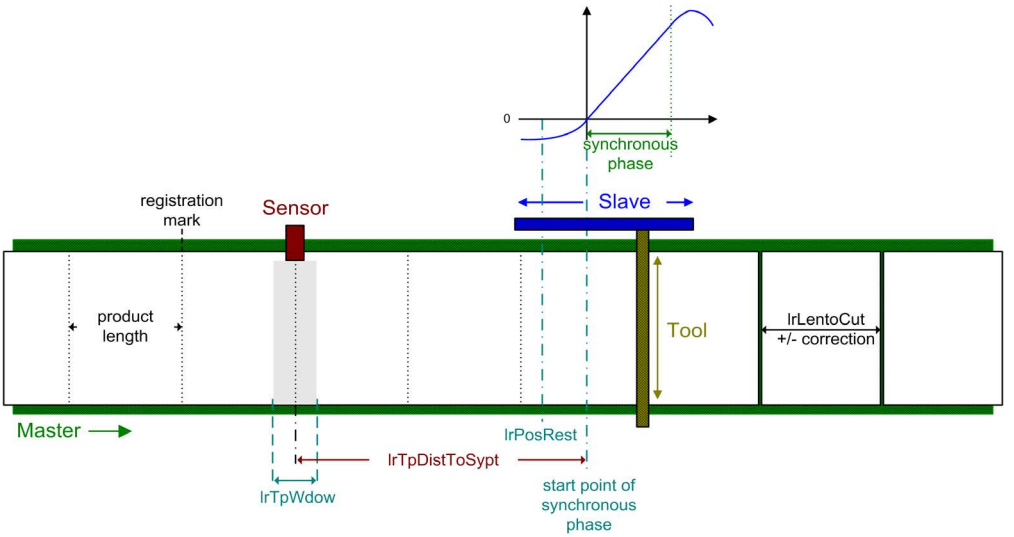
A tolerance capture window, `lrTpWdow`, can be used to define an interval of valid position.

## Rotary Knife Representation




## Flying Shear Representation

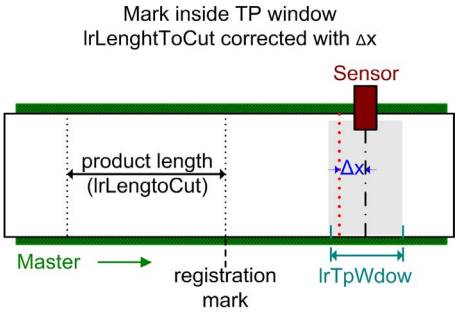


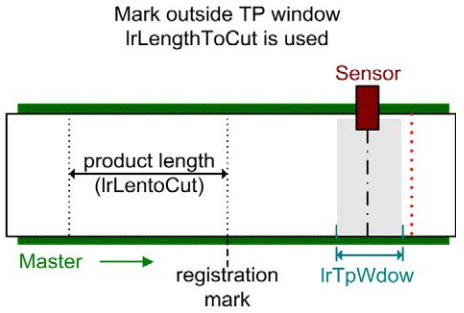



## Tolerance Capture Window (Operating Mode 1)

In help to avoid interfering signals (detection of false labels) or limit the stretching tolerance of elastic products, a tolerance window for capturing can be defined by `lrTpWdow`:

`i_lrLentoCut` - `lrTpWdow` /2< Range of tolerance < `i_lrLentoCut` + `lrTpWdow` /2






| If... | Then... |
|---|---|
| a mark is detected inside this range, | the measured length is used for cutting. Any mark before this range will be ignored. |
| the position leaves this range without seeing a mark, | the default length is used and the output `q_diTpAlrmCnt` is incremented by 1. |

A maximum number of successive captures outside this range can be configured with the parameter `diTpMisdMax`.

An alarm occurs if `q_diTpAlrmCnt` > `diTpMisdMax`. The slave is stopped with the ramp `lrDecStop`.

The alarm counter `q_diTpAlrmCnt` is set back to 0 with the following conditions:
- a valid capture signal before an alarm occurs
- the function block is de-activated
- with rising edge of input `i_xAlrmRst` while no alarm is active

With a rising edge of the input `i_xTpUseNext` the next capture signal is valid without any consideration to the `lrTpWdow` and the alarm counter `q_diTpAlrmCnt` is set back to 0.

# Operating Mode 2: Non Continuous Product (Cut on Mark)

## Overview

This operating mode description is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

In the operating mode 2, the product arrives on the conveyor in a discontinuous and random way. The size and the format can vary. The cutting condition is started exclusively by the position capture.

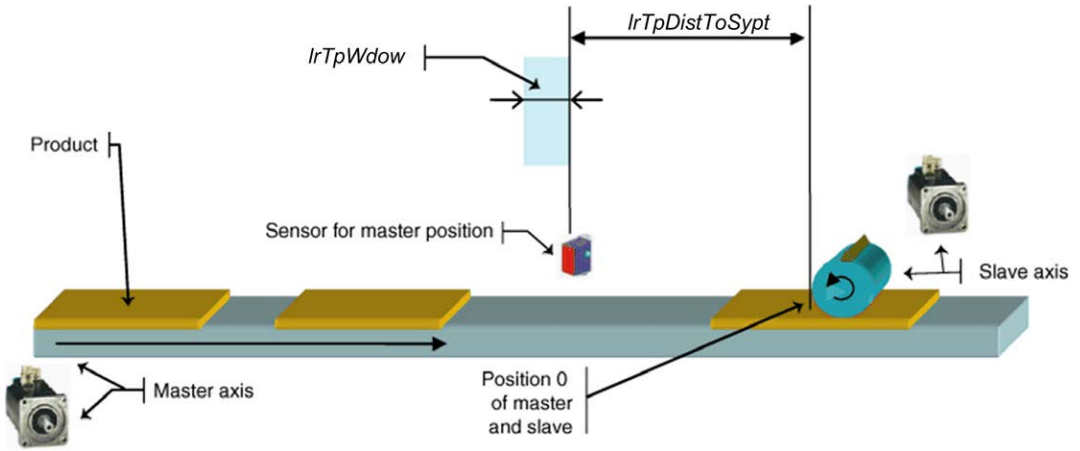
The parameter `lrTpDistToSypt` corresponds to the distance between the sensor for position capture and the beginning of the synchronous phase (position 0 of the slave).

Depending upon this distance, several labels may be located between the sensor and the beginning of the synchronous phase. Those captures are stored in a buffer. The RotaryKnife/FlyingShear functions allow the storage of up to 15 positions. In case more than 15 positions are detected an alarm will be raised.

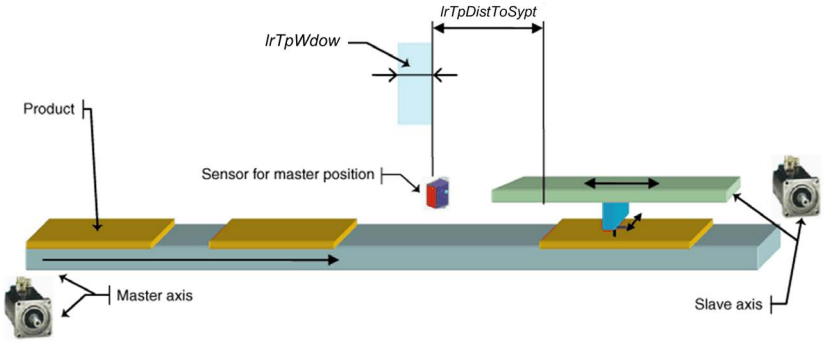
`lrTpDistToSypt` has to be higher than the minimum value `lrTpDistToSyptMin` calculated by the function block.

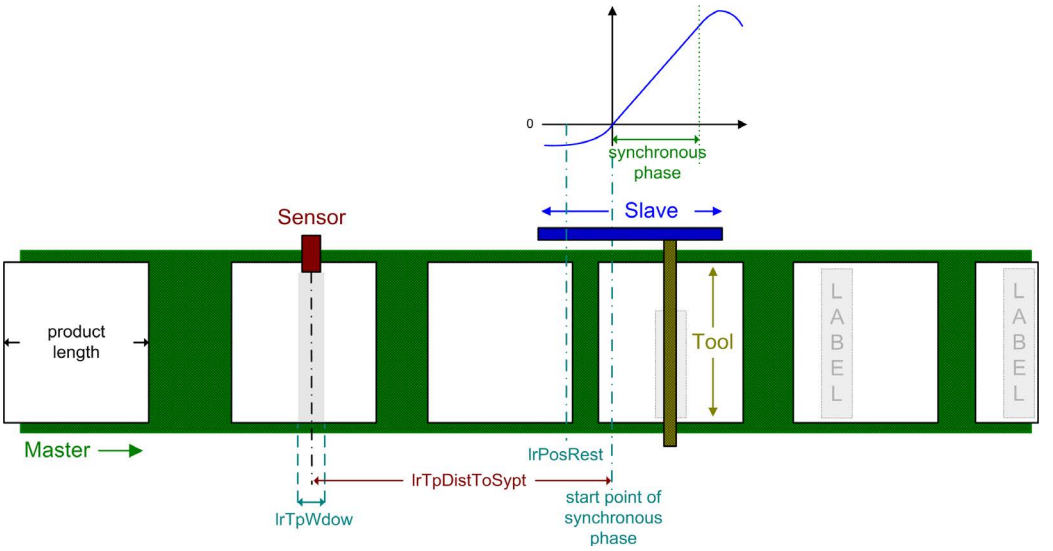
A capture ignorance window can be used to define an interval of valid position.

## Rotary Knife Representation



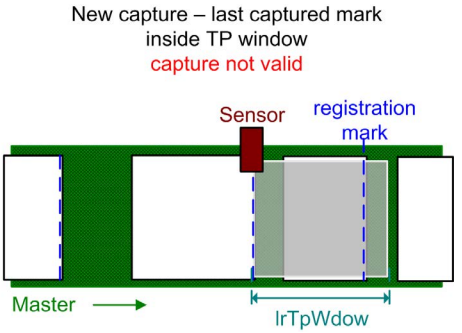
## Flying Shear Representation

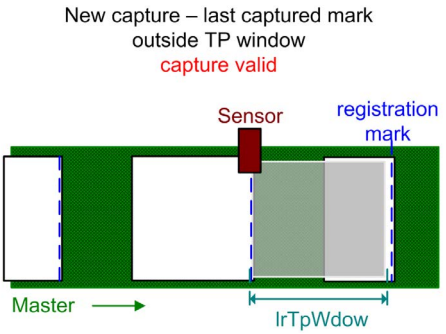


## Capture Ignorance Window (Operating Mode 2)

In order to avoid detection of false signals, a tolerance for position captures can be defined by `lrTpWdow`. The range begins with the previous captured position.




To be a valid mark, the position has to be greater then the last accepted mark + `lrTPWdow`: position of previous + `lrTpWdow`.

If a mark is detected inside this range, the product length is calculated depending on this mark.

Any detection outside this range is ignored.

# Selection of the Capture Mode

## Selection of the Capture Mode

**NOTE:** This mode description is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

Position capture is used with operating mode 1 and 2.

According to the architecture and the requirements of precision for the application, several capture modes are possible.

| Mode | Description |
|---|---|
| Detection with standard input | The sensor is wired to a standard input. The capturing is within the cycle of the motion controller. |
| Detection with fast input | The sensor is wired to a Touch Probe input<br>• `TP1`, `TP2`, … if the master is an external encoder<br>• a capture input of a servo drive, if the master is a drive |

Capture input can be configured with parameter `diCptrNb`:

| diCptrNb | i_xTpIputStd |
|---|---|
| 0:<br>Capture is done with a standard input | Associated standard input. If input `DI3` is used, then `DI3` is wired to `i_xTpIputStd`. |
| 1 to 2:<br>Capture is done on Touch Probe 1 to 2 | No effect |

If a Touch Probe is selected, the edge (rising edge or falling edge) has to be selected outside the application function block.

# Section 9.6
## Start, Stop and Starting Modes

### What Is in This Section?

This section contains the following topics:

## Start and Stop

### Overview

**NOTE:** This description is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

The correct sequence for the application function block is as follows:

- activate the application function block (`i_xActv`=TRUE)
- start the movement (`i_xExe`=TRUE)
- de-execute the application function block (`i_xExe`=FALSE)
- de-activate the application function block (`i_xActv`FALSE)

### Activate the Application Function Block

The function block is activated with a rising edge of `i_xActv`.

- the input parameters (defined in `iq_stParam`) are confirmed for consistency.
- `q_xActv` is set to TRUE
- in case of incorrect or inconsistent inputs or input parameters, output `q_xAlrm` becomes TRUE. Troubleshooting *(see page 325)*

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Do not send any other commands to the slave axis when it is under control of the `RotaryKnife_Motion`/`FlyingShear_Motion` application function blocks.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

### Start the Movement

Movement is started with a rising edge of `i_xExe`.

- verify slave axis state (slave has to be standstill)
- inputs are confirmed for validity
- if no alarm occurred, movement is started depending on the following conditions Starting Modes *(see page 303)*:
  - with the first rising edge of `i_xExe` after activation of the application application function block (rising edge of `i_xActv`) a cold start is automatically executed. Input `i_uiStrtMode` is ignored.
  - with every following rising edge of `i_xExe` a warm start is executed depending on the input `i_uiStrtMode`
- output `q_xCoup` is set to TRUE when the slave is coupled to the master.
- in case of incorrect or inconsistent inputs, output `q_xAlrm` becomes TRUE. Troubleshooting *(see page 325)*

### Stop the Movement

On a falling edge of `i_xExe` the movement is stopped under the following conditions:

- If outside the synchronous phase (`q_xInSync`=FALSE), the slave is uncoupled from the master and returns directly to its rest position (`lrPosRest`).
- If inside the synchronous phase (`q_xInSync`=TRUE), the slave remains coupled with the master until the end of synchronous phase. Then the slave is uncoupled from the master and returns directly to its rest position (`lrPosRest`).

Therefore, `i_xExe`= FALSE does not guarantee that the slave stops immediately or remain stationary. To stop immediately or to block any movement of the slave controlled by the application function block, `i_xActv` has to be FALSE.

**NOTE:** In operating mode 1 and 2 and with a falling edge of `i_xExe`, the application function block continues to capture and buffer positions as long as the application function block is activated.

Other conditions that stop the slave:

- when the slave is coupled with the master (`q_xCoup`=TRUE), the slave stops if the master stops. Nevertheless, the slave remains coupled and restarts if the master restarts.
- on a falling edge of `i_xActv` the slave is stopped with the deceleration ramp `lrDecStop`. The master / slave synchronization is lost.
- if an alarm occurs, the slave is stopped with the deceleration ramp `lrDecStop`. The master / slave synchronization is lost.

### De-Activate the Application Function Block

On a falling edge of `i_xActv` the block is de-activated and:

- if moving, the slave is stopped with the deceleration ramp `lrDecStop`
- when the slave is stopped, all outputs are reset
- the current shift is reset (`q_lrShiftActl`)
- the position capture is stopped and the position capture buffer (for operating mode 1 and 2) is reset.

### Slave Movement to Rest Position (`lrPosRest`)

**NOTE:** For Rotary Knife only.

The slave goes to its rest position in the following situations:

Inside the synchronous phase:

- with a cold start
- with a warm start 0 if the master and slave are uncoupled

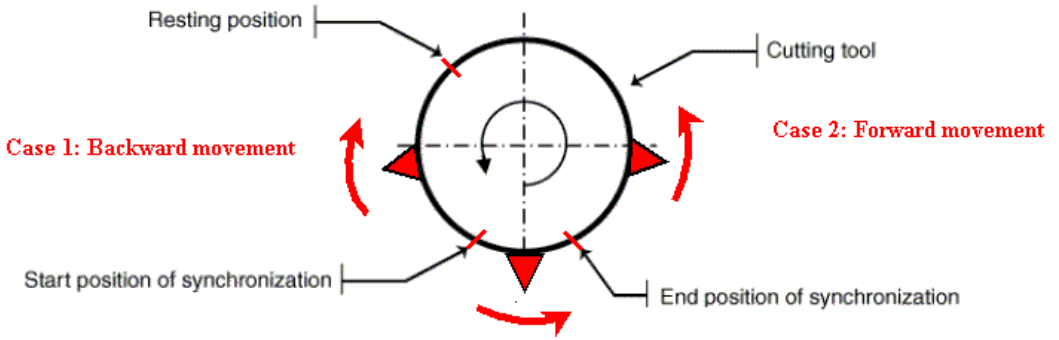
Outside the synchronous phase:

- with a cold start
- with a warm start 0 if the master and slave are uncoupled
- with a falling edge of execute input

The following rules are applied:

- In any case the slave does not cross the synchronous phase to move to its rest position (case 1 and 2, see figure below). In case 2 the slave moves backward.
- If the slave is in the synchronous phase, it moves forward to its rest position (case 3, see figure below).

The following figure shows the slave movement to the rest position.



# Starting Modes

## Overview

**NOTE:** This description is valid for the `RotaryKnife_Motion` and the `FlyingShear_Motion` function blocks.

There are 2 different types of starting modes:

| Mode | Description |
|---|---|
| Cold Start | is the starting method automatically used for the first execution of the application function block after activation. A cold start is necessary when there is no context coming from a previous execution of the application function block (No position captured inside the buffer, no shift applied, no output information…). |
| Warm Start | is the starting method used for a re-start of the application function block, following a temporary interruption (with a falling edge on `i_xExe`), an error-stop or an emergency stop without de-activation of the application function block. In this case the previous context is kept and considered for the re-start of the movement. |

Main differences between cold start and warm start:

In operating mode 1 and 2:

- at cold start the position buffered is empty. The first product to be cut will be the first detected product. Therefore, if the position sensor is 5 products before the cutting tool, the 5 intermediate products will not be cut.
- at warm start, the buffer is kept active. Therefore the application can restart immediately without loosing any product.

With a rising edge of `i_xExe` the movement is started depending on the following conditions:

- with the first rising edge of `i_xExe` after activation of the application function block (rising edge of `i_xActv`) a cold start is automatically executed. Input `i_uiStrtMode` is ignored
- with every following rising edge of `i_xExe` a warm start is executed depending on the input `i_uiStrtMode`

# Cold Start

## Overview

**NOTE:** This description is valid for the `RotaryKnife_Motion` and the `FlyingShear_Motion` function blocks.

A cold start is the starting method used for the first execution of the `RotaryKnife_Motion`/`FlyingShear_Motion` application function blocks after activation. In other word a cold start is necessary when there is no context coming from a previous execution of the application function block (No position captured inside the buffer, no shift applied, no outputs information…).

Therefore, with the first rising edge of `i_xExe` after activation of the application function block (rising edge of `i_xActv`) a cold start is automatically executed. Input `i_uiStrtMode` is ignored.

At cold start the slave moves to the rest position (`lrPosRest`). When the rest position has been reached, the slave is coupled to the master (using the start phase) depending on the operating mode (`diOpMode`) and the position capture buffer.

| Modes | Description |
|---|---|
| Operating Mode 0 | the slave will start immediately the Rotary Knife/FlyingShear profile when the master is started. |
| Operating Mode 1 and 2 | the first product to be cut will be the first detected product. Therefore, if the position capture sensor is placed 5 products before the cutting tool the 5 intermediates products will not be cut. |

# Warm Start

## Overview

**NOTE:** This description is valid for the `RotaryKnife_Motion` and the `FlyingShear_Motion` function blocks.

A warm start is the starting method used for a re-start of the application function block following a temporary interruption (with a falling edge on `i_xExe`), an error-stop or an emergency stop without de-activation of the application function block.

In this case the previous context is kept (buffer is still active, shift is maintained, outputs are relevant). Therefore in operating mode 1 and 2, the application can restart using valid positions from the buffer.

Six warm start modes are available depending on the application requirements. For each warm start mode there are certain conditions to be fulfilled. Warm start mode is selected with input `i_uiStrtMode`.

How to select a warm start mode:

| Mode | Description |
|---|---|
| Warm Start 0 (`i_uiStrtMode=0`) | is the only warm start possible with a moving master. |
| | has no tolerance window and can be used if warm start 1 to 5 is not possible (if slave is outside the tolerance window…). In this case the slave goes first to the rest position. |
| Warm Start 1 to 5 (`i_uiStrtMode=1` to `5`) | only possible if the master is stopped otherwise an alarm is created. |
| | is mainly used when there is a mechanical constraint between master and slave during the synchronization phase (a blade into a metal sheet…). In case of accidental de-coupling of the master and slave during the synchronization phase (emergency stop…) the slave can be re-coupled with the master following different conditions. |

For warm start modes 1 to 5, output `q_xPosWsRch` indicates that the warm start position has been reached. It signals to the application, that the master can be started.

### Warm Start 0 (`i_uiStrtMode = 0`)

**NOTE:** Warm start 0 is the only warm start possible with a moving master.

With a rising edge of `i_xExe`:

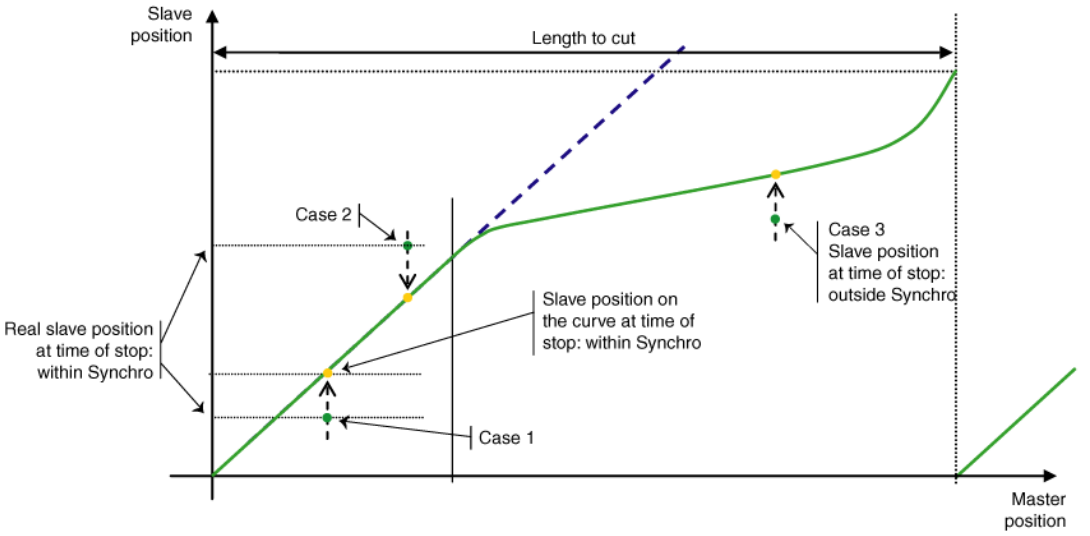
| If... | Then... |
|---|---|
| the slave is still coupled to the master (ex: `i_xExe=FALSE` with a master stopped into the synchronous phase) | the slave stays coupled to the master and restarts normally. |
| the slave is in rest position (`lrPosRest`) | the slave restarts with the next cycle if the distance is long enough to accomplish the start phase. |
| the slave is not coupled and not in rest position (in or out of the synchronous phase) | the slave goes to rest position and restart with the next cycle if the distance is long enough to accomplish the start phase. |

### Warm Start 1 (`i_uiStrtMode = 1`)

With a rising edge at `i_xExe`:

- The slave returns to the curve position regardless where (inside or outside synchronous phase) it has been stopped.

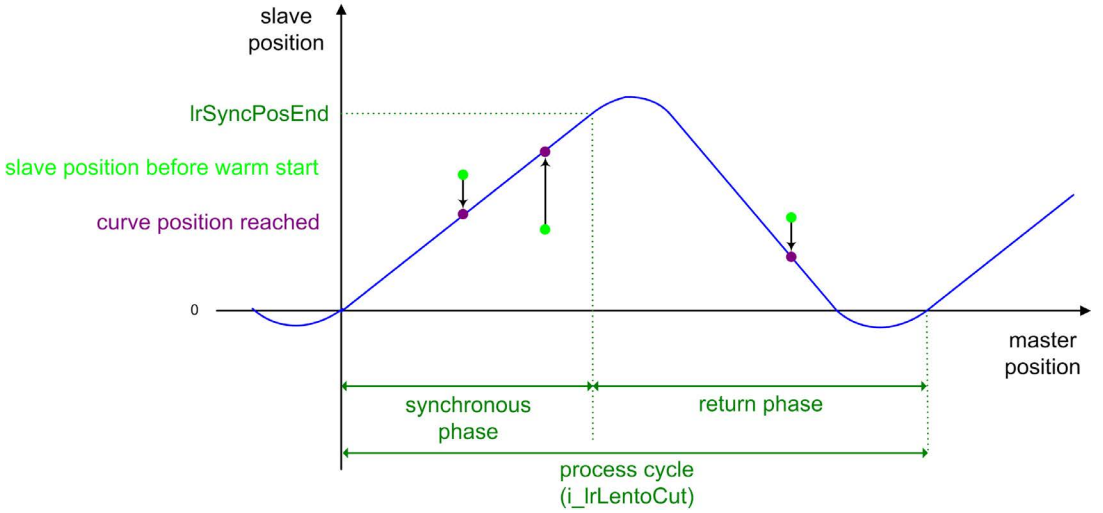
RotaryKnife Representation




Warm start 1 allows backward movement of the slave into the synchronous phase.

| ⚠ WARNING |
| --- |
| **UNINTENDED EQUIPMENT OPERATION**<br>Use warm start mode 3 or 5, if the machine design does not account for a backward movement of the slave (Rotary Knife).<br>**Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

FlyingShear Representation

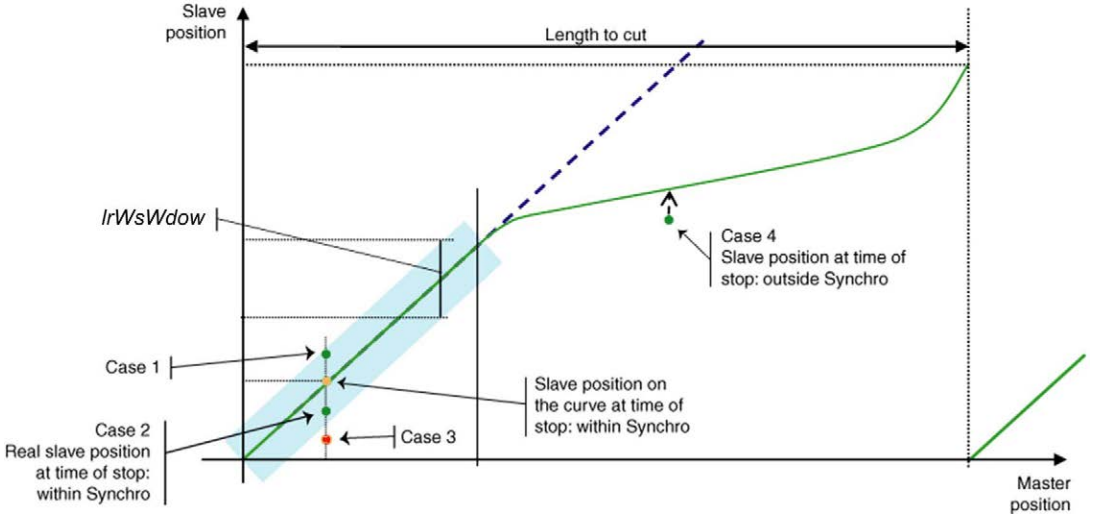


### Warm Start 2 (`i_uiStrtMode` = 2)

With a rising edge at `i_xExe`:

| RotaryKnife Case | FlyingShear Case | If.. | Then.. |
|---|---|---|---|
| 1 and 2 | 1 | the slave is in the synchronous phase and located inside the tolerance window defined by `lrWsWdow`<br>● slave position ≥ curve position - `lrWsWdow` / 2<br>AND<br>● slave position ≤ curve position + `lrWsWdow` / 2 | the slave returns to its curve position. |
| 3 | 2 | the slave is in the synchronous phase and located outside the tolerance window defined by `lrWsWdow`<br>● slave position > curve position + `lrWsWdow` / 2<br>OR<br>● slave position < curve position - `lrWsWdow` / 2 | the slave remains in position and an alarm is produced. Troubleshooting *(see page 325)* |
| 4 | 3 | the slave is outside the synchronous phase | the slave returns to its curve position. |

RotaryKnife Representation




Warm start 2 allows backward movement of the slave into the synchronous phase.

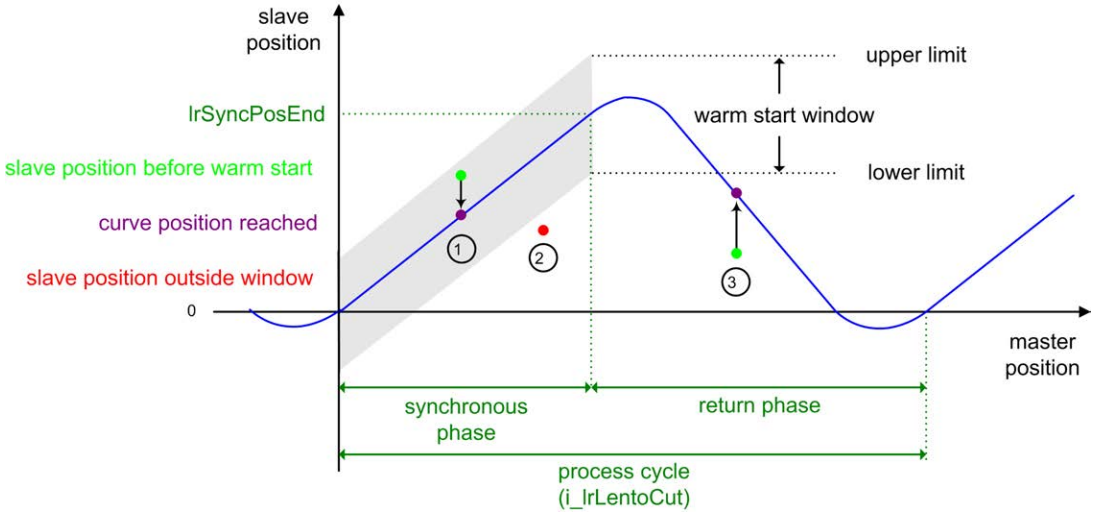
**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Use warm start mode 3 or 5, if the machine design does not account for a backward movement of the slave (Rotary Knife).

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

FlyingShear Representation




## Warm Start 3 (`i_uiStrtMode = 3`)

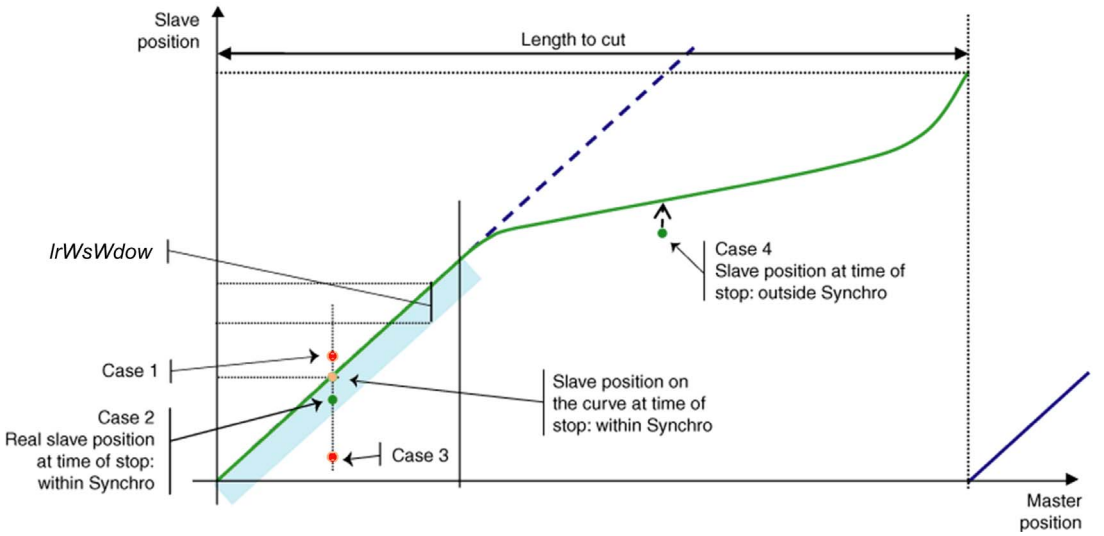
With a rising edge at `i_xExe`:

| RotaryKnife Case | FlyingShear Case | If.. | Then.. |
|---|---|---|---|
| 1 and 3 | 1 | the slave is in the synchronous phase and located outside the tolerance window defined by `lrWsWdow`<br>• LREAL position > curve position (see figure RotaryKnife, case 1)<br>OR<br>• LREAL position < curve position - `lrWsWdow` (see figure RotaryKnife, case 3) | the slave remains in position and an alarm is detected. Troubleshooting *(see page 325)* |

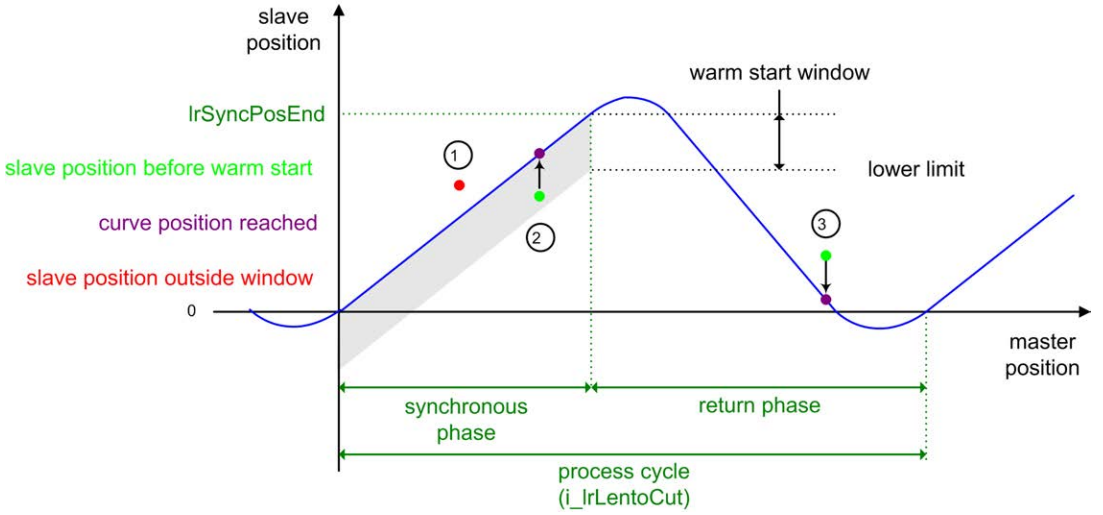
| RotaryKnife Case | FlyingShear Case | If.. | Then.. |
|---|---|---|---|
| 2 | 2 | the slave is in the synchronous phase and located inside the tolerance window defined by `lrWsWdow`<br>● slave position ≤ curve position<br>AND<br>● slave position ≥ curve position - `lrWsWdow` | the slave returns to its curve position. |
| 4 | 3 | the slave is outside the synchronous phase (see figure RotaryKnife, case 4) | the slave returns to its curve position. |

**NOTE:** In this warm start mode, the slave can move only in the positive direction in the event of a stop within the synchronous phase.

RotaryKnife Representation



FlyingShear Representation



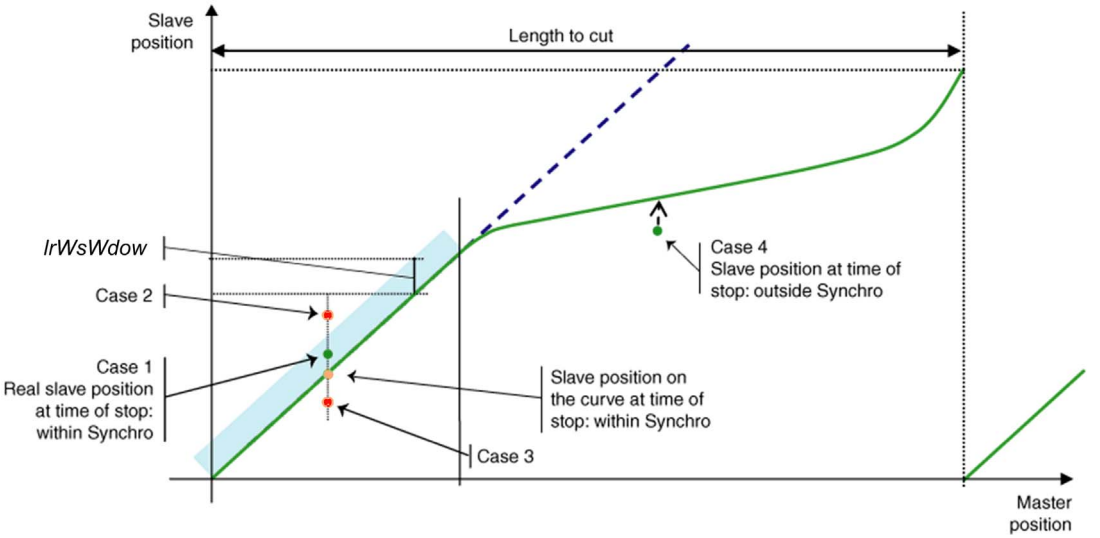

## Warm Start 4 (`i_uiStrtMode = 4`)

With a rising edge at `i_xExe`:

| RotaryKnife Case | FlyingShear Case | If.. | Then.. |
|---|---|---|---|
| 1 | 2 | the slave is in the synchronous phase and located inside the tolerance window defined by `lrWsWdow` (see figure RotaryKnife, case 2)<br>• LREAL position ≤ curve position + `lrWsWdow`<br>AND<br>• LREAL position ≥ curve position | the slave returns to its curve position. |
| 2 and 3 | 1 | the slave is in the synchronous phase and located outside the tolerance window defined by `lrWsWdow`<br>• LREAL position ≥ curve position + `lrWsWdow` (see figure RotaryKnife, case 1)<br>OR<br>• LREAL position < curve position (see figure RotaryKnife, case 3) | the slave remains in position and an alarm is detected. Troubleshooting *(see page 325)*. |
| 4 | 3 | the slave is outside the synchronous phase (see figure RotaryKnife, case 4) | the slave returns to its curve position. |

**NOTE:** With this warm start mode, the slave can move only in the negative direction in the event of a stop within the synchronous phase.

RotaryKnife Representation




Warm start 4 allows backward movement of the slave into the synchronous phase.

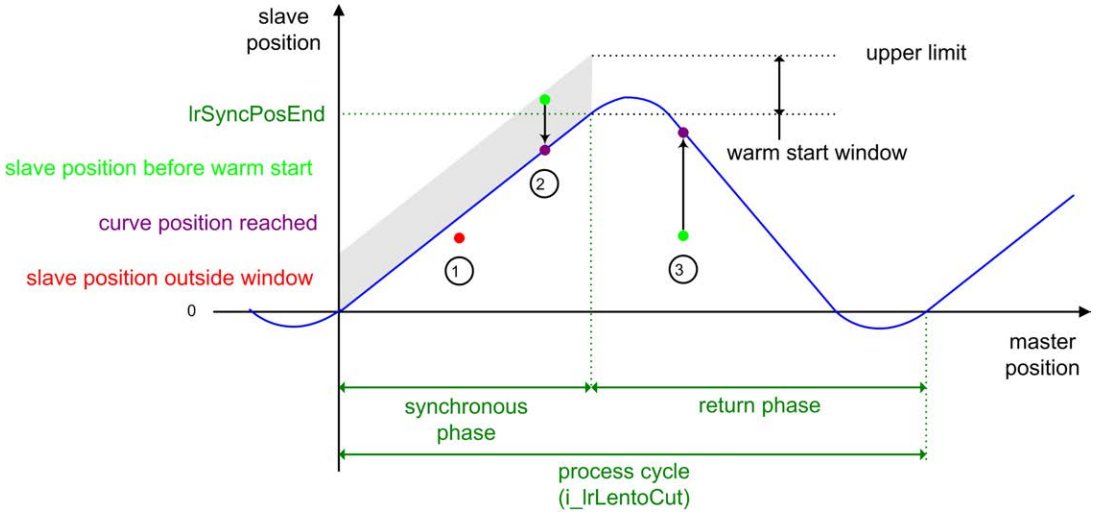
**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Use warm start mode 3 or 5, if the machine design does not account for a backward movement of the slave (Rotary Knife).

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

FlyingShear Representation



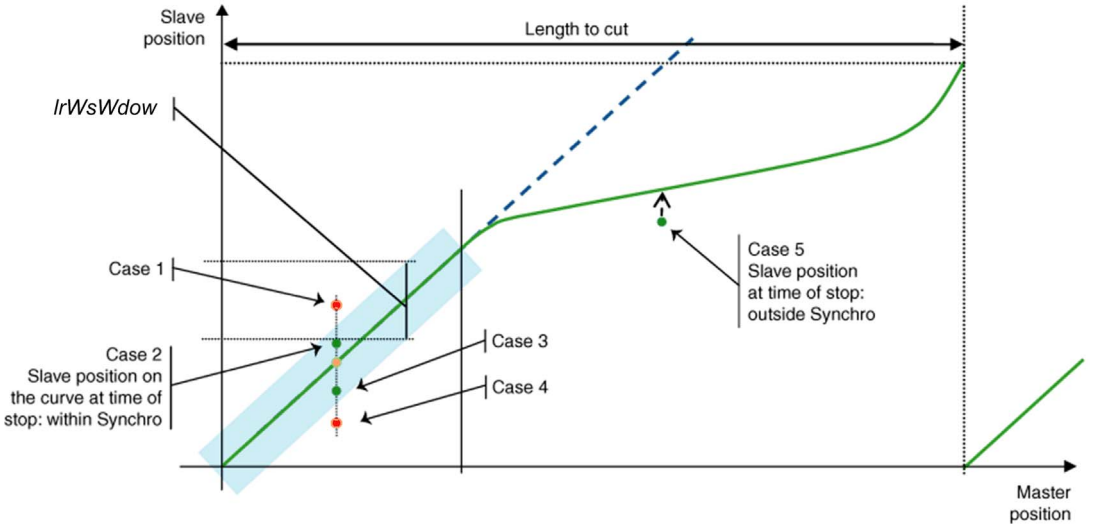

## Warm Start 5 (`i_uiStrtMode = 5`)

With a rising edge at `i_xExe`:

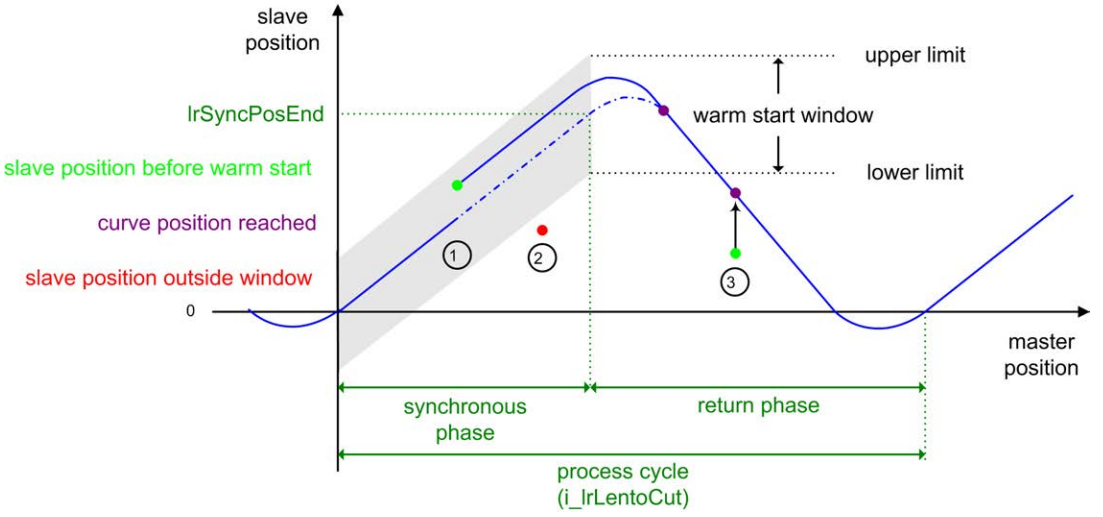
| RotaryKnife Case | FlyingShear Case | If.. | Then.. |
|---|---|---|---|
| 1 and 4 | 2 | the slave is in the synchronous phase and located outside the tolerance window defined by `lrWsWdow`<br>• LREAL position > curve position + `lrWsWdow` / 2 (see figure RotaryKnife, case 1)<br>OR<br>• LREAL position < curve position - `lrWsWdow` / 2 (see figure RotaryKnife, case 4) | the slave remains in position and an alarm is detected. Troubleshooting *(see page 325)*. |
| 2 and 3 | 1 | the slave is in the synchronous phase and located inside the tolerance window defined by `lrWsWdow`<br>• LREAL position ≥ curve position - `lrWsWdow` / 2 (see figure RotaryKnife, case 2)<br>AND<br>• LREAL position ≤ curve position + `lrWsWdow` / 2 (see figure RotaryKnife, case 3) | the slave remains on its position and follow the master as with a normal profile until the end of the synchronous phase. |
| 5 | 3 | the slave is outside the synchronous phase (see figure RotaryKnife, case 5) | the slave returns to its curve position. |

**NOTE:** In this warm start mode, the slave keeps its position shift with the master until the end of synchronous phase in the event of a stop within the synchronous phase.

RotaryKnife Representation




FlyingShear Representation



# Section 9.7
## Special Modes

### What Is in This Section?

This section contains the following topics:

# Immediate Cut

## General

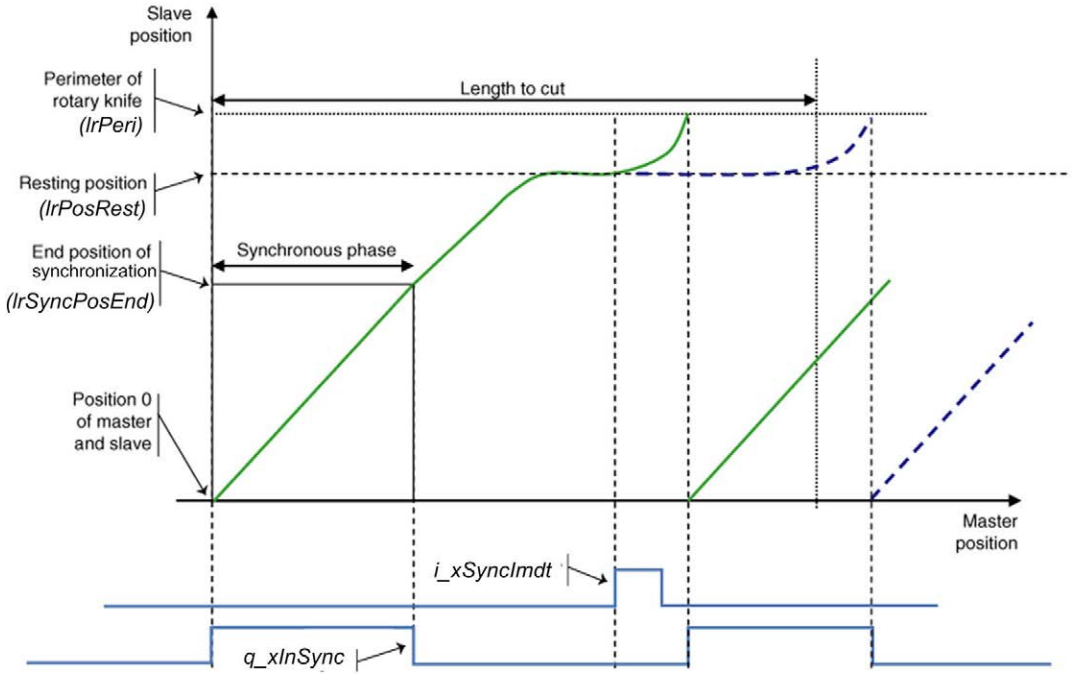
**NOTE:** This description is valid for the `RotaryKnife_Motion` and the `FlyingShear_Motion` function blocks.

With input `i_xSyncImdt` it is possible to engage an immediate cut in operating mode 0.

The slave does not await the end of the cycle in progress. It begins immediately a new cycle with the start phase. The length of this new cycle will be equal to the length `lrLentoCut`.

The input reacts on a rising edge and is considered only if the slave is in `lrPosRest`.This input will be ignored in operating mode 1 and 2.

RotaryKnife Representation



# Offset Mode

## General

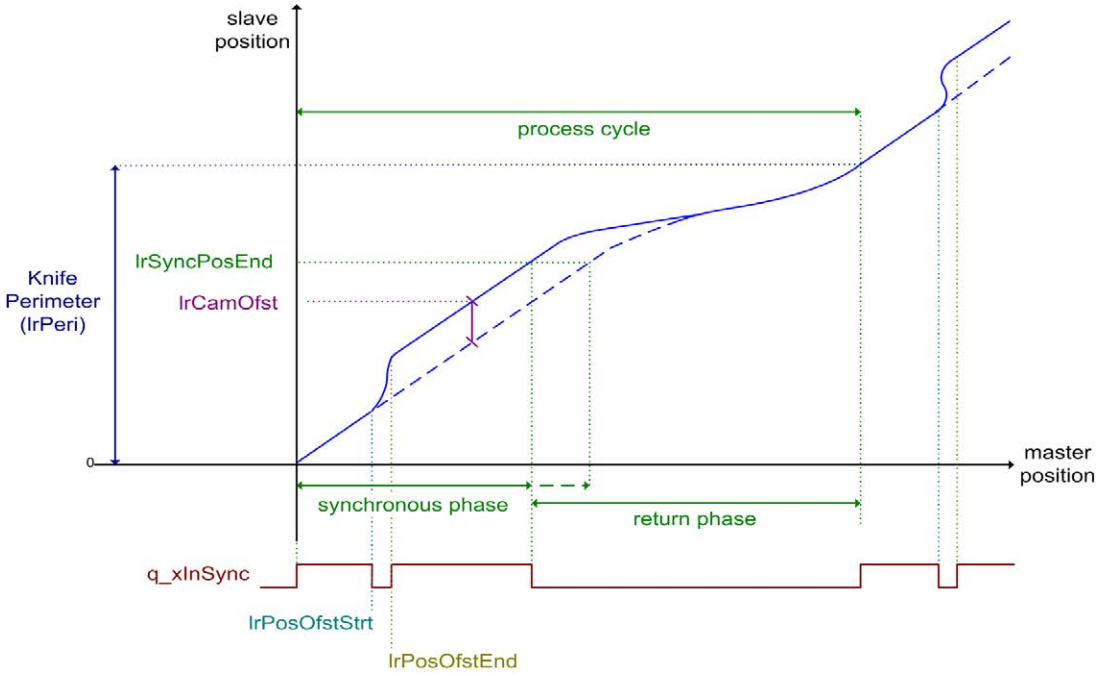
**NOTE:** This description is valid for the `RotaryKnife_Motion` and the `FlyingShear_Motion` function blocks.

The offset mode:

- allows the shift of the slave axis a certain distance (`lrCamOfst`) during the synchronous phase.
- can only be activated when the application function block is de-activated.
- When activated, the offset mode is applied at every cycle.
- is activated with the parameter input `xOfstMode`.
- does not influence the length of the process cycle.
- value is defined with the parameter input `lrCamOfst`.
- output `q_xInSync` is TRUE during the complete synchronous phase although slave and master are not synchronized during the transition phase needed to apply the offset.

The start and end positions of this transition are defined by the following parameters:

- `lrPosOfstStrt`
- `lrPosOfstEnd`



# Oblique Cut

## General

**NOTE:** This description is valid for the `RotaryKnife_Motion` and the `FlyingShear_Motion` function blocks.

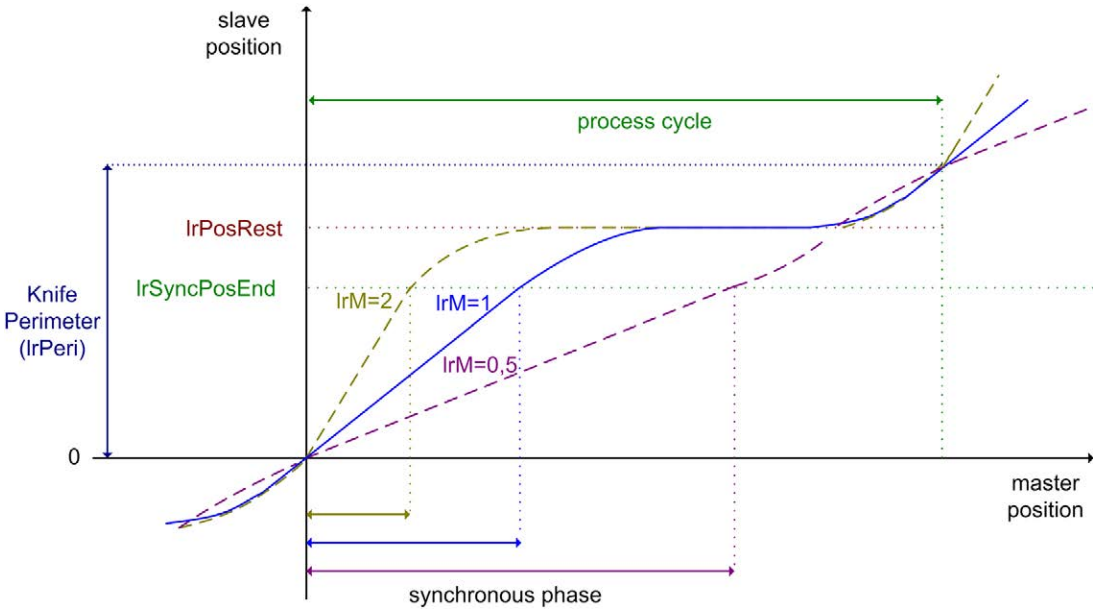
With the parameter `rm`, it is possible to adjust the ratio of the position between the slave and the master during the synchronous phase.

`rm = ΔY / ΔX`

This coefficient should be used in the applications, in which the paths of master and slave are not parallel.

`lrm` can only be modified when the application function block is de-activated.

- `lrm` = 1, the speed of slave and master are equal
- `lrm` < 1, slaves moves slower than the master
- `lrm` > 1, slave moves quicker than the master



# Shift Function

## General

**NOTE:** This description is valid for the `RotaryKnife_Motion` and the `FlyingShear_Motion` function blocks.

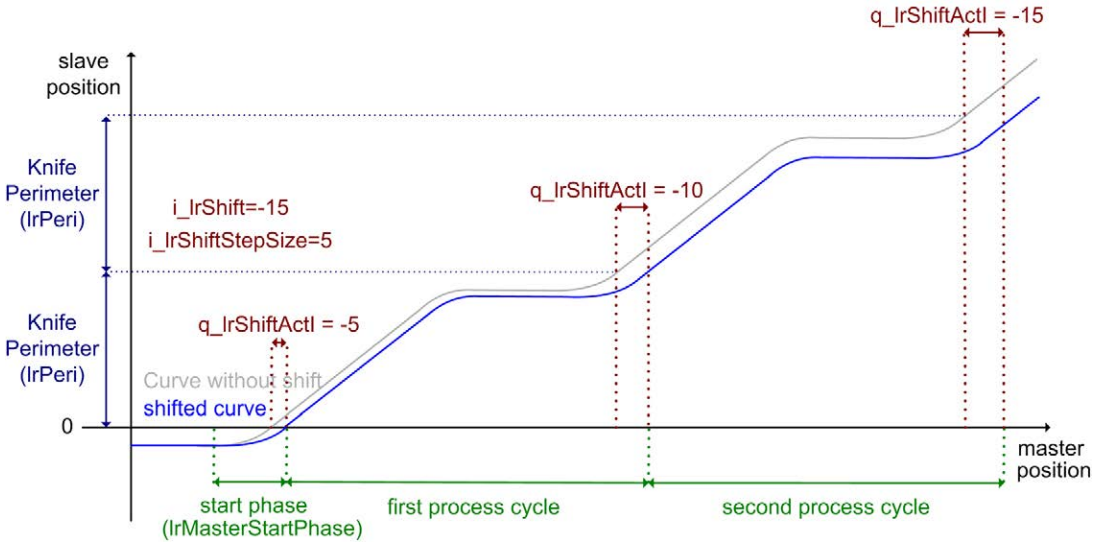
The shift function allows to adjust manually the cut on the product. This shift (`i_lrShift`) is applied to the master position in order to shift the starting point of cutting on the product.

**NOTE:**
The shift necessarily creates a variation of the `i_lrLentoCut` during the transition phase:

- a positive shift leads to a shorter product during the transition and moves the start point of the synchronous phase in the direction of the master movement.
- a negative shift leads to a longer product during the transition and moves the start point of the synchronous phase against the direction of the master movement.

In order to limit the `i_lrLentoCut` variation during the transition phase, it is possible to apply the total shift divided into several steps.

This can be set-up with `i_lrShiftStepSize`.



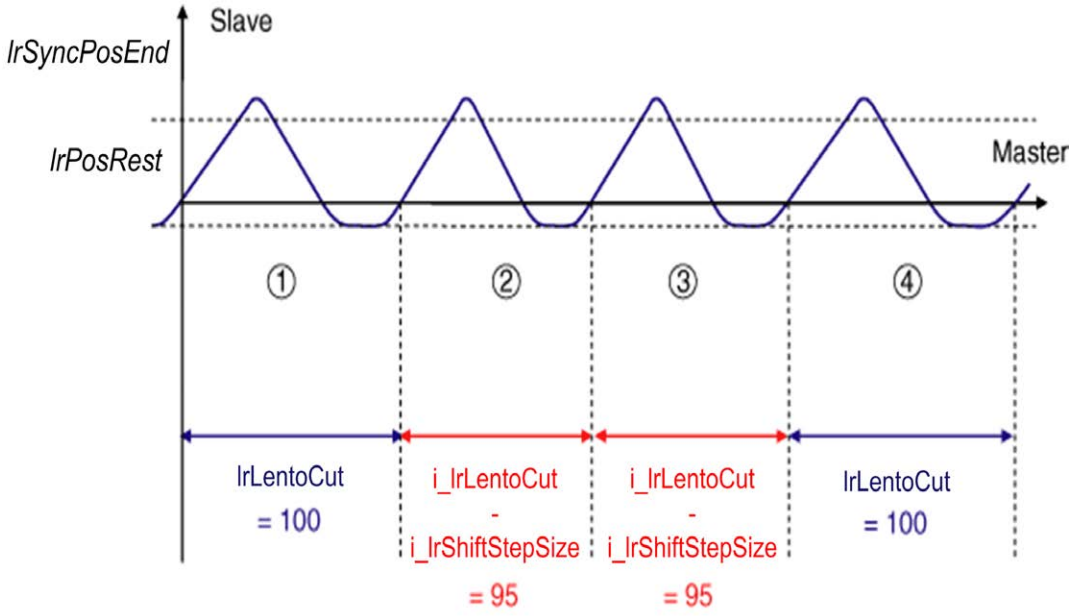
## Example

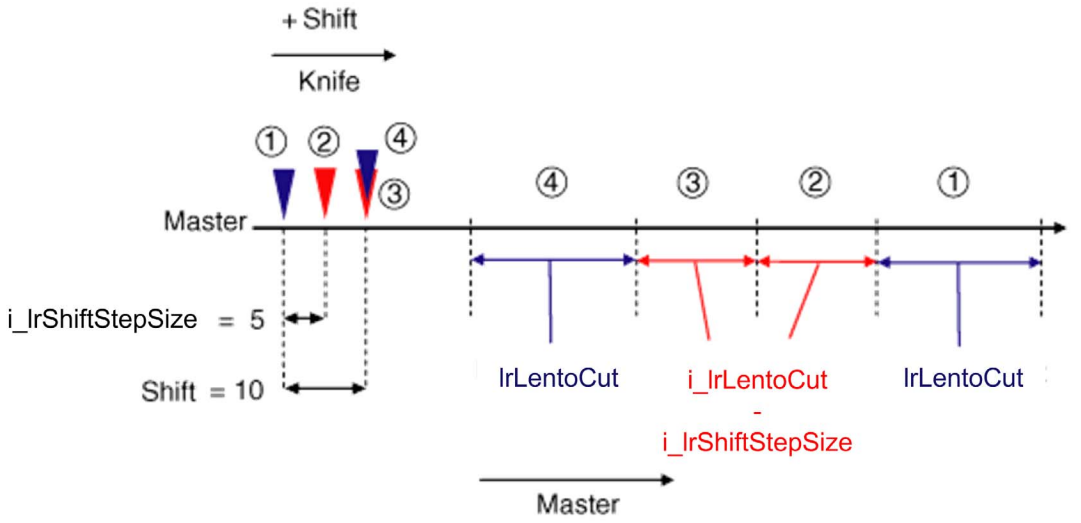
Example with a positive shift

Length to cut: `i_lrLentoCut` = 100

Steps 1-4 refer to figure below

| Step | Description |
|---|---|
| 1 | Length of the product = 100, the product is cut normally<br>`i_lrShift` = + 10 user units<br>`i_lrShiftStepSize` = 5 user units<br>**NOTE:** In this case the total shift will be applied in 2 steps, corresponding to 2 slave cycles. |
| 2 | Length of the product = 95, the product is shorter. |
| 3 | Length of the product = 95, the product is shorter.<br>The length of the product comes back to the normal length. |
| 4 | Length of the product = 100<br>The product is cut to the normal length but the starting point of the tool has been shifted of 10 user units. |






## Start Modes

Depending on the start mode (cold start or warm start), the behavior of the shift is different:

| If... | Then... |
|---|---|
| the slave is restarted with a cold start | the current shift is set to zero. No initial shift is applied. |
| the slave is restarted with a warm start | the current shift is kept and is applied as soon as the slave restarts. The positioning of the slave on the curve (depending on `i_uiStrtMode`) is done with the shift. |

Remarks:

The following positions and distances are not affected by the shift function:

- length to cut
- rest position

The following windows are not affected by the shift function:

- Touch Probe window
- warm start window

**NOTE:** `i_lrShift` and `i_lrShiftStepSize` are checked with every rising edge on `i_xActv` and `i_xExe`. When the slave is running (AFB active) `i_lrShift` and `i_lrShiftStepSize` are checked at every machine cycle.

# Section 9.8
## Quick Reference Guide

### What Is in This Section?

This section contains the following topics:

# Visualization

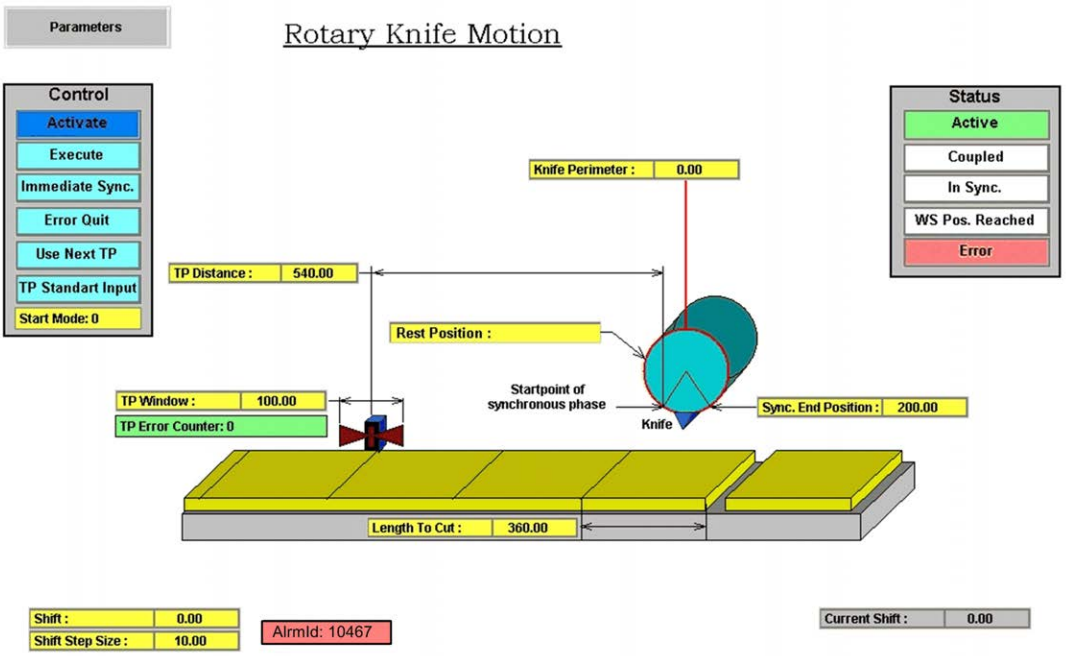
## Overview

Visualization screens are embedded in the Rotary Knife library. They allow to configure and to start the application function block very easily and quickly.

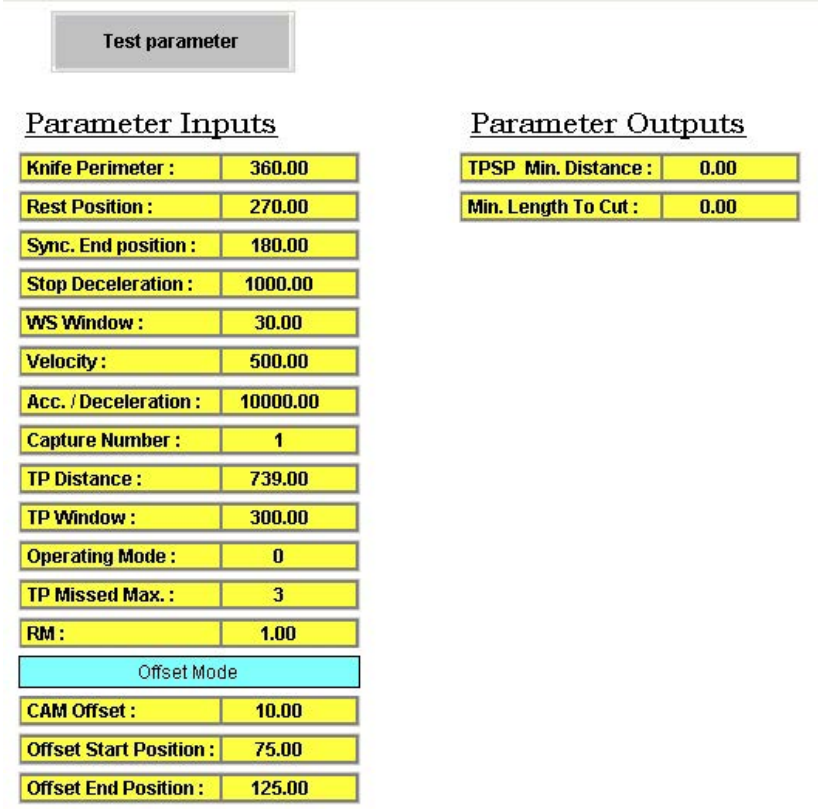
All the inputs/outputs and all the parameters are accessible.

For more information please refer to How to Integrate the Visualization into an Application *(see page 433)*.

Rotary Knife template visualization



Rotary Knife parameters visualization



# Troubleshooting

## Troubleshooting

### Troubleshooting

**NOTE:** This description is valid for the `RotaryKnife_Motion` and `FlyingShear_Motion` function blocks.

| Issue | `q_diAlrmId` Offset | Example | Remark |
|---|---|---|---|
| Application function block alarm detection | 10000 (Factory setting) | Application function blocks alarm Id 451 ("Length to cut invalid") will be: `q_diAlrmId` = 10000 + 451 = 10451 | If several application libraries are used in the same application, this offset can be modified with the global variable: `GC_MotionLib_ErrorOffset` |
| Other alarms (SoftMotion, Motion Controller…) | No offset | SoftMotion alarm Id 12 ("The position limits of the axis have been exceeded") will be: `q_diAlrmId` = 12 | - |

### Notification

When an alarm is detected:

- the slave is stopped immediately with the deceleration ramp `lrDecStop`. The master and slave synchronization is lost.
- `q_xAlrm` is set to TRUE
- `q_diAlrmId` display the notification number according to the table below
- detected alarms have to be reset with a rising edge on the input `i_xAlrmRst`

**NOTE:** Output `q_diAlrmId` only displays the first detected alarm. Therefore a new problem is reported only if the last active problem has been reset.

| Detected AlrmId | Notification |
|---|---|
| 14 | Slave axis state is not in standstill at execution.<br>Enable slave axis with MC_Power. |
| 50 | Invalid capture input (parameter `diCptrNb`).<br>Verify parameter.<br>Range: 0 ≤ `diCptrNb` ≤ 2 |
| 51 | Invalid operating mode (parameter `diOpMode`).<br>Verify parameter.<br>Range: 0 ≤ `diOpMode` ≤ 2 |
| 52 | Invalid start mode (input `i_uiStrtMode`)<br>Verify input.<br>Range: 0 ≤ `i_uiStrtMode` ≤ 5 |

| Detected AlrmId | Notification |
|---|---|
| 53 | Master moved at warm start.<br>Start modes 1 to 5 (`i_uiStrtMode` = 1 to 5) are only possible if the master is stopped.<br>Stop the master or switch to start mode 0. |
| 55 | Slave is outside warm start window configured with parameter `lrWsWdow`. Start modes 2 to 5 (`i_uiStrtMode` = 2 to 5) are not possible. |
| 56 | Application function blocks runs on wrong controller.<br>This application function block requires a Motion Controller. |
| 129 | Invalid deceleration ramp (parameter `lrDecStop`).<br>Verify parameter.<br>Range: 0 ≤ `lrDecStop` |
| 130 | Invalid warm start window (parameter `lrWsWdow`).<br>Verify parameter.<br>Range: 0 ≤ `lrWsWdow` |
| 131 | Invalid velocity (parameter `lrVel`).<br>Verify parameter.<br>Range: 0 ≤ `lrVel` |
| 132 | Invalid acceleration / deceleration (parameter `lrAccDec`)<br>Verify parameter.<br>Range: 0 ≤ `lrAccDec` |
| 451 | Invalid length to cut (input `i_lrLentoCut`).<br>Verify input.<br>Range: `lrLentoCutMin` ≤ `i_lrLentoCut`<br>Operating mode 0: `lrLentoCutMin` = `lrSyncPosEnd`<br>Operating mode 1: `lrLentoCutMin` = `lrSyncPosEnd` + `lrTpWdow/2`<br>Operating mode 2:<br>● For Rotary Knife: `lrLentoCutMin` = `lrSyncPosEnd` + ABS (((`lrPeri - lrPosRest`) / `lrM`) * 1.875)<br>● For Flying Shear: `lrLentoCutMin` = `lrSyncPosEnd` + ABS ((`lrPosRest` / `lrM`) * 1.875) |
| 452 | Invalid synchronization end position (parameter `lrSyncPosEnd`).<br>Verify parameter.<br>Range: 0 ≤ `lrSyncPosEnd` |
| 453 | Invalid TP distance (parameter `lrTpDistToSypt`).<br>Verify parameter.<br>Range: `lrTpDistToSyptMin` ≤ `lrTpDistToSypt`<br>● For Rotary Knife:<br>`lrTpDistToSyptMin` = ABS (((`lrPeri - lrPosRest`) / `lrM`) * 1.875) + `lrTpWdow/2`<br>● For Flying Shear<br>`lrTpDistToSyptMin` = ABS ((`lrPosRest` / `lrM`) * 1.875) + `lrTpWdow/2` |

| Detected AlrmId | Notification |
|---|---|
| 454 | Invalid TP window (parameter lrTpWdow).<br>Verify parameter.<br>Range: lrTpWdow > ((lrPerim-lrPosRest)/lrm)*1,875 |
| 455 | There are more than 15 products between the position capture sensor and the knife, the position buffer is full. Reduce the distance between the sensor and the knife. |
| 456 | Invalid RM (slope of oblique cut, parameter lrM).<br>Verify parameter.<br>Range: 0 ≤ lrM ≤ 10 |
| 457 | Invalid CAM offset (parameter lrCamOfst).<br>Verify parameter.<br>Range: 0 ≤ lrCamOfst ≤ lrSyncPosEnd/2 |
| 458 | Invalid TP missed max. (parameter diTpMisdMax).<br>Verify parameter.<br>Range: 0 ≤ diTpMisdMax |
| 459 | Invalid offset start position (parameter lrPosOfstStrt).<br>Verify parameter.<br>Range: 0 < lrPosOfstStrt < lrPosOfstEnd |
| 460 | Invalid offset end position (parameter lrPosOfstEnd).<br>Verify parameter.<br>Range: lrPosOfstStrt < lrPosOfstEnd < lrSyncPosEnd |
| 461 | Invalid shift (input i_lrShift).<br>Verify input.<br>Range: - lrSyncPosEnd ≤ i_lrShift ≤ + lrSyncPosEnd |
| 462 | Invalid shift step size (input i_lrShiftStepSize).<br>Verify input.<br>Range:<br>Operating mode 0: 0 ≤ i_lrShiftStepSize < (i_lrLentoCut-lrSyncPosEnd/lrM)*0.9)<br>Operating mode 1: 0 ≤ i_lrShiftStepSize < (i_lrLentoCut - lrSyncPosEnd/lrM - lrTpWdow/2)*0.9)<br>Operating mode 1: 0 ≤ i_lrShiftStepSize |
| 463 | Number of missed marks (out of the TP window). Exceeded the maximum value defined by parameter diTpMisdMax (q_diTpAlrmCnt > diTpMisdMax)<br>Only for operating mode 1.<br>Verify the product / verify the position sensor / increase parameter diTpMisdMax |
| 465<br>(for Rotary Knife only) | Invalid knife perimeter (input lrPeri).<br>Verify input.<br>Range: lrPeri ≥ 0 |

| Detected AlrmId | Notification |
|---|---|
| 466 | Invalid rest position (parameter `lrPosRest`).<br>Verify parameter.<br>Range:<br>● For Rotary Knife: `lrSyncPosEnd` ≤ `lrPosRest` ≤ `lrPeri`<br>● For Flying Shear: `lrPosRest` ≤ `0` |
| 467<br>(for Rotary Knife only) | Invalid Rotary Knife slave axis configuration. Rotary Knife slave axis must be configured as a modulo axis. |
| 468<br>(for Flying Shear only) | Invalid Flying Shear slave axis configuration. Flying Shear slave axis must be configured as a linear axis, not as a modulo axis |
| 469<br>(for Rotary Knife only) | Invalid Rotary Knife slave axis configuration. Rotary Knife slave axis must be configured as a modulo axis with a modulo higher than 2 times the knife perimeter (Slave modulo ≥ 2*`lrPeri`). |
| 470 | Invalid master axis configuration. Master axis has to be configured as a modulo axis, with a modulo higher than maximum possible length to cut + TP distance to the synchronization point. (`> max i_lrLentoCut + lrTpDistToSypt`). |
| 471 | The function block was interrupted by a PLCopen function block. |

# Part VII
## FlyingShear

# Chapter 10
# `FlyingShear_Motion`: Synchronization of 2 Linear Axis to Perform On The Fly Operations

## What Is in This Chapter?

This chapter contains the following sections:

# Section 10.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Why Use the `FlyingShear_Motion` Function Block?

The `FlyingShear_Motion` function block controls a machine that performs an operation, on the fly, on a moving part.

Typical operations can include:
- Cutting
- Clamping
- Stamping
- Marking

## Solution with the `FlyingShear_Motion` Function Block

The Flying Shear function is required for moving the operational axis to synchronize it with the forward motion of the part.

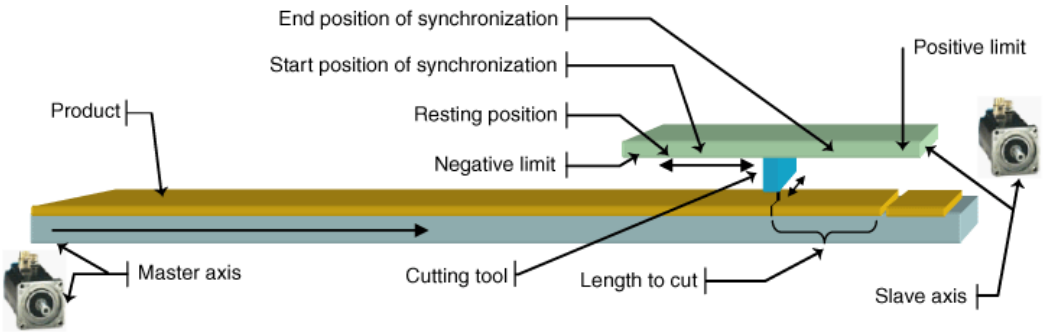
This introduces the concept of master and slave axes.
- Master: moves the part forward
- Slave: performs the operation

**NOTE:** Although the Flying Shear function can be used for many different applications, the examples shown in this document refer to a typical cutting application.

# Machine Overview

## Machine View

The figure below gives an example of a Flying Shear application. Both master and slave axes are linear axis types.




## Note

Please also refer to:

- Master Axis *(see page 266)*
- Slave Axis *(see page 267)*
- Cutting Tool *(see page 267)*
- Capture Position of the Product *(see page 267)*

# Section 10.2
# Architecture

## Hardware and Software Architecture

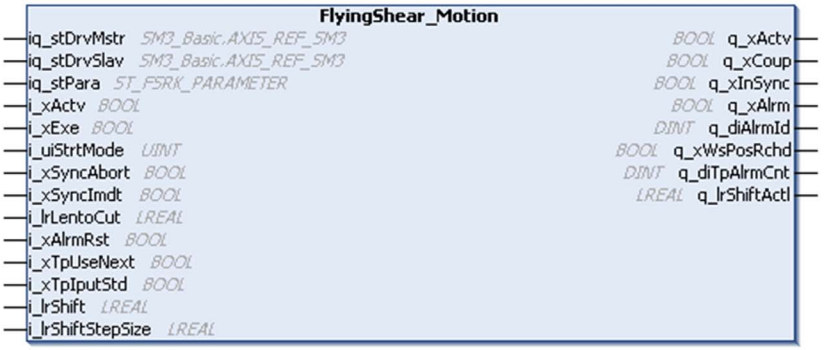
### Note

For the hardware and software architecture please refer to the Architecture chapter *(see page 265)*.

# Section 10.3
## Function Block Description

## `FlyingShear_Motion` Function Block

### Pin Diagram



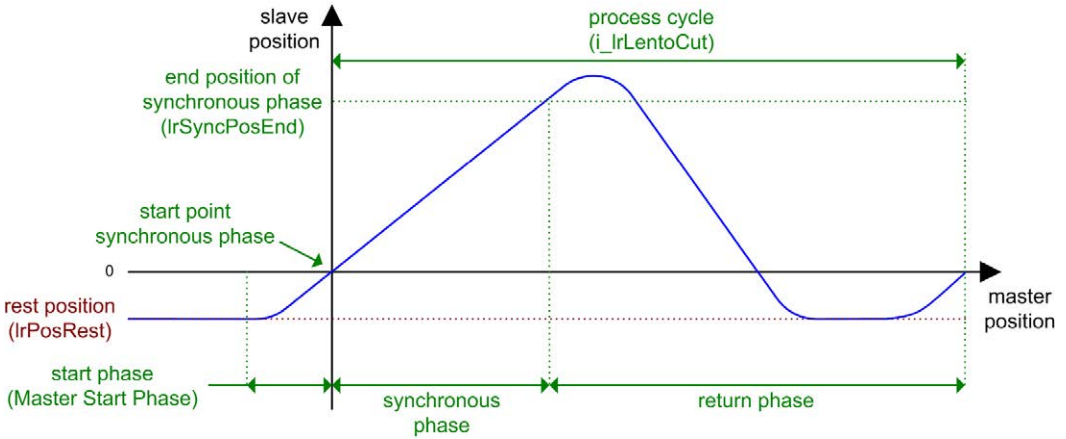

### Profiles of Master/Slave Positions

The curves (see case 1 and 2 below) show the position profile of the slave axis in relation to the master during the execution of the Flying Shear function.

A Flying Shear cycle consists of 3 phases:

| Phase | Description |
|---|---|
| Start Phase | The slave accelerates up to the master velocity to be synchronized at the beginning of the synchronous phase.<br>● The start phase distance on the slave is equal to:<br>Slave Start Phase Distance = ABS (`lrPosRest`)<br>● The start phase distance on the master can be calculated with the following formula:<br>Master Start Phase Distance = (ABS (`lrPosRest`) / `lrM`) * 1.875 |
| Synchronous Phase | The cut (or the operation) is performed. The synchronous phase always starts at position 0 of the slave. |
| Return Phase | The slave returns and makes a half-turn before it begins a new synchronous phase.<br>According to the length to be cut and the length of the synchronous phase, 2 cases are distinguished. |

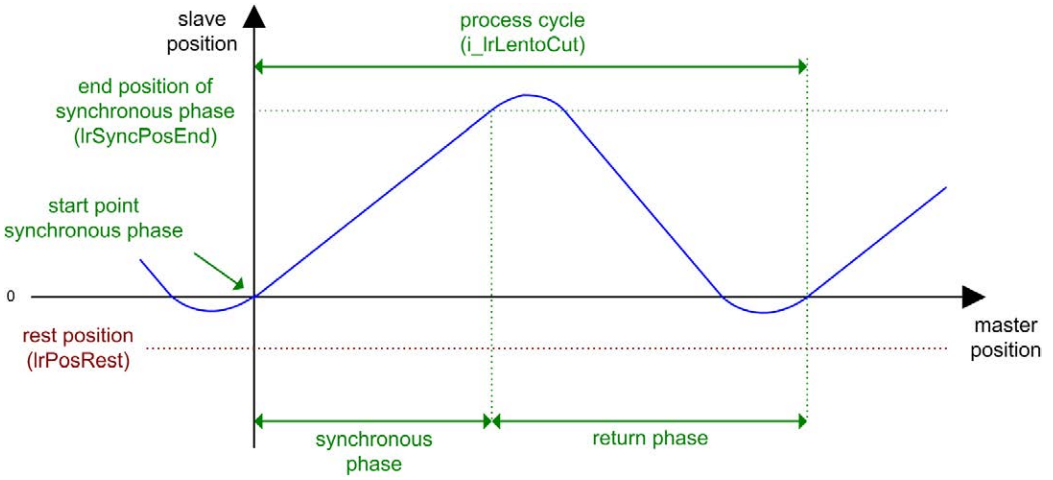
Case 1: return to resting position (long cut)




When the ratio `i_lrLentoCut` / `lrSyncPosEnd` is > 3, the slave returns, crosses its position of origin (position 0) and stops at its resting position (`lrPosRest`) before it begins a new cycle.

The duration of the stop depends on the length to cut (`i_lrLentoCut`) and the speed of the master.

Case 2: sequence of cuts (short cut)




When the ratio `i_lrLentoCut` / `lrSyncPosEnd` is < 3, the slave returns, crosses its position of origin and begins a new cycle immediately.

The distance covered by the master during this cycle corresponds to the length to cut (`i_lrLentoCut`).

# Section 10.4
# Pin Description

## Pin Description

### Notes

For the input pin description, please refer to the Input Pin Description *(see page 278)* of `RotaryKnife_Motion` function block.

For the output pin description, please refer to the Output Pin Description *(see page 281)* of `RotaryKnife_Motion` function block.

For input/output pin description, please refer to the Input/Output Pin Description *(see page 282)* of `RotaryKnife_Motion` function block.

For structured parameter description, please refer to the Structured Parameter *(see page 283)* of `RotaryKnife_Motion` function block.

# Section 10.5
# Operating Modes

## Operating Modes

### Note

For the description of operating modes, please refer to the Operating Modes chapter *(see page 288)*.

# Section 10.6
# Start, Stop and Starting Modes

## Start, Stop and Starting Modes

### Note

For the descriptions of start, stop and starting modes, refer to the Start, Stop and Starting Modes chapter *(see page 299)*.

# Section 10.7
## Special Modes

### What Is in This Section?

This section contains the following topics:

## Special Modes

### Note

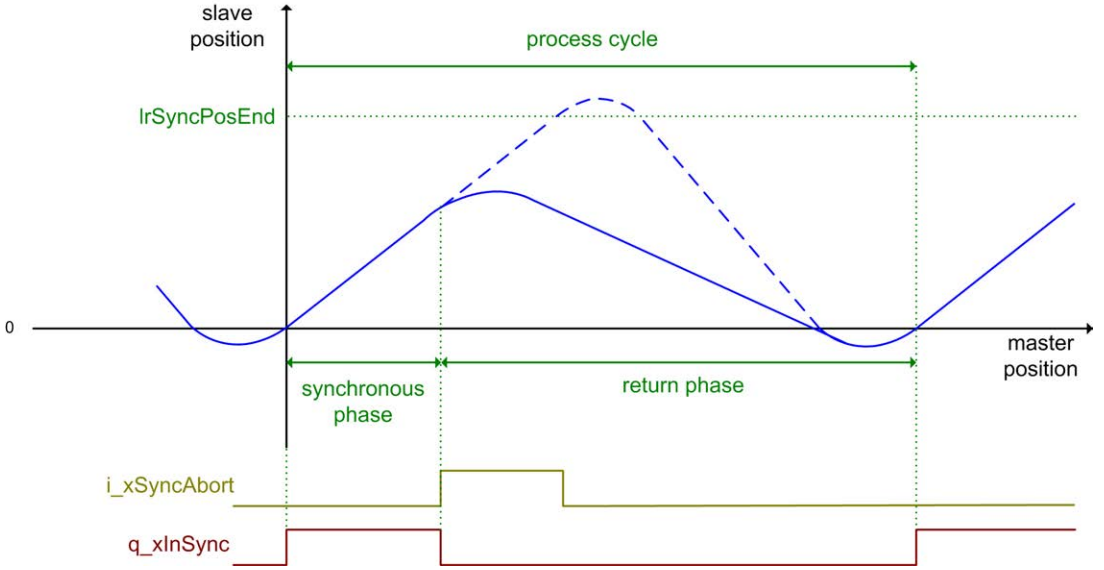
For the descriptions of other special modes, refer to the Special Modes chapter *(see page 315)*.

## Interruption of the Synchronous Phase

### Overview

With the input `i_xSyncAbort`, it is possible to prematurely interrupt the synchronous phase of the cycle. Depending upon the length of the return path, the slave returns to its resting position (`lrPosRest`) or begins a new cycle.

The input reacts on a rising edge and is considered only when the cycle is in the synchronous phase.



# Section 10.8
## Quick Reference Guide

### What Is in This Section?

This section contains the following topics:

# Visualization

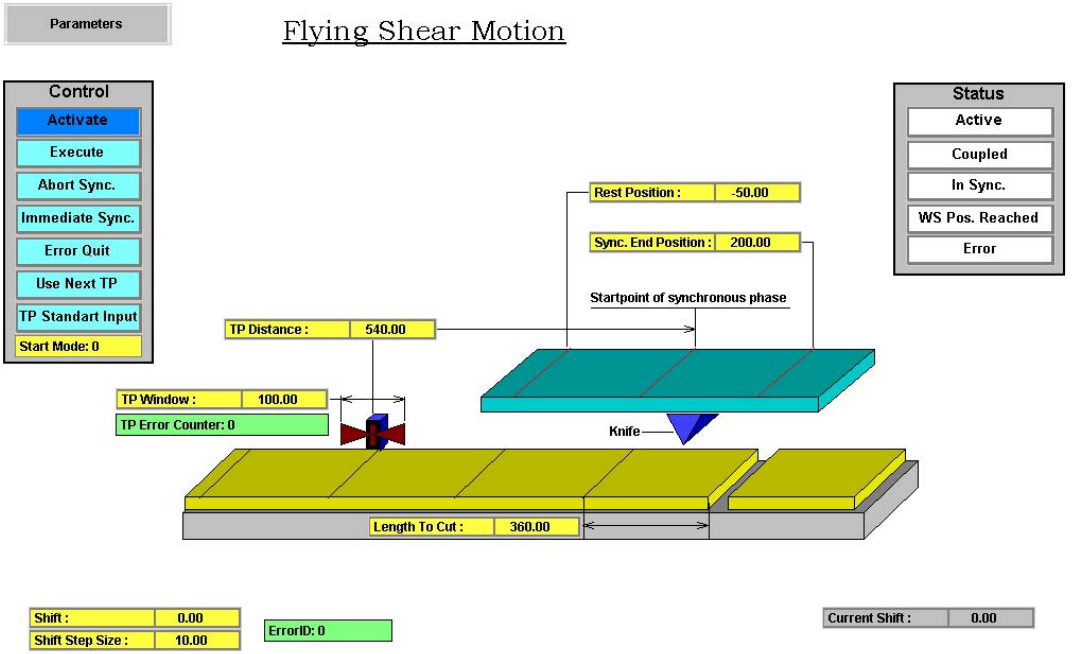
## Overview

Visualization screens are embedded in the FlyingShear library. They allow configuring and starting the application function block easily and quickly.

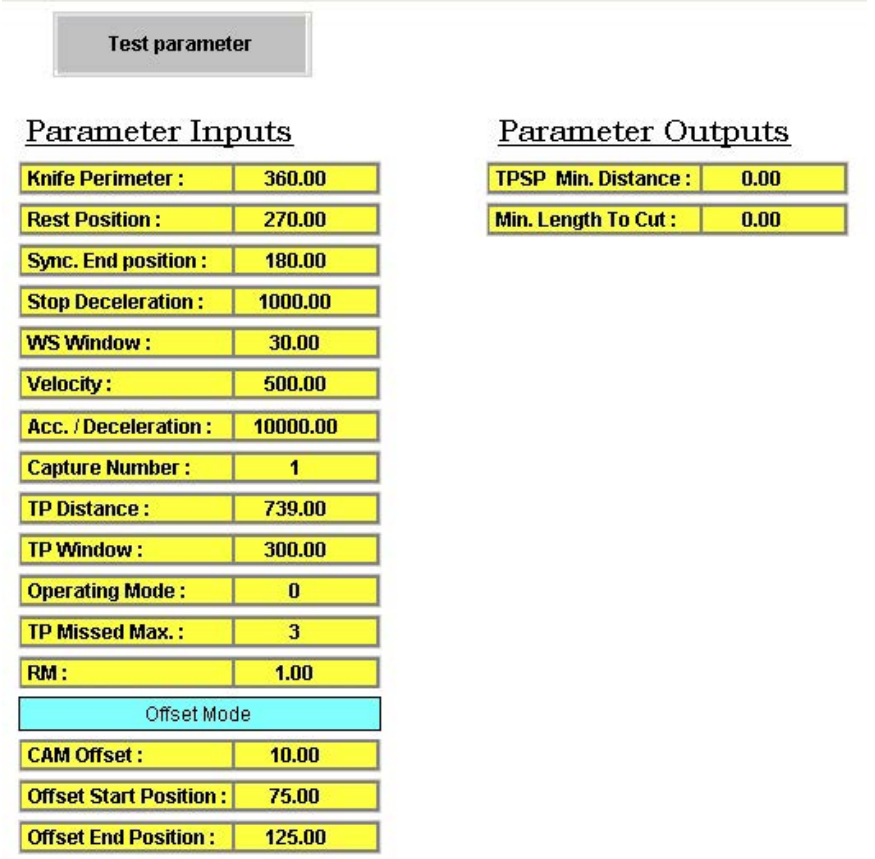
All the inputs/outputs and all the parameters are accessible.

For more information please refer to How to Integrate the Visualization into an Application *(see page 433)*.

Flying Shear template visualization



Flying Shear parameters visualization



# Troubleshooting

## Note

For troubleshooting, please refer to the Troubleshooting section *(see page 325)*.

# Part VIII
# GroupingUngrouping

## What Is in This Part?

This part contains the following chapters:

# Chapter 11
## Grouping - Ungrouping: Synchronization of Linear Axes to Organize Products on a Conveyor

### What Is in This Chapter?

This chapter contains the following sections:

# Section 11.1
## Functional and Machine Overview

### What Is in This Section?

This section contains the following topics:

# Functional Overview

## Usage of Grouping/Ungrouping Function

Grouping/Ungrouping involves the synchronization of several conveyors to sort and organize products in a predefined way (groups).

## Solution with the Grouping/Ungrouping Function

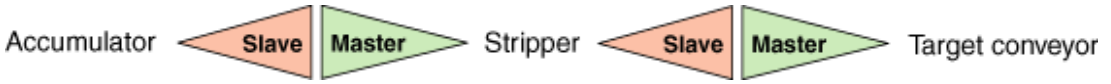
Appropriate function blocks allow a storing and a subsequent delivery of products. A desired distance between the products can be set and be modified at each cycle.

Those function blocks are:

- `GroupingAccumulator_Motion`
- `GroupingStripper_Motion`

## Function Description

The `Grouping` function blocks can be used separately or together. In combination they form an unsteady product flow into an evenly spaced product flow. This introduces the concept of master and slave.

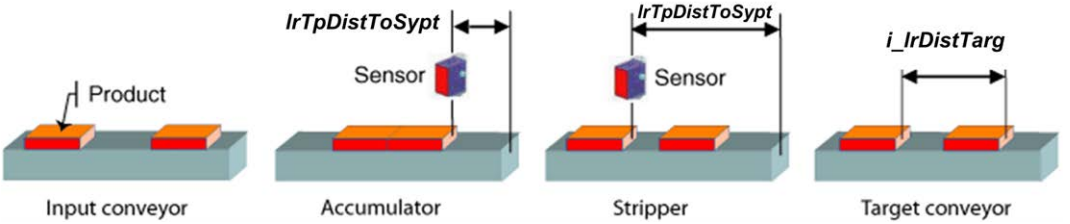



For further information, refer to Machine Overview *(see page 354)*.

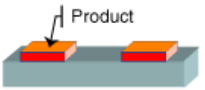
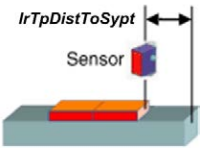
# Machine Overview

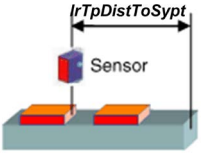
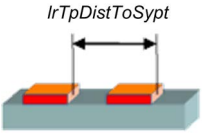
## Machine View

General Overview:

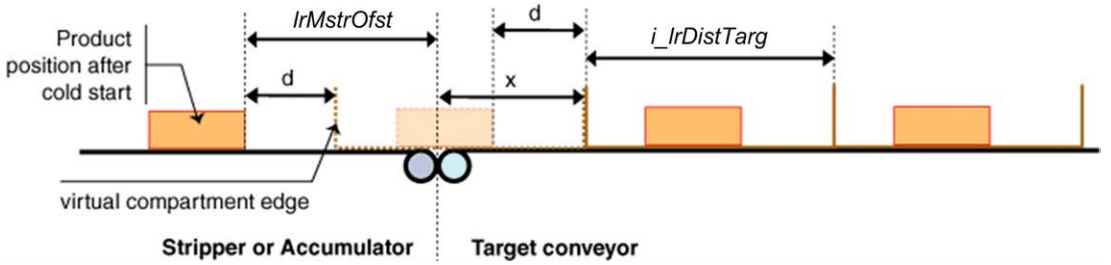



## Principle of Operation

| Conveyor | Figure | Product delivery to following conveyor | Description | Controlled by AFB |
|---|---|---|---|---|
| Input Conveyor |  | Input conveyor Velocity > Accumulator Velocity | The input conveyor delivers products with a random product flow.<br>The velocity of the input conveyor must be greater than the velocity of a following accumulation conveyor.<br>A too high back pressure however may push the products on the accumulation conveyor. | No |
| Accumulator Conveyor |  | Delivery dependent on `iOpMode`<br>Product Delivery *(see page 357)* | The accumulation conveyor generates a product jam. Ensure that no gaps occur as they are not managed by the function block.<br>Product delivery to the following conveyor can be set with `iOpMode` = 0..3. | `GroupingAccumulator_Motion` *(see page 368)* |

| Conveyor | Figure | Product delivery to following conveyor | Description | Controlled by AFB |
|---|---|---|---|---|
| Stripper Conveyor |  | During delivery:<br>Distance = `i_lrDistTarg`<br>Velocity = MasterVelocity<br>Product Delivery<br>*(see page 357)* | The stripper is the master of the accumulator.<br>At the start position of the stripper, the accumulator generates an evenly spaced product flow which is approximately the final target distance.<br>At the end the stripper has corrected these distances to the accurate target distance and delivers the products with a synchronous speed to the target conveyor. | `GroupingStripper_Motion`<br>*(see page 378)* |
| Target Conveyor |  | - | The target conveyor is the master of the stripper, or the accumulator if no stripper is used.<br>On this conveyor the products must have a defined distance, and/or position where compartments are used. | No |

## Compartments on the Target Conveyor




If compartments are used on the target conveyor, the position of this conveyor at the beginning of a movement is important for correct product placement. The parameter `lrMstrOfst` must be determined once before the first start-up.

Its value remains constant if with every restart of the system a reference movement is executed:

- After a system start-up, execute a reference movement of the master (target conveyor) to an arbitrary compartment edge. The position of the target conveyor is then 0.

| NOTICE |
| --- |
| **INOPERABLE EQUIPMENT**<br>For a proper referencing in the controller configuration, set the modulo value of the target conveyor to the value of `i_lrDistTarg`.<br>**Failure to follow these instructions can result in equipment damage.** |

- Measure the distance x as the distance from the start of delivery position to the first compartment edge on the target conveyor (see figure *Compartments on the Target Conveyor*).

Calculate the `lrMstrOfst` as a function of x and d:

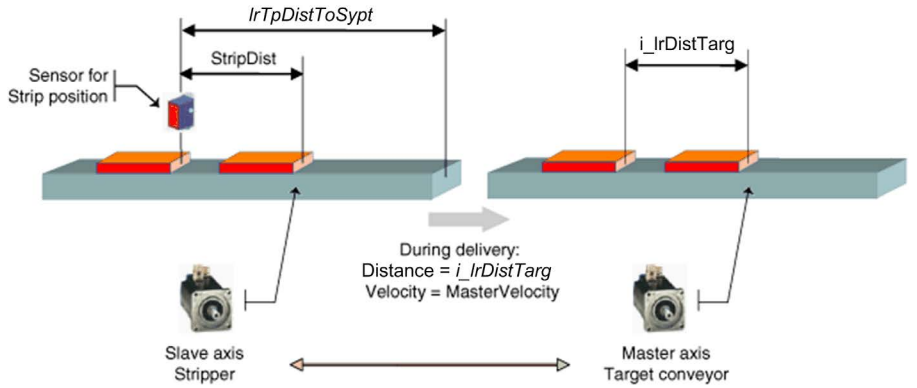
| If... | Then... |
| --- | --- |
| x > d | `rMsterOfst = i_lrDistTarg - x + d` |
| x < d | `rMsterOfst = d - x` |
| d = desired distance of the product from the compartment edge | |

When using compartments, `xCmpt` must be set TRUE. However this parameter is only evaluated with a cold start. Then the virtual encoder is set to the actual position of the target conveyor minus `lrMstrOfst`.

For an accumulator (if no stripper is used), only `iOpMode` = 2 (periodic geared pulsed mode) is allowed.

For further information refer to Cold Start Accumulator *(see page 402)* and Cold Start Stripper *(see page 404)*.

## Product Delivery (Stripper)




Requirements:

- The stripping conveyor must deliver each product with an accurate distance (`i_lrDistTarg`) to its predecessor.
- For the duration of the delivery, the stripper and the target conveyor must have exactly the same velocity (Velocity = MasterVelocity).

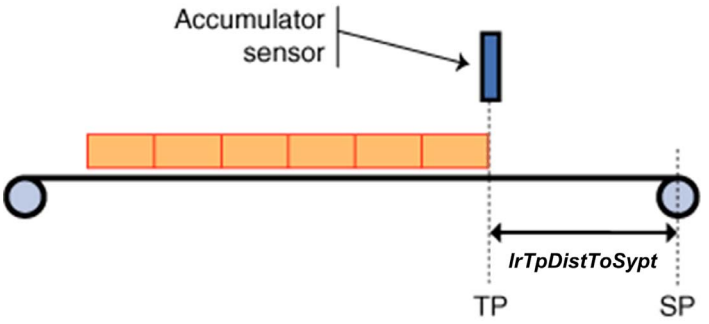
To fulfill both requirements even high-dynamically, the accumulator conveyor must deliver the products to the stripper conveyor with a distance `i_lrDistTarg`.

For this you can select between 4 different delivery modes (`iOpMode` = 0..3).

The stripper only requires the position information (where compartments are on the target conveyor) and velocity information of the target conveyor (using a cam). Therefore, the master can be an encoder.

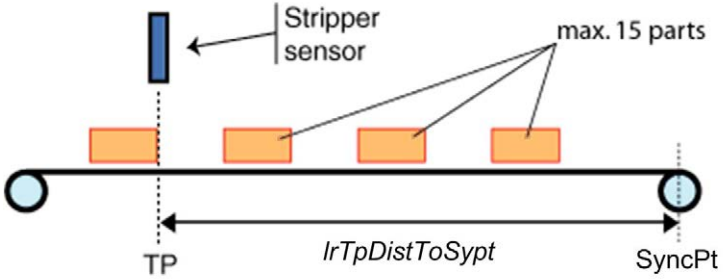
For further information, refer to Curve Diagrams *(see page 378)*, Input Pin Description *(see page 382)*, Output Pin Description *(see page 384)* and Parameter *(see page 388)*.

### Sensor for Product Detection (Accumulator, optional)




The sensor becomes active with the arrival of the first product. Therefore, this sensor is relevant only for a cold start. The parameter `lrTpDistToSypt` sets the distance between the touch probe sensor (TP) and the point of delivery (SyncPt) from which a product is started to be delivered to the following stripper and/or target conveyor. The delivery mode can be selected with `iOpMode`.

### Sensor for Product Detection (Stripper)




The sensor becomes active with every arrival of a product. The parameter `lrTpDistToSypt` sets the distance between the sensor (TP) and the point of delivery (SyncPt) from which a product is started to be delivered to the following target conveyor.

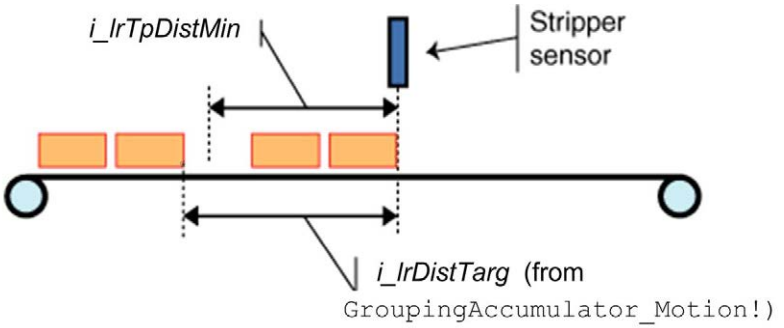
During the delivery the stripper conveyor moves synchronously with the target conveyor *(see page 420)*.

Up to 15 parts may be buffered between TP and SyncPt.

When the distance between the signals is greater than `i_lrTpDistMin`, the product detection is accepted.

For the compartments on the target conveyor: if the signal distance is greater than `i_lrDistTarg` + `i_lrTpDistMin`, a compartment is left out and the product detected is delivered to a next compartment.

By applying `i_lrTpDistMin` with the TP detection, the following type of grouping can be accomplished:



# Section 11.2
## Architecture

### What Is in This Section?

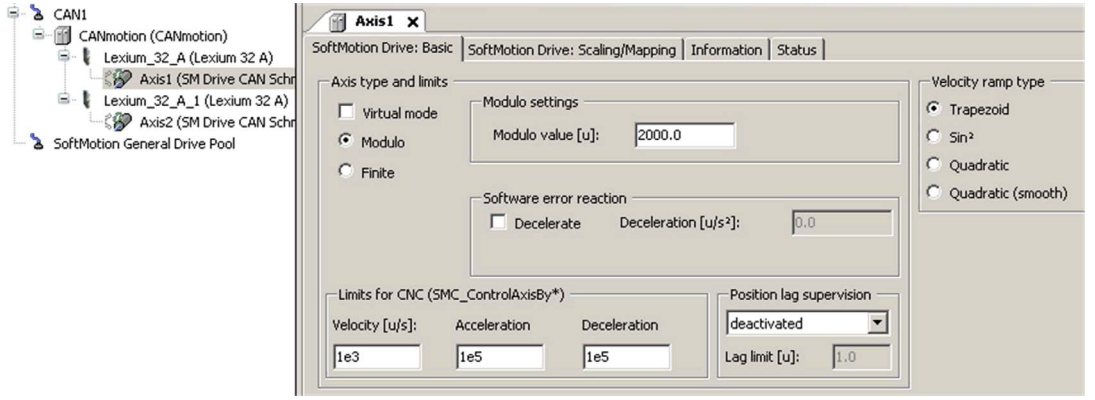
This section contains the following topics:

# Software Architecture

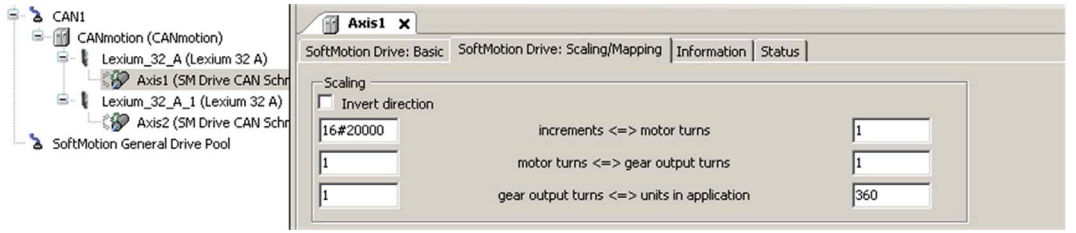
## Configuration of the Axes with the Motion Controller

The master axis must be configured as a rotary axis.

The modulo value must be set higher than the application's maximum `i_lrDistTarg`. The modulo value is a configuration parameter and cannot be modified by the application




All inputs/outputs for position, velocity and acceleration in user units are defined in the controller configuration.




Example:

Gear ratio = 1: 1

1 motor revolution = 360 mm

### Configuration of the Slave Axis with the Drive

In addition to the communication parameters (addresses on the bus, speed transmission), the following parameter adjustments should be made to optimize the response of the servo drive that controls the slave axis:

- increase the maximum slopes of acceleration and deceleration.
- set the parameter `N_ref_Filter` to OFF.
- set various loop parameters. They include proportional gain speed loop, integral time speed loop and the proportional gain position.

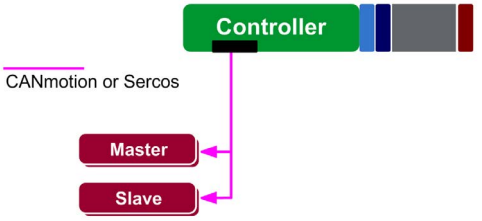
# Hardware Architecture

## Hardware Architecture Overview

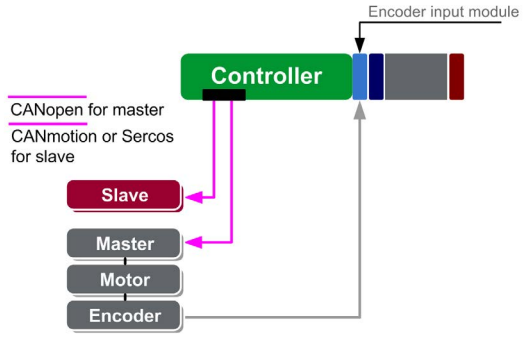
The target conveyor (master axis) moves the products forward continuously. It is an infinite axis.

The controller of this axis can be of any type (servo drive, variable speed drive, etc.). The movement of this axis is not controlled by a Grouping function block.

Both master and slave may be axes on CANmotion or Sercos:




If the master axis is not a servo drive on the same CANmotion or Sercos as the slave, then the position of the master is determined by an encoder signal connected to the input of the Motion Controller:




## Sensor Wiring (Accumulator)

The sensor is evaluated at a cold start and only once with the arrival of the first product.

Sensor wiring (accumulator) is only possible over a standard input by wiring one of the digital inputs to `i_xTpIputStd`.

**NOTE:** The accuracy depends on the master velocity and the program cycle time and/or the cycle time of the Motion Controller.

## Sensor Wiring (Stripper)

Perform the sensor wiring (stripper) over a TP input as follows.

A TP input is used where a short acquisition time is required. The TP position is captured via CANmotion or Sercos. The attainable accuracy is higher than with the capturing over a standard input (see below).

- Set the parameter `xTpMode` to TRUE.
- Connect the sensor to one of the capture inputs (`CAP1` or `CAP2`) of the Lexium servo drive.

**NOTE:** The TP inputs of the Controller cannot be used as they can only trigger encoder positions (master) and not drive positions.

- Set the parameter `iCptrNb` to 0 or 1.
- The input `i_xTpIputStd` is unused.

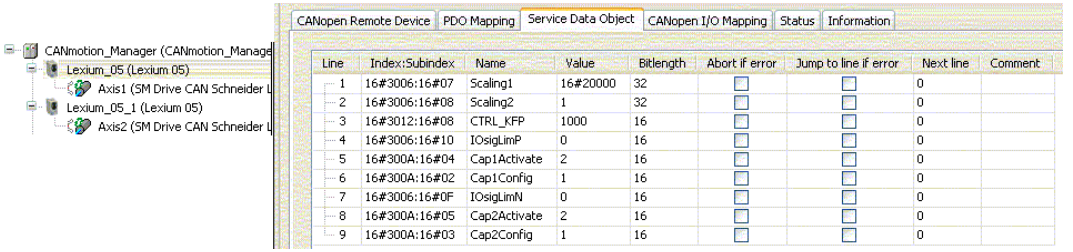
The edge to be evaluated must configured directly in the Lexium drive.

Using for Sercos:

- the `FB_SercosWriteServiceDataAsync`

Using for CANmotion either:

- the `SMC3_CAN_WriteParameter`
- the CAN Drive list in the Controller configuration




- Configure the parameter `Cap1Config` for capture unit 1, or `Cap2Config` for capture unit 2.

| Parameter Name<br>HMI menu | Description | Unit<br>Minimum value<br>Default value<br>Maximum value | Data type<br>R/W<br>persistent<br>Expert | Parameter address via fieldbus |
|---|---|---|---|---|
| Cap1Config | Configuration of capture unit 1 (240) | -<br>0<br>0<br>1 | UINT16<br>UINT16<br>R/W<br>-<br>- | CANopen 300A:$2_h$<br>Modbus 2564 |
| - | **0 / 1->0**: position capture with 1->0 switch | | | |
| - | **1 / 0->1**: position capture with 0->1 switch | | | |

The Sercos identification number (IDN) for `Cap1Config` is P-O-3010.0.2

| Parameter Name<br>HMI menu | Description | Unit<br>Minimum value<br>Default value<br>Maximum value | Data type<br>R/W<br>persistent<br>Expert | Parameter address<br>via fieldbus |
|---|---|---|---|---|
| Cap2Config<br><br>-<br><br>- | Configuration of capture unit 2 (240)<br><br>**0 / 1->0**: position capture with 1->0 switch<br>**1 / 0->1**: position capture with 0->1 switch | -<br>0<br>0<br>1 | UINT16<br>UINT16<br>R/W<br>-<br>- | CANopen 300A:3$_h$<br>Modbus 2566 |

The Sercos identification number (IDN) for `Cap2Config` is P-O-3010.0.3

**NOTE:** This parameter cannot be saved in the Lexium persistent Memory (EEPROM).

Perform the sensor wiring (stripper) over a standard input as follows:

Alternatively to a TP input, an digital input of the controller may be used. However, in this case the attainable accuracy depends on the master velocity and the program cycle time and/or the cycle time of the Motion Controller. A captured position will be evaluated by the function block.

- sets the parameter `xTpMode` to FALSE.
- wire 1 of the digital inputs to `i_xTpIputStd`.

# Chapter 12
# GroupingAccumulator_Motion

## What Is in This Chapter?

This chapter contains the following sections:

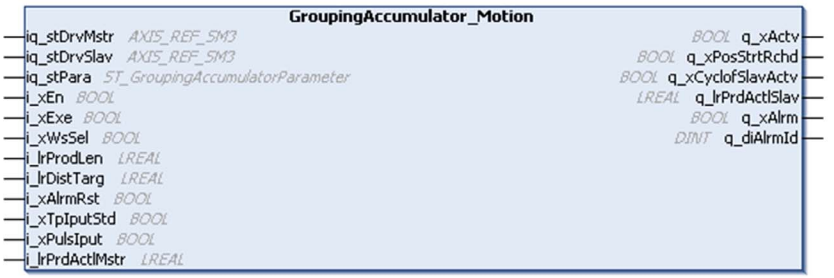
# Section 12.1
## Function Block Description

## GroupingAccumulator_Motion Function Block

### Pin Diagram

The Accumulator function block is made up of:

- Inputs and outputs (they can be changed on the fly.)
- Parameters of inputs / outputs



# Section 12.2
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Enables the function block<br>● With a rising edge, the parameters are checked for validity and consistency.<br>● A falling edge stops a movement with deceleration lrDecEmgy.<br>● The parameters defined in iq_stPara are checked. Incorrect or inconsistent input parameters affect the outputs q_xAlrm and q_diAlrmId.<br>FALSE: The current movement stops with lrDecEmgy<br>● The output q_xActv stays TRUE until stopping is finished.<br>● While q_xActv = TRUE, the function block must be called cyclically.<br>Factory setting: FALSE |
| i_xExe | BOOL | TRUE: The movement starts (steady signal required)·<br>● A falling edge stops a movement with deceleration lrAccDecStrt.<br>Factory setting: FALSE |
| i_xWsSel | BOOL | FALSE: Cold start<br>TRUE: Warm start<br>Different cold and warm start modes are available, see Starting Modes *(see page 402)*<br>Factory setting: FALSE |
| i_lrProdLen | LREAL | Length of product<br>● Must be << slave modulo (configured in Controller configuration)·<br>● Must be ≤ i_lrDistTarg, otherwise an interruption is generated.<br>Factory setting: 0 |

| Input | Data Type | Description |
|---|---|---|
| i_lrDistTarg | LREAL | Desired distance for products (length of product plus gap) on master (stripper or target conveyor) of accumulator.<br>$\text{Accumulator velocity} = \frac{\text{i\_lrProdLen}}{\text{i\_lrDistTarg}} \bullet \text{Target velocity}$<br>● Must be << slave modulo (configuration in Controller configuration).<br>● Must be >= i_lrProdLen, otherwise an interruption is generated<br>Factory setting: 0<br>Range: > 0 |
| i_xAlrmRst | BOOL | TRUE: Detected alarm is acknowledged, function block leaves alarm state.<br>Factory setting: FALSE |
| i_xTpIputStd | BOOL | For product detection (first product edge at cold start), see Sensor Wiring *(see page 363)*<br>Factory setting: FALSE |
| i_xPulsIput | BOOL | Only with iOpMode = 1 or 3 pulse for product delivery.<br>Factory setting: FALSE |
| i_lrPrdActlMstr | LREAL | The position period of the master has to be set here as input. The master cannot make any position jumps (because of modulo or set pos) different from this input or the slave can not run.<br>Range: > 0 |

# Output Pin Description

## Output Pin Description

### Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xActv | BOOL | TRUE: Function block is active. |
| q_xPosStrtRchd | BOOL | TRUE: Cold / warm start position reached.<br>Reset with function block disabled (i_xEn = FALSE) |
| q_xCyclofSlavActv | BOOL | Only with iOpMode = 1 or 3<br>TRUE: While a slave cycle is in progress.<br>FALSE: When slave is ready to start a new cycle. |
| q_lrPrdActlSlav | LREAL | On this output the actual period of the CAM curve is given.<br>If the drive is a master for another drive, this signal has to be given to the control function block of the slave. |
| q_xAlrm | BOOL | TRUE: Not OK |
| q_diAlrmId | DINT | Notification number:<br>Troubleshooting *(see page 413)*<br>Range: 0..99999 |

### Troubleshooting

There are 2 types of notifications given on the output of the function block.

The first types of notifications come from internal (SoftMotion (SM3 library)) function blocks. These notifications are given directly to the output q_diAlrmId. For these notifications see the notifications list of the SoftMotion (SM3 library) function blocks.

The second types of notifications are generated by the function block itself. These notifications are added to the global constant GC_MotionLib_ErrorOffset (standard 1000).

| Detected Alarm ID (added to GC_MotionLib_ErrorOffset) | Notification |
|---|---|
| 10 | controller axis detected alarm |
| 14 | controlled axis in wrong state |
| 15 | controlled axis is not a modulo axis |
| 50 | invalid capture input |
| 51 | invalid iOpMode |
| 52 | invalid iWsMode |
| 53 | master moved during start |
| 54 | warm start not possible |
| 55 | position outside warm start window |

| Detected Alarm ID (added to GC_MotionLib_ErrorOffset) | Notification |
|---|---|
| 129 | invalid lrDecEmgy |
| 130 | invalid lrWsWdow |
| 453 | lrTpDistToSypt < 0 |
| 455 | FIFO full - too many products detected |
| 501 | FIFO empty - no products detected |
| 502 | no TP found |
| 504 | invalid i_lrSyncLen |
| 505 | invalid i_lrTpDistMin |
| 506 | invalid i_lrProdLen |
| 507 | invalid i_lrDistTarg |
| 508 | i_lrProdLen greater i_lrDistTarg |
| 509 | i_lrSyncLen greater i_lrDistTarg |
| 510 | i_lrSyncLen greater i_lrDistDflt |
| 511 | invalid i_lrDistDflt |
| 512 | lrTpDistToSypt smaller i_lrDistDflt plus i_lrTpDistMin |
| 513 | i_lrTpDistMin smaller i_lrSyncLen |
| 514 | invalid iCsMode |
| 515 | invalid lrCsDistMax |
| 516 | invalid lrVelStrt |
| 517 | invalid lrAccDecStrt |
| 519 | invalid lrVelOpMode1 |
| 520 | invalid lrAccOpMode1 |
| 521 | invalid lrDecOpMode1 |
| 522 | invalid lrPulsSyncPhas |
| 523 | invalid Slop |
| 524 | invalid lrVelStsl |
| 525 | invalid lrCorrMax |
| 526 | invalid lrMstrOfst |
| 527 | invalid warm start data |

# Input/Output Pin Description

## Input/Output Pin Description

### Input/Output Pin Description

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stDrvMstr` | `AXIS_REF_SM3` | Master axis (stripper or target conveyor) reference |
| `iq_stDrvSlav` | `AXIS_REF_SM3` | Slave axis (accumulator) reference |
| `iq_stPara` | STRUCT `ST_GroupingAccumulatorParameter` | Block parameters<br>For further information, refer to Structured Parameters *(see page 375)*. |

## Structured Parameter

### iq_stPara

| Structure Parameter | Data Type | Description |
|---|---|---|
| iCsMode | INT | Cold start mode<br>0: first product is moved to delivery position<br>1: dimension setting (no movement)<br>Cold Start Mode *(see page 402)*<br>Factory setting: 0 |
| iWsMode | INT | Selecting warm start mode *(see page 406)*.<br>Factory setting: 0 |
| iOpMode | INT | 0: Geared mode *(see page 394)*<br>1: Triggered timed pulsed mode *(see page 396)*<br>2: Periodic geared pulsed mode *(see page 397)*<br>3: Triggered geared pulsed mode *(see page 398)*<br>Factory setting: 0 |
| lrCsDistMax | LREAL | Maximum position allowed for detecting the first product (i_xTpIputStd = TRUE)<br>= 0: no limitation<br>> 0: axis moves maximum this distance<br>Factory setting: 1000 |
| lrWsWdowFwd | LREAL | Only with iWsMode = 1 or 4.<br>Maximum position that the slave is allowed to move forward to return to the cam position.<br>Factory setting: 200 |
| lrWsWdowBack | LREAL | Only with iWsMode = 1 or 4.<br>Maximum position that the slave is allowed to move backwards in order to return to the cam position.<br>Factory setting: 10 |
| lrVelStrt | LREAL | Start velocity<br>Factory setting: 100 |
| lrAccDecStrt | LREAL | Start acceleration / deceleration (also for falling edge at i_xExe)<br>Factory setting: 1000 |
| lrTpDistToSypt | LREAL | Distance between TP (sensor) and SyncPt (start of delivery).<br>Must be greater than master modulo (i_lrDistTarg) + lrMstrOfst<br>Factory setting: 180 |
| lrDecEmgy | LREAL | Stop ramp for detected error stop or falling edge at i_xEn.<br>Factory setting: 10000 |

| Structure Parameter | Data Type | Description |
|---|---|---|
| lrVelOpMode1 | LREAL | Only with iOpMode = 1<br>Maximum velocity for moving to the cam position.<br>Factory setting: 360 |
| lrAccOpMode1 | LREAL | Only with iOpMode = 1.<br>Maximum acceleration for moving to the cam position.<br>Factory setting: 1000 |
| lrDecOpMode1 | LREAL | Only with iOpMode = 1.<br>Maximum deceleration for moving to the cam position.<br>Factory setting: 1000 |
| lrSlop | LREAL | Only with iOpMode = 2 or 3.<br>Gear ratio between master and slave during the synchronous phase.<br>Examples:<br>1: velocity ratio between accumulator and stripper is 1:1.<br>2: velocity ratio between accumulator and stripper is 2:1 (accumulator delivers the products to the stripper faster than with 1:1).<br>Factory setting: 1 |
| lrPulsSyncPhas | LREAL | Only with iOpMode = 2 or 3.<br>Length of the synchronous phase in percentage of the product length.<br>Factory setting: 80<br>Range: 0..99% |
| lrMstrOfst | LREAL | Only with cold start and xCmpt = TRUE.<br>Product distance from a compartment edge.<br>Factory setting: 0 |
| xCmpt | BOOL | Only with cold start<br>TRUE: Target conveyor with compartments.<br>FALSE: Target conveyor without compartments.<br>Compartments on the Target Conveyor *(see page 355)*.<br>Factory setting: FALSE |
| lrVelStsl | LREAL | Velocity window in which the drive is considered to be standing still.<br>Factory setting: 10 |

# Chapter 13
# GroupingStripper_Motion

## What Is in This Chapter?

This chapter contains the following sections:

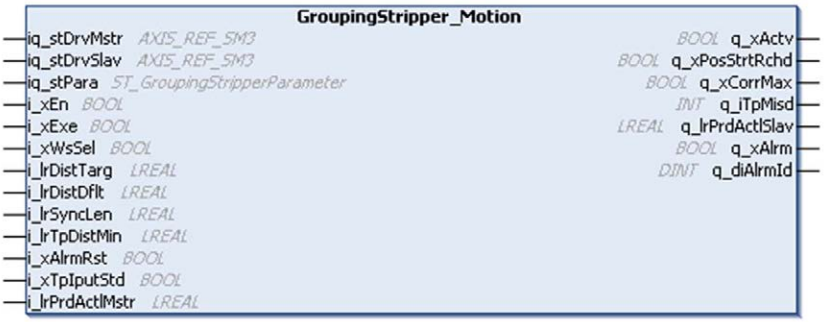
# Section 13.1
## Function Block Description

## GroupingStripper_Motion Function Block

### Pin Diagram

The Stripper function block is made up of:

- Inputs and outputs (they can be changed on the fly.)
- Parameters of inputs / outputs

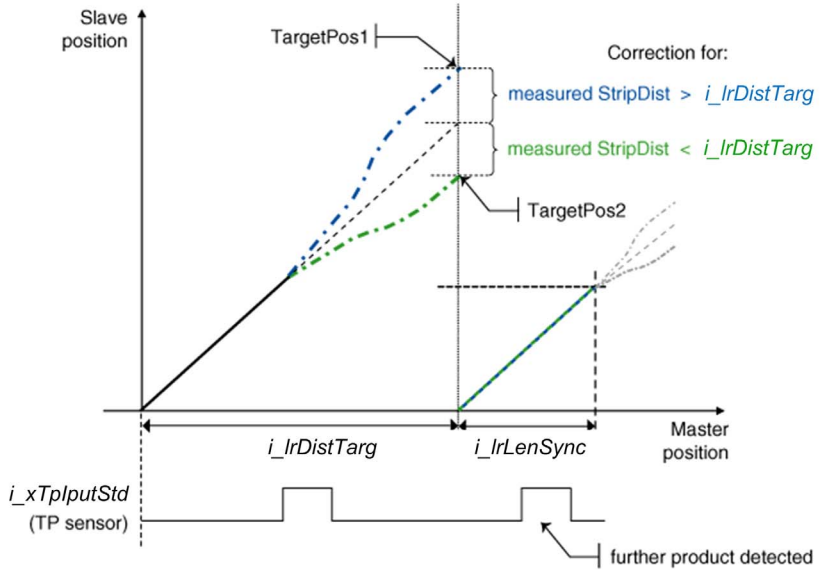



### Curve Diagrams

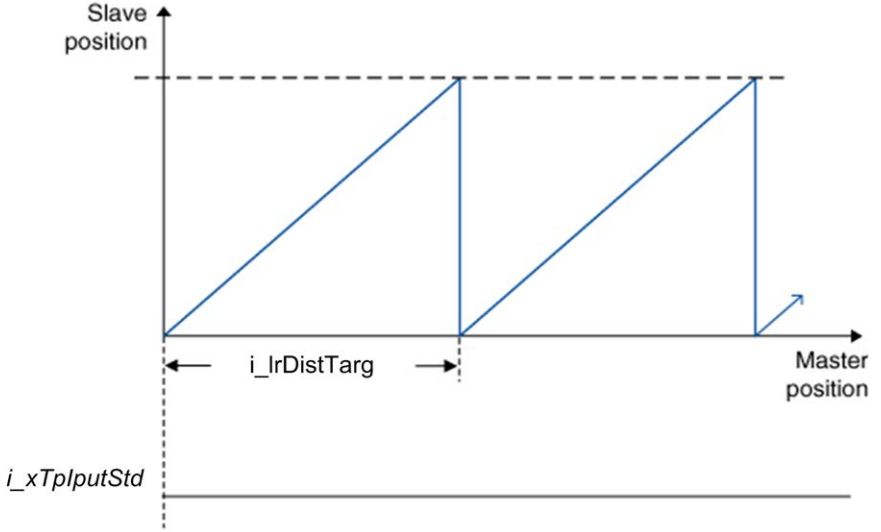
Trajectory for correcting the product distance:

The motion profile indicates how products are passed from the stripper to the target conveyor.

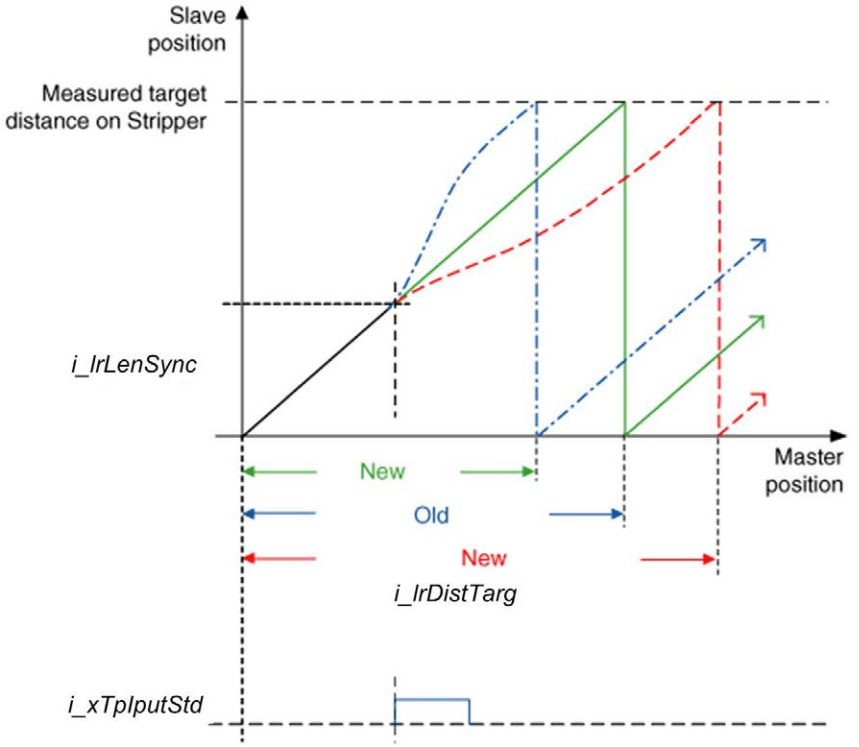
The distance of a product to its predecessor measured by the TP sensor of the stripper is corrected. This is done so that the distance is exactly = `i_lrDistTarg` at the position of delivery to the target conveyor. During the delivery (`i_lrSyncLen`) the stripper (slave) and the target conveyor (master) move synchronously (Velocity = MasterVelocity). The TP sensor may have detected another product (a maximum of 15 are possible) during this time.




Special case: No product on conveyor



Special case: Changing `i_lrDistTarg` on the fly



# Section 13.2
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

### Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Enables the function block<br>● With a rising edge, the parameters are checked for validity and consistency.<br>● A falling edge stops a movement with deceleration lrDecEmgy.<br>● The parameters defined in iq_stPara are checked. Incorrect or inconsistent input parameters will affect the outputs q_xAlrm and q_diAlrmId.<br>FALSE: The current movement stops with lrDecEmgy<br>● The output q_xActv stays TRUE until a stopping is finished.<br>● While q_xActv = TRUE, the function block must be called cyclically.<br>Factory setting: FALSE |
| i_xExe | BOOL | TRUE: The movement starts (steady signal required)<br>● A falling edge stops a movement with deceleration lrAccDecStrt.<br>FALSE: Stripper in cyclic (automatic) mode<br>● The stripper axis stops at the end of the synchronous phase (position > i_rLenSync).<br>● If i_xExe goes TRUE again within the same synchronous phase, the previous FALSE will be ignored.<br>Factory setting: FALSE |
| i_xWsSel | BOOL | FALSE: Cold start<br>TRUE: Warm start<br>Different cold and warm start modes are available, refer<br>Cold Start *(see page 404)* and Warm Start *(see page 406)*<br>Factory setting: FALSE |

| Input | Data Type | Description |
|---|---|---|
| i_lrDistTarg | LREAL | Desired distance for products (length of product plus gap) on master (stripper or target conveyor) of accumulator.<br>● Must be << slave modulo (configured in controller configuration).<br>Factory setting: 0<br>Range: > 0 |
| i_lrDistDflt | LREAL | Default distance on stripper. It should be equal to i_lrDistTarg of accumulator and is used if a part is missing.<br>Factory setting: 0<br>Range: > 0 |
| i_lrSyncLen | LREAL | Distance that the slave (stripper) moves synchronously with the master (target conveyor).<br>Factory setting: 0<br>Range: > 0 |
| i_lrTpDistMin | LREAL | Minimum distance between 2 edges at the TP input.<br>Factory setting: 0<br>Range: > 0 |
| i_xAlrmRst | BOOL | TRUE: Detected alarm is acknowledged, function block leaves alarm state.<br>Factory setting: FALSE |
| i_xTpIputStd | BOOL | Standard input is used for product detection if xTpMode = FALSE.<br>Sensor Wiring *(see page 364)*<br>Factory setting: FALSE |
| i_lrPrdActlMstr | LREAL | The position period of the master has to be set here as input. The master cannot make any position jumps (because of modulo or set pos) different from this input or the slave can not run.<br>Factory setting: 0<br>Range: > 0 |

# Output Pin Description

## Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xActv | BOOL | TRUE: Function block is active. |
| q_xPosStrtRchd | BOOL | TRUE: Cold / warm start position reached.<br>Reset with function block disabling (i_xEn = FALSE) |
| q_xCorrMax | BOOL | TRUE: The current product would require a correction > lrCorrMax.<br>Instead a backward correction is executed (=> empty place on the target conveyor) and this output q_xCorrMax is set. |
| q_iTpMisd | INT | Number of consecutive missing parts.<br>Range: 0...n |
| q_lrPrdActlSlav | LREAL | On this output the actual period of the CAM curve is given.<br>If the drive is a master for another drive, this signal has to be given to the control function block of the slave.<br>E.g. to the input i_lrPrdActlMstr of the Accumulator function block. |
| q_xAlrm | BOOL | TRUE: Not OK |
| q_diAlrmId | DINT | Notification number:<br>Notification *(see page 413)*<br>Range: 0...99999 |

## Troubleshooting

There are 2 types of notifications given on the output of the function block.

The first types of notifications are coming from internal used SoftMotion (SM3 library) function blocks. These notifications are given directly to the output `q_diAlrmId`. For these notifications see the notifications list of the SoftMotion (SM3 library) function blocks.

The second types of notifications are generated by the function block itself. These notifications are added to the global constant `GC_MotionLib_ErrorOffset` (standard 1000).

| Detected Alarm ID (added to `GC_MotionLib_ErrorOffset`) | Notification |
|---|---|
| 10 | controlled axis not working |
| 14 | controlled axis in wrong state |
| 15 | controlled axis is no modulo axis |
| 50 | invalid capture input |
| 51 | invalid `iOpMode` |
| 52 | invalid `iWsMode` |
| 53 | master moved during start |
| 54 | warm start not poosible |
| 55 | position outside warm start window |
| 129 | invalid `lrDecEmgy` |
| 130 | invalid `lrWsWdow` |
| 453 | `lrTpDistToSypt` < 0 |
| 455 | FIFO full - too much products detected |
| 501 | FIFO empty - no products detected |
| 502 | no TP found |
| 504 | invalid `i_lrSyncLen` |
| 505 | invalid `i_lrTpDistMin` |
| 507 | invalid `i_lrDistTarg` |
| 509 | `i_lrSyncLen` greater `i_lrDistTarg` |
| 510 | `i_lrSyncLen` greater `i_lrDistDflt` |
| 511 | invalid `i_lrDistDflt` |
| 512 | `lrTpDistToSypt` smaller `i_lrDistDflt` plus `i_lrTpDistMin` |
| 513 | `i_lrTpDistMin` smaller `i_lrSyncLen` |
| 514 | invalid `iCsMode` |

| Detected Alarm ID (added to `GC_MotionLib_ErrorOffset`) | Notification |
|---|---|
| 515 | invalid `lrCsDistMax` |
| 516 | invalid `lrVelStrt` |
| 517 | invalid `lrAccDecStrt` |
| 519 | invalid `lrVelOpMode1` |
| 520 | invalid `lrAccOpMode1` |
| 521 | invalid `lrDecOpMode1` |
| 522 | invalid `lrPulsSyncPhas` |
| 523 | invalid `Slop` |
| 524 | invalid `lrVelStsl` |
| 525 | invalid `lrCorrMax` |
| 526 | invalid `lrMstrOfst` |
| 527 | invalid warm start data |

# Input/Output Pin Description

## Input/Output Pin Description

### Input/Output Pin Description

| Input/Output | Data Type | Description |
|---|---|---|
| `iq_stDrvMstr` | `AXIS_REF_SM3` | Master axis (stripper or target conveyor) reference |
| `iq_stDrvSlav` | `AXIS_REF_SM3` | Slave axis (accumulator) reference |
| `iq_stPara` | STRUCT `ST_GroupingStripperParameter` | Block parameters<br>For further information refer to Structured Parameters *(see page 375)*. |

## Structured Parameter

### iq_stPara

Changes of parameters are accepted with a positive edge at input i_xEn

At the same time the parameters are checked. If their values are invalid or inconsistent, an interruption is generated which affects the outputs q_xAlrm and q_diAlrmId.

| Structure Parameter | Data Type | Description |
|---|---|---|
| iWsMode | INT | Selecting warm start mode *(see page 406)*.<br>Factory setting: 0 |
| lrCsDistMax | LREAL | Maximum distance allowed for detecting the first product (i_xTpIputStd = TRUE)<br>= 0: no limitation<br>> 0: axis maximum distance<br>Factory setting: 1000 |
| lrCorrMax | LREAL | Maximum distance for forward correction.<br>If the distance to the next product is to high (correction > lrCorrMax), a place on the target conveyor is left empty.<br>This prevents excessive stripper acceleration and velocity.<br>Factory setting: 20 |
| lrWsWdowFwd | LREAL | Only with iWsMode = 1 or 4.<br>Maximum distance that the slave is allowed to move forward in order to return to the cam position.<br>Factory setting: 200 |
| lrWsWdowBack | LREAL | Only with iWsMode = 1 or 4.<br>Maximum distance that the slave is allowed to move backwards in order to return to the cam position.<br>Factory setting: 10 |
| lrVelStrt | LREAL | Start velocity<br>Factory setting: 200 |
| lrAccDecStrt | LREAL | Start acceleration / deceleration (also for falling edge at i_xExe)<br>Factory setting: 1000 |
| xTpMode | BOOL | TRUE: Use TP input (iCptrNb)<br>FALSE: Use standard input (i_xTpIputStd)<br>Factory setting: FALSE |

| Structure Parameter | Data Type | Description |
|---|---|---|
| lrTpDistToSypt | LREAL | Distance between TP (sensor) and SyncPt (start of delivery).<br>Must be greater than master modulo (i_lrDistTarg) + lrMstrOfst<br>Factory setting: 500 |
| lrDecEmgy | LREAL | Stop ramp for error stop or falling edge at i_xEn.<br>Factory setting: 5000 |
| lrMstrOfst | LREAL | Only with cold start and xCmpt = TRUE.<br>Product distance from a compartment edge.<br>Factory setting: FALSE |
| xCmpt | BOOL | Only with cold start<br>TRUE: Target conveyor with compartments<br>FALSE: Target conveyor without compartments<br>Compartments of the Target Conveyor *(see page 355)*<br>Factory setting: FALSE |
| lrVelStsl | LREAL | Velocity window in which the drive is regarded as standing still<br>Factory setting: 10 |
| iCptrNb | INT | Number of the TP (0 or 1) to be used.<br>Factory setting: 0 |

# Chapter 14
# Operation Description

## What Is in This Chapter?

This chapter contains the following sections:

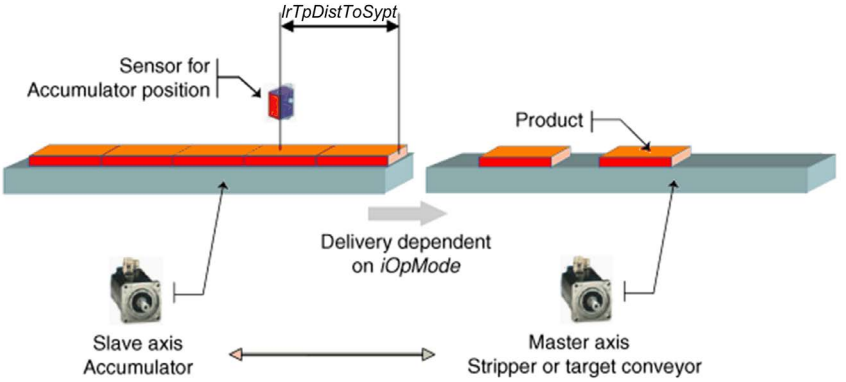
# Section 14.1
# Operating Modes

## What Is in This Section?

This section contains the following topics:

# General Description

## Overview




`iOpMode` sets how the products are passed from the accumulator to the stripper (and/or the target conveyor).

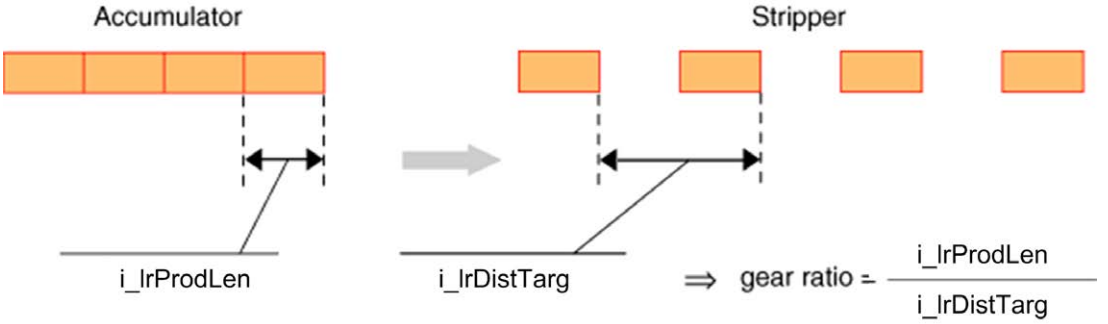
| `iOpMode` | Name | Principle | Description |
|---|---|---|---|
| 0 | Geared Mode |  | Master/slave have constant velocity ratio (during delivery master and slave run with different velocities). |
| 1 | Triggered Timed Pulsed Mode |  | Pulse triggered delivery of one product length (independent of master; no synchronous phase) |
| 2 | Periodic Geared Pulsed Mode |  | Master/slave mode with synchronous phase (during delivery master and slave run with the same velocity).<br>The delivery is triggered cyclically and depends on `i_lrDistTarg`. |
| 3 | Triggered Geared Pulsed Mode |  | Master/slave mode with synchronous phase (as with `iOpMode` = 2).<br>Here a pulse triggers the synchronous phase |

## Geared Mode

### Overview

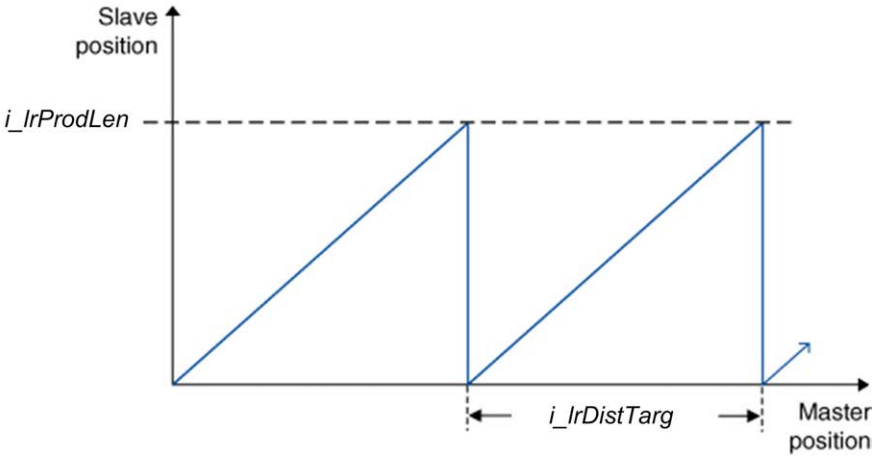
The accumulator follows its master with the velocity ratio:

$$\text{Accumulator velocity} = \frac{\text{i\_lrProdLen}}{\text{i\_lrDistTarg}} \bullet \text{Target velocity}$$




`i_lrProdLen` and `i_lrTargDis` are both inputs of the `GroupingAccumulator_Motion` block.

The master may be a stripper or target conveyor. The change of the velocity is accepted cyclically.




With a rising edge at `i_xExe`, the accumulator synchronizes with its master axis.

If `i_lrProdLen` or `i_lrDistTarg` are changed on the fly (during the movement), a (Poly5) transition cam is used.

# Triggered Timed Pulsed Mode

## Overview

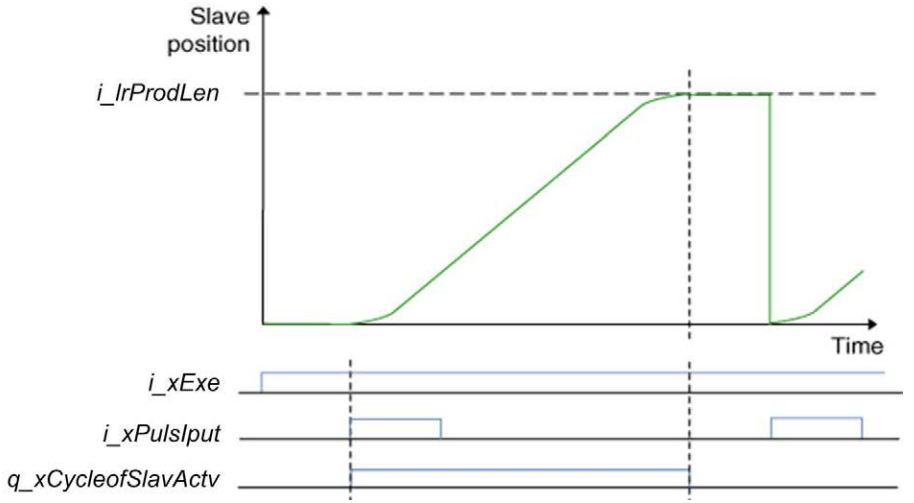
PulseInput

On a rising edge at `i_xPulsIput` (and `i_xExe` = TRUE) one product is delivered. I.e., the accumulator is positioned `i_lrProdLen` forward. There is no synchronization with the master.

Repetitive pulses (rising edges at `i_xPulsIput`) during one cycle are ignored.

The motion of the accumulator is defined by the following parameters:

- `lrVelOpMode1` (velocity)
- `lrAccOpMode1` (acceleration)
- `lrDecOpMode1` (deceleration)



## Periodic Geared Pulsed Mode

### Overview

When `i_xExe` = TRUE, the accumulator executes a periodic movement. With every master period `i_lrDistTarg` one product is delivered.

Setting the delivery cam.

The accumulator follows its master over an electronic cam:
- X-Factor is `i_lrDistTarg`
- Y-Factor is `i_lrProdLen`

Within a master period (`i_lrDistTarg`) the slave movement is divided into 3 phases:
- acceleration phase
- synchronous phase
- deceleration phase

These curves can be adjusted by `lrPulsSyncPhas` and `lrSlop`. Their values must be consistent. For further informations refer to Special Modes *(see page 408)*.

## Triggered Geared Pulsed Mode

### Overview


On a rising edge at `i_xPulsIput` (and `i_xExe` = TRUE) one product is delivered. The accumulator performs one motion cycle and is positioned `i_lrProdLen` forward. Here if `iOpMode` ≠ 1, the motion cycle is defined by an electronic cam.

Repetitive pulses (rising edges at `i_xPulsIput`) during one cycle are ignored.

The delivery movement is the same as with `iOpMode` = 2.

# Section 14.2
## Start, Stop and Special Modes

### What Is in This Section?

This section contains the following topics:

# Start and Resume Modes

## Start and Resume Modes

### Start and Resume Modes

When the input `i_xEn` is activated, the parameters defined in `iq_stPara` are checked for consistency. With incorrect or inconsistent input parameters, `q_xAlrm` becomes TRUE. In `q_diAlrmId` the notification number is indicated.

When the input `i_xEn` is activated, the control of the slave is managed by the Grouping function block.

When the input `i_xExe` is activated, the slave is launched according to the start mode defined by the input `i_xWsSel`.

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Do not send any other commands to the slave axis when it is under control of the Grouping/Ungrouping application function block.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

The start modes describe the slave behavior on a rising edge at `i_xExe`.

For the Accumulator, refer to Cold Start *(see page 402)* and Warm Start *(see page 406)*.

For the Stripper, refer to Cold Start *(see page 404)* and Warm Start *(see page 406)*.

The axes behavior when starting is dependent upon the conditions for stopping, which include:

- Falling edge at input `i_xEn`
- Falling edge at `i_xExe`
- Detected alarm in the block function

For further details, refer to Stop and Resume Modes *(see page 401)*.

# Stop and Resume Modes

## Stop and Resume Modes

The stop and resume modes describe the slave behavior according to the cause of the stop.

With normal execution of the function, when the master stops, the slave stops following the master slope.

With the restart of the master (without modification of the block inputs) the slave will follow accordingly.

Other causes for a stop may be:

| Cause for Stop | Stop Ramp | Possible Resume Modes |
|---|---|---|
| Falling Edge at `i_xEn` | `lrDecEmgy` | Cold start or warm start |
| Falling Edge at `i_xExe` | `lrAccDecStrt` | Cold start or warm start |
| Block Problem | `lrDecEmgy` | Cold start |

In all 3 cases the master (target conveyor and/or stripper) must be stopped immediately.

If this is not possible, the following events may occur:

| If... | Then... |
|---|---|
| conveyors are within SyncPhase | with a standing or moving conveyor, the product currently being delivered may be shifted in an undefined manner. |
| conveyors are outside SyncPhase | at the next warm start the next product can be pushed onto a standing master conveyor. |

Take into consideration the actions and events listed in the following table for a restart.

| With an... | Then... |
|---|---|
| cold start within SyncPhase | you have to remove the products from the delivery position before the cold start, otherwise a collision may occur. |
| cold start outside SyncPhase | the start can be managed without any specific consideration. |
| warm start within SyncPhase after the master stopped immediately | the start can be managed without any specific consideration. |
| warm start within SyncPhase after the master had moved on | the product currently being delivered may be shifted in an undefined manner. |
| warm start outside SyncPhase after the master stopped immediately | the start can be managed without any specific consideration. |
| warm start outside SyncPhase after the master had moved on | during the warm start the products may be pushed onto a standing master conveyor. |

# Cold Start - GroupingAccumulator_Motion

## Overview

There are 2 different cold start modes for the behavior of the accumulator (see iCsMode in the table below).

When the cold start is finished, q_xPosStrtRchd becomes TRUE.

For further informations refer to Warm Start Modes *(see page 406)*.

## Requirements for a Cold Start

A cold start is the first start after switching on the machine. The master (stripper or target conveyor) must wait for the arrival of the first product before moving. Otherwise, at the end of the cold start movement an interrupt is generated.

## Settings for a Cold Start

For a cold start, the following inputs and parameters must be set:

| Input and/or Parameter | Description | |
|---|---|---|
| i_xWsSel | TRUE: Warm start<br>FALSE: Cold start | |
| iCsMode | Cold start mode | |
| | Setting | Description |
| | 0 | the axis moves until the TP sensor detects the first product (i_xTpIputStd = TRUE). Then the axis moves forward lrTpDistToSypt:<br>• When exceeding the lrCsDistMax for detecting a product, an alarm is generated.<br>• When iOpMode = 1 or iOpMode = 3 and there is a rising edge at i_xPulsIput, the conveyor moves forward i_lrProdLen.<br>For further informations, refer to Sensor Wiring *(see page 363)*, Geared Mode *(see page 394)* and Triggered Geared Pulsed Mode *(see page 398)*. |
| | 1 | setting dimension: the axis is set to a defined start position (constant internal value). There is no movement taking place.<br>• This mode is used where no TP sensor exists or is taken into account. The user positions the first product at the delivery position.<br>• The axis is set to the calculated slave position (iOpMode = 1 or iOpMode = 3) and/or to i_lrDistTarg (iOpMode = 0 or iOpMode = 2).<br>For further informations, refer to Geared Mode *(see page 394)*, Triggered Timed Pulsed Mode *(see page 396)*, Periodic Geared Pulsed Mode *(see page 397)* and Triggered Geared Pulsed Mode *(see page 398)*. |

| Input and/or Parameter | Description | |
|---|---|---|
| lrTpDistToSypt | Measured distance between the sensor (TP) and the synchronous start point (SyncPt, end of conveyor, start of delivery) | |
| lrCsDistMax | maximum permissible distance for detecting the first product (i_xTpIputStd = TRUE) | |
| | Setting | Description |
| | = 0 | no limitation |
| | > 0 | maximum axis movement |
| lrVelStrt | start velocity | |
| lrAccDecStrt | start acceleration / deceleration | |
| lrMstrOfst | only if no stripper is used<br>only with xCmpt = TRUE<br>Product distance from a compartment edge | |
| xCmpt | only if no stripper is used<br>TRUE: Compartments on target conveyor. Refer to Compartments of the Target Conveyor *(see page 355)*<br>FALSE: No compartments on target conveyor. The synchronization starts immediately. | |

# Cold Start - GroupingStripper_Motion

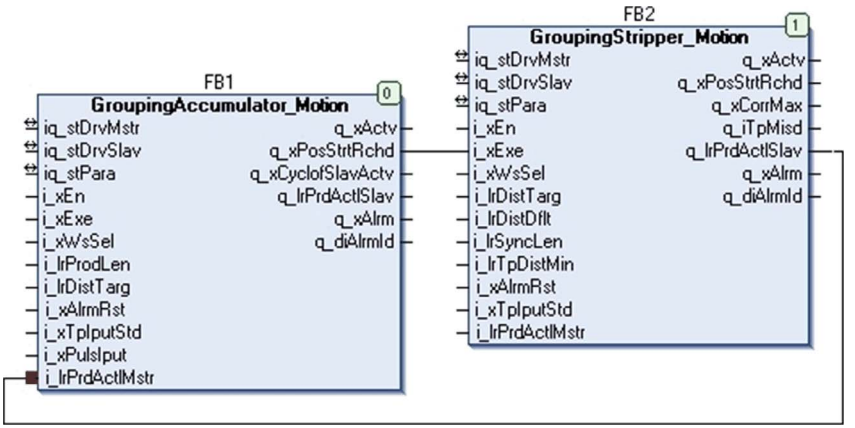
## Overview

The stripper moves until the TP sensor detects a product. Then q_xPosStrtRchd becomes TRUE.

## Requirements for a Cold Start

A cold start is the first start after switching on the machine. The position of the products are unknown.

The cold start of the accumulator (if applicable) must be finished, i.e., the accumulator operates in automatic mode. For further information, refer to Cold Start Accumulator *(see page 402)* and Sensor Wiring *(see page 364)*.




The master (target conveyor) must wait for the arrival of the first product.

Otherwise, at the end of the cold start movement an interruption is generated.

If compartments are used, the target conveyor must wait in its start position.

### Settings for a Cold Start

For a cold start, the following inputs and parameters must be set:

| Input and/or Parameter | Description | |
|---|---|---|
| `i_xWsSel` | TRUE: Warm start<br>FALSE: Cold start | |
| `lrTpDistToSypt` | Measured distance between the sensor (TP) and the synchronous start point (SyncPt, end of conveyor, start of delivery) | |
| `lrCsDistMax` | maximum permissible distance for detecting the first product (`i_xTpIputStd` = TRUE) | |
| | Setting | Description |
| | = 0 | no limitation |
| | > 0 | maximum axis movement |
| `lrVelStrt` | start velocity | |
| `lrAccDecStrt` | start acceleration / deceleration | |
| `lrMstrOfst` | only with `xCmpt` = TRUE<br>Product distance from a compartment edge | |
| `xCmpt` | TRUE: Compartments on target conveyor. Refer to Compartments of the Target Conveyor *(see page 355)*<br>FALSE: No compartments on target conveyor. The synchronization starts immediately. | |

# Warm Start

## Requirements for a Warm Start

You execute a warm start in order to continue an interrupted cycle at the current position of the master (after an emergency stop, for example). The master (stripper or target conveyor) has stopped and its position is known. The slave (accumulator or stripper) usually stops outside of its cam position.

## Settings for a Warm Start

- At start of the movement `i_xWsSel` must be TRUE.
- The warm start mode is selected. (see below)
- The master axis must not be moved.
- A warm start is not possible with `iOpMode` = 1. Refer to Tiggered Timed Pulsed Mode *(see page 396)*

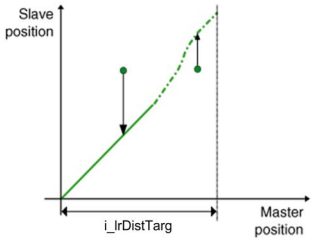
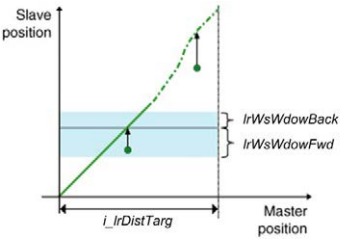
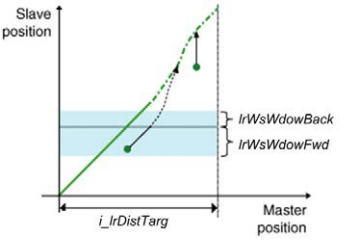
## Warm Start Behavior

- The slave (accumulator) returns to its cam position that depends on the warm start mode.
- When the cam position is reached, `q_xPosStrtRchd` becomes TRUE.

For further informations, refer to Accumulator Cold Start *(see page 402)* and Selecting Warm Start Mode.

## Selecting a Warm Start Mode

The parameter `iWsMode` selects between 3 different warm start modes:

| iWsMode | Description | Representation |
|---|---|---|
| 0 | The slave moves directly to its cam position.<br>**NOTE:** The slave may go backwards to reach its cam position. |  |
| 1 | The slave moves to its cam position only if it is within the range `lrWsWdowBack` < warm start window < `lrWsWdowFwd`<br>If slave position is outside of this warm start window an alarm is generated. |  |
| 4 | The slave keeps its distance to the cam. A position alarm on target conveyor must be acceptable.<br>The initial slave position must be within the warm start window, otherwise an alarm is generated.<br>This warm start window is the range between `lrWsWdowFwd` and `lrWsWdowBack` in relation to the initial master position. |  |

## Special Modes

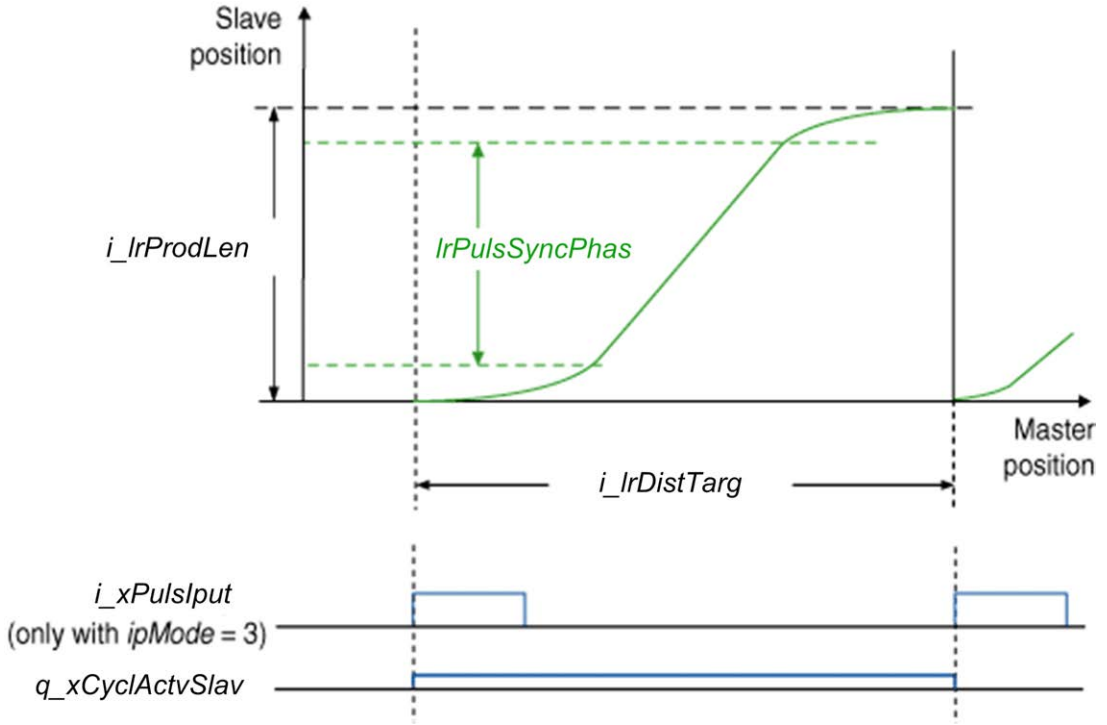
### lrPulsSyncPhas

The length of the synchronous phase is configured by the parameter `lrPulsSyncPhas`, which represents a percentage of `i_lrProdLen`.

The remainder of `i_lrProdLen` is evenly divided between acceleration and deceleration.

Example:

`lrPulsSyncPhas` = 80 means that the accumulator conveyor moves synchronously (to be exact: they have the same velocity only if `lrSlop`= 1) with its master for 80% of `i_lrProdLen`. 10% is used for acceleration and 10% for deceleration.



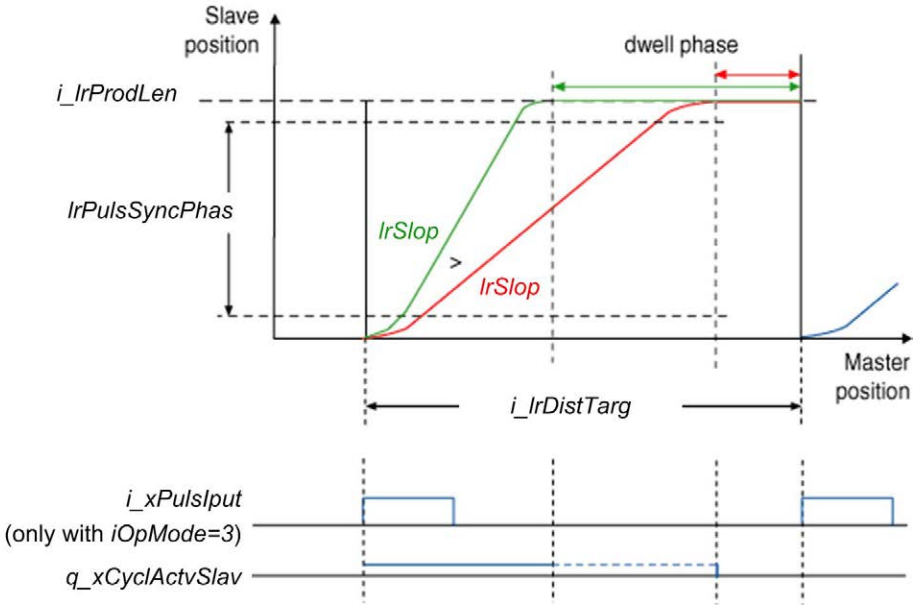
### lrSlop

The master/slave velocity ratio during the synchronous phase is configured with the parameter `lrSlop`. The cam may then have a dwell phase. `lrSlop` does not affect the length of the synchronous phase. This is set with `lrPulsSyncPhas`.

Examples:

`lrSlop` = 1 means a velocity ratio between accumulator and stripper of 1:1.

`lrSlop` = 2 increases this ratio to 2:1. The accumulator delivers the products to the stripper faster.



# Chapter 15
# Quick Reference Guide

## What Is in This Chapter?

This chapter contains the following topics:

# Visualization

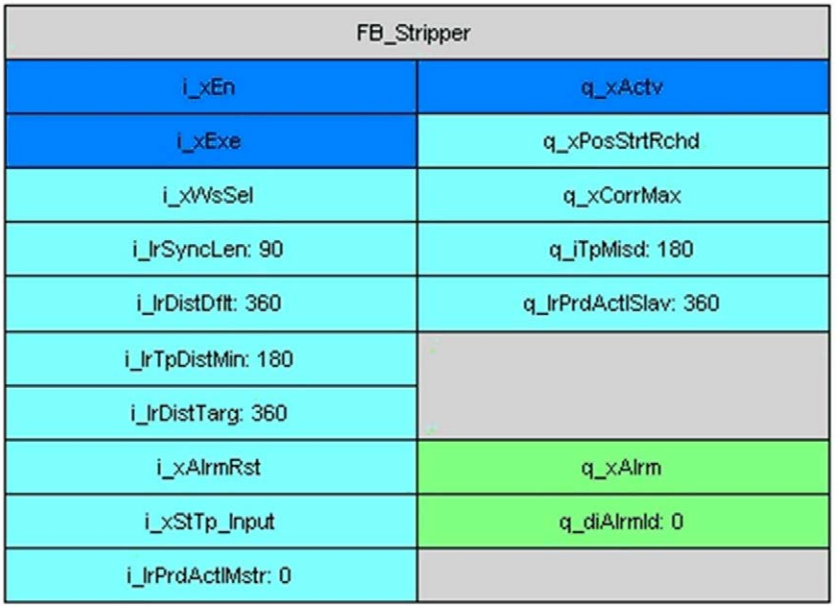
## Overview

Visualization screens are embedded in the Packaging library. They help to configure and to start the application function blocks easily and quickly.

All the inputs/outputs and all the parameters are accessible.

For more information please refer to How to integrate the Visualization into an Application *(see page 433)*.

Stripper visualization



# Troubleshooting

## Troubleshooting

This table describes some general issues and their solutions:

| Issue | Cause | Solution |
|---|---|---|
| The cycle is stopped and detected alarm outputs are set. | Parameter is invalid or inconsistent. | • Correct the parameter.<br>• Set new rising edge at `i_xEn`. |
| | Inputs are invalid or inconsistent. | • Correct the inputs and acknowledge with `i_xAlrmRst`.<br>• Set new rising edge at `i_xExe`. |
| | Application during operation (general alarm) | • Correct the application and acknowledge with `i_xAlrmRst`.<br>• After `i_xEn` -> FALSE, execute a warm start and/or a cold start. For further information, refer to Starting Modes *(see page 402)*. |
| The SoftMotion (SM3 library) function block detected alarm is passed on to the alarm outputs of the Grouping function block. | SoftMotion (SM3 library) (e.g., incorrect following) | • Correct the SoftMotion (SM3 library) function block and acknowledge with `i_xAlrmRst`.<br>• Restart SoftMotion (SM3 library) function block.<br>• After `i_xEn` -> FALSE, execute a warm start and/or a cold start. |

# Part IX
## Clamping

# Chapter 16
# `Clamping_Motion`: Logical Sequence of Product Clamping with Torque Limitation

## What Is in This Chapter?

This chapter contains the following sections:

# Section 16.1
# Functional and Machine Overview

## Functional Overview

### Usage of `Clamping_Motion` Function Block?

The `Clamping_Motion` function block is required for clamping arbitrary products.

### Solution with the `Clamping_Motion` Function Block

When `i_xCls` is activated, the clamping starts a fast positioning up to a defined position or until the recognition of an input signal. After this, the axis brakes and continues driving slowly until a set reference current or the goal position is reached. With the signal `i_xOpen`, the clamping returns to a specified position.

**NOTE:** The function block only runs on type S controllers.

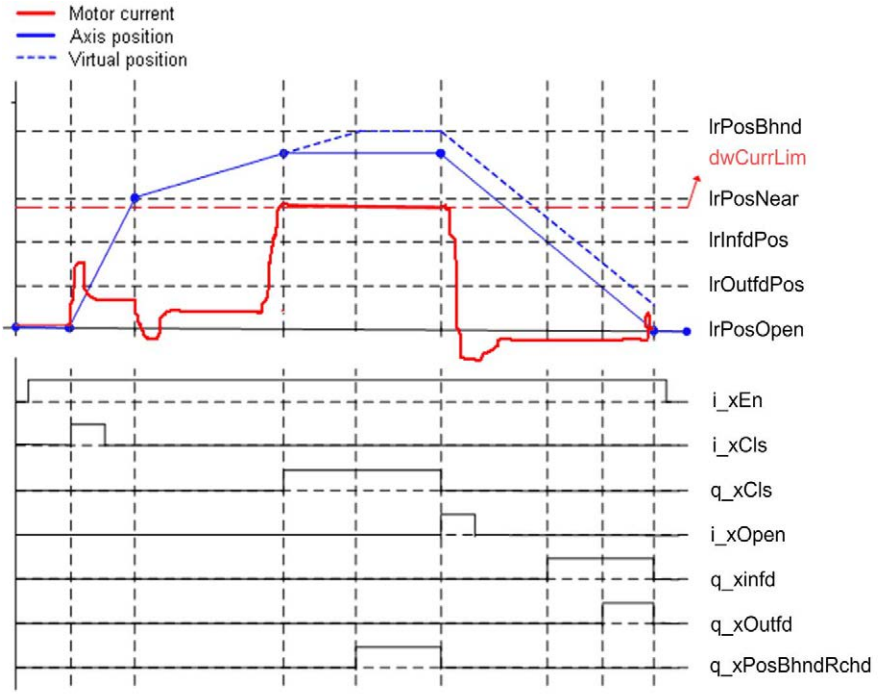
**NOTE:** The closing movement must be in positive direction of the drive.

| ⚠ CAUTION |
| --- |
| **UNITENDED EQUIPMENT OPERATION**<br>The Clamping function works only with drives of type Lexium 05 or Lexium 32.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

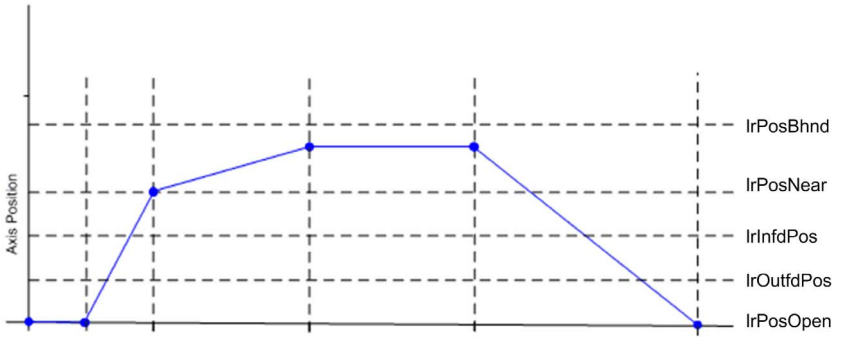
# Section 16.2
## Architecture

## Software Architecture

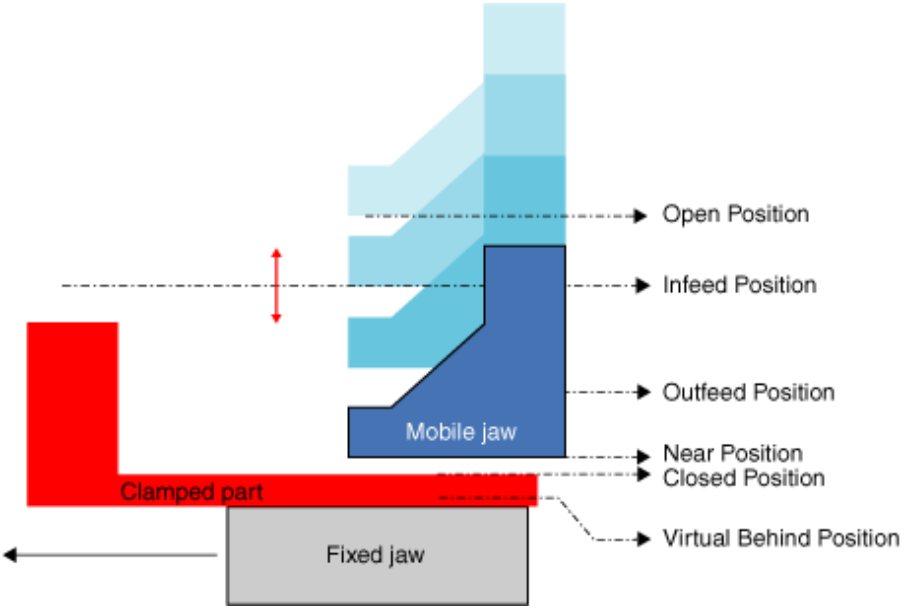
### Profile in Principle



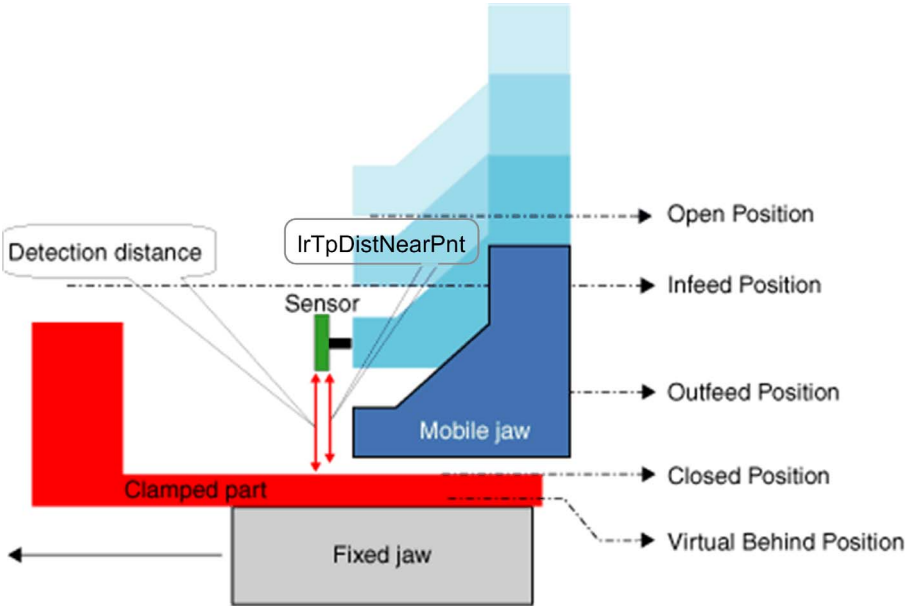
## Motion Profile




## Clamping without Product Detection



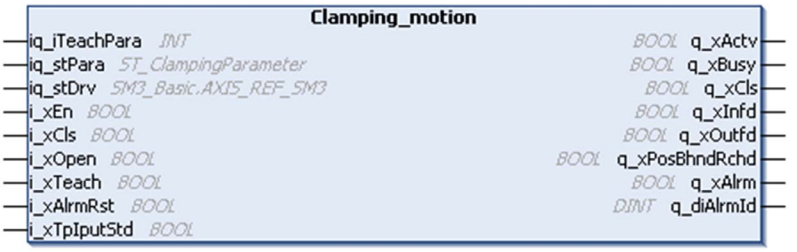
## Clamping with Product Detection



# Section 16.3
## Function Block Description

## `Clamping_motion` Function Block

### Pin Diagram




### Function Block Description

When `i_xEn` is activated, the block is activated. The parameters defined in `iq_stPara` are examined. If the input parameters are incorrect or inconsistent, `q_xAlrm` becomes TRUE. `q_diAlrmId` indicates the alarm number.

When `i_xCls` is activated, the axis is started and moves with `lrVelHigh` until it reaches `lrPosNear`. After this, the axis decelerates and moves with the slow speed `lrVelLow`. When the current specified in `dwCurrLim` is reached, the axis stops. Alternatively, switching to slow speed can be accomplished with a TP input or a standard input (`i_xTpIputStd`).

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Do not send any other commands to the slave axis when it is under control of the `Clamping_Motion` application function block.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

# Section 16.4
# Pin Description

## What Is in This Section?

This section contains the following topics:

# Input Pin Description

## Input Pin Description

| Input | Data Type | Description |
|---|---|---|
| i_xEn | BOOL | TRUE: Block is activated<br>Factory setting: FALSE |
| i_xCls | BOOL | TRUE: Closing block function is started (edge evaluation)<br>Factory setting: FALSE |
| i_xOpen | BOOL | TRUE: Opening block function is started (edge evaluation)<br>Factory setting: FALSE |
| i_xTeach | BOOL | TRUE: Start of teach procedure<br>Factory setting: FALSE |
| i_xAlrmRst | BOOL | TRUE: Alarm reset on rising edge.<br>Factory setting: FALSE |
| i_xTpIputStd | BOOL | Capture input (standard) of master position.<br>Significant only if xTpMode = FALSE<br>Factory setting: FALSE |

# Output Pin Description

## Output Pin Description

### Output Pin Description

| Output | Data Type | Description |
|---|---|---|
| q_xActv | BOOL | TRUE: Block is activated |
| q_xBusy | BOOL | TRUE: Cycle is in progress |
| q_xCls | BOOL | TRUE: Jaw is in closed position. Torque is constant and product is clamped |
| q_xInfd | BOOL | TRUE: Axis has reached lrInfdPos position |
| q_xOutfd | BOOL | TRUE: Axis has reached lrOutfdPos position |
| q_xPosBhndRchd | BOOL | TRUE: Axis has reached lrPosBhnd position (no product for clamping found) |
| q_xAlrm | BOOL | TRUE: There is a detected alarm |
| q_diAlrmId | DINT | Notification number.<br>For detailed information see table below. |

### Notifications

There are 2 types of notifications numbers given on the output q_diAlrmId of the function block.

The first types of notifications are coming from internal used (SoftMotion (SM3 library)) function blocks. These notifications are given directly to the output q_diAlrmId. For these notifications see the list of notifications of the SoftMotion (SM3 library) function blocks.

The second types of notifications are generated by the function block itself. These notifications are added to the global constant GC_MotionLib_ErrorOffset (standard 1000).

| Detected Alarm ID (added to GC_MotionLib_ErrorOffset) | Notification |
|---|---|
| 10 | controlled axis not working |
| 14 | controlled axis in wrong state |
| 15 | controlled axis is no modulo axis |
| 16 | invalid drive - the Clamping_Motion function block supports only the LXM05, LXM32A and LXM32M drive |
| 50 | invalid capture input |
| 400 | teach not possible |
| 401 | invalid dwCurrLim |
| 403 | invalid lrAccHigh |

| Detected Alarm ID (added to `GC_MotionLib_ErrorOffset`) | Notification |
|---|---|
| 404 | invalid `lrDecHigh` |
| 405 | invalid `lrVelHigh` |
| 406 | invalid `lrAccLow` |
| 407 | invalid `lrDecLow` |
| 408 | invalid `lrVelLow` |
| 409 | invalid `lrPosNear` |
| 410 | invalid `lrAccOpen` |
| 411 | invalid `lrDecOpen` |
| 412 | invalid `lrVelOpen` |
| 413 | invalid `lrPosBhnd` |
| 414 | invalid teach input |
| 415 | close speed too fast |
| 416 | close speed too slow |
| 418 | `lrPosNear` reached without detecting TP signal |
| 419 | `lrTpDistNearPnt` greater than `lrPosBhnd` - `IrPosOpen` |
| 420 | invalid `lrInfdPos` |
| 421 | invalid `lrOutfdPos` |

# Input/Output Pin Description

## Input/Output Pin Description

| Input/Output | Data Type | Description | |
|---|---|---|---|
| iq_iTeachPara | INT | Position for the teaching | |
| | | 0 | (none) |
| | | 1 | OutFeed position |
| | | 2 | InFeed position |
| | | 3 | Near position |
| | | 4 | Open position |
| iq_stPara | STRUCT ST_ClampingParameter | Block parameters<br>Refer to the input structured parameter iq_stPara *(see page 428)*. | |
| iq_stDrv | SM3_Basic.AXIS_REF_SM3 | Axis to be controlled.<br>Axis has to be configured as linear axis. | |

## Structured Parameter

### iq_stPara

| Input | Data Type | Description |
|---|---|---|
| dwCurrLim | DWORD | Value of current limitation during clamping procedure in steps of 1/100 A<br>Factory setting: 25 |
| lrPosOpen | REAL | Position when the clamping is opened<br>Factory setting: 0 |
| lrInfdPos | REAL | Position for display of output q_xInfd (only displayed on opening)<br>Can be used to start infeed of the product.<br>Factory setting: 50 |
| lrOutfdPos | REAL | Position for display of output q_xOutfd (only displayed on opening)<br>Can be used to start outfeed of the product.<br>Factory setting: 80 |
| lrPosBhnd | REAL | Virtual behind position. If this position is reached, q_xPosBhndRchd goes to TRUE and an alarm notification is generated.<br>It means that no product has been found.<br>Factory setting: 240<br>Range: >lrPosNear, <lrPosOpen |
| lrPosNear | REAL | Near position: position for changeover from fast to slow speed on closing. This parameter is used only when clamping is done without product detection (xTpUse = FALSE).<br>Factory setting: 180<br>Range: >lrPosOpen |
| lrVelHigh | REAL | Speed for fast movements (closing)<br>Factory setting: 100<br>Range: >= 0.1 |
| lrAccHigh | REAL | Acceleration for fast movements (closing)<br>Factory setting: 1000<br>Range: >= 0.1 |
| lrDecHigh | REAL | Brake ramp for fast movements (closing)<br>Factory setting: 1000<br>Range: >= 0.1 |
| lrVelLow | REAL | Speed for slow movements (closing)<br>Factory setting: 10<br>Range: >= 0.1 |
| lrAccLow | REAL | Acceleration for slow movements (closing)<br>Factory setting: 1000<br>Range: >= 0.1 |

| Input | Data Type | Description |
|---|---|---|
| lrDecLow | REAL | Brake ramp for slow movements (closing)<br>Factory setting: 1000<br>Range: >= 0.1 |
| lrVelOpen | REAL | Speed for opening the clamping<br>Factory setting: 500<br>Range: >= 0.1 |
| lrAccOpen | REAL | Acceleration for opening the clamping<br>Factory setting: 5000<br>Range: >= 0.1 |
| lrDecOpen | REAL | Brake ramp for opening the clamping<br>Factory setting: 500<br>Range: >= 0.1 |
| lrDecEmgy | REAL | Brake ramp in case of interruption or block de-selection<br>Factory setting: 5000<br>Range: >= 0.1 |
| xTpUse | BOOL | TRUE: Clamping with product detection.<br>Use product detection (TP) to start slow approach.<br>Factory setting: FALSE |
| xTpMode | BOOL | Used only if xTpUse = TRUE<br>FALSE: TP over standard input (i_xTpIputStd)<br>TRUE: TP over TP input<br>Factory setting: FALSE |
| iCptrNb | INT | Touch Probe number<br>Used only if xTpUse = TRUE and xTpMode = TRUE.<br>Factory setting: 1<br>Range: 0...1 |
| xTpEdge | BOOL | Used only if xTpUse = TRUE<br>FALSE = negative edge<br>TRUE = positive edge<br>Factory setting: FALSE |
| lrTpDistNearPnt | REAL | Used only if xTpUse = TRUE<br>Distance between sensor (captor) and slow approach position.<br>Factory setting: 20 |

If no product detection is used, then due to movement behavior, the parameters have to fulfil the following formulas:

```
lrPosNear - lrPosOpen < lrVelLow² / (2 * lrAccHigh)
lrPosBhnd - lrPosOpen < lrVelLow² / (2* lrAccHigh)
```

# Section 16.5
# Operating Mode

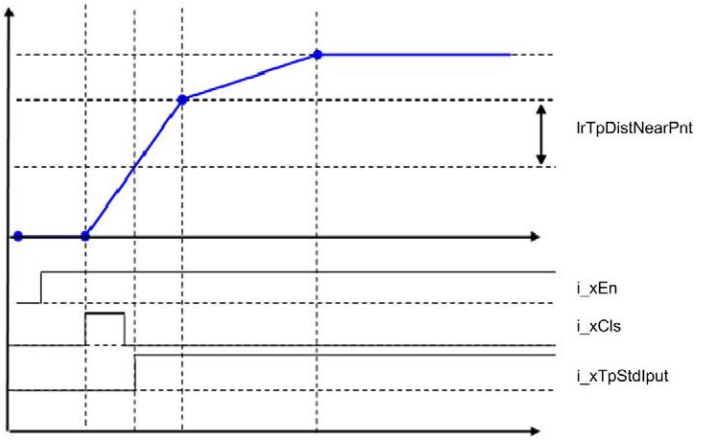
## Selection of the Capture Mode

### Overview

According to the architecture and the precision requirements of the application, several capture modes are possible:

| Capture Mode | Description |
|---|---|
| Detection with standard input | The sensor is wired to a standard input. The capturing is done in the cycle of the motion task in which the function block is called. |
| Detection with special input | The sensor is wired to a special input:<br>● The inputs of the controller are captured if the master axis is an encoder.<br>● A input of a servo drive is captured if the master is a servo drive. |

| **xTpMode** | **iCptrNb** | **i_xTpIputStd** |
|---|---|---|
| FALSE: Capturing with standard input | No effect | Associated with the standard input.<br>The digital input has to be connected to the input `i_xTpStdIput`. |
| TRUE: Capturing with special input | Touch Probe input number. If TP2 is used, then `iCptrNb` = 1 | No effect |


lrTpDistNearPnt
i_xEn
i_xCls
i_xTpStdIput

# Section 16.6
# Quick Reference Guide

## Visualization

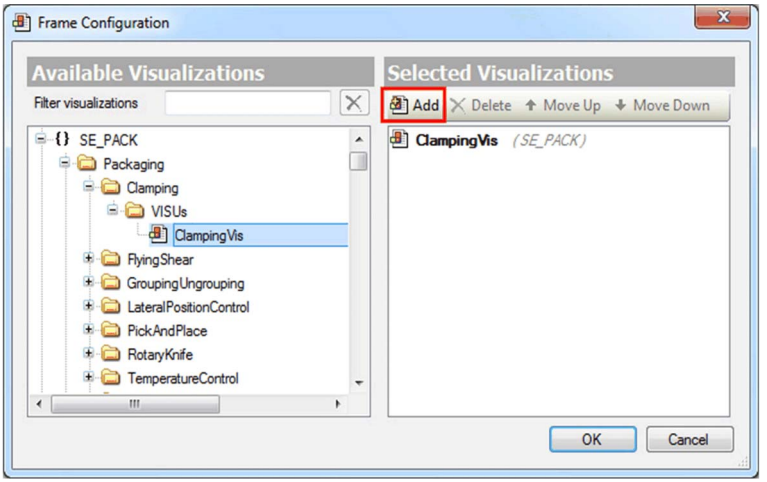
### Overview

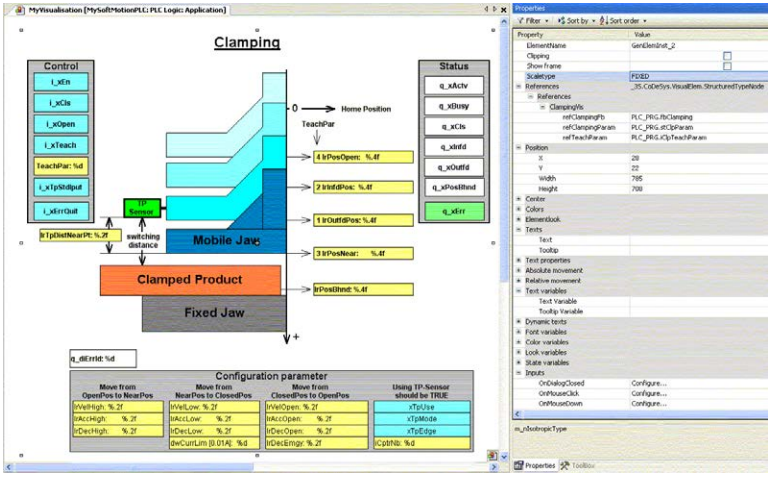
Visualization screens are in the Clamping folder. They help to configure and start the application function block easily and quickly. All the inputs/outputs and all the parameters are accessible.

How to start the visualization is described in the following.

## How to Integrate the Visualization into an Application

The following table describes the procedure of integrating the visualization into an application:

| Step | Action |
|---|---|
| 1 | In the **Tools Tree** right click **Application**. |
| 2 | Select **Add Object**. |
| 3 | Select the **Visualization object** and give it a name (e.g., **MyVisualization**) and press **Add**.<br>**Result**: The visualization and the **Visualization Manager** are added to the project. |
| 4 | In the screen of **MyVisualization** add a new frame by selecting **Frame** in the toolbox. Drag the **Frame** onto the visualization screen.<br>**Result**: The Frame Configuration window appears. |
| 5 | Select for the visualization **ClampingVis** and move it with the **Add** button and press **OK**.<br>  |

| Step | Action |
|---|---|
| 6 | To avoid deformation of the visualization, set the scale type in the properties of the frame to FIXED.<br>To connect the visualization to the project set 3 references to the instances of your project:<br>● `refClampingFb` -> the instance of the function block `Clamping_Motion`.<br>● `refClampingParam` -> the instance of the structure which is connected to the input `iq_stPara` of the clamping instance.<br>● `refTeachParam` -> the instance of the variable which is connected to the input `iq_iTeachPara` of the clamping instance.<br>  <br>**Result**: The visualization is ready to work. |

# Glossary

## A

**AFB**

(*application function block*)

**analog output**

Converts numerical values within the logic controller and sends out proportional voltage or current levels.

**application**

A program including configuration data, symbols, and documentation.

## C

**CANmotion**

A CANopen-based motion bus with an additional mechanism that provides synchronization between the motion controller and the drives.

**CANopen**

An open industry-standard communication protocol and device profile specification (EN 50325-4).

**closed loop**

A closed loop control is a motion control system that used both positional feedback and velocity feedback to generate a correction signal. It does this by comparing its position and velocity to the values of specified parameters. The devices providing the feedback are typically encoders, resolvers, LVTDs, and tachometers.

See also: *open loop*

**closing on stack**

Process of filling the grab with bulk cargo

**configuration**

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

**controller**

Automates industrial processes (also known as programmable logic controller or programmable controller).

# E

**EEPROM**

(*electrically erasable programmable read-only memory*) A type of non-volatile memory to store required data even when power is removed.

**element**

The short name of the ARRAY element.

**equipment**

A part of a machine including sub-assemblies such as conveyors, turntables, and so on.

**expansion bus**

An electronic communication bus between expansion I/O modules and a controller.

# F

**FB**

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

**function**

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

Examples: boolean (AND) operators, calculations, conversions (BYTE_TO_INT)

# H

**HMI**

(*human machine interface*) An operator interface (usually graphical) for human control over industrial equipment.

**HSP**

(*high speed*) The motor frequency at maximum reference of a drive.

## I

**I/O**

(*input/output*)

**ID**

(*identifier/identification*)

**input/output**

The index of the ARRAY.

## K

**Kp**

The proportional gain parameter of a PID controller.

## L

**LSP**

The motor frequency at minimum reference of a drive.

## M

**machine**

Consists of several *functions* and/or *equipment*.

**master/slave**

The single direction of control in a network that implements the master/slave mode.

**ms**

(*millisecond*)

## N

**node**

An addressable device on a communication network.

# O

**open loop**

Open loop control refers to a motion control system with no external sensors to provide position or velocity correction signals.

See also: *closed loop*.

**OTB**

(*optimized terminal block*) Used in the context of STB I/O distributed modules.

# P

**pallet**

A portable platform, which is used for storing or moving goods.

**PID**

(*proportional, integral, derivative*) A generic control loop feedback mechanism (controller) widely used in industrial control systems.

**PLCopen**

For more information, refer to *http://www.plcopen.org/*.

**program**

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

**protocol**

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

**Pt100/Pt1000**

(*platinum 100/1000*) Resistance thermometers, also referred to as resistance temperature detectors, are sensors used to measure temperature by correlating electrical resistance with temperature. As the temperature changes, the resistance to an electrical current passing through them predictably changes likewise. They are characterized by their nominal resistance R0 at a temperature of 0 °C.

- Pt100 (R0 = 100 Ω)
- Pt1000 (R0 = 1 kΩ)

**PWM**

(*pulse width modulation*) A fast output that oscillates between off and on in an adjustable duty cycle, producing a rectangular wave form (though you can adjust it to produce a square wave).

# R

**RPM**

(*revolutions per minute*)

**run**

A command that causes the controller to scan the application program, read the physical inputs, and write to the physical outputs according to solution of the logic of the program.

# S

**scan**

A function that includes:

- reading inputs and placing the values in memory
- executing the application program 1 instruction at a time and storing the results in memory
- using the results to update outputs

**SDO**

(*service data object*) A message used by the field bus master to access (read/write) the object directories of network nodes in CAN-based networks. SDO types include service SDOs (SSDOs) and client SDOs (CSDOs).

**Sercos**

(*serial real-time communications system*) A digital control bus that interconnects, motion controls, drives, I/Os, sensors, and actuators for numerically controlled machines and systems. It is a standardized and open controller-to-intelligent digital device interface, designed for high-speed serial communication of standardized closed-loop real-time data.

**setpoint**

In a PID controller, the target value set by the user. The main objective of the PID controller is to ensure that the process value reaches the setpoint.

See also *process value*.

**STOP**

A command that causes the controller to stop running an application program.

# T

**task**

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

**TP**

(*touch probe*) A position capture that is triggered by a fast input signal (quick sensor). On the rising edge of the touch probe input, the position of an encoder is captured.

For example, this is used for packaging machines to capture the position of a printmark on a film to cut on the same position.

# V

**variable**

A memory unit that is addressed and modified by a program.

**VSD**

(*variable speed drive*) An equipment that makes a variable and regulates the speed and rotational force, or torque output, of an electric motor.

# Index