EcoStruxure Automation Expert - Motion
EcoStruxure Machine Expert

MotionApplicationFunctionBlocks

Library Guide

EIO0000004585.03
12/2024

www.se.com

Legal Information

The information provided in this document contains general descriptions, technical characteristics and/or recommendations related to products/solutions.

This document is not intended as a substitute for a detailed study or operational and site-specific development or schematic plan. It is not to be used for determining suitability or reliability of the products/solutions for specific user applications. It is the duty of any such user to perform or have any professional expert of its choice (integrator, specifier or the like) perform the appropriate and comprehensive risk analysis, evaluation and testing of the products/solutions with respect to the relevant specific application or use thereof.

The Schneider Electric brand and any trademarks of Schneider Electric SE and its subsidiaries referred to in this document are the property of Schneider Electric SE or its subsidiaries. All other brands may be trademarks of their respective owner.

Schneider Electric does not grant any right or license for commercial use of the document or its content, except for a non-exclusive and personal license to consult it on an "as is" basis.

Schneider Electric reserves the right to make changes or updates with respect to or in the content of this document or the format thereof, at any time without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this document, as well as any non-intended use or misuse of the content thereof.

Table of Contents

Safety Information

Important Information

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.

Qualification of Personnel

A qualified person is one who has the following qualifications:

- Skills and knowledge related to the construction and operation of electrical equipment and the installation.
- Knowledge and experience in industrial control programming.
- Received safety-related training to recognize and avoid the hazards involved.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

PROPER USE

This product is a library to be used together with the control systems and servo amplifiers intended solely for the purposes as described in the present documentation as applied in the industrial sector.

Always observe the applicable safety-related instructions, the specified conditions, and the technical data.

Perform a risk evaluation concerning the specific use before using the product. Take protective measures according to the result.

Since the product is used as a part of an overall system, you must ensure the safety of the personnel by means of the design of this overall system (for example, machine design).

Any other use is not intended and may be hazardous.

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 are 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:

(In case of divergence or contradiction between any translation and the English original, the original text in the English language will prevail.)

- 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 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

Document Scope

This document describes the MotionApplicationFunctionBlocks library.

Validity Note

For more information on the validity of the present document, see the online help for the product.

Related Documents

Document title	Reference
Cybersecurity Best Practices	CS-Best-Practices-2019-340
Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment	EIO0000004242
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert, Functions and Libraries User Guide	EIO0000002829 (ENG); EIO0000002830 (FRE); EIO0000002831 (GER); EIO0000002832 (ITA); EIO0000002833 (SPA); EIO0000002834 (CHS);
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert Programming Guide	EIO0000002854 (ENG); EIO0000002855 (FRE); EIO0000002856 (GER); EIO0000002857 (ITA); EIO0000002858 (SPA); EIO0000002859 (CHS);

To find documents online, visit the Schneider Electric download center (www.se.com/ww/en/download/).

Product Related Information

⚠WARNING

LOSS OF CONTROL

- Perform a Failure Mode and Effects Analysis (FMEA), or equivalent risk analysis, of your application, and apply preventive and detective controls before implementation.
- Provide a fallback state for undesired control events or sequences.
- Provide separate or redundant control paths wherever required.
- Supply appropriate parameters, particularly for limits.
- Review the implications of transmission delays and take actions to mitigate them.
- Review the implications of communication link interruptions and take actions to mitigate them.
- Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and error conditions) according to your risk assessment, and applicable codes and regulations.
- Apply local accident prevention and safety regulations and guidelines.[1]
- Test each implementation of a system for proper operation before placing it 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.

Before you attempt to provide a solution (machine or process) for a specific application using the POUs found in the library, you must consider, conduct and complete best practices. These practices include, but are not limited to, risk analysis, functional safety, component compatibility, testing and system validation as they relate to this library.

⚠WARNING

IMPROPER USE OF PROGRAM ORGANIZATION UNITS

- Perform a safety-related analysis for the application and the devices installed.
- Ensure that the Program Organization Units (POUs) are compatible with the devices in the system and have no unintended effects on the proper functioning of the system.
- Ensure that the axis is homed and that the homing is valid before usage of absolute movements or POUs using absolute movements.
- Use appropriate parameters, especially limit values, and observe machine wear and stop behavior.
- Verify that the sensors and actuators are compatible with the selected POUs.
- Thoroughly test all functions during verification and commissioning in all operation modes.
- Provide independent methods for critical control functions (emergency stop, conditions for limit values being exceeded, etc.) according to a safety-related analysis, respective rules, and regulations.

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

⚠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.

Incomplete file transfers, such as data files, application files and/or firmware files, may have serious consequences for your machine or controller. If you remove power, or if there is a power outage or communication interruption during a file transfer, your machine may become inoperative, or your application may attempt to operate on a corrupted data file. If an interruption occurs, reattempt the transfer. Be sure to include in your risk analysis the impact of corrupted data files.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION, DATA LOSS, OR FILE CORRUPTION

- Do not interrupt an ongoing data transfer.
- If the transfer is interrupted for any reason, re-initiate the transfer.
- Do not place your machine into service until the file transfer has completed successfully, unless you have accounted for corrupted files in your risk analysis and have taken appropriate steps to prevent any potentially serious consequences due to unsuccessful file transfers.

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

Care must be taken and provisions made for use of this library for machine control to avoid inadvertent consequences of commanded machine operation, state changes, or alteration of data memory or machine operating elements.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

- Place operator devices of the control system near the machine or in a place where you have full view of the machine.
- Protect operator commands against unauthorized access.
- If remote control is a necessary design aspect of the application, ensure that there is a local, competent, and qualified observer present when operating from a remote location.
- Configure and install the Run/Stop input, if so equipped, or, other external means within the application, so that local control over the starting or stopping of the device can be maintained regardless of the remote commands sent to it.

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

Information on Non-Inclusive or Insensitive Terminology

As a responsible, inclusive company, Schneider Electric is constantly updating its communications and products that contain non-inclusive or insensitive terminology. However, despite these efforts, our content may still contain terms that are deemed inappropriate by some customers.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in the information contained herein, 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
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2023	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2020	Safety of machinery: Electro-sensitive protective equipment. 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
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2021	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 61784-3:2021	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
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.

General Information

What’s in This Part

Presentation of the Library

What's in This Chapter

General Information

Library Overview

The MotionApplicationFunctionBlocks library implements function blocks to control a machine that performs operations on a moving part. Typical applications can include cutting, clamping, stamping, marking, sealing.

Characteristics of the Library

The table indicates the characteristics of the library:

Characteristic	Value
Library title	MotionApplicationFunctionBlocks
Company	Schneider Electric
Category	**Application**
Component	**Motion**
Default namespace	**SE_MAFB**
Language model attribute	Qualified-access-only (see Functions and Libraries User Guide)
Forward compatible library	Yes (see Functions and Libraries User Guide)

NOTE: For this library, qualified-access-only is set. The POUs, data structures, enumerations, and constants must be accessed using the namespace of the library. The default namespace of the library is **SE_MAFB**.

Configuration

The following configuration is supported:

- The master can be configured as a physical axis or as an encoder.
- The master can be configured as a virtual axis.
- More than one slave can be synchronized with the same master.
- In rotary knife applications, the slave must be defined as a rotary axis with the modulo value set equal to the value of the parameter *lrPerimeter*, page 28.
- Several parallel instances of the function blocks *FlyingShear* and *RotaryKnife* are allowed per application.
- The function blocks *FlyingShear* and *RotaryKnife* can be combined with additional motion application function blocks such as clamping or grouping in one application.
- The master axis must be defined as a rotary axis (modulo axis).

NOTICE
UNSATISFACTORY EQUIPMENT OPERATION Set the modulo value of the rotary master axis greater than the maximum distance between the products plus the value of the parameter *lrTpDistToSyncPoint* depending on the operating mode. **Failure to follow these instructions can result in equipment damage.**

NOTE: For more information, refer to the *lrTpDistToSyncPoint* parameter description.

Task Concept

The *FlyingShear* and the *RotaryKnife* function blocks are based on the electronic cam functionality of the PLCopen MC part 1 library. Therefore, consider the task concept described in the *Motion Control* chapter of the M262 Synchronized Motion Control Library Guide. The cycle time of the user application task in which the function block *FlyingShear* or *RotaryKnife* is called depends on the maximum velocity of the master axis and is typically 10 ms. The higher the task cycle time, the slower the reaction time. This has an impact on the processing period of input commands and cam calculations inside the *FlyingShear* and the *RotaryKnife* function blocks.

General Considerations

NOTE: Schneider Electric adheres to industry best practices in the development and implementation of control systems. This includes a "Defense-in-Depth" approach to secure an Industrial Control System. This approach places the controllers behind one or more firewalls to restrict access to authorized personnel and protocols only.

⚠WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

- Evaluate whether your application environments are connected to your critical infrastructure and, if so, take appropriate steps in terms of prevention, based on Defense-in-Depth, before connecting the automation system to any network.
- Limit the number of devices connected to a network to the minimum necessary.
- Isolate your industrial network from other networks inside your company.
- Protect any network against unintended access by using firewalls, VPN, or other, proven security measures such as an Intrusion Prevention System or Intrusion Detection System.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Install certificates that are issued by publicly known Trusted Certificate Authorities.
- Keep your systems up to date and rely only on legitimate sources.
- Prepare a recovery plan including backup of your system and process information.

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

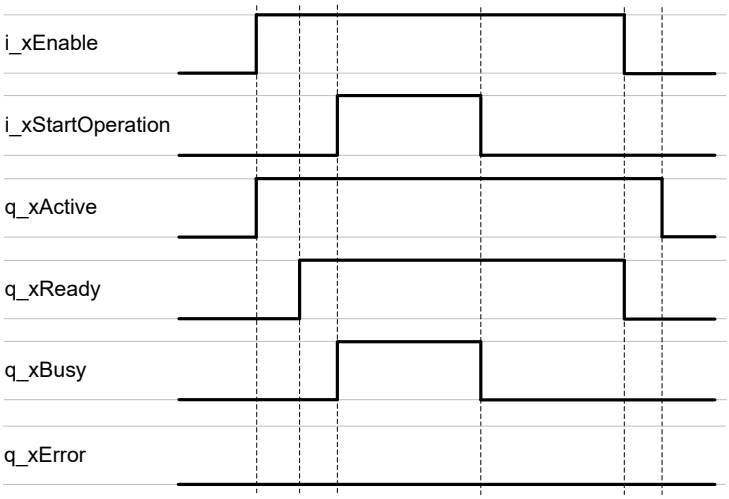
For more information on organizational measures and rules covering access to infrastructures, refer to ISO/IEC 27000 series, Common Criteria for Information Technology Security Evaluation, ISO/IEC 15408, IEC 62351, ISA/IEC 62443, NIST Cybersecurity Framework, Information Security Forum - Standard of Good Practice for Information Security and refer to Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment.

Common Inputs and Outputs

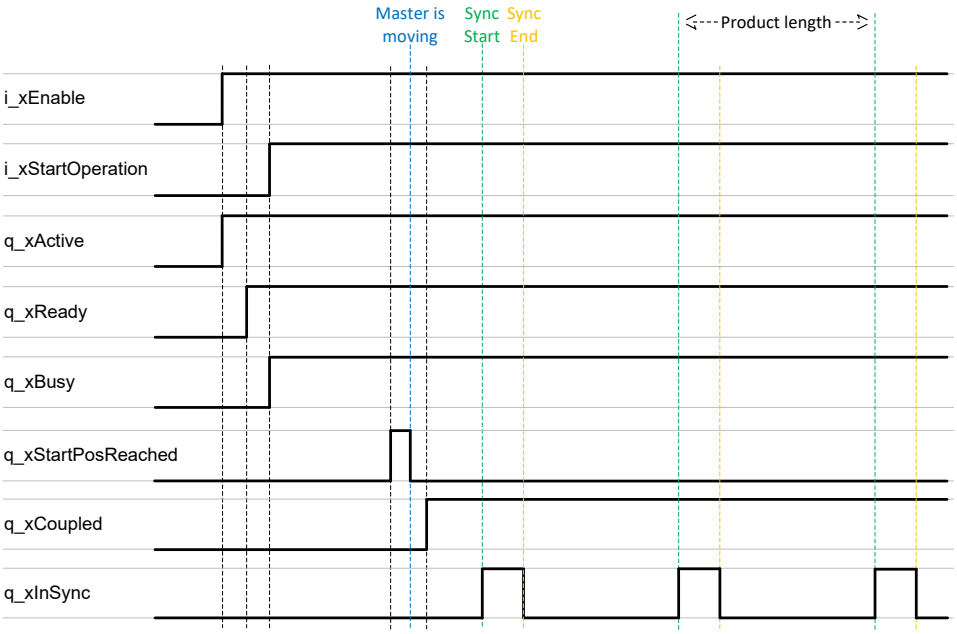
What's in This Chapter

Signal Diagrams

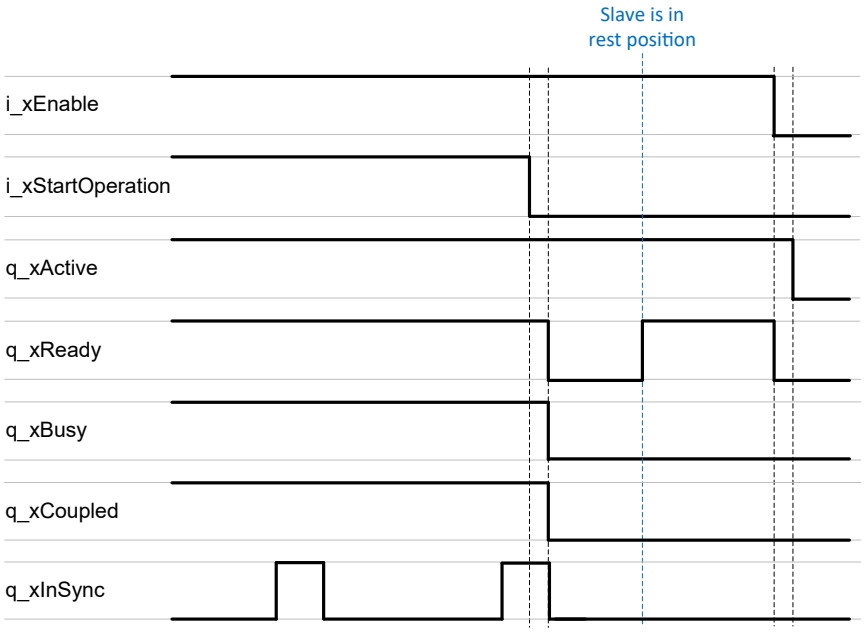
Start Operation



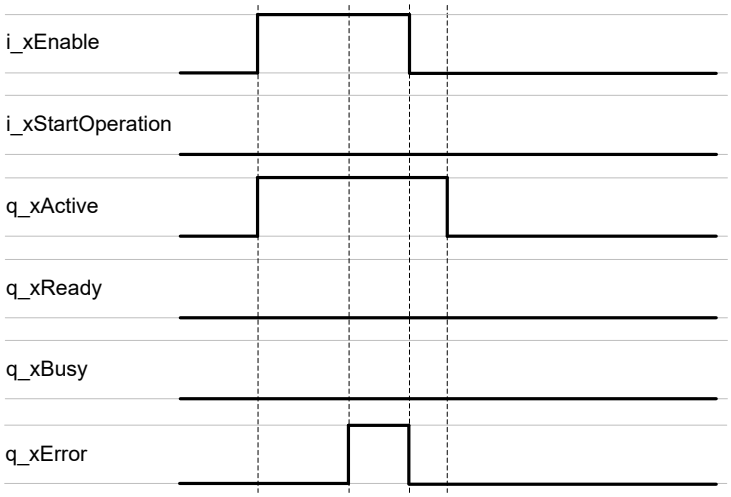
Operating



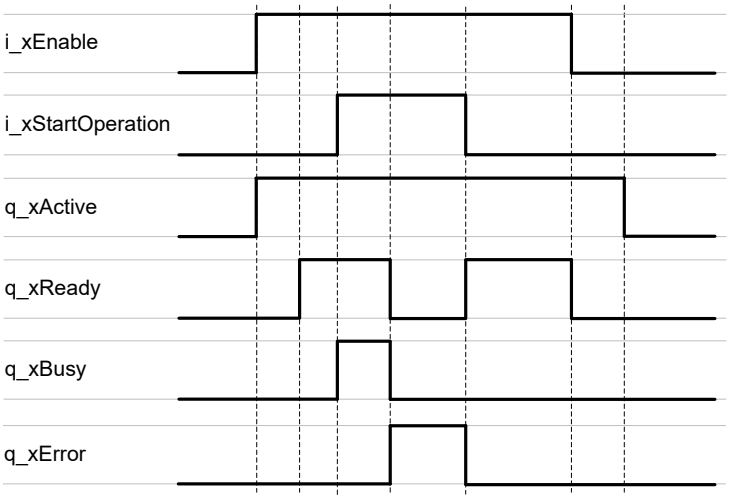
Stop Operation



Error Detected While Enabling



Error Detected While Starting Operation



Diagnostic Concept

What's in This Chapter

Diagnostic Concept

Overview

EcoStruxure Automation Expert - Motion and EcoStruxure Machine Expert provide a three-layer diagnostic concept for the libraries. This concept is valid for the Technology/Module libraries (for example, the library CommonToolbox) and uses enumerations for diagnostic coding.

In principle, the diagnostic information has the following layers:

1. General information on the exception. No specific knowledge about the POU functionality is necessary.
2. POU-specific diagnostic and status messages (part 1): Detailed information on the source triggering the diagnostic or status messages.
3. POU-specific diagnostic and status messages (part 2): Detailed and dynamic information on the source triggering the diagnostic or status messages.

   This information changes at runtime (for example, information about the condition of the input parameters). This diagnostic output is optional for the POUs.

The diagnostic concept offers the following advantages:

- Online display of the diagnostic messages
- Detailed information on diagnostic events with the availability of the diagnostic codes
- Overview on the status or exceptional condition of a POU
- Pertinent suggested solutions to correct the causes for exceptional conditions
- Enumerated diagnostic messages to facilitate multi-language support for HMI displays

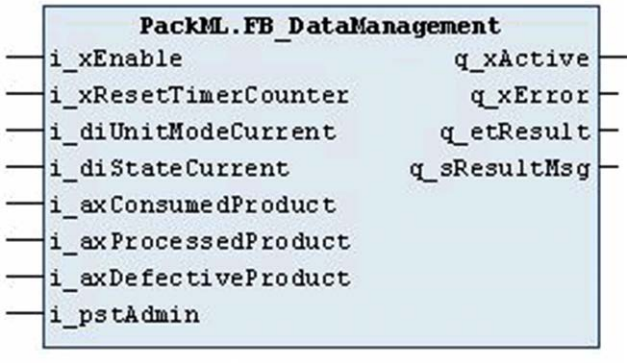
Diagnostic Outputs

Function blocks and functions or methods can have the three diagnostic outputs *q_xError*, *q_etResult*, and *q_sResultMsg*. The outputs are defined in the POU one after another.

Output	Data type	Meaning
q_xError	*BOOL*	The output *q_xError* is set to TRUE when an error has been detected during execution of the function block. The outputs *q_etResult* and *q_sResultMsg* provide the corresponding error code and a plain text information.
q_etResult	*ET_Result*(1)	Provides diagnostic and status information as a numeric value. *If q_xError* = FALSE, *q_etResult* provides status information. *If q_xError* = TRUE, *q_etResult* provides diagnostic/error information. The enumeration *ET_Result* contains the possible values of the POU operation results.

Output	Data type	Meaning
q_sResultMsg	STRING	Provides additional information about the present status of the POU. If the POU is in error state (*q_xError* = TRUE), the message provides additional information for the error cause and, if meaningful, a possible solution. If the POU is busy the present process or state is issued at this output.
(1) Each library provides its own, specific enumeration *ET_Result* which contains the possible *q_etResult* outputs for that library.		

Diagnostic information example:



Data Unit Types

What’s in This Part

Enumerations

What's in This Chapter

ET_Result

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *ET_Result* contains the values that indicate the result of operations executed by the function blocks of this library.

Enumeration Elements

Name	Description
OK	No error has been detected.
GPL_lrLengthToCutMinFactorInvalid	The global parameter *GPL.lrLengthToCutMinFactor*, page 31 is outside the value range.
InitializingMasterFailed	The conversion of the input *i_Master* was unsuccessful.
InitializingSlaveFailed	The conversion of the input *i_Slave* was unsuccessful.
InitializingTriggerRefFailed	The initialization of the touch probe input was unsuccessful.
ReadingTaskCycleFailed	Reading the task cycle time was unsuccessful.
SlaveAxisNotHomed	The slave axis is not referenced.
SlaveAxisNotStandstill	The slave axis is not in the PLCopen state standstill.
MasterAxisIsNoModuloAxis	The master axis is not a modulo axis.
MasterAxisNotHomed	The master axis is not referenced.
MasterAxisNotStandstill	The master axis is not in the PLCopen state standstill.
MasterAxisModuloError	Another master modulo jump occurred before the previous one was processed.
ConnectingAxisMovementMonitorFailed	The *Connect* method of *AxisMovementMonitor* was unsuccessful.
SetPosAxisMovementMonitorFailed	The *SetPosition* method of *AxisMovementMonitor* was unsuccessful.
TouchProbeHandlingError	The initialization of the touch probe handler was unsuccessful.
FifoBufferFull	The FIFO buffer exceeds the limit. It is not possible to add detected touch probes.
FifoBufferEmpty	The FIFO buffer is empty. It is not possible to retrieve the product length.
SlaveIsNoFiniteAxis	This slave is not configured as a linear axis.
SlaveIsNoModuloAxis	This slave is not configured as a modulo axis.
PerimeterInvalid	The parameter *lrPerimeter* is outside the value range.
RestPositionInvalid	The parameter *lrRestPosition*, page 28 is outside the value range.
SyncEndPositionInvalid	The parameter *lrSyncEndPosition*, page 28 is outside the value range.
StopDecelerationInvalid	The parameter *lrStopDeceleration*, page 28 is outside the value range.
VelocityInvalid	The parameter *lrVelocity*, page 28 is outside the value range.
AccDecInvalid	The parameter *lrAccDec*, page 28 is outside the value range.
JerkInvalid	The parameter *lrJerk*, page 28 is outside the value range.
TouchProbeWindowInvalid	The parameter *lrTouchProbeWindow*, page 28 is outside the value range.
SlopeInvalid	The parameter *lrSlope*, page 28 is outside the value range.
RestPositionFactorInvalid	The parameter *lrRestPositionFactor*, page 28 is outside the value range.
LengthToCutInvalid	The input *i_lrLengthToCut*, page 33 is outside the value range.
TPDistToSyncPointInvalid	The parameter *lrTpDistToSyncPoint*, page 28 is outside the value range.
McStopError	An error has been detected in the PLCopen function block *MC_Stop*. Also refer to the M262 Synchronized Motion Control Library Guide.
McMoveAbsoluteError	An error has been detected in the PLCopen function block *MC_MoveAbsolute*. Also refer to the M262 Synchronized Motion Control Library Guide.
McMoveAbsoluteAborted	The PLCopen function block *MC_MoveAbsolute* has been aborted.
McCamInError	An error has been detected in the PLCopen function block *MC_CamIn*. Also refer to the M262 Synchronized Motion Control Library Guide.
McCamInAborted	The PLCopen function block *MC_CamIn* has been aborted.
FcEvaluateCamError	An error has been detected in the function *FcEvaluateCam*.
SlaveWarmStartPositionInvalid	The warmstart position is outside the value range. Also refer to: • *Warm Start Mode ContinueWithOffsetCurve* for *FlyingShear*, page 46 • *Warm Start Mode ContinueWithOffsetCurve* for *RotaryKnife*, page 71

ET_StartMode

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *ET_StartMode* is used to define the warm start mode, page 44.

Enumeration Elements

Name	Value (USINT)	Description
MoveToRestPosition	1	The slave moves to the rest position and stores the FIFO values. For further information, refer to: • Warm start mode *MoveToRestPosition* for *FlyingShear*, page 45 • Warm Start Mode *MoveToRestPosition* for *RotaryKnife*, page 70
MoveToCurvePosition	2	The slave moves to the position on the curve. For further information, refer to: • Warm start mode *MoveToCurvePosition* for *FlyingShear*, page 45 • Warm Start Mode *MoveToCurvePosition* for *RotaryKnife*, page 70
ContinueWithOffsetCurve	3	The slave continues with an offset curve. For further information, refer to: • Warm start mode *ContinueWithOffsetCurve* for *FlyingShear*, page 46 • Warm Start Mode *ContinueWithOffsetCurve* for *RotaryKnife*, page 71

ET_OperatingMode

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *ET_OperatingMode* is used to define the operating mode, page 47.

Enumeration Elements

Name	Value (USINT)	Description
Continuous	0	The operating mode *Continuous* is selected for processing products with a constant length. Also refer to: • Operating Mode *Continuous* for *FlyingShear*, page 48 • Operating Mode *Continuous* for *RotaryKnife*, page 73
ContinuousWithCorrection	1	The operating mode *ContinuousWithCorrection* is selected for processing products with small differences in length. It includes an automatic correction of the product length, calculated from a position capture. Also refer to: • Operating Mode *ContinuousWithCorrection* for *FlyingShear*, page 49 • Operating Mode *ContinuousWithCorrection* for *RotaryKnife*, page 73
CutOnTouchProbe	2	The operating mode *CutOnTouchProbe* is selected for processing products with variable length and format, calculated from a position capture. Also refer to: • Operating Mode *CutOnTouchProbe* for *FlyingShear*, page 50 • Operating Mode *CutOnTouchProbe* for *RotaryKnife*, page 75

Structures

What's in This Chapter

ST_Parameters

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-

Description

The structure *ST_Parameters* specifies the parameters that cannot be modified while the function block is active.

Structure Elements

Name	Data type	Description
lrPerimeter	LREAL	Exclusive to *RotaryKnife*: • Defines the perimeter of the rotary knife. • Defines the distance between the blades for multiple blades. Default value: 0 Value range: > 0
lrRestPosition	LREAL	Rest position of the slave. Default value: 0 Value range: • *FlyingShear*: < 0 • *RotaryKnife*: > *lrSyncEndPosition* < *lrPerimeter*
lrSyncEndPosition	LREAL	End position of the synchronous phase (related to the slave axis). Default value: 0 Value range: • *FlyingShear*: > 0 Also refer to the curves illustrating the position profiles for *FlyingShear*, page 39. • *RotaryKnife*: > 0 < *lrPerimeter* Also refer to the curves illustrating the position profiles for *RotaryKnife*, page 63.
lrTpDistToSyncPoint	LREAL	Distance between the center point of the touch probe sensor and the starting point of the synchronous phase. It is required for the calculation of the curves and the validation of the touch probe signals. The parameter is valid for operating modes *ContinuousWithCorrection*, page 49 and *CutOnTouchProbe*, page 50. Default value: 0 Value range: > *q_lrTpDistToSyncPointMin*
lrRestPositionFactor	LREAL	Factor used to calculate the distance of the master stop phase. For further information, refer to Rest Position Factor, page 55. Default value: 1.5 Value range: > 1.0
lrStopDeceleration	LREAL	Deceleration ramp to stop the slave in case an error is detected or the function block is disabled (*i_xEnable* = FALSE). Default value: 1000 Value range: > 0
lrVelocity	LREAL	The velocity of the slave axis that is applied for the following movement: • In warm start mode to move to the start position. • In cold start mode to move to the rest position. • With a falling edge of *i_xStartOperation* to move to the rest position. Default value: 100 Value range: > 0
lrAccDec	LREAL	The acceleration/deceleration of the slave axis that is applied for the following movement: • In warm start mode to move to the start position. • In cold start mode to move to the rest position. • With a falling edge of *i_xStartOperation* to move to the rest position. Default value: 1000 Value range: > 0

Name	Data type	Description
lrJerk	LREAL	The jerk that is applied to the slave axis for the following movement: • In warm start mode to move to the start position. • In cold start mode to move to the rest position. • With a falling edge of *i_xStartOperation* to move to the rest position. Default value: 0 Value range: ≥ 0
ifTriggerRef	*PLCO.MC.TriggerRef*	The trigger reference for capturing the position of the master axis. Default value: 0 (not connected, no trigger reference): The digital input *i_xTpDigitalInput*, page 33 is used to capture the position For further information on activating capture inputs in the Logic Builder, refer to the Appendices, page 87.
lrTouchProbeWindow	LREAL	Tolerance window of the touch probe position depending on the operating mode. Default value: 50 • The operating mode *ContinuousWithCorrection* defines a touch probe acceptance window, that is a range before and after the sensor in which detected touch probe signals are valid. Value range: 0...*i_lrLengthToCut* • The operating mode *CutOnTouchProbe* defines a touch probe ignorance window, that is a range after the sensor in which detected touch probe signals are ignored as long as the registration mark of the previous touch probe signal is inside this window. Value range: maximum value of *lrMasterStartPhase* and *lrLengthToCutMin*
etOperatingMode	*ET_OperatingMode*, page 25	Selection of the operating mode. Default value: 0 Value range: 0...2
lrSlope	LREAL	A slope coefficient for the synchronous phase to define a different velocity for the master and the slave. For further information, refer to *Slope of the Synchronous Phase* for *FlyingShear*, page 55 and to *Slope of the Synchronous Phase* for *RotaryKnife*, page 80. Default value: 1: The master and the slave axis are moving with the same velocity. 0.0 < value range < 10.0

Global Variables

What’s in This Part

Global Parameter List

What's in This Chapter

Global Parameter List (GPL)

Overview

Type:	Global parameters
Available as of:	V1.0.0.0

Description

The global parameter list (GPL) contains global constants which are used by certain internal functions of this library. The parameters can be edited individually for each application where the library is used. The modification must be done within the **Library Manager** of the project where the library is referenced.

The global parameter list (GPL) has the attribute *qualified access only*.

Global Parameters

Variable	Data type	Default value	Range	Description
Gc_usiFiFoMaxNumberOfProducts	USINT	15	1...255	The maximum number of products stored in the FIFO buffer before an error is detected. Also refer to *FIFO Buffer Saving Touch Probe Positions*, page 52.
Gc_lrMasterPositionOffsetFactor	LREAL	2.5	1...20	Factor to calculate the position offset for overwriting curves. For further information, refer to: • *Warm Start Mode ContinueWithOffsetCurve* for *FlyingShear*, page 46 • *Warm Start Mode ContinueWithOffsetCurve* for *RotaryKnife*, page 71
Gc_lrLengthToCutMinFactor	LREAL	1.2	≥1	Factor to calculate the value for the minimum length to cut.

Inputs and Outputs

What’s in This Part

The MotionApplicationFunctionBlocks library implements the function blocks *FlyingShear*, page 35 and *RotaryKnife*, page 59 which provide nearly identical inputs and outputs that are described in this common chapter.

Pin Description

What's in This Chapter

Input Pin Description

Input Pin Description

Inputs and outputs (except *i_stParameters*) can be modified while the function block is active as they are refreshed cyclically. This is the main difference with the *ST_Parameters*, page 27.

Input	Data type	Description
i_Master	*PLCO.Axis_Ref*	The reference to the master axis.
i_Slave	*PLCO.Axis_Ref*	The reference to the slave axis.
i_stParameters	*ST_Parameters*, page 27	The structure containing the parameters that cannot be modified while the function block is active.
i_xEnable	BOOL	A rising edge of this input enables the function block (motion is not started). If the input is set to FALSE, the function block is disabled. Default value: FALSE
i_xStartOperation	BOOL	A rising edge of this input starts the motion depending on the starting mode and the operating mode. If the input is set to FALSE, the motion is stopped. Default value: FALSE
i_etStartMode	*ET_StartMode*, page 25	Defines the warm start mode. For further information, refer to Warm Start, page 44. Default value: *MoveToRestPosition*
i_xAbortInSync	BOOL	Exclusive to *FlyingShear*: A rising edge of this input aborts the synchronous phase. For further information, refer to Interruption of the Synchronous Phase, page 54.
i_xImmediateSync	BOOL	A rising edge of this input starts the synchronization of the slave axis to the master axis. For further information, refer to Immediate Synchronization, page 54. This input is only applicable in operating mode *Continuous*.
i_lrLengthToCut	LREAL	Defines the length of one process cycle. This input is only applicable in operating modes *Continuous* and *ContinuousWithCorrection*. Default value: 0 Value range: > *q_lrLengthToCutMin*, page 34 Also refer to the calculations of the Minimum Value for Length to Cut (*q_lrLengthToCutMin*), page 86.
i_xUseNextTp	BOOL	Upon a rising edge of this input, the next touch probe input will be accepted without considering the *lrTouchProbeWindow* range, page 49. This input is only applicable in Operating Mode *ContinuousWithCorrection*, page 49. Default value: FALSE
i_xTpDigitalInput	BOOL	Upon a rising edge of this input, the position of the master axis is captured. Also refer to Touch Probe Function to Capture the Position of the Master Axis, page 52. Default value: FALSE

Output Pin Description

Output Pin Description

Inputs and outputs can be modified while the function block is active as they are refreshed cyclically. This is the main difference to the *ST_Parameters*, page 27.

Output	Data Type	Description
q_xActive	BOOL	If the output is set to TRUE, the function block is active.
q_xReady	BOOL	If the output is set to TRUE, the initialization is successful and the function block is ready for operation.
q_xBusy	BOOL	If this output is set to TRUE, the slave axis is started.
q_xError	BOOL	If this output is set to TRUE, an error has been detected. For details, refer to *q_etResult* and *q_etResultMsg*.
q_etResult	*ET_Result*, page 23	Provides diagnostic and status information as a numeric value.
q_sResultMsg	STRING [80]	Provides additional diagnostic and status information as a text message.
q_xInSync	BOOL	If this output is set to TRUE, the slave axis is synchronized with the master axis (synchronous phase).
q_xCoupled	BOOL	If this output is set to TRUE, the slave axis is coupled with the master axis on the flying shear profile curve (= *MC_CamIn.InSync*).
q_xStartPosReached	BOOL	If this output is set to TRUE, the slave axis has reached the start position for the warm start.
q_xTpOutsideWindow	BOOL	If this output is set to TRUE, no registration mark has been detected inside the touch probe acceptance window. This output is only applicable in operating mode *ContinuousWithCorrection*, page 49. It is useful in applications to react on missing or invalid registration marks (for example, by counting invalid touch probe positions and stopping the process, if a limit is exceeded).
q_lrTpDistToSyncPointMin	LREAL	The minimum distance between the touch probe sensor and the start point of the synchronous phase. For calculating this parameter for the different operating modes, refer to the Calculations, page 85.
q_lrLengthToCutMin	LREAL	The minimum value for the length to cut (*i_lrLengthToCut*). This output is only applicable in operating modes *Continuous* and *ContinuousWithCorrection*. For calculating this parameter for the different operating modes, refer to the Calculations, page 86.

FlyingShear

What's in This Part

Functional and Machine Overview

What's in This Chapter

Functional Overview

Functional Description

The *FlyingShear* function block controls a machine that performs operations on a moving product.

Typical operations can include:

- Cutting
- Clamping
- Stamping
- Marking

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

NOTE: Most parameters involving distance, position, velocity, acceleration, torque values, etc., are in user-defined units unless otherwise indicated.

Master/Slave Axis Concept

This introduces the concept of a master and a slave axis using the cam functionality.

- The master axis moves the product forward.
- The slave axis is synchronized with the master axis during the working process to perform an operation on the product.
- The master axis must be defined as a rotary axis (modulo axis).

NOTICE
UNSATISFACTORY EQUIPMENT OPERATION Set the modulo value of the rotary master axis greater than the maximum distance between the products plus the value of the parameter *lrTpDistToSyncPoint* depending on the operating mode. **Failure to follow these instructions can result in equipment damage.**

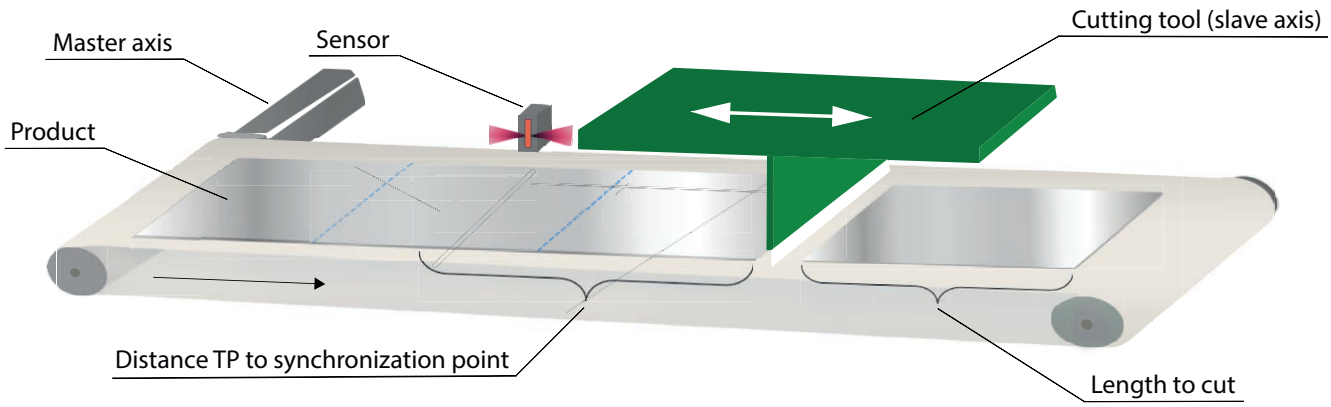
NOTE: For more information, refer to the *lrTpDistToSyncPoint* parameter description.

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

Machine Overview

Machine View

The figure provides an example of a flying shear application.



Function Block Description

What's in This Chapter

FlyingShear

Phases of the Flying Shear Process

A flying shear cycle consists of 3 phases:

Phase	Description
Start phase	The slave axis accelerates up to the velocity of the master axis (depending on the value of the parameter *lrSlope*) to be synchronized at the beginning of the synchronous phase. The movement from the rest position to the start point of the synchronous phase is calculated as follows (in units defined for the master axis): $\frac{ABS(ST\_Parameters.lrRestPosition)}{ST\_Parameters.lrSlope} * 1.875$ The parameter *lrRestPosition*, page 28 must have a value < 0.
Synchronous phase	The slave axis is synchronized with the master axis and the operation on the product is performed. The synchronous phase starts at master position 0 and slave position 0. A new process cycle is started with the start point of the synchronous phase.
Return phase	The slave axis is decoupled from the master axis and can continue operation in two different ways. • It returns to the rest position, page 39. • It continues with the next process cycle without reaching the rest position, page 39.

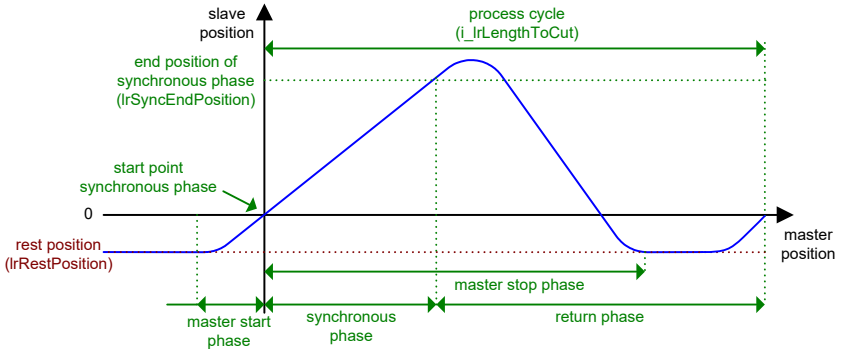
During the flying shear process, the slave axis can return to the rest position or short process cycles can be performed without returning to the rest position as described in the following sections.

Profile of Master/Slave Positions with Rest Position

In general, the process is such that the slave axis returns to the rest position according to the following procedure:

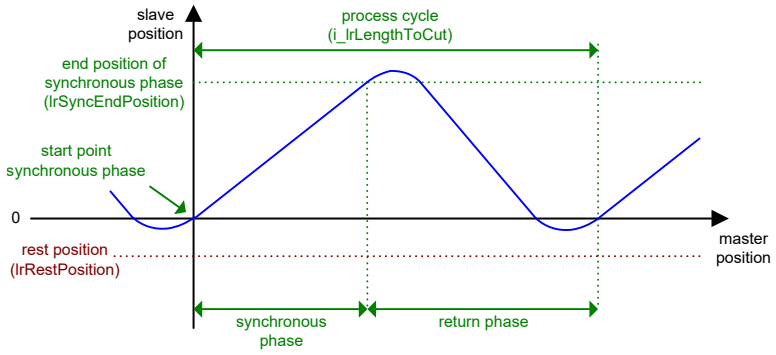
1. The slave axis starts in the rest position.
2. The slave axis accelerates to the velocity of the master axis.
3. The slave axis follows the master synchronously during the working process (synchronous phase).
4. The slave axis returns to the rest position.

The curve illustrates the position profile of the slave axis in relation to the master axis with return to the rest position.



Profile of Master/Slave Positions without Rest Position

The curve illustrates the position profile of the slave axis in relation to the master axis for short process cycles without returning to the rest position. The position profile of the slave is changed to short process cycles if the product length is shorter than *lrMasterStopPhase* + *lrMasterStartPhase* + 1.0.



NOTE: The total working area (including overshoot after synchronous phase) of the FlyingShear axis depends on its parameterization, for example, lrRestPosition, lrSyncEndPosition and lrLengthToCut. In order to help avoid mechanical damage by exceeding a defined limited working area, it is a good practice to limit movement by incorporating limit switches in your design to stop the axis if need be.

Pin Description

What's in This Chapter

Input and Output Pin Description

As the function blocks *FlyingShear*, page 35 and *RotaryKnife*, page 59 provide nearly identical inputs and outputs, they are described in the following common chapters:

- Input Pin Description, page 33
- Output Pin Description, page 34

Start and Stop and Starting Modes

What's in This Chapter

Start and Stop

Overview

The processing sequence of the function block is as follows:

- Enabling the function block (*i_xEnable* = TRUE).
- The function block is active (*q_xActive* = TRUE).
- Starting the movement (*i_xStartOperation* and *q_xReady* = TRUE).
- Stopping the movement (*i_xStartOperation* = FALSE or *i_xEnable* = FALSE).
- Disabling the function block (*i_xEnable* = FALSE).
- The function block is deactivated (*q_xActive* = FALSE).

Enabling the *FlyingShear* Function Block

Upon a rising edge of the input *i_xEnable*, the following actions are executed:

- The output *q_xActive* is set to TRUE.
- The references to the master and slave axis are verified and stored. Modifications while the function block is active are ignored.
- The parameter configuration is verified for validity.
- Cam data is calculated and curves are created.
- The outputs *q_lrTpDistToSyncPointMin* and *q_lrLengthToCutMin* are calculated.
- If no error is detected, the function block is ready to start the Flying Shear process and the output *q_xReady* is set to TRUE.
- If an error is detected, the output *q_xError* is set to TRUE and error information is provided at *q_etResult* and *q_sResultMsg*. A renewed execution of the function block is not possible as long as the error state is present. The function block must be disabled in order to reset the error state.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not send any other commands to the slave axis when it is under control of the *FlyingShear* function block.

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

Starting the Movement

Upon a rising edge of the input *i_xStartOperation* and when *q_xReady* is set to TRUE, the following actions are executed:

- The input parameters are verified for consistency.
- In cold start: The slave is moved to the *lrRestPosition*.

  In warm start, the action depends on the warm start mode selected:

  MoveToRestPosition: The slave moves to the rest position and stores the FIFO values. The slave is coupled to the master axis and the start is performed at the next possible process cycle without loss of synchronization between master and slave.

  MoveToCurvePosition: The slave is moved to the position on the curve.

  ContinueWithOffsetCurve: The new curve is calculated with offset.
- If no error is detected, the output *q_xBusy* is set to TRUE.
- If an error is detected, the output *q_xError* is set to TRUE and error information is provided at *q_etResult* and *q_sResultMsg*.
- After the slave has reached *lrRestPosition* or the curve position, depending on the start mode, the corresponding curve is loaded and the cam is activated.
- Modifiable inputs are verified during the movement.

Stopping the Movement

Upon a falling edge of the input *i_xStartOperation*, the slave does not stop immediately or remain stationary. The following actions are executed:

- Depending on the process phase:
  - Outside the synchronous phase: The slave is uncoupled from the master and returns to the rest position (*lrRestPosition*).
  - Inside the synchronous phase: The slave remains coupled with the master until the end of the synchronous phase. Then the slave is uncoupled from the master and returns to the rest position (*lrRestPosition*).
- The output *q_xBusy* is set to FALSE.

Immediately Stopping the Movement

The slave is stopped immediately with the deceleration ramp *lrStopDeceleration*, page 28 if one of the following conditions applies:

- An error is detected.
- A falling edge of the input *i_xEnable*.
- The FIFO buffer is full. Also refer to FIFO Buffer Saving Touch Probe Positions, page 52.

Disabling the *FlyingShear* Function Block

Upon a falling edge of the input *i_xEnable*, the following actions are executed:

- The output *q_xReady* is set to FALSE.
- If the slave is moving, it is stopped immediately with the deceleration ramp *lrStopDeceleration*.
- The output variables are reset after the slave has come to a halt.
- A cold start is prepared as no warm start is possible after disabling.
- The FIFO buffer is reset, the stored touch probe positions are deleted.
- After successful disabling the function block, the output *q_xActive* is set to FALSE.

Starting Modes

Starting Modes Available

Two types of starting modes are available:

- Cold Start, page 43
- Warm Start, page 44

Differences Between Cold Start and Warm Start

The main differences between cold start and warm start in operating modes *ContinuousWithCorrection* and *CutOnTouchProbe* are as follows:

- When a cold start is performed, the FIFO buffer is empty and no touch probe position is available. The first product to be processed will be the product that is detected first. Thus, if the position sensor is located five products before the processing tool, the five intermediate products will not be processed.
- When a warm start is performed, the FIFO buffer contains touch probe position data. Thus, the application can resume operation without omitting products.

Cold Start

Overview

The cold start is executed directly after the activation for the first execution of the function block. It is necessary whenever there is no context available from a previous execution of the application function block, such that no touch probe positions are available inside the buffer.

Upon the first rising edge of *i_xStartOperation* after activation of the function block (rising edge of the input *i_xEnable*), a cold start is executed. The input *i_etStartMode* is ignored. The slave axis is moved to the rest position. After it has reached the rest position, it is coupled to the master axis and starts the movement depending on the operating mode, page 47.

In a *Continuous* operating mode, the slave axis starts the flying shear process when the master axis is started. In *ContinuousWithCorrection* or *CutOnTouchProbe* operating modes, the first product to be processed will be the product that is detected first.

NOTE: Products located between the sensor and the processing tool are not processed after cold start.

Warm Start

Overview

A warm start is the starting method used for a re-start of the function block following a temporary interruption (upon a falling edge on *i_xStartOperation*), an error-stop or an emergency stop without de-activation of the application function block. The warm start mode according to the input *i_etStartMode* is applied.

The previous context is preserved (touch probe positions captured inside the buffer) and considered for the restart of the movement. Thus, in operating modes *ContinuousWithCorrection* and *CutOnTouchProbe* the application can resume using valid positions from the buffer.

Warm Start Modes

Three warm start modes are available depending on the application requirements. In the three modes, the master position determines whether the curve position is inside or outside the synchronous phase. For each warm start mode, certain conditions must be fulfilled. The warm start mode is selected with the input *i_etStartMode*, page 33.

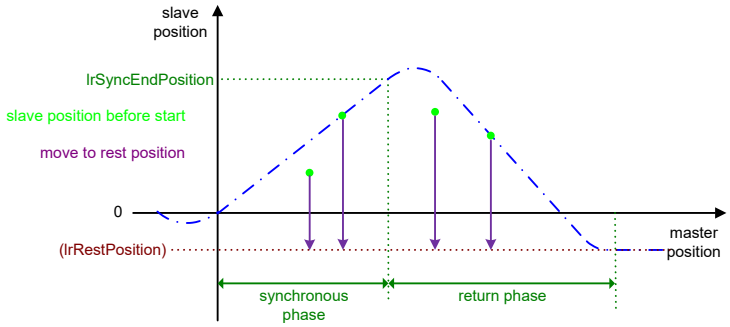
Warm start modes	Description
MoveToRestPosition	The *MoveToRestPosition* mode is the only warm start that can be applied to a moving master axis. The *MoveToRestPosition* mode can be used if the warm start modes *MoveToCurvePosition* and *ContinueWithOffsetCurve* are not applicable. As initial step, the slave axis is moved to the rest position.
MoveToCurvePosition *ContinueWithOffsetCurve*	These warm start modes can only be executed if the master axis is not moving. Otherwise an error message is created. These warm start modes are mainly used when there is a mechanical constraint between master axis and slave axis during the synchronization phase (such as a blade cutting into a metal sheet). In case of unintentional de-coupling of the master axis from the slave axis during the synchronization phase (for example, emergency stop), the slave axis can be re-coupled with the master axis according to different conditions.

Warm Start Mode *MoveToRestPosition*

The *MoveToRestPosition* mode is the only warm start mode that can be applied to a moving master axis.

Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The slave axis is moved to the rest position. The output *q_xStartPosReached* indicates that the warm start position has been reached. Thus, the master axis can start the movement.
2. The slave axis is coupled to the master axis.
3. The next process cycle starts, depending on the operating mode.



Warm Start Mode *MoveToCurvePosition*

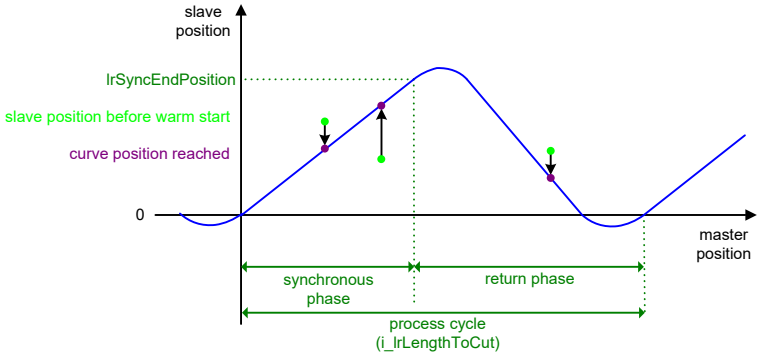
With the *MoveToCurvePosition* warm start mode, the slave axis returns to the position on the loaded curve corresponding to the position of the master axis.

The master axis must remain in the PLCopen state standstill until the slave axis has reached the position on the curve. If the master axis is moving or started before the slave axis has reached the position on the curve, the error *MasterAxisIsNotStandstill* is created.

Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The slave axis returns to the position on the curve.
2. The interrupted process cycle is continued as soon as the master axis has been restarted.

The output *q_xStartPosReached* indicates that the slave axis has reached the position on the curve and the master axis can start the movement. If the slave axis leaves this position, the output *q_xStartPosReached* is set to FALSE.



Warm Start Mode *ContinueWithOffsetCurve*

The slave axis continues with an offset or returns to the position of the active CAM profile depending on the position of the master axis. To continue with an offset curve, the slave axis must be within the synchronous phase (see scenario 1 below).

The master axis must remain in the PLCopen state standstill until the slave axis has reached the position on the curve. If the master axis is moving or started before the slave axis has reached the position on the curve, the error *MasterAxisIsNotStandstill* is created. The process is different, depending on whether the master axis is inside the synchronous phase.

Scenario 1: The master and slave axis are inside the synchronous phase:

Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The output *q_xStartPosReached* is set to TRUE.
2. The slave axis remains in the position offset to the curve during the synchronous phase.
3. After the synchronous phase, the slave axis returns back to the position on the curve.
4. The interrupted process cycle is continued as soon as the master axis is restarted.

NOTE: The total working area (including overshoot after synchronous phase) of the axis depends on its parameterization. In order to help avoid mechanical damage by exceeding a defined limited working area, it is a good practice to limit movement by incorporating limit switches in your design to stop the axis if need be.

Scenario 2: The master and slave axis are outside the synchronous phase:

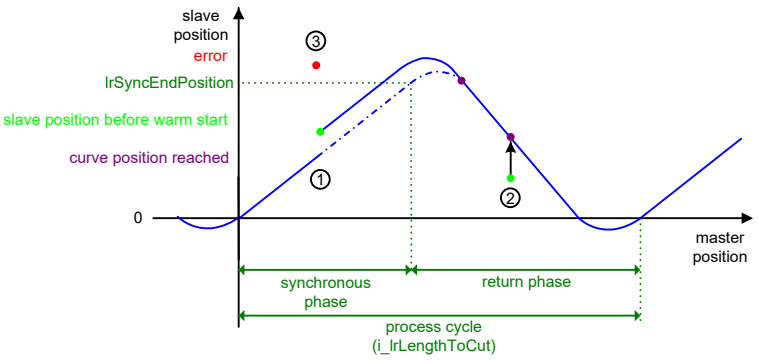
Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The slave axis returns to the position on the curve.

   The output *q_xStartPosReached* indicates that the slave axis has reached the position on the curve and the master axis can start the movement. If the slave axis leaves this position, the output *q_xStartPosReached* is set to FALSE.
2. The interrupted process cycle is continued as soon as the master axis is restarted.

Scenario 3: The master axis is inside and the slave axis is outside the synchronous phase:

If the slave axis is outside the synchronous phase, an error is detected and a corresponding message is presented by the function block. In this case, execute the warm start mode *MoveToCurvePosition* or a cold start.



Operating Modes

What's in This Chapter

Overview of the Flying Shear Process

In general, the flying shear process is such that the slave axis returns to the rest position at every process cycle. For further information, refer to Profile of Master/Slave Positions with Rest Position, page 39.

With short process cycles, the slave does not return to the rest position but accelerates before reaching the rest position to start the next synchronous phase. For further information, refer to Profile of Master/Slave Positions without Rest Position, page 39.

This chapter provides further information on the process cycles depending on the selected operating mode (*ET_OperatingMode*, page 25). Additional functions are described in the section Functions for Operating Modes *ContinuousWithCorrection* and *CutOnTouchProbe*, page 52 at the end of this chapter.

Operating Mode *Continuous*

Overview

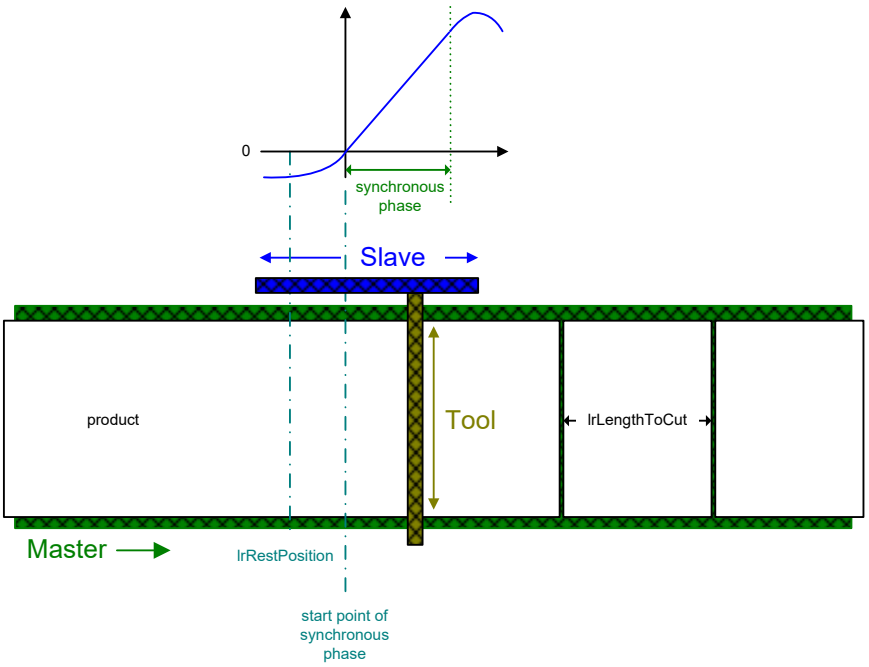
The operating mode *Continuous* is used to process products with a constant length. This product length corresponds to the length of one process cycle and is defined via the input *i_lrLengthToCut*, page 33.

Since the inputs of the function block are refreshed cyclically, the length can be modified during the running application.

NOTE: To help ensure that a change of the product length is considered for the next product cycle, modify the length to cut during the synchronous phase of the process cycle. Although it is possible to change length to cut outside of the synchronous phase, it may be that the modification would not be activated in next product cycle, but in the cycle thereafter.

The modified length is verified with each process cycle:

If the value is ...	**Then...**
valid: >= *q_lrLengthToCutMin*	The next product is processed with the new value of *i_lrLengthToCut*.
invalid: <*lrMinLengthToCutMin*	• An error is detected. • The slave stops with the ramp *lrStopDeceleration*, page 28.



Operating Mode *ContinuousWithCorrection*

Overview

The operating mode *ContinuousWithCorrection* provides the same functionality as the operating mode *Continuous*, plus an automatic correction of the product length, calculated from a position capture.

The product length correction is designed for applications processing products with small differences in length or products not rigid and strechable (for example, plastic film). To manage different product lengths, a sensor is placed on the master axis to capture the position of registration marks on the product. In combination with a touch probe function, the registration marks define the distances between the products.

For further information, refer to Touch Probe Function to Capture the Position of the Master Axis, page 52.

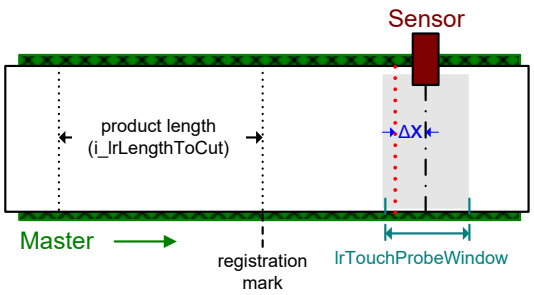
Touch Probe Acceptance Window: *lrTouchProbeWindow*

The touch probe acceptance window allows you to limit the product length allowed. The length of the product must be within the touch probe acceptance window that corresponds to the parameter *lrTouchProbeWindow*, page 28. It defines a range (that equals to half of the *lrTouchProbeWindow* before and half of the *lrTouchProbeWindow* after the sensor) in which detected touch probe signals are valid. The minimum value of this parameter is 0 and the maximum value is *i_lrLengthToCut* .

> **NOTE:** Do not modify the value of the parameter *lrTouchProbeWindow* while the function block is active. Modifications of *ST_Parameters* values during operation are ignored and the function block continues to use the value from the last activation.

Scenario 1: A touch probe signal is detected inside the touch probe acceptance window:

If a touch probe signal is valid (a registration mark has been detected inside the *lrTouchProbeWindow* range), the length of the product is corrected from the expected length (*i_lrLengthToCut*) to the detected length, i.e. the distance between the two registration marks. For further information on calculating the minimum value for the length to cut, refer to Minimum Value for Length to Cut (*q_lrLengthToCutMin*), page 86.

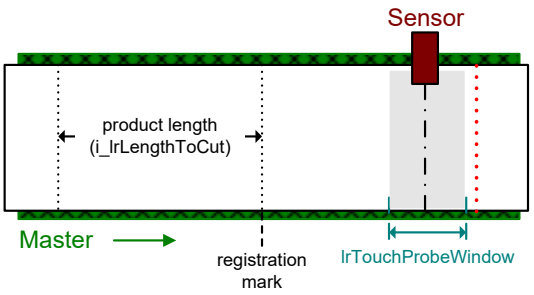


Registration marks detected outside the *lrTouchProbeWindow* are ignored.

Scenario 2: NO touch probe signal is detected inside the touch probe acceptance window:

If no registration mark is detected within the window:

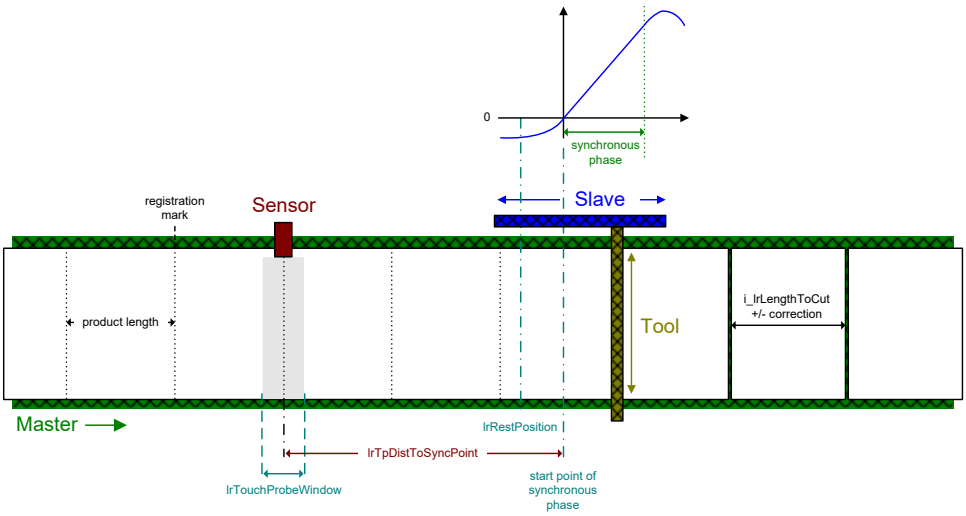
- The length of the product is set to the value of *i_lrLengthToCut*.
- The output *q_xTpOutsideWindow* is set to TRUE to indicate that no registration mark has been detected inside the touch probe acceptance window.
- The output is reset to FALSE after the next call of the function block.



Upon a rising edge of the input *i_xUseNextTp*, page 33, the next touch probe signal becomes valid without considering the *lrTouchProbeWindow* range.

lrTpDistToSyncPoint for Calculation and Validation

For the calculation of the curves and the validation of the touch probe signals, the parameter *lrTpDistToSyncPoint*, page 28 is required. This parameter defines the distance between the center point of the sensor and the start point of the synchronous phase.



Operating Mode *CutOnTouchProbe*

Overview

The operating mode *CutOnTouchProbe* is used for processing products with variable length and format calculated from a position capture. The length of the process cycle is defined via registration marks corresponding to the distances between the products. These distances are calculated by the touch probe positions in connection with a touch probe ignorance window.

Touch Probe Ignorance Window: *IrTouchProbeWindow*

The touch probe ignorance window corresponds to the parameter *IrTouchProbeWindow*, page 28. It defines a range after the sensor in which detected touch probe signals are ignored as long as the registration mark of the previous touch probe signal is inside this window.

A new touch probe signal is only valid if the touch probe position is greater than the last touch probe position plus the value of the parameter *IrTouchProbeWindow*:

touch probe position > last touch probe position + touch probe window

The minimum value of the parameter *IrTouchProbeWindow* is calculated depending on the parameters of the function block configuration:

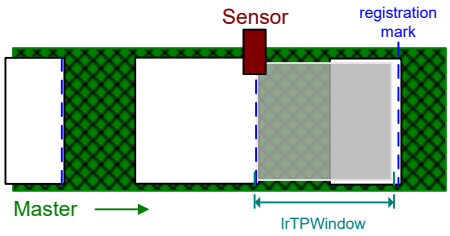
master start phase + synchronous end position

For further information on calculating the master start phase, refer to Master Start Phase, page 85.

> **NOTE:** Do not modify the value of the parameter *IrTouchProbeWindow* while the function block is active. Modifications of *ST_Parameters* values during operation are ignored and the function block continues to use the value from the last activation.

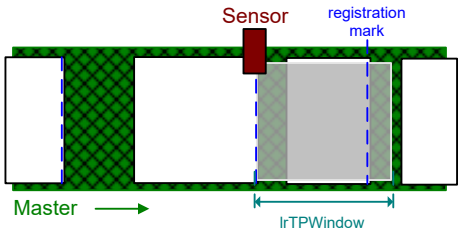
Scenario 1: The touch probe signal is valid:

new touch probe position - last touch probe position = outside the touch probe window



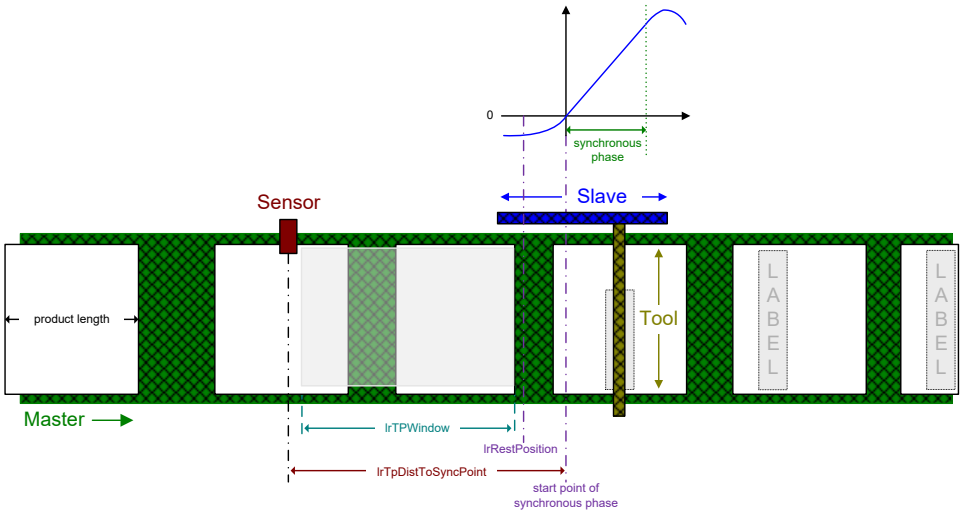
Scenario 2: The touch probe signal is invalid:

new touch probe position - last touch probe position = inside the touch probe window



lrTpDistToSyncPoint for Calculation and Validation

For the calculation of the curves and the validation of the touch probe signals, the parameter *lrTpDistToSyncPoint*, page 28 is required. This parameter defines the distance between the center point of the sensor and the start point of the synchronous phase.



Functions for Operating Modes *ContinuousWithCorrection* and *CutOnTouchProbe*

Overview

For the operating modes *ContinuousWithCorrection* and *CutOnTouchProbe*, specific functions are available.

Touch Probe Function to Capture the Position of the Master Axis

It is possible to capture the position of the master axis for calculating the distance between two products in units defined for the master axis as follows:

- Via a digital input by setting the parameter *ifTriggerRef* to 0:

  Upon a rising edge at the input *i_xTpDigitalInput*, the present position of the master axis is captured and used as touch probe position.

- Via a capture unit by setting the parameter *ifTriggerRef* = *DRV_Lexium32S.triggerCap1* / *Cap2* / *Cap3*:

  The position of the master axis is captured via the configured capture input (number of the capture input and rising / falling edge) of the master axis. For further information, refer to Activating Capture Inputs, page 87.

FIFO Buffer Saving Touch Probe Positions

The function block provides a FIFO buffer to save the touch probe positions of products if more than one product is detected by the sensor before the product can be processed. The maximum number of products that can be stored in the FIFO buffer is defined with the parameter *GPL.Gc_usiFiFoMaxNumberOfProducts*, page 31.

The FIFO buffer is activated when the slave axis reaches the rest position after a cold start.

The FIFO buffer is deactivated and the entries are deleted with a falling edge at the input *i_xEnable*, page 33.

With a falling edge at the input *i_xStartOperation*, page 33, the slave moves to the rest position. The FIFO buffer as well as the touch probe function are kept active allowing the process to continue without operator intervention upon a rising edge at *i_xStartOperation* and allowing restart of the motion. New captured positions are written to the FIFO buffer and at every start of a new process cycle, the corresponding position is read from the FIFO. The FIFO and the touch probe function continue operation as if the slave axis was moving.

If the maximum number of products that can be stored in the FIFO buffer (*GPL. Gc_usiFiFoMaxNumberOfProducts*) is exceeded, the function block detects an error and stops the slave axis. The last captured position is not stored.

If the function block becomes inactive, the entries of the FIFO buffer are deleted.

Special Functions and Modes

What's in This Chapter

General Information

As a prerequisite for executing special functions, the function block must be active.

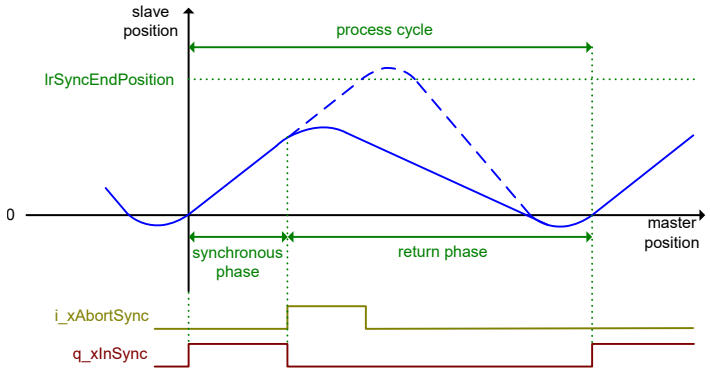
If the function block is disabled while a function is active, the function is also disabled and needs to be restarted.

Interruption of the Synchronous Phase

Overview

As a prerequisite for this function, the process must be in the synchronous phase.

A rising edge at the input *i_xAbortSync*, page 33 aborts the synchronous phase: The slave returns to the rest position or continues with the next process cycle without modifying the length of the process cycle.



Immediate Synchronization

Overview

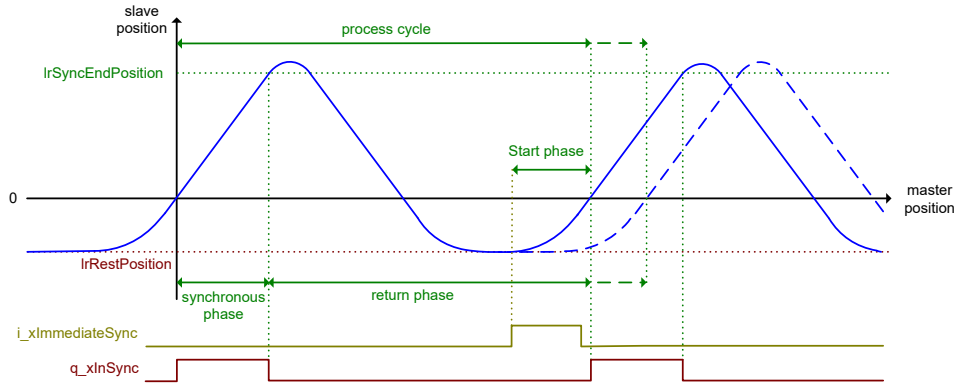
Prerequisites for this function:

- The slave is in rest position.
- The operating mode is *Continuous*.

A rising edge of the input *i_xImmediateSync*, page 33 starts a new process cycle by initiating the synchronization of the slave to the master.

After the *lrMasterStartPhase*, the synchronous phase starts and the input *i_lrLengthToCut* is used for the length of the process cycle.

A rising edge of the input *i_xImmediateSync* is ignored when the slave is not in *lrRestPosition*.



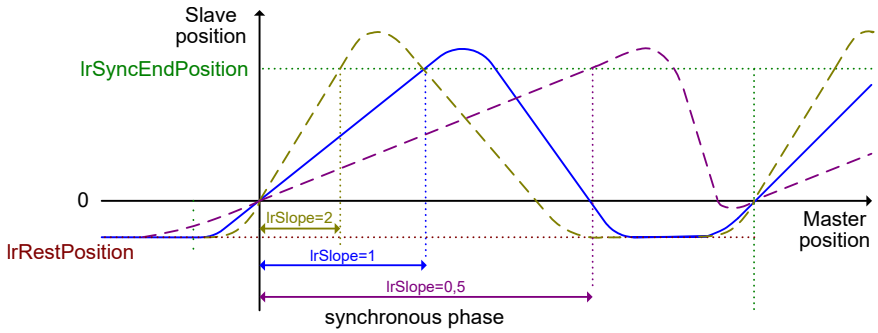
Slope of the Synchronous Phase

Overview

As a prerequisite for modifying the slope of the synchronous phase, the function block must be inactive.

Use the parameter *lrSlope*, page 28 to introduce a slope coefficient for the synchronous phase:

- *lrSlope* = 1: The master and the slave axis are moving with the same velocity.
- *lrSlope* > 1: The slave is moving slower than the master.
- *lrSlope* < 1: The slave is moving faster than the master.



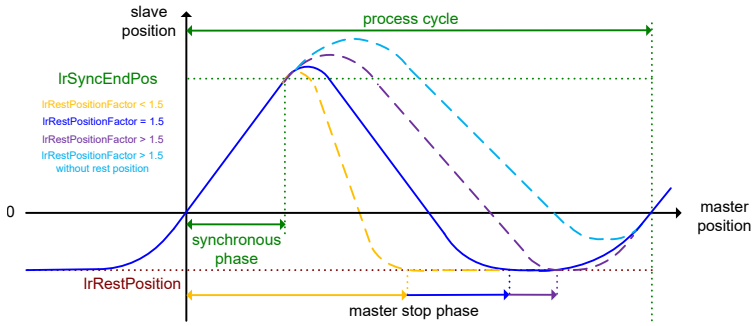
Rest Position Factor

Overview

The parameter *lrRestPositionFactor* allows you to modify the distance between the synchronous start position and the point when the slave axis reaches the rest position (master stop phase).

- The smaller the value, the sooner the slave axis reaches the rest position. This short master stop phase can lead to errors detected in the slave axis (following errors that are originally generated by the drive).
- The higher the value, the later the slave axis reaches the rest position. This long master stop phase leads to a smoother movement and can result in a profile without rest position, page 39.

For calculating the master stop phase, refer to the Calculations, page 85.


slave
position
process cycle
lrSyncEndPos
lrRestPositionFactor < 1.5
lrRestPositionFactor = 1.5
lrRestPositionFactor > 1.5
lrRestPositionFactor > 1.5
without rest position
0
master
position
synchronous
phase
lrRestPosition
master stop phase

Quick Reference Guide

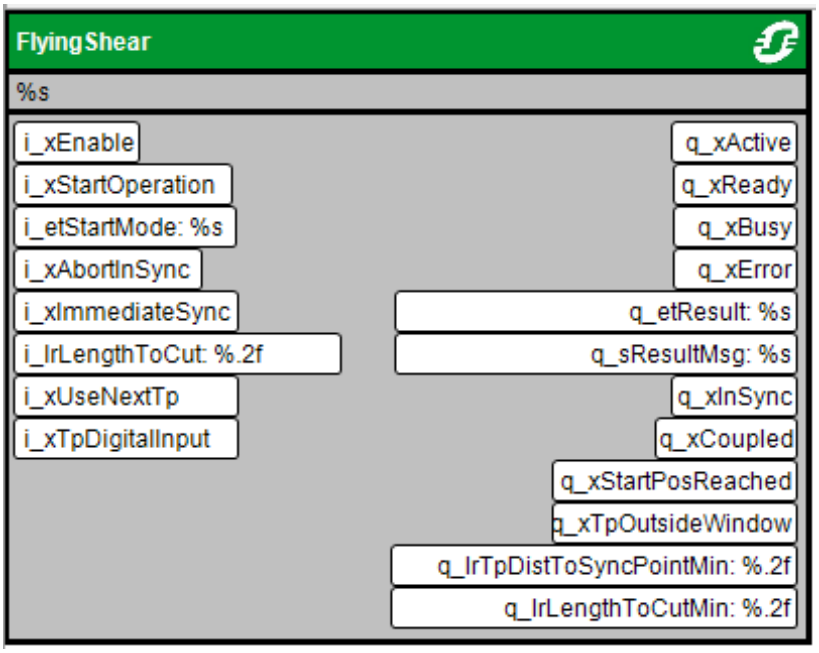
What's in This Chapter

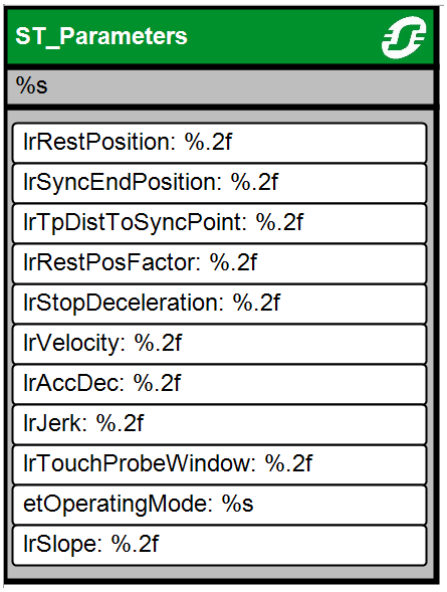
Visualization

Overview

Visualization screens are embedded in the MotionApplicationFunctionBlocks library to configure and start the function block.

Inputs and outputs as well as *ST_Parameters* are accessible.





Troubleshooting

Types of Detected Errors

The function block can detect two types of errors:

Type of detected error	Example	Description	Reset
Axis errors	Following error	The function block reacts on detected axis errors and provides the dedicated information.	Axis errors must be reset outside the function block.
Function block errors	A parameter is outside the value range.	These errors are detected by the function block when: • the *ST_Parameters*, page 27 are verified (upon a rising edge of the input *i_xEnable*). • the inputs, page 33 are verified (upon a rising edge of the input *i_xStartOperation*) or modified during the operation.	Function block errors are reset with a falling edge of the input *i_xEnable* or *i_xStartOperation*.

Notification

If an error is detected, the output *q_xError* is set to TRUE and error information is provided at *q_etResult* and *q_sResultMsg*. The output *q_xError* remains TRUE and subsequent errors are not indicated until the function block is disabled and the error state is reset.

In general, two types of errors can be detected:

- Axis errors (such as following errors)

  The function block detects errors that are related to the axis and provides the corresponding error information but does not reset them. The reset must be performed outside the function block.

- Function block-specific errors (such as parameters outside the value range)

  The function block detects such errors while verifying the parameters (with a rising edge of the input *i_xEnable*) or while verifying the inputs (with a rising edge of the input *i_xStartOperation*) or while modifying during the operation. These errors are reset upon a falling edge of the input *i_xEnable* or *i_xStartOperation* respectively.

As a consequence, the function block does not have a specific input to reset detected errors.

If an error is detected while the axis is moving, the movement is stopped immediately with the deceleration ramp (*lrStopDeceleration*) without considering the process cycle.

RotaryKnife

What's in This Part

Functional and Machine Overview

What's in This Chapter

Functional Overview

Functional Description

The *RotaryKnife* function block controls a machine that performs operations on a moving product.

Typical operations can include:

- Cutting
- Sealing
- Marking

The rotary knife function is required for moving the operational axis to synchronize it with the forward motion of the product.

NOTE: Most parameters involving distance, position, velocity, acceleration, torque values, etc., are in user-defined units unless otherwise indicated.

Master/Slave Axis Concept

This introduces the concept of a master and a slave axis using the cam functionality.

- The master axis moves the product forward.
- The slave axis is synchronized with the master axis during the working process to perform an operation on the product.
- The master axis must be defined as a rotary axis (modulo axis).

NOTICE
UNSATISFACTORY EQUIPMENT OPERATION Set the modulo value of the rotary master axis greater than the maximum distance between the products plus the value of the parameter *IrTpDistToSyncPoint* depending on the operating mode. **Failure to follow these instructions can result in equipment damage.**

NOTE: For more information, refer to the *IrTpDistToSyncPoint* parameter description.

- The slave axis must be configured as a rotary axis.

NOTICE
UNSATISFACTORY EQUIPMENT OPERATION Set the modulo value of the slave axis equal to the value of the *IrPerimeter*. **Failure to follow these instructions can result in equipment damage.**

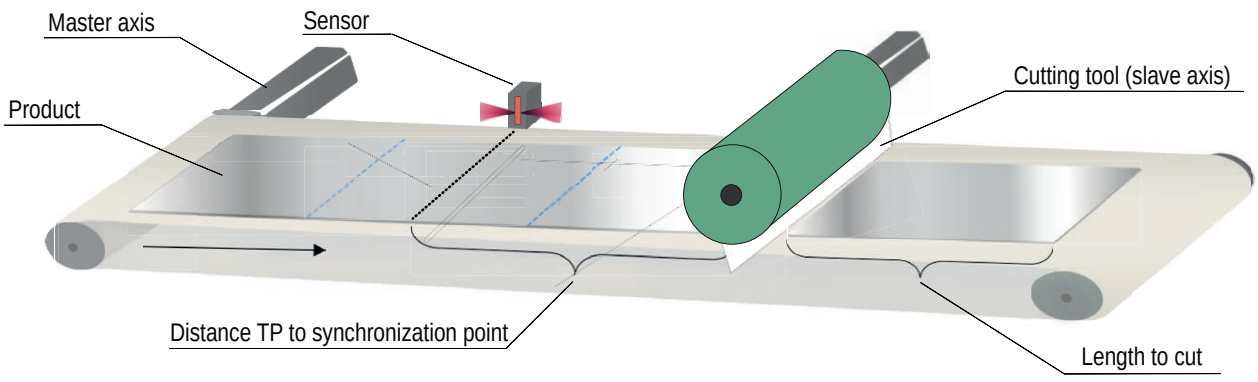
NOTE: For more information, refer to the *IrPerimeter* parameter description.

NOTE: Although the rotary knife function can be used for many different applications, the examples provided in this document refer to a typical cutting application.

Machine Overview

Machine View

The figure provides an example of a rotary knife application.



Function Block Description

What's in This Chapter

RotaryKnife

Phases of the Rotary Knife Process

A rotary knife cycle consists of 3 phases:

Phase	Description
Start phase	The slave axis accelerates up to the velocity of the master axis (depending on the value of the parameter *lrSlope*) to be synchronized at the beginning of the synchronous phase. The movement from the rest position to the start point of the synchronous phase is calculated as follows (in units defined for the master axis): $\frac{(ST\_Parameters.lrPerimeter - ST\_Parameters.lrRestPosition)}{ST\_Parameters.lrSlope} * 1.875$ The parameter *lrRestPosition*, page 28 must have a value > *lrSyncEndPosition* < *lrPerimeter*.
Synchronous phase	The slave axis is synchronized with the master axis and the operation on the product is performed. The synchronous phase starts at master position 0 and slave position 0. A new process cycle is started with the start point of the synchronous phase.
Return phase	The slave axis is decoupled from the master axis and can continue operation in two different ways. • It returns to the rest position, page 63. • It continues with the next process cycle without reaching the rest position, page 63.

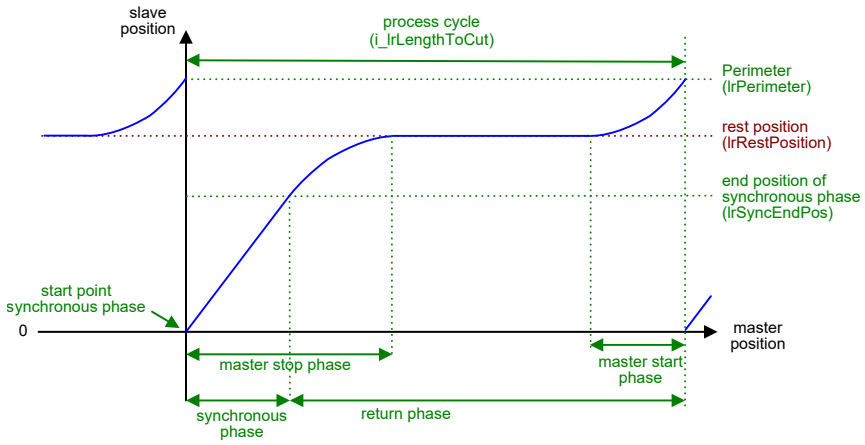
During the rotary knife process, the slave axis can return to the rest position or short process cycles can be performed without returning to the rest position as described in the following sections.

Profile of Master/Slave Positions with Rest Position

In general, the process is such that the slave axis returns to the rest position according to the following procedure:

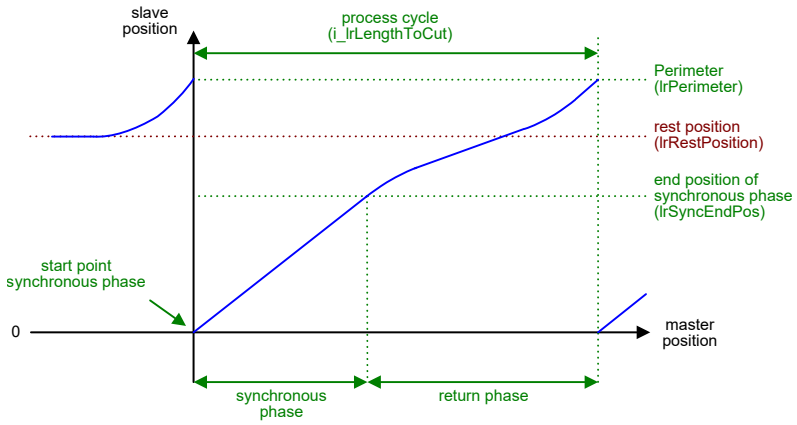
1. The slave axis starts in the rest position.
2. The slave axis accelerates to the velocity of the master axis.
3. The slave axis follows the master synchronously during the working process (synchronous phase).
4. The slave axis returns to the rest position.

The curve illustrates the position profile of the slave axis in relation to the master axis with return to the rest position.



Profile of Master/Slave Positions without Rest Position

The curve illustrates the position profile of the slave axis in relation to the master axis for short process cycles without returning to the rest position. The position profile of the slave is changed to short process cycles if the product length is shorter than *i_lrLengthToCut* < *lrMasterStartPhase* + *lrMasterStopPhase* + 1.0.



NOTE: The total working area (including overshoot after synchronous phase) of the RotaryKnife axis depends on its parameterization, for example, *lrRestPosition*, *lrSyncEndPosition* and *lrLengthToCut*. In order to help avoid mechanical damage by exceeding a defined limited working area, it is a good practice to limit movement by incorporating limit switches in your design to stop the axis if need be.

Movement to Rest Position

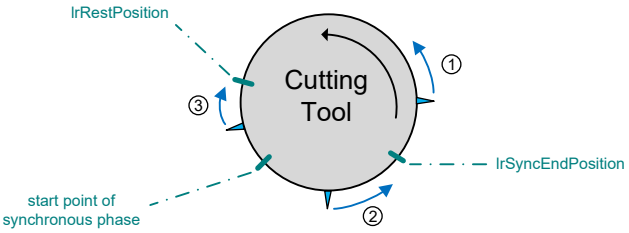
The slave axis moves to the rest position if one of the following situation applies:

- A cold start is executed.
- The warm start mode *MoveToRestPosition* is executed.
- The operation is stopped (*i_xStartOperation* = FALSE) when the slave axis is outside the synchronous phase.

As the slave axis must not move through the synchronous area, there are two different ways to reach the rest position depending on the position of slave axis:

- If the slave axis is after the end position of the synchronous phase (*lrSyncEndPos*), it moves in forward direction to reach the rest position.
- If the slave axis is before the start point of the synchronous phase (within the master start phase), it moves in backward direction to reach the rest position.

  NOTE: Backward movement of the rotary knife axis is only and exclusively allowed in this case.



1 After synchronous phase and before rest position: **forward** movement to rest position

2 Inside synchronous phase: **forward** movement to rest position

3 After rest position and before synchronous phase: **backward** movement to rest position

Pin Description

What's in This Chapter

Input and Output Pin Description

As the function blocks *FlyingShear*, page 35 and *RotaryKnife*, page 59 provide nearly identical inputs and outputs, they are described in the following common chapters:

- Input Pin Description, page 33
- Output Pin Description, page 34

Start and Stop and Starting Modes

What's in This Chapter

Start and Stop

Overview

The processing sequence of the function block is as follows:

- Enabling the function block (*i_xEnable* = TRUE).
- The function block is active (*q_xActive* = TRUE).
- Starting the movement (*i_xStartOperation* and *q_xReady* = TRUE).
- Stopping the movement (*i_xStartOperation* = FALSE or *i_xEnable* = FALSE).
- Disabling the function block (*i_xEnable* = FALSE).
- The function block is deactivated (*q_xActive* = FALSE).

Enabling the *RotaryKnife* Function Block

Upon a rising edge of the input *i_xEnable*, the following actions are executed:

- The output *q_xActive* is set to TRUE.
- The references to the master and slave axis are verified and stored. Modifications while the function block is active are ignored.
- The parameter configuration is verified for validity.
- Cam data is calculated and curves are created.
- The outputs *q_lrTpDistToSyncPointMin* and *q_lrLengthToCutMin* are calculated.
- If no error is detected, the function block is ready to start the rotary knife process and the output *q_xReady* is set to TRUE.
- If an error is detected, the output *q_xError* is set to TRUE and error information is provided at *q_etResult* and *q_sResultMsg*. A renewed execution of the function block is not possible as long as the error state is present. The function block must be disabled in order to reset the error state.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not send any other commands to the slave axis when it is under control of the *RotaryKnife* function block.

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

Starting the Movement

Upon a rising edge of the input *i_xStartOperation* and when *q_xReady* is set to TRUE, the following actions are executed:

- The input parameters are verified for consistency.
- In cold start: The slave is moved to the *lrRestPosition*.

  In warm start, the action depends on the warm start mode selected:

  MoveToRestPosition: The slave moves to the rest position and stores the FIFO values. The slave is coupled to the master axis and the start is performed at the next possible process cycle without loss of synchronization between master and slave.

  MoveToCurvePosition: The slave is moved to the position on the curve.

  ContinueWithOffsetCurve: The new curve is calculated with offset.
- If no error is detected, the output *q_xBusy* is set to TRUE.
- If an error is detected, the output *q_xError* is set to TRUE and error information is provided at *q_etResult* and *q_sResultMsg*.
- After the slave has reached *lrRestPosition* or the curve position, depending on the start mode, the corresponding curve is loaded and the cam is activated.
- Modifiable inputs are verified during the movement.
- If a movement in negative direction is required to execute the start mode, an error is detected. In this case, either the start mode *MoveToRestPosition* or a cold start is required.

  NOTE: Before executing a warm start, ensure that the slave axis is still in the same modulo period as the master. If the slave axis has moved outside the module period in which the desynchronization from the master axis occurred, a cold start is required.

Stopping the Movement

Upon a falling edge of the input *i_xStartOperation*, the slave does not stop immediately or remain stationary. The following actions are executed:

- Depending on the process phase:
  - Outside the synchronous phase: The slave is uncoupled from the master and returns to the rest position (*lrRestPosition*).
  - Inside the synchronous phase: The slave remains coupled with the master until the end of the synchronous phase. Then the slave is uncoupled from the master and returns to the rest position (*lrRestPosition*).
- The output *q_xBusy* is set to FALSE.

Immediately Stopping the Movement

The slave is stopped immediately with the deceleration ramp *lrStopDeceleration*, page 28 if one of the following conditions applies:

- An error is detected.
- A falling edge of the input *i_xEnable*.
- The FIFO buffer is full. Also refer to FIFO Buffer Saving Touch Probe Positions, page 77.

Disabling the *RotaryKnife* Function Block

Upon a falling edge of the input *i_xEnable*, the following actions are executed:

- The output *q_xReady* is set to FALSE.
- If the slave is moving, it is stopped immediately with the deceleration ramp *lrStopDeceleration*.
- The output variables are reset after the slave has come to a halt.
- A cold start is prepared as no warm start is possible after disabling.
- The FIFO buffer is reset, the stored touch probe positions are deleted.
- After successful disabling the function block, the output *q_xActive* is set to FALSE.

Starting Modes

Starting Modes Available

Two types of starting modes are available:

- Cold Start, page 68
- Warm Start, page 69

Differences Between Cold Start and Warm Start

The main differences between cold start and warm start in operating modes *ContinuousWithCorrection* and *CutOnTouchProbe* are as follows:

- When a cold start is performed, the FIFO buffer is empty and no touch probe position is available. The first product to be processed will be the product that is detected first. Thus, if the position sensor is located five products before the processing tool, the five intermediate products will not be processed.
- When a warm start is performed, the FIFO buffer contains touch probe position data. Thus, the application can resume operation without omitting products.

Cold Start

Overview

The cold start is executed directly after the activation for the first execution of the function block. It is necessary whenever there is no context available from a previous execution of the application function block, such that no touch probe positions are available inside the buffer.

Upon the first rising edge of *i_xStartOperation* after activation of the function block (rising edge of the input *i_xEnable*), a cold start is executed. The input *i_etStartMode* is ignored. The slave axis is moved to the rest position. After it has reached the rest position, it is coupled to the master axis and starts the movement depending on the operating mode, page 47.

In a *Continuous* operating mode, the slave axis starts the rotary knife process when the master axis is started. In *ContinuousWithCorrection* or *CutOnTouchProbe* operating modes, the first product to be processed will be the product that is detected first.

NOTE: Products located between the sensor and the processing tool are not processed after cold start.

Warm Start

Overview

A warm start is the starting method used for a re-start of the function block following a temporary interruption (upon a falling edge on *i_xStartOperation*), an error-stop or an emergency stop without de-activation of the application function block. The warm start mode according to the input *i_etStartMode* is applied.

The previous context is preserved (touch probe positions captured inside the buffer) and considered for the restart of the movement. Thus, in operating modes *ContinuousWithCorrection* and *CutOnTouchProbe* the application can resume using valid positions from the buffer.

Warm Start Modes

Three warm start modes are available depending on the application requirements. In all modes, the master position defines, if the curve position is inside or outside the synchronous phase. For each warm start mode, certain conditions must be fulfilled. The warm start mode is selected with the input *i_etStartMode*, page 33.

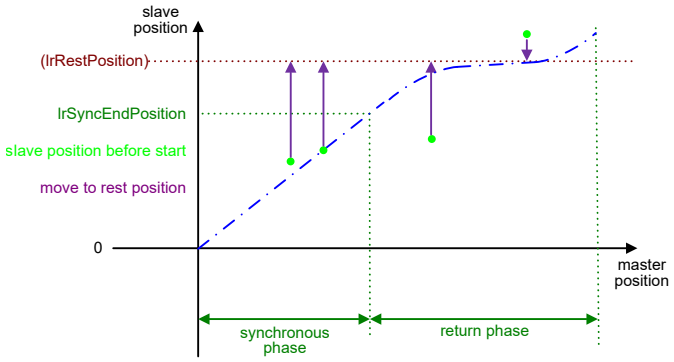
Warm start modes	Description
MoveToRestPosition	The *MoveToRestPosition* mode is the only warm start that can be applied to a moving master axis. The *MoveToRestPosition* mode can be used if the warm start modes *MoveToCurvePosition* and *ContinueWithOffsetCurve* are not applicable. As initial step, the slave axis is moved to the rest position.
MoveToCurvePosition *ContinueWithOffsetCurve*	These warm start modes can only be executed if the master axis is not moving. Otherwise an error message is created. These warm start modes are mainly used when there is a mechanical constraint between master axis and slave axis during the synchronization phase (such as a blade cutting into a metal sheet). In case of unintentional de-coupling of the master axis from the slave axis during the synchronization phase (for example, emergency stop), the slave axis can be re-coupled with the master axis according to different conditions.

Warm Start Mode *MoveToRestPosition*

The *MoveToRestPosition* mode is the only warm start mode that can be applied to a moving master axis.

Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The slave axis is moved to the rest position. The output *q_xStartPosReached* indicates that the warm start position has been reached. Thus, the master axis can start the movement.
2. The slave axis is coupled to the master axis.
3. The next process cycle starts, depending on the operating mode.



Warm Start Mode *MoveToCurvePosition*

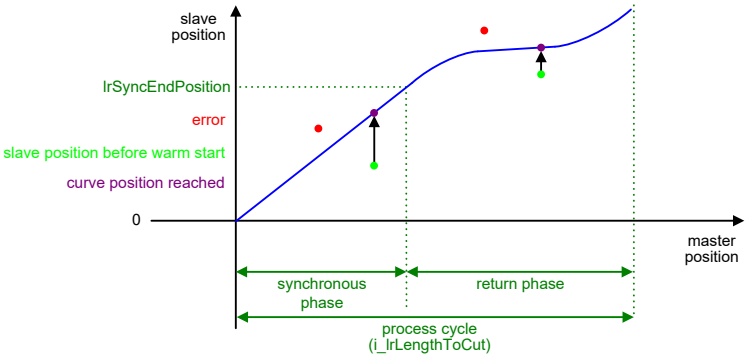
With the *MoveToCurvePosition* warm start mode, the slave axis returns to the position on the loaded curve corresponding to the position of the master axis exclusively in forward direction. If the position of the slave axis is greater than the position on the curve, the error *SlaveWarmStartPositionInvalid* is created.

The master axis must remain in the PLCopen state standstill until the slave axis has reached the position on the curve. If the master axis is moving or started before the slave axis has reached the position on the curve, the error *MasterAxisIsNotStandstill* is created.

Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The slave axis returns to the position on the curve.
2. The interrupted process cycle is continued as soon as the master axis has been restarted.

The output *q_xStartPosReached* indicates that the slave axis has reached the position on the curve and the master axis can start the movement. If the slave axis leaves this position, the output *q_xStartPosReached* is set to FALSE.



Warm Start Mode *ContinueWithOffsetCurve*

The slave axis continues with an offset or returns to the position on the loaded curve depending on the position of the master axis. To continue with an offset curve, the slave axis must be within the synchronous phase (see scenario 1 below).

The master axis must remain in the PLCopen state standstill until the slave axis has reached the position on the curve. If the master axis is moving or started before the slave axis has reached the position on the curve, the error *MasterAxisIsNotStandstill* is created.

The process is different, depending on whether the master axis is inside the synchronous phase.

Scenario 1: The master and slave axis are inside the synchronous phase:

Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The output *q_xStartPosReached* is set to TRUE.
2. The slave axis remains in the position offset to the curve during the synchronous phase.
3. After the synchronous phase, the slave axis returns back to the position on the curve.
4. The interrupted process cycle is continued as soon as the master axis is restarted.

NOTE: The total working area (including overshoot after synchronous phase) of the axis depends on its parameterization. In order to help avoid mechanical damage by exceeding a defined limited working area, it is a good practice to limit movement by incorporating limit switches in your design to stop the axis if need be.

Scenario 2: The master and slave axis are outside the synchronous phase:

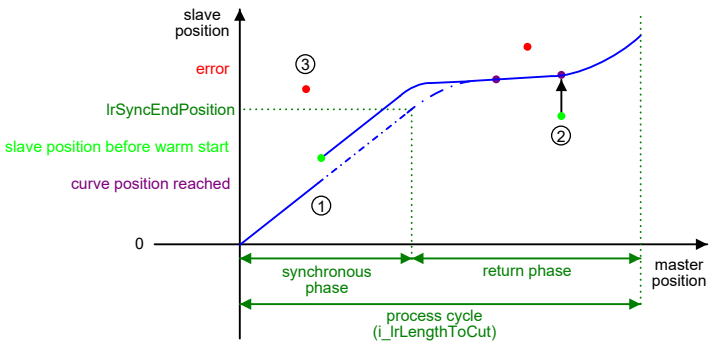
Upon a rising edge at the input *i_xStartOperation*, the following process is initiated:

1. The slave axis returns to the position on the curve exclusively in forward direction. If the position of the slave axis is greater then the position on the curve, an error message is created.

   The output *q_xStartPosReached* indicates that the slave axis has reached the position on the curve and the master axis can start the movement. If the slave axis leaves this position, the output *q_xStartPosReached* is set to FALSE.
2. The interrupted process cycle is continued as soon as the master axis is restarted.

Scenario 3: The master axis is inside and the slave axis is outside the synchronous phase:

If the slave axis is outside the synchronous phase, the error message *SlaveWarmStartPositionInvalid* is created. In this case, execute the warm start mode *MoveToCurvePosition* or a cold start.



Operating Modes

What's in This Chapter

Overview of the Rotary Knife Process

In general, the rotary knife process is such that the slave axis returns to the rest position at every process cycle. For further information, refer to Profile of Master/Slave Positions with Rest Position, page 63.

With short process cycles, the slave does not return to the rest position but accelerates before reaching the rest position to start the next synchronous phase. For further information, refer to Profile of Master/Slave Positions without Rest Position, page 63.

This chapter provides further information on the process cycles depending on the selected operating mode (*ET_OperatingMode*, page 25). Additional functions are described in the section Functions for Operating Modes *ContinuousWithCorrection* and *CutOnTouchProbe*, page 77 at the end of this chapter.

Operating Mode *Continuous*

Overview

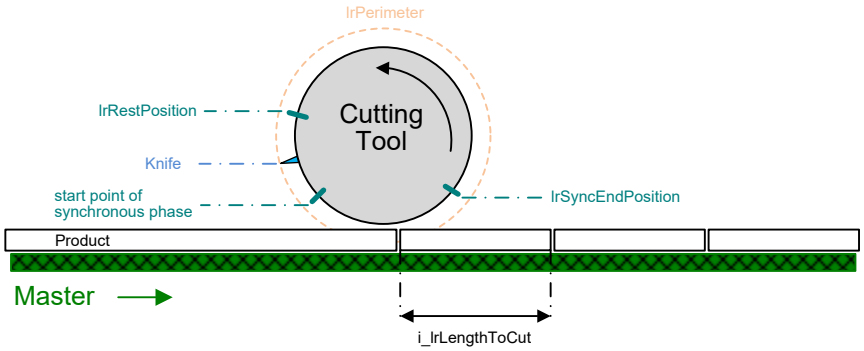
The operating mode *Continuous* is used to process products with a constant length. This product length corresponds to the length of one process cycle and is defined via the input *i_lrLengthToCut*, page 33.

Since the inputs of the function block are refreshed cyclically, the length can be modified during the running application.

NOTE: To help ensure that a change of the product length is considered for the next product cycle, modify the length to cut during the synchronous phase of the process cycle. Although it is possible to change length to cut outside of the synchronous phase, it may be that the modification would not be activated in next product cycle, but in the cycle thereafter.

The modified length is verified with each process cycle:

If the value is ...	Then...
valid: >= *q_lrLengthToCutMin*	The next product is processed with the new value of *i_lrLengthToCut*.
invalid: <*lrMinLengthToCutMin*	• An error is detected. • The slave stops with the ramp *lrStopDeceleration*, page 28.



Operating Mode *ContinuousWithCorrection*

Overview

The operating mode *ContinuousWithCorrection* provides the same functionality as the operating mode *Continuous*, plus an automatic correction of the product length, calculated from a position capture.

The product length correction is designed for applications processing products with small differences in length or products not rigid and strechable (for example, plastic film). To manage different product lengths, a sensor is placed on the master axis to capture the position of registration marks on the product. In combination with a touch probe function, the registration marks define the distances between the products.

For further information, refer to Touch Probe Function to Capture the Position of the Master Axis, page 77.

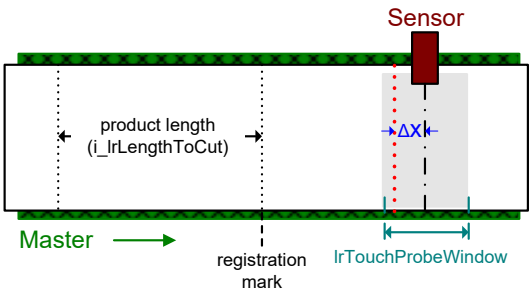
Touch Probe Acceptance Window: *lrTouchProbeWindow*

The touch probe acceptance window allows you to limit the product length allowed. The length of the product must be within the touch probe acceptance window that corresponds to the parameter *lrTouchProbeWindow*, page 28. It defines a range (that equals to half of the *lrTouchProbeWindow* before and half of the *lrTouchProbeWindow* after the sensor) in which detected touch probe signals are valid. The minimum value of this parameter is 0 and the maximum value is *i_lrLengthToCut* .

NOTE: Do not modify the value of the parameter *lrTouchProbeWindow* while the function block is active. Modifications of *ST_Parameters* values during operation are ignored and the function block continues to use the value from the last activation.

Scenario 1: A touch probe signal is detected inside the touch probe acceptance window:

If a touch probe signal is valid (a registration mark has been detected inside the *lrTouchProbeWindow* range), the length of the product is corrected from the expected length (*i_lrLengthToCut*) to the detected length, i.e. the distance between the two registration marks. For further information on calculating the minimum value for the length to cut, refer to Minimum Value for Length to Cut (*q_lrLengthToCutMin*), page 86.

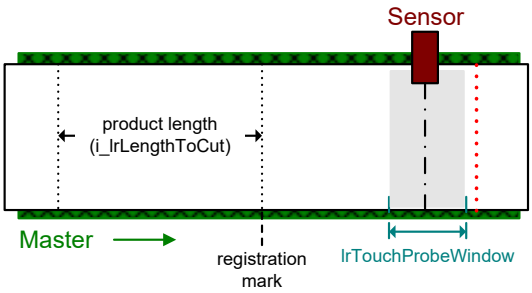


Registration marks detected outside the *lrTouchProbeWindow* are ignored.

Scenario 2: NO touch probe signal is detected inside the touch probe acceptance window:

If no registration mark is detected within the window:

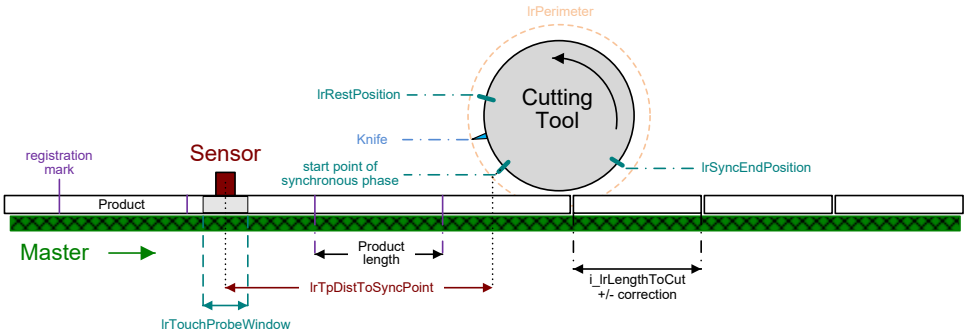
- The length of the product is set to the value of *i_lrLengthToCut*.
- The output *q_xTpOutsideWindow* is set to TRUE to indicate that no registration mark has been detected inside the touch probe acceptance window.
- The output is reset to FALSE after the next call of the function block.



Upon a rising edge of the input *i_xUseNextTp*, page 33, the next touch probe signal becomes valid without considering the *lrTouchProbeWindow* range.

lrTpDistToSyncPoint for Calculation and Validation

For the calculation of the curves and the validation of the touch probe signals, the parameter *lrTpDistToSyncPoint*, page 28 is required. This parameter defines the distance between the center point of the sensor and the start point of the synchronous phase.



Operating Mode *CutOnTouchProbe*

Overview

The operating mode *CutOnTouchProbe* is used for processing products with variable length and format calculated from a position capture. The length of the process cycle is defined via registration marks corresponding to the distances between the products. These distances are calculated by the touch probe positions in connection with a touch probe ignorance window.

Touch Probe Ignorance Window: *IrTouchProbeWindow*

The touch probe ignorance window corresponds to the parameter *IrTouchProbeWindow*, page 28. It defines a range after the sensor in which detected touch probe signals are ignored as long as the registration mark of the previous touch probe signal is inside this window.

A new touch probe signal is only valid if the touch probe position is greater than the last touch probe position plus the value of the parameter *IrTouchProbeWindow*:

touch probe position > last touch probe position + touch probe window

The minimum value of the parameter *IrTouchProbeWindow* is calculated depending on the parameters of the function block configuration:

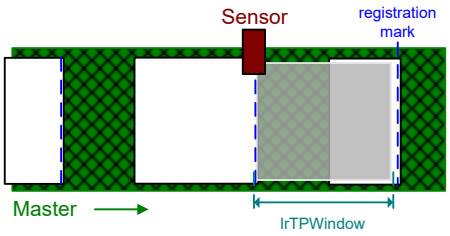
master start phase + synchronous end position

For further information on calculating the master start phase, refer to Master Start Phase, page 85.

> **NOTE:** Do not modify the value of the parameter *IrTouchProbeWindow* while the function block is active. Modifications of *ST_Parameters* values during operation are ignored and the function block continues to use the value from the last activation.

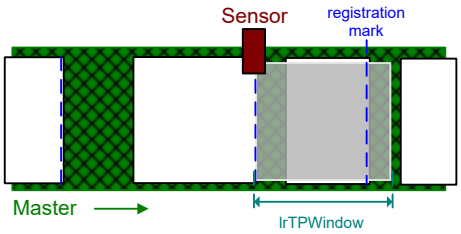
Scenario 1: The touch probe signal is valid:

new touch probe position - last touch probe position = outside the touch probe window



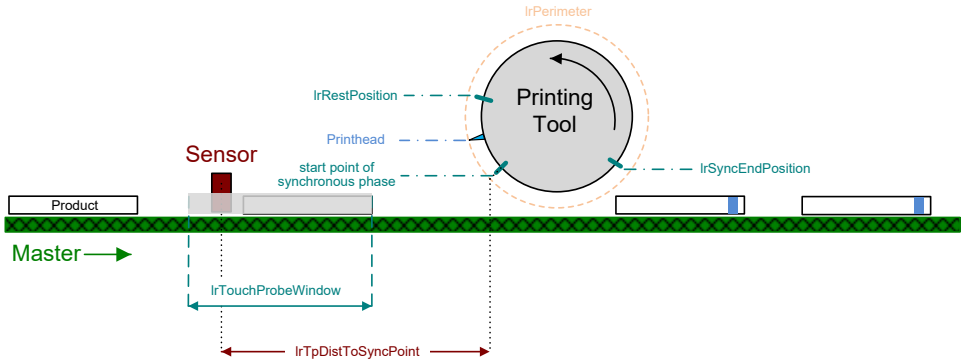
Scenario 2: The touch probe signal is invalid:

new touch probe position - last touch probe position = inside the touch probe window



lrTpDistToSyncPoint for Calculation and Validation

For the calculation of the curves and the validation of the touch probe signals, the parameter *lrTpDistToSyncPoint*, page 28 is required. This parameter defines the distance between the center point of the sensor and the start point of the synchronous phase.



Functions for Operating Modes *ContinuousWithCorrection* and *CutOnTouchProbe*

Overview

For the operating modes *ContinuousWithCorrection* and *CutOnTouchProbe*, specific functions are available.

Touch Probe Function to Capture the Position of the Master Axis

It is possible to capture the position of the master axis for calculating the distance between two products in units defined for the master axis as follows:

- Via a digital input by setting the parameter *ifTriggerRef* to 0:

  Upon a rising edge at the input *i_xTpDigitalInput*, the present position of the master axis is captured and used as touch probe position.

- Via a capture unit by setting the parameter *ifTriggerRef* = *DRV_Lexium32S. triggerCap1* / *Cap2* / *Cap3*:

  The position of the master axis is captured via the configured capture input (number of the capture input and rising / falling edge) of the master axis. For further information, refer to Activating Capture Inputs, page 87.

FIFO Buffer Saving Touch Probe Positions

The function block provides a FIFO buffer to save the touch probe positions of products if more than one product is detected by the sensor before the product can be processed. The maximum number of products that can be stored in the FIFO buffer is defined with the parameter *GPL.Gc_ usiFiFoMaxNumberOfProducts*, page 31.

The FIFO buffer is activated when the slave axis reaches the rest position after a cold start.

The FIFO buffer is deactivated and the entries are deleted with a falling edge at the input *i_xEnable*, page 33.

With a falling edge at the input *i_xStartOperation*, page 33, the slave moves to the rest position. The FIFO buffer as well as the touch probe function are kept active

allowing the process to continue without operator intervention upon a rising edge at *i_xStartOperation* and allowing restart of the motion. New captured positions are written to the FIFO buffer and at every start of a new process cycle, the corresponding position is read from the FIFO. The FIFO and the touch probe function continue operation as if the slave axis was moving.

If the maximum number of products that can be stored in the FIFO buffer (*GPL. Gc_usiFiFoMaxNumberOfProducts*) is exceeded, the function block detects an error and stops the slave axis. The last captured position is not stored.

If the function block becomes inactive, the entries of the FIFO buffer are deleted.

Special Functions and Modes

What's in This Chapter

General Information

As a prerequisite for executing special functions, the function block must be active.

If the function block is disabled while a function is active, the function is also disabled and needs to be restarted.

Immediate Synchronization

Overview

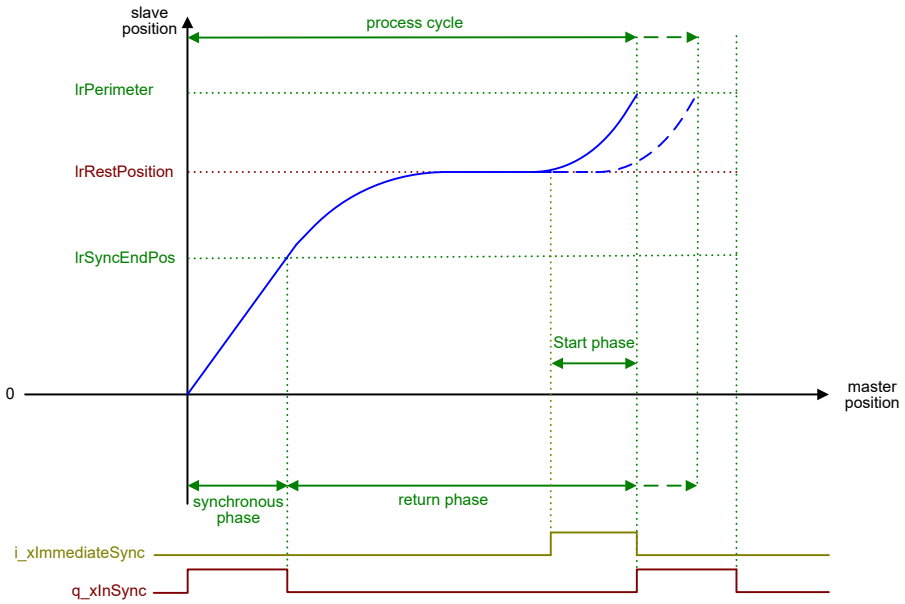
Prerequisites for this function:

- The slave is in rest position.
- The operating mode is *Continuous*.

A rising edge of the input *i_xImmediateSync*, page 33 starts a new process cycle by initiating the synchronization of the slave to the master.

After the *lrMasterStartPhase*, the synchronous phase starts and the input *i_lrLengthToCut* is used for the length of the process cycle.

A rising edge of the input *i_xImmediateSync* is ignored when the slave is not in *lrRestPosition*.



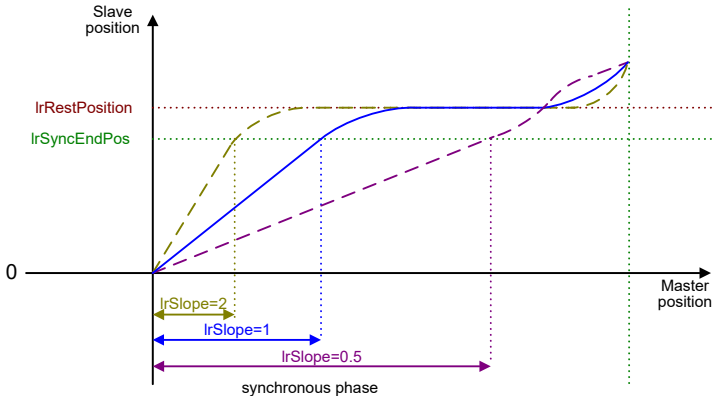
Slope of the Synchronous Phase

Overview

As a prerequisite for modifying the slope of the synchronous phase, the function block must be inactive.

Use the parameter *lrSlope*, page 28 to introduce a slope coefficient for the synchronous phase:

- *lrSlope* = 1: The master and the slave axis are moving with the same velocity.
- *lrSlope* > 1: The slave is moving slower than the master.
- *lrSlope* < 1: The slave is moving faster than the master.



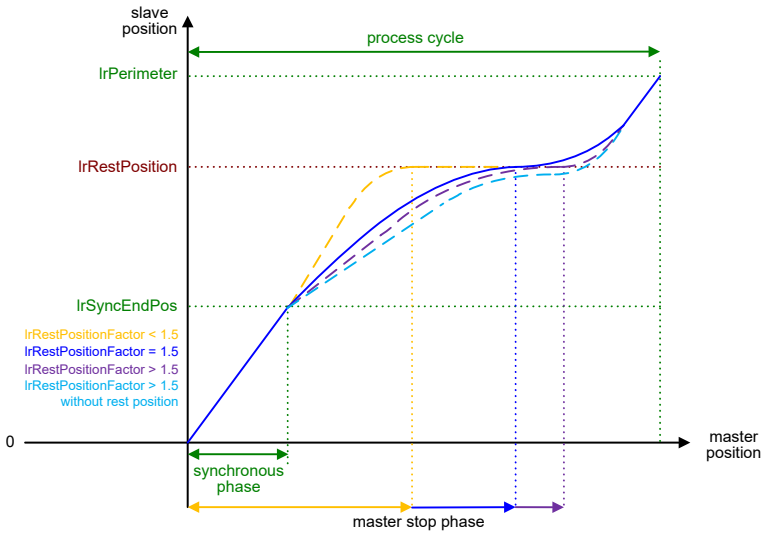
Rest Position Factor

Overview

The parameter *lrRestPositionFactor* allows you to modify the distance between the synchronous start position and the point when the slave axis reaches the rest position (master stop phase).

- The smaller the value, the sooner the slave axis reaches the rest position. This short master stop phase can lead to errors detected in the slave axis (following errors that are originally generated by the drive).
- The higher the value, the later the slave axis reaches the rest position. This long master stop phase leads to a smoother movement and can result in a profile without rest position, page 63.

For calculating the master stop phase, refer to the Calculations, page 85.


slave position
process cycle
lrPerimeter
lrRestPosition
lrSyncEndPos
lrRestPositionFactor < 1.5
lrRestPositionFactor = 1.5
lrRestPositionFactor > 1.5
lrRestPositionFactor > 1.5 without rest position
0
master position
synchronous phase
master stop phase

Quick Reference Guide

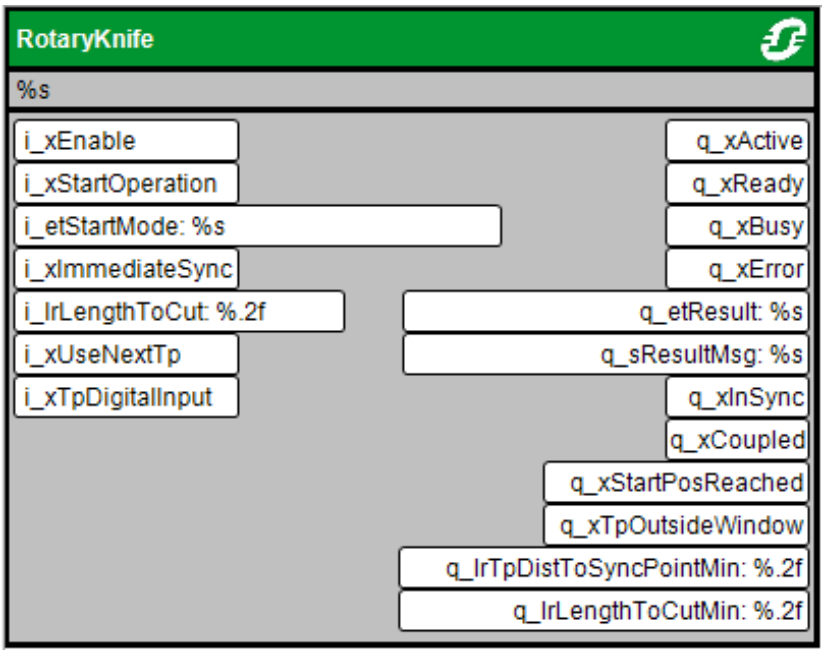
What's in This Chapter

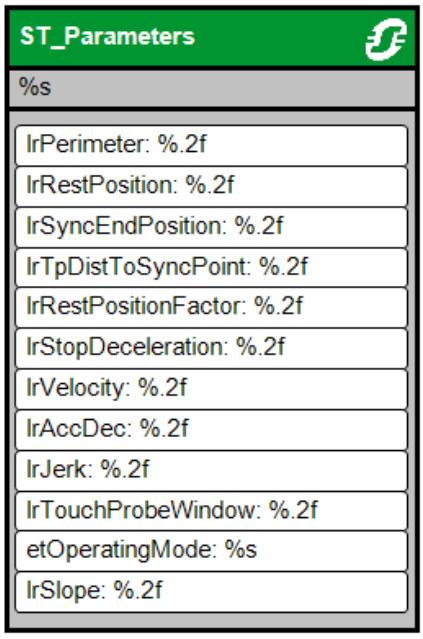
Visualization

Overview

Visualization screens are embedded in the MotionApplicationFunctionBlocks library to configure and start the function block.

Inputs and outputs as well as *ST_Parameters* are accessible.





Troubleshooting

For types of detected errors and error notifications, refer to the *Troubleshooting* chapter of the *FlyingShear* function block, page 58.

Appendices

What’s in This Part

Calculations

What's in This Chapter

Calculations

Master Start Phase

The movement from the rest position to the start point of the synchronous phase is calculated as follows (in units defined for the master axis).

For *FlyingShear*:

$$\frac{ABS(ST\_Parameters.IrRestPosition)}{ST\_Parameters.IrSlope} * 1.875$$

For *RotaryKnife*:

$$\frac{(ST\_Parameters.IrPerimeter - ST\_Parameters.IrRestPosition)}{ST\_Parameters.IrSlope} * 1.875$$

Master Stop Phase

The movement from the start point of the synchronous phase to the rest position is calculated as follows (in units defined for the master axis):

$$\frac{ST\_Parameters.IrSyncEndPosition}{ST\_Parameters.IrSlope} * ST\_Parameters.IrRestPositionFactor$$

Minimum Distance Between Touch Probe and Synchronous Point (*q_IrTpDistToSyncPointMin*)

The output *q_IrTpDistToSyncPointMin* that defines the minimum distance between the touch probe sensor and the start point of the synchronous phase depends on the operating mode.

Operating mode *ContinuousWithCorrection*:

$$IrMasterStartPhase + \frac{ST\_Parameters.IrTouchProbeWindow}{2}$$

Operating mode *CutOnTouchProbe*:

$$IrMasterStartPhase$$

Minimum Value for Length to Cut (*q_IrLengthToCutMin*)

The minimum value for the length to cut (*q_IrLengthToCutMin*) is calculated as follows, depending on the operating mode:

Operating mode *Continuous*:

$$IrSyncEndPosX * GPL.Gc\_IrLengthToCutMinFactor$$

Operating mode *ContinuousWithCorrection*:

$$\left(IrSyncEndPosX + \frac{stParameters\_loc.IrTouchProbeWindows}{2}\right) * GPL.Gc\_IrLengthToCutMinFactor$$

Operating mode *CutOnTouchProbe*:

$$IrSyncEndPosX * GPL.Gc\_IrLengthToCutMinFactor$$

Where:

$$IrSyncEndPosX = \frac{ST\_Parameters.IrSyncEndPosition}{ST\_Parameters.IrSlope}$$

Minimum Value for Touch Probe Window

The minimum value for the touch probe window depends on the operating mode.

In operating mode *ContinuousWithCorrection*, the value must be greater than 0.

In operating mode *CutOnTouchProbe*, the value is equal to the maximum value of *IrMasterStartPhase* and *IrLengthToCutMin*.

Activating Capture Inputs

What's in This Chapter

Activating Capture Inputs in the Logic Builder

Activating a Capture Input of the Sercos Device

To activate a capture input of the drive in the Logic Builder, proceed as follows:

Step	Action
1	In the **Devices** tree, double-click the drive subnode of the **SercosMaster** node. **Result**: The drive editor opens on the right-hand side.
2	From the drive editor, select the **Feature Configuration** tab.
3	Select the option **TouchProbe**.
4	Select one of the capture inputs **Cap1**, **Cap2** or **Cap3**.

Example

Example for using capture input 1 of the drive as *Master* = *DRV_Lexium32S*:

```
ifTriggerRef = DRV_Lexium32S.triggerCap1
```

Activating a Capture Input of the Encoder

To activate a capture input of the encoder in the Logic Builder, proceed as follows:

Step	Action
1	In the **Devices** tree, double-click the **DI (Digital Inputs)** node.
2	From the **DI** editor on the right-hand side, select the **Capture Inputs** tab.
3	Select one of the capture inputs **Cap1**, **Cap2** or **Cap3**.

Example

Example for using capture input 1 of the encoder as *Master* = *Incremental_Encoder*:

```
ifTriggerRef = DI.triggerCap1
```

Index

S

T

W

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

EIO0000004585.03