

SoMachine

Hoisting Application Functions Hoisting Library Guide

09/2016

E100000000620.09

www.schneider-electric.com

Schneider
 **Electric**

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

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

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

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

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

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

© 2016 Schneider Electric. All rights reserved.

Table of Contents



	Safety Information	17
	About the Book	21
Part I	Hoisting Library System Requirements	23
Chapter 1	System Requirements	25
	Hoisting System Requirements	26
	Function Block Location in the Hoisting Library	29
	Hoisting Segment	31
Chapter 2	Crane Environment and Architecture	37
	Construction Cranes	38
	Industrial Cranes	40
	Special Cranes	43
	Hardware Architecture	45
Part II	Advanced Positioning	47
Chapter 3	AdvancedPositioning: Position Control of Single Axis Driven by a Variable Speed Drive	49
3.1	Functional Overview	50
	Functional Overview	50
3.2	Architecture	52
	Architecture	52
3.3	Function Block Description	54
	AdvancedPositioning Function Block	54
3.4	Pin Description	58
	Input Pin Description	59
	Output Pin Description	66
3.5	Structured Parameters	71
	Structured Parameter - stMotPara	72
	Structured Parameter - stPosPara	73
	Structured Parameter - stJogPara	77
	Structured Parameter - stHmngPara	78
	Structured Parameter - stEncPara	84

3.6	Quick Reference Guide	93
	Function Block Visualization	94
	Instantiation and Usage Example	95
	Commissioning Procedure	97
	Troubleshooting	100
Part III	Anti-Crab	101
Chapter 4	AntiCrab: Correction of Bridge Movement to Solve Wear of Rail and Wheels	103
4.1	Functional and Machine Overview	104
	Functional Overview	105
	Machine Overview	108
4.2	Architecture	109
	Hardware Architecture	110
	Software Architecture	111
4.3	Function Block Description	113
	AntiCrab Function Block	113
4.4	Pin Description	114
	Input Pin Description	115
	Output Pin Description	121
4.5	Troubleshooting	123
	Troubleshooting	123
Chapter 5	AntiCrab_2: Correction of Bridge Movement to Solve Wear of Rail and Wheels	125
5.1	Functional and Machine Overview	126
	Functional Overview	127
	Machine Overview	128
5.2	Architecture	129
	Hardware Architecture	130
	Software Architecture	131
5.3	Function Block Description	132
	AntiCrab_2 Function Block	132
5.4	Pin Description	137
	Input Pin Description	138
	Output Pin Description	145

5.5	Quick Reference Guide	148
	Visualization	149
	Instantiation and Usage Example	151
	Commissioning Procedure	153
	Troubleshooting	155
Part IV	Anti-Sway	157
Chapter 6	AntiSwayOpenLoop: Correction of Sway in Horizontal Movements without Sensor	159
6.1	Functional and Machine Overview	160
	Functional Overview	161
	Machine Overview	164
6.2	Architecture	165
	Hardware Architecture	166
	Software Architecture	167
6.3	Function Block Description - Speed Reference	169
	Speed Reference - Overview	170
	SpeedRef_2 Function Block	171
	Input Pin Description - SpeedRef_2	172
	SpeedRef_4 Function Block	175
	Input Pin Description - SpeedRef_4	176
	SpeedRef_AI Function Block	179
	Input Pin Description - SpeedRef_AI	180
	Output Pin Description - SpeedRef_2, SpeedRef_4 and SpeedRef_AI	183
6.4	Function Block Description - Cable Length	186
	Cable Length - Overview	187
	CableLength_2Pos Function Block	189
	Input Pin Description - CableLength_2Pos	190
	Output Pin Description - CableLength_2Pos	192
	CableLength_3Pos Function Block	194
	Input Pin Description - CableLength_3Pos	195
	Output Pin Description - CableLength_3Pos	197
	CableLength_Enc Function Block	199
	Input Pin Description - CableLength_Enc	200
	Output Pin Description - CableLength_Enc	202
	Calibration Procedure for the CableLength_Enc Function Block	205

	CableLength_Enc_2 Function Block	207
	Input Pin Description - CableLength_Enc_2	209
	Output Pin Description - CableLength_Enc_2	212
	Instantiation and Usage Example	213
6.5	Function Block Description - AntiSwayOpenLoop	214
	Anti-Sway - Overview	215
	AntiSwayOpenLoop Function Block	216
	Input Pin Description - AntiSwayOpenLoop	217
	Structure Parameter - AntiSwayOpenLoop	219
	Output Pin Description - AntiSwayOpenLoop	222
	AntiSwayOpenLoop Main Parameters	224
6.6	Function Block Description - AntiSwayOpenLoop_2	231
	AntiSwayOpenLoop_2 - Overview	232
	AntiSwayOpenLoop_2 Function Block	234
	Input Pin Description - AntiSwayOpenLoop_2	235
	Structure Parameter - AntiSwayOpenLoop_2	240
	Output Pin Description - AntiSwayOpenLoop_2	245
	Instantiation and Usage Example - AntiSwayOpenLoop_2	248
	Commissioning Procedure	249
6.7	Troubleshooting	254
	Troubleshooting	254
Part V	Diagnostic Coverage	255
Chapter 7	DiagnosticCoverage: Increase Diagnostic Coverage of Used Control System	257
7.1	Functional Overview	258
	Functional Overview	258
7.2	Architecture	260
	Architecture	260
7.3	Function Block Description	262
	DiagnosticCoverage Function Block	262
7.4	Pin Description	263
	Input/Output Pin Description	264
	Input Pin Description	265
	Output Pin Description	269
7.5	Quick Reference Guide	271
	Instantiation and Usage Example	272
	Commissioning Procedure	273
	Troubleshooting	274

Part VI	Electronic Potentiometer	275
Chapter 8	ElectronicPotentiometer: Set Speed and Direction of a Movement	277
8.1	Functional Overview	278
	Functional Overview	278
8.2	Architecture	279
	Hardware Architecture	280
	Software Architecture	281
8.3	Function Block Description	282
	ElectronicPotentiometer Function Block	282
8.4	Pin Description	286
	Input Pin Description	287
	Output Pin Description	290
8.5	Quick Reference Guide	292
	Instantiation and Usage Example	293
	Commissioning Procedure	294
	Troubleshooting	295
Part VII	Grab Control	297
Chapter 9	GrabControl: Control of a Four Cable Grab	299
9.1	Functional Overview	300
	Functional Overview	300
9.2	Architecture	302
	Architecture	302
9.3	Function Block Description	304
	GrabControl Function Block	304
9.4	Pin Description	309
	Input Pin Description	310
	Structured Variable Description	316
	Output Pin Description	321
9.5	Quick Reference Guide	325
	Instantiation and Usage Example	326
	Commissioning Procedure	328
	Troubleshooting	334

Part VIII	Hoisting Position Synchronization	335
Chapter 10	HoistPositionSync: Synchronization of Two Identical Axes for Hoist or Trolley Movement with Encoder.	337
10.1	Functional and Machine Overview	338
	Functional Overview	339
	Machine Overview	341
10.2	Architecture	342
	Hardware Architecture	343
	Software Architecture	344
10.3	Function Block Description	345
	HoistPositionSync Function Block	345
10.4	Pin Description	346
	Input Pin Description	347
	Acceleration and Deceleration Parameter	352
	Output Pin Description	353
10.5	Troubleshooting	356
	Troubleshooting	356
Chapter 11	HoistPositionSync_2: Position Synchronization of Two Crane Axes	359
11.1	Functional Overview	360
	Functional Overview	360
11.2	Architecture	362
	Software Architecture	362
11.3	Function Block Description	363
	HoistPositionSync_2 Function Block	363
11.4	Pin Description	364
	Input Pin Description	365
	Structured Variable Description	370
	Output Pin Description	373
11.5	Commissioning Guide	375
	Commissioning Guide	375
11.6	Troubleshooting	376
	Troubleshooting	376

Part IX	Limit Switch Management.	377
Chapter 12	LimitSwitch: Surveillance of the Used Limit Switches of a Crane with Fast/Slow Zone Identification	379
12.1	Functional and Machine Overview	380
	Functional Overview	381
	Machine Overview	383
12.2	Architecture	384
	Hardware Architecture	385
	Software Architecture	387
12.3	Function Block Description	388
	LimitSwitch Function Block	389
	Configuration of the Movement Positions	390
	Timing Chart	394
12.4	Pin Description	396
	Input Pin Description	397
	Output Pin Description	399
12.5	Troubleshooting	401
	Troubleshooting	401
Chapter 13	LimitSwitch_AR: Monitoring of the Limit Switches of a Crane with Adaptive Speed Reference in Slow Zone	403
13.1	Functional Overview	404
	Functional Overview	404
13.2	Architecture	406
	Hardware Architecture	407
	Software Architecture	408
13.3	Function Block Description	409
	LimitSwitch_AR Function Block	409
13.4	Pin Description	413
	Input Pin Description	414
	Output Pin Description	418
13.5	Quick Reference Guide	420
	Instantiation and Usage Example	421
	Commissioning Procedure	422
	Troubleshooting	423
Chapter 14	DoubleLimitSwitch_AR: Administers Two Devices in Synchronous Mode.	425
14.1	Functional Overview	426
	Functional Overview	426

14.2	Architecture	428
	Hardware Architecture	429
	Software Architecture	430
14.3	Function Block Description	431
	DoubleLimitSwitch_AR Function Block	431
14.4	Pin Description	436
	Input Pin Description	437
	Structured Variable Description	440
	Output Pin Description	442
14.5	Quick Reference Guide	445
	Instantiation and Usage Example	446
	Commissioning Procedure	447
	Troubleshooting	448
Chapter 15	DoubleLimitSwitch_AR_2: Administers Two Devices in Synchronous Mode	449
15.1	Functional Overview	450
	Functional Overview	450
15.2	Architecture	453
	Hardware Architecture	454
	Software Architecture	455
15.3	Function Block Description	456
	DoubleLimitSwitch_AR_2 Function Block	456
15.4	Pin Description	462
	Input Pin Description	463
	Structured Variable Description	466
	Output Pin Description	468
15.5	Quick Reference Guide	471
	Instantiation and Usage Example	472
	Commissioning Procedure	473
	Troubleshooting	474
Part X	Load Overspeed Control	475
Chapter 16	LoadOverspeedCtrl: Detects Load Overspeed, Brake Wear and Sensor Feedback on Hoist.	477
16.1	Functional and Machine Overview	478
	Functional Overview	479
	Machine Overview	481

16.2	Architecture	482
	Hardware Architecture	483
	Software Architecture	484
16.3	Function Block Description	485
	LoadOverspeedCtrl Function Block	486
	Load Overspeed Detection and Sensor Feedback Alarm Detection	488
	Brake Wear Detection	491
16.4	Pin Description	493
	Input Pin Description	494
	Output Pin Description	498
16.5	Troubleshooting	500
	Troubleshooting	500
Chapter 17	LoadOverspeedCtrl_2: Detects Load Overspeed, Brake Wear and Sensor Feedback on Hoist	501
17.1	Functional and Machine Overview	502
	Functional Overview	503
	Machine Overview	505
17.2	Architecture	507
	Hardware Architecture	508
	Software Architecture	509
17.3	Function Block Description	510
	LoadOverspeedCtrl_2 Function Block	511
	Load Overspeed Detection	512
	Brake Wear Detection	514
17.4	Pin Description	515
	Input Pin Description	516
	Output Pin Description	519
17.5	Quick Reference Guide	521
	Instantiation and Usage Example	522
	Troubleshooting	523
Part XI	Monitoring Data Storage	525
Chapter 18	Monitoring Data Storage Function: Collect and Monitor the Operational Data of Cranes	527
18.1	Functional and Machine Overview	528
	Functional Overview	529
	Machine Overview	533
18.2	Architecture	534
	Hardware Architecture	534

18.3	Function Block Description - StatisticDataStorage	535
	StatisticDataStorage Function Block	536
	Pin Description - StatisticDataStorage.....	539
18.4	Function Block Description - StatisticDataStorage_2	544
	StatisticDataStorage_2 Function Block.....	545
	Pin Description - StatisticDataStorage_2	546
18.5	Function Block Description - AlarmDataStorage.....	549
	AlarmDataStorage Function Block	550
	Pin Description - AlarmDataStorage.....	551
18.6	Function Block Description - AlarmDataStorage_2.....	553
	AlarmDataStorage_2 Function Block	554
	Pin Description - AlarmDataStorage_2.....	555
18.7	Function Block Description - OvldAlarmDataStorage	557
	OvldAlarmDataStorage Function Block	558
	Pin Description - OvldAlarmDataStorage.....	559
18.8	Function Block Description - OvtqAlarmDataStorage	561
	OvtqAlarmDataStorage Function Block	562
	Pin Description - OvtqAlarmDataStorage.....	563
18.9	Function Block Description - OvspAlarmDataStorage	565
	OvspAlarmDataStorage Function Block	566
	Pin Description - OvspAlarmDataStorage.....	567
18.10	Function Block Description - LdslAlarmDataStorage	569
	LdslAlarmDataStorage Function Block	570
	Pin Description - LdslAlarmDataStorage.....	571
18.11	Function Block Description - EncAlarmDataStorage	573
	EncAlarmDataStorage Function Block	574
	Pin Description - EncAlarmDataStorage.....	575
18.12	Function Block Description - MaintenanceDataStorage	577
	MaintenanceDataStorage Function Block.....	578
	Pin Description - MaintenanceDataStorage	582
	Commissioning Procedure.....	585
18.13	Function Block Description - MaintenanceDataStorage_2	587
	MaintenanceDataStorage_2 Function Block.....	588
	Pin Description - MaintenanceDataStorage_2	590
	Commissioning Procedure.....	595
18.14	Function Block Description - PasswordDataStorage	597
	PasswordDataStorage Function Block	598
	Pin Description - PasswordDataStorage.....	600

18.15	Quick Reference Guide	601
	Instantiation and Usage Example	602
	Troubleshooting	603
Part XII	Overload Control	605
Chapter 19	Overload Control Function: Detect Mechanical Overload on Hoisting Devices	607
19.1	Functional and Machine Overview	608
	Functional Overview	609
	Machine Overview	615
19.2	Architecture	616
	Hardware Architecture	617
	Software Architecture	618
19.3	Function Block Description - OverloadCtrlTrq	620
	OverloadCtrlTrq Function Block	621
	Pin Description - OverloadCtrlTrq	623
19.4	Function Block Description - OverloadCtrlDist	625
	OverloadCtrlDist Function Block	626
	Pin Description - OverloadCtrlDist	628
19.5	Function Block Description - OverloadCtrlEnc	630
	OverloadCtrlEnc Function Block	631
	Pin Description - OverloadCtrlEnc	633
19.6	Troubleshooting	635
	Troubleshooting	635
Chapter 20	Overload_EN15011: Overload Calibration With the Requirements of EN15011	637
20.1	Functional Overview	638
	Functional Overview	638
20.2	Architecture	640
	Hardware Architecture	641
	Software Architecture	642
20.3	Function Block Description	643
	Overload_EN15011 Function Block	643
20.4	Pin Description	644
	Input Pin Description	645
	Structured Variable Description	646
	Output Pin Description	651

20.5	Quick Reference Guide	654
	Instantiation and Usage Example	655
	Commissioning Procedure	656
	Calibration Procedure	657
	Troubleshooting	658
Part XIII	Scale Input	659
Chapter 21	ScaleInput: Change Incoming Input to a Scaled Value ..	661
21.1	Functional Overview	662
	Functional Overview	662
21.2	Architecture	663
	Software Architecture	663
21.3	Function Block Description	665
	ScaleInput Function Block	665
21.4	Pin Description	666
	Input Pin Description	667
	Output Pin Description	669
21.5	Troubleshooting	670
	Troubleshooting	670
Part XIV	Smooth Slewing	671
Chapter 22	Smooth Slewing Function: Stepwise Smooth Crane Movement Based on Speed or Torque Value of the Slewing Drive	673
22.1	Functional and Machine Overview	674
	Functional Overview	675
	Machine Overview	677
22.2	Architecture	678
	Hardware Architecture	679
	Software Architecture	680
22.3	Function Block Description - SmoothSlewingSpd	682
	SmoothSlewingSpd Function Block	683
	Input Pin Description	686
	Output Pin Description	687
22.4	Function Block Description - SmoothSlewingTrq	688
	SmoothSlewingTrq Function Block	689
	Input Pin Description	692
	Output Pin Description	694
22.5	Troubleshooting	695
	Troubleshooting	695

Part XV	Speed Optimization and Rope Slack	697
Chapter 23	SpeedOptRopeSlack: Reduces Work Cycle Time and Avoid Start Up with Slack on the Rope	699
23.1	Functional and Machine Overview	700
	Functional Overview	701
	Machine Overview	703
23.2	Architecture	704
	Hardware Architecture	705
	Software Architecture	706
23.3	Function Block Description	708
	SpeedOptRopeSlack Function Block	709
	Timing Chart	712
23.4	Pin Description	716
	Input Pin Description	717
	Output Pin Description	721
23.5	Troubleshooting	723
	Troubleshooting	723
Part XVI	Speed Select	725
Chapter 24	SpeedSelect: Provides the Option to Choose between 8 Predefined and one Analog Speed	727
24.1	Functional Overview	728
	Functional Overview	728
24.2	Architecture	729
	Software Architecture	729
24.3	Function Block Description	730
	SpeedSelect Function Block	730
24.4	Pin Description	731
	Input Pin Description	732
	Output Pin Description	734
24.5	Troubleshooting	735
	Troubleshooting	735
Part XVII	Wind Speed Control	737
Chapter 25	WindSpeedCtrl: Monitors and Provides Alarms for High Wind Conditions	739
25.1	Functional and Machine Overview	740
	Functional Overview	741
	Machine Overview	742

25.2	Architecture	743
	Hardware Architecture	744
	Software Architecture	745
25.3	Function Block Description	746
	windSpeedCtrl Function Block	746
25.4	Pin Description	748
	Input Pin Description	749
	Output Pin Description	751
Glossary	753
Index	759

Safety Information



Important Information

NOTICE

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



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



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

DANGER

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

WARNING

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

CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

BEFORE YOU BEGIN

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

WARNING

UNGUARDED EQUIPMENT

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

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

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

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

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

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

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

START-UP AND TEST

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

CAUTION

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 injury or equipment damage.

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

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

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

Before energizing equipment:

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

OPERATION AND ADJUSTMENTS

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

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

About the Book



At a Glance

Document Scope

This document describes the functions of the SoMachine Hoisting Library.

Validity Note

This document has been updated with the release of SoMachine 4.2.

Product Related Information

WARNING

LOSS OF CONTROL

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

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

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

WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Part I

Hoisting Library System Requirements

Overview

This part describes the system requirements for the Hoisting Library function blocks.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	System Requirements	25
2	Crane Environment and Architecture	37

Chapter 1

System Requirements

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Hoisting System Requirements	26
Function Block Location in the Hoisting Library	29
Hoisting Segment	31

Hoisting System Requirements

Overview

This section describes the 2 library versions, hardware and software requirements, and user interface compatibility for Hoisting Library functions.

Hoisting Library Versions

The Hoisting Library is being delivered in 2 versions:

- Hoisting_Basic Library
- advanced Hoisting Library

The Hoisting_Basic library contains the following function blocks:

- ElectronicPotentiometer
- DoubleLimitSwitch_AR_2
- LimitSwitch
- LimitSwitch_AR
- LoadOverspeedCtrl_2
- AlarmDataStorage_2
- MaintenanceDataStorage_2
- StatisticDataStorage_2
- PasswordDataStorage_2
- OverloadCtrlTrq
- OverloadCtrlDist
- OverloadCtrlEnc
- SmoothSlewingSpd
- SmoothSlewingTrq
- SpeedOptRopeSlack
- WindSpeedCtrl

The Hoisting_Basic Library is part of the SoMachine software delivery, but needs to be registered without any additional charge inside SoMachine.

The advanced Hoisting Library contains all available function blocks from the Hoisting_Basic Library plus the following ones:

- SpeedRef_2
- SpeedRef4
- SpeedRef_AI
- CableLength_2Pos
- CableLength_3Pos
- CableLength_Enc
- CableLength_Enc_2
- AntiSwayOpenLoop
- AntiSwayOpenLoop_2
- Overload_EN15011
- AntiCrab
- AntiCrab_2

- AdvancedPositioning
- DiagnosticCoverage
- HoistingPositionSync
- HoistingPositionSync_2
- GrabControl

The advanced Hoisting Library needs to be registered inside SoMachine.

NOTE: Both libraries cannot be installed at the same time. When upgrading from the basic to the advanced version, the Hoisting_Basic Library needs to be uninstalled.

System Requirements

The targeted controller for the segment Hoisting are:

- Modicon M241 Logic controller
- Modicon M258 Logic controller
- Modicon LMC058 Motion controller
- Altivar ATV IMC Drive controller

NOTE: All function blocks work with internally created information. Only the controller is needed. Altivar ATV IMC Drive controller is associated with ATV71.

NOTE: Modicon LMC058 Motion controller can be used with 2 CANopen fieldbuses. In this case the CANmotion bus is configured as a CANopen fieldbus.

Using the Library

WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

For more detailed information, see Schneider Electric Libraries.

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

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

 WARNING
--

UNINTENDED MACHINE OPERATION

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

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

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

Function Block Location in the Hoisting Library

Function Block Location in the Hoisting Library of SoMachine

Folder	Subfolder	Application Function Block
Advanced Positioning	–	AdvancedPositioning
Anti-crab	–	AntiCrab_2
Anti-sway	AntiSway Open Loop	AntiSwayOpenLoop_2
–	Cable Length Method	CableLength_Enc_2 CableLength_2Pos CableLength_3Pos
–	Speed Reference	SpeedRef_2 SpeedRef_4 SpeedRef_A1
Diagnostic Coverage	–	DiagnosticCoverage
Electronic Potentiometer	–	ElectronicPotentiometer
Grab Control	–	GrabControl
Hoisting position synchronization	–	HoistPositionSync HoistPositionSync_2
Limit switch management	–	DoubleLimitSwitch_AR_2 LimitSwitch LimitSwitch_AR
Load overspeed control	–	LoadOverspeedCtrl_2
Monitoring data storage	Alarm Data Storage	AlarmDataStorage_2
–	Maintenance Data Storage	MaintenanceDataStorage_2
–	Statistic Data Storage	StatisticDataStorage_2
–	Supporting Function Blocks	PasswordDataStorage
Over load control	–	Overload_EN15011
Smooth slewing	–	SmoothSlewingSpd SmoothSlewingTrq
Speed optimization and rope slack	–	SpeedOptRopeSlack
Supporting function blocks	–	ScaleInput SpeedSelect
Wind speed control	–	WindSpeedCtrl

Folder	Subfolder	Application Function Block
Legacy	-	AntiCrab AntiSwayOpenLoop CableLength_Enc DoubleLimitSwitch_AR LoadOverspeedCtrl AlarmDataStorage EncAlrmDataStorage LdslAlrmDataStorage OvldAlrmDataStorage OvspAlrmDataStorage OvtqAlrmDataStorage MaintenanceDataStorage StatisticDataStorage OverloadCtrlDist OverloadCtrlEnc OverloadCtrlTrq

Hoisting Segment

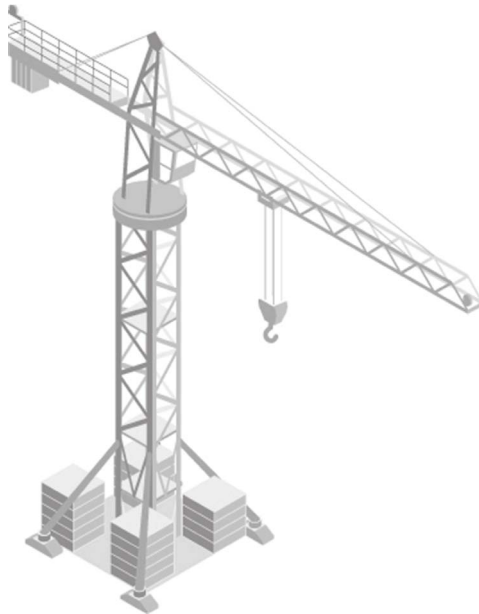
Overview

The Hoisting segment is separated into 3 targeted areas:

- Construction cranes
- Industrial cranes
- Special cranes

Construction Cranes

Construction cranes are cranes for construction sites, such as tower cranes, hammerhead cranes, self erecting cranes etc.

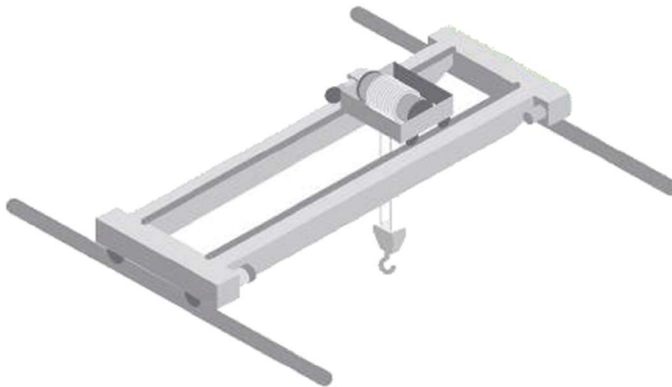


Construction cranes are used mainly external to buildings and used for constructing buildings.



Industrial Cranes

Industrial cranes are cranes used for production, such as bridge cranes, etc.



Industrial cranes are mainly used inside buildings or production halls and used to support the production work. However, they can also be used externally, for example in storage areas, loading or unloading processes, etc.



Special Cranes

Special cranes are defined as being neither industrial nor construction cranes. A special crane could be a Grab crane to load/unload bulk goods or an Rubber Tire Crane (RTC) used in a harbor to move containers, etc.





Chapter 2

Crane Environment and Architecture

What Is in This Chapter?

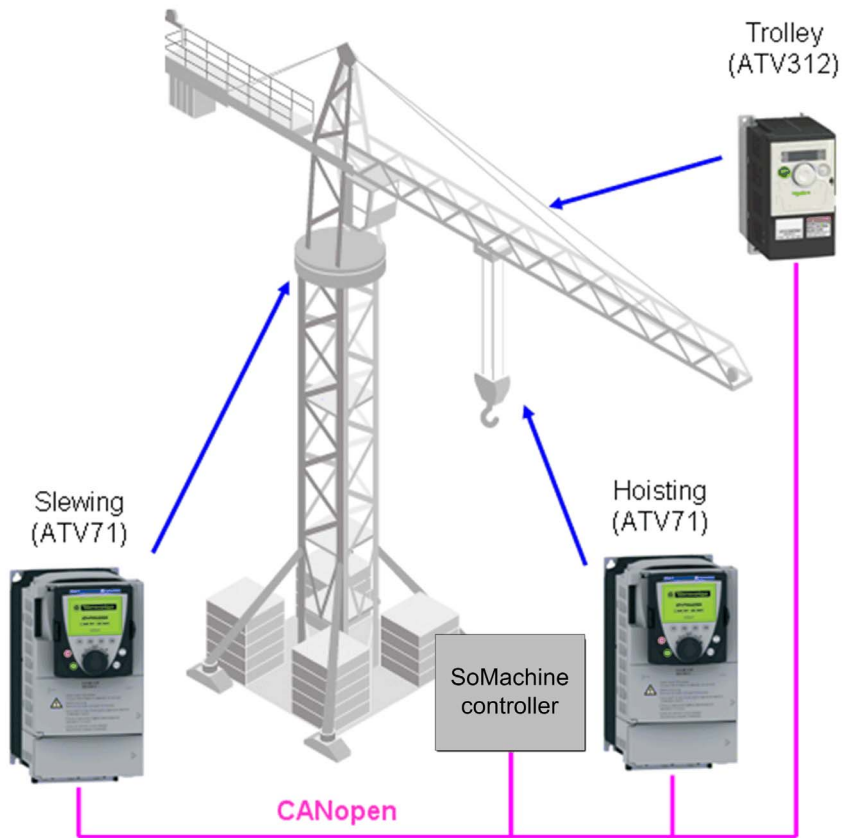
This chapter contains the following topics:

Topic	Page
Construction Cranes	38
Industrial Cranes	40
Special Cranes	43
Hardware Architecture	45

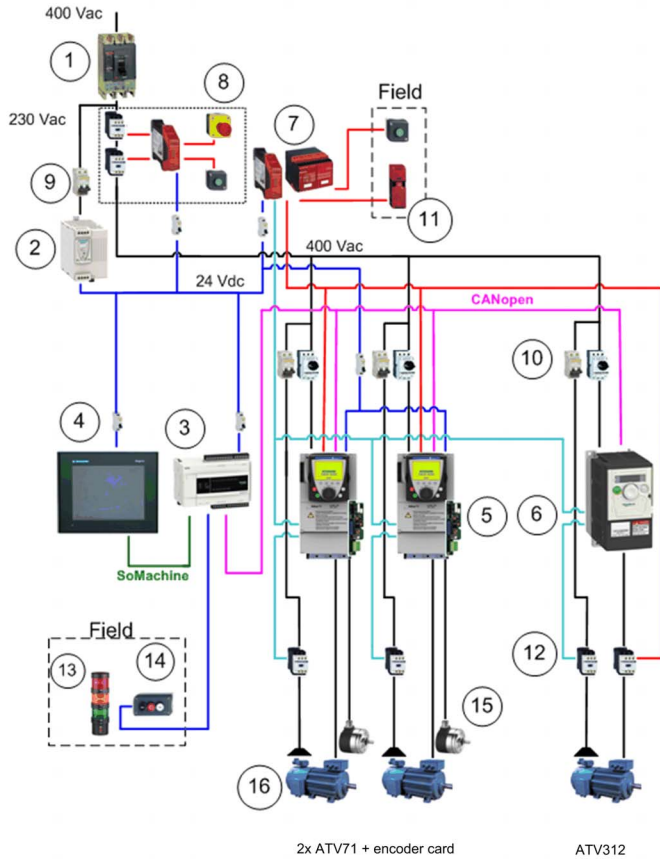
Construction Cranes

Overview

Simple construction crane with 1 drive per movement and a Schneider Electric controller:



Construction Crane Hoisting Architecture With M238

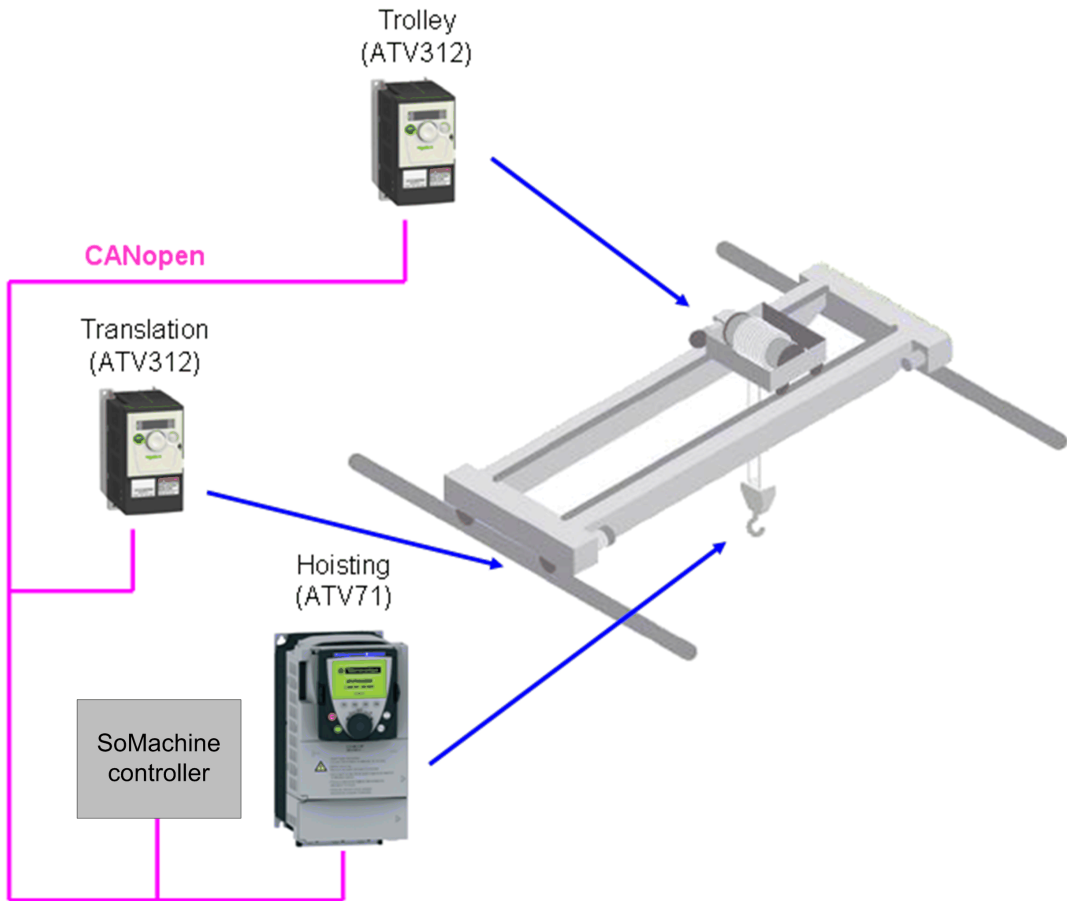


1	Compact NSX main switch	9	Multi9 circuit breaker
2	Phaseo power supply ABL8	10	TeSys motor protection GV2L
3	Modicon M238 Logic controller	11	Preventa door guard XCS
4	Magelis graphic HMI XBTGT	12	TeSysD contactor LC1D
5	Altivar drive ATV71 + encoder card	13	Harmony tower light XVBC
6	Altivar drive ATV312	14	Harmony pushbuttons enclosure XALD
7	Preventa safety module XPS	15	Absolute encoder XCC
8	Harmony e-stop enclosure XALK	16	AC-motor

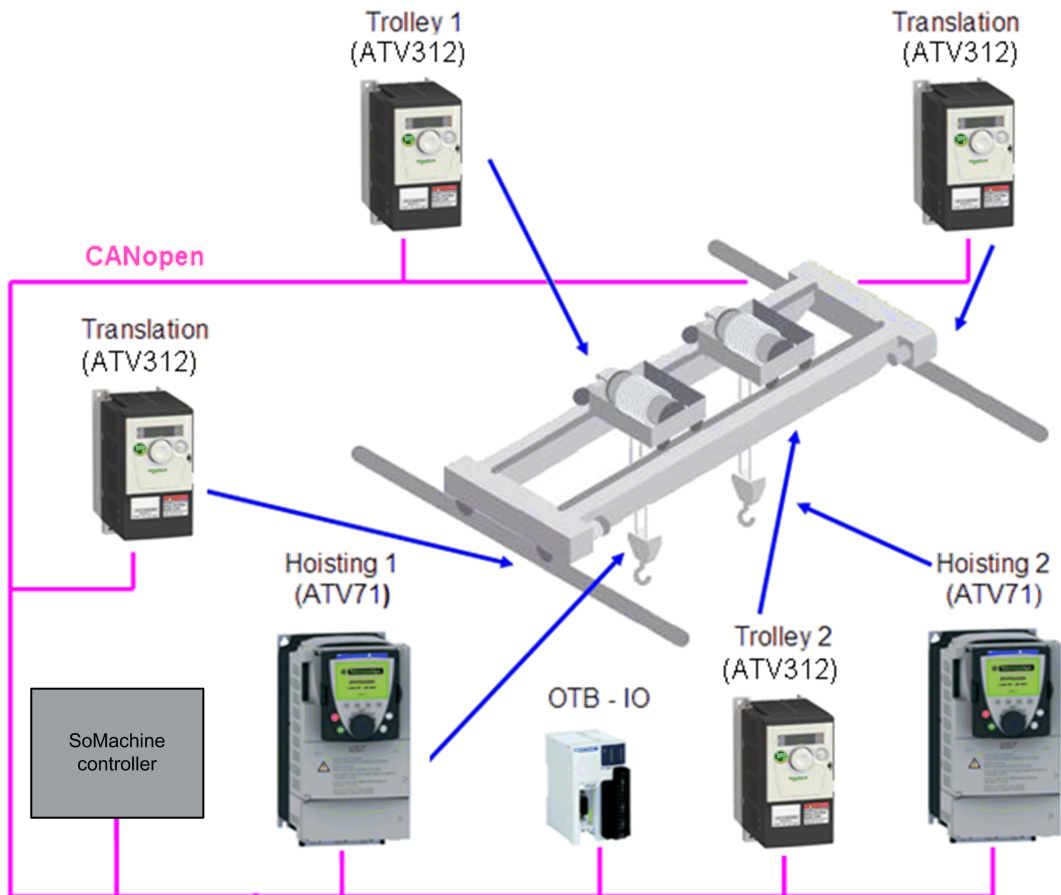
Industrial Cranes

Overview

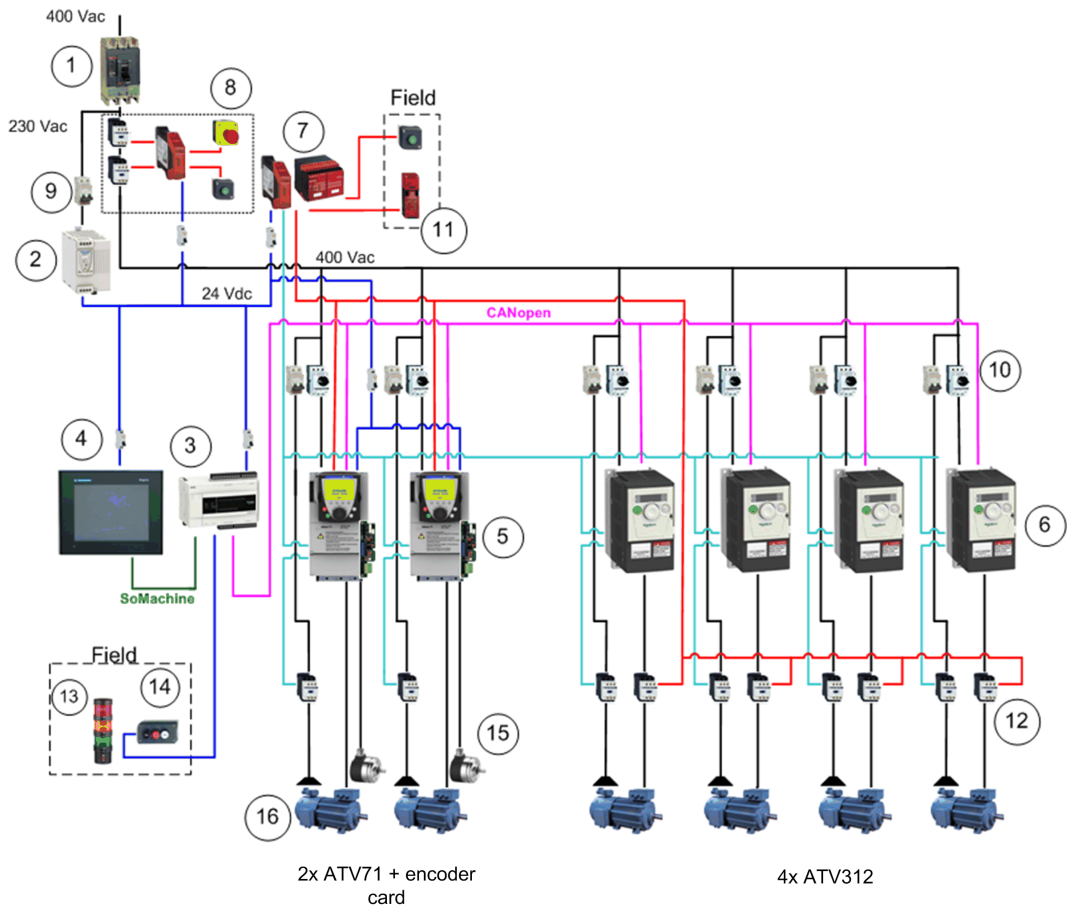
Simple industrial crane with 1 drive per movement and a Schneider Electric controller:



Industrial crane with 2 hoists and 2 trolleys, 1 drive on each side for the translation and a Schneider Electric controller with an I/O Expansion (OTB Island):



Hoisting Optimized CANopen Architecture With M238



1	Compact NSX main switch	9	Multi9 circuit breaker
2	Phaseo power supply ABL8	10	TeSys motor protection GV2L
3	Modicon M238 Logic controller	11	Preventa door guard XCS
4	Magelis graphic HMI XBTGT	12	TeSysD contactor LC1D
5	Altivar drive ATV71 + encoder card	13	Harmony tower light XVBC
6	Altivar drive ATV312	14	Harmony pushbuttons enclosure XALD
7	Preventa safety module XPS	15	Absolute encoder XCC
8	Harmony e-stop enclosure XALK	16	AC-motor

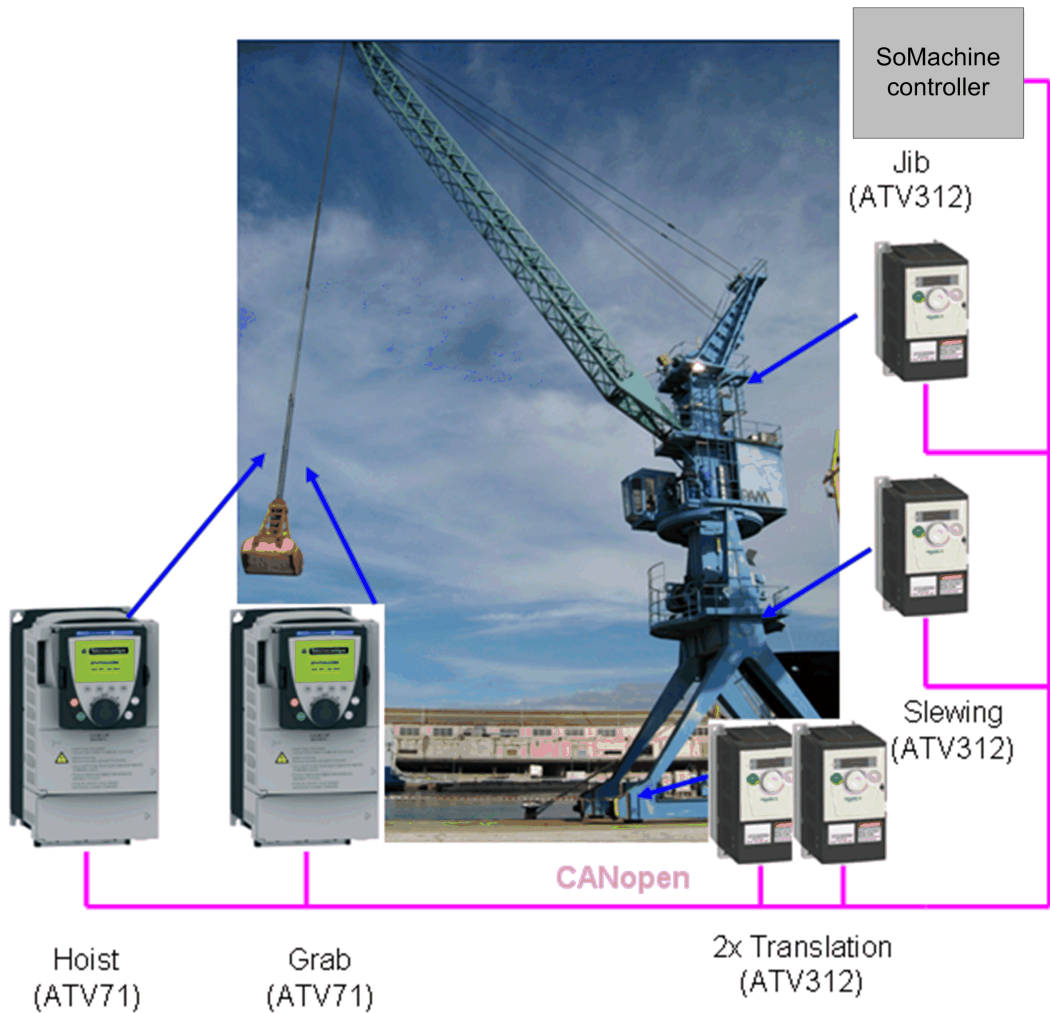
Special Cranes

Overview

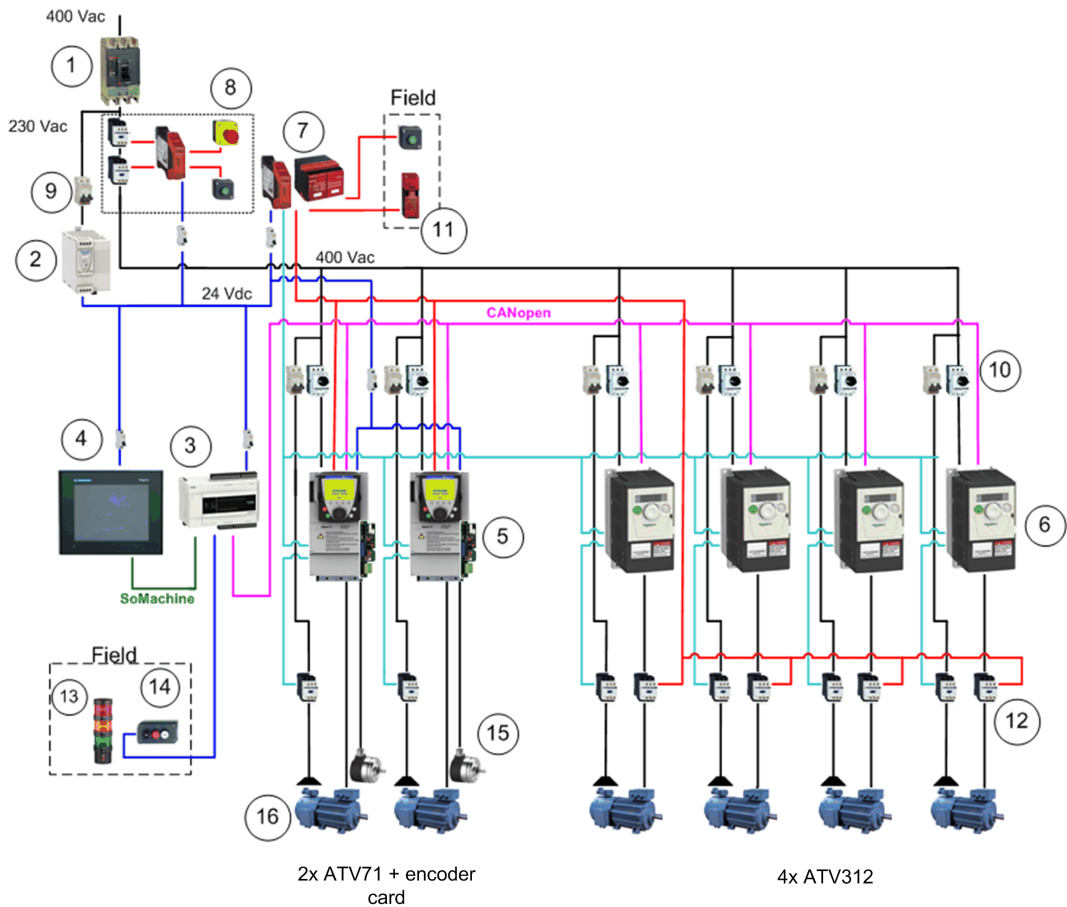
These are cranes with specialized features such as a Grab crane or a crane to move liquids.

The architecture of this type of crane can be similar to the Tower/Bridge cranes but may be completely different.

As an example here is the architecture for a Grab crane:

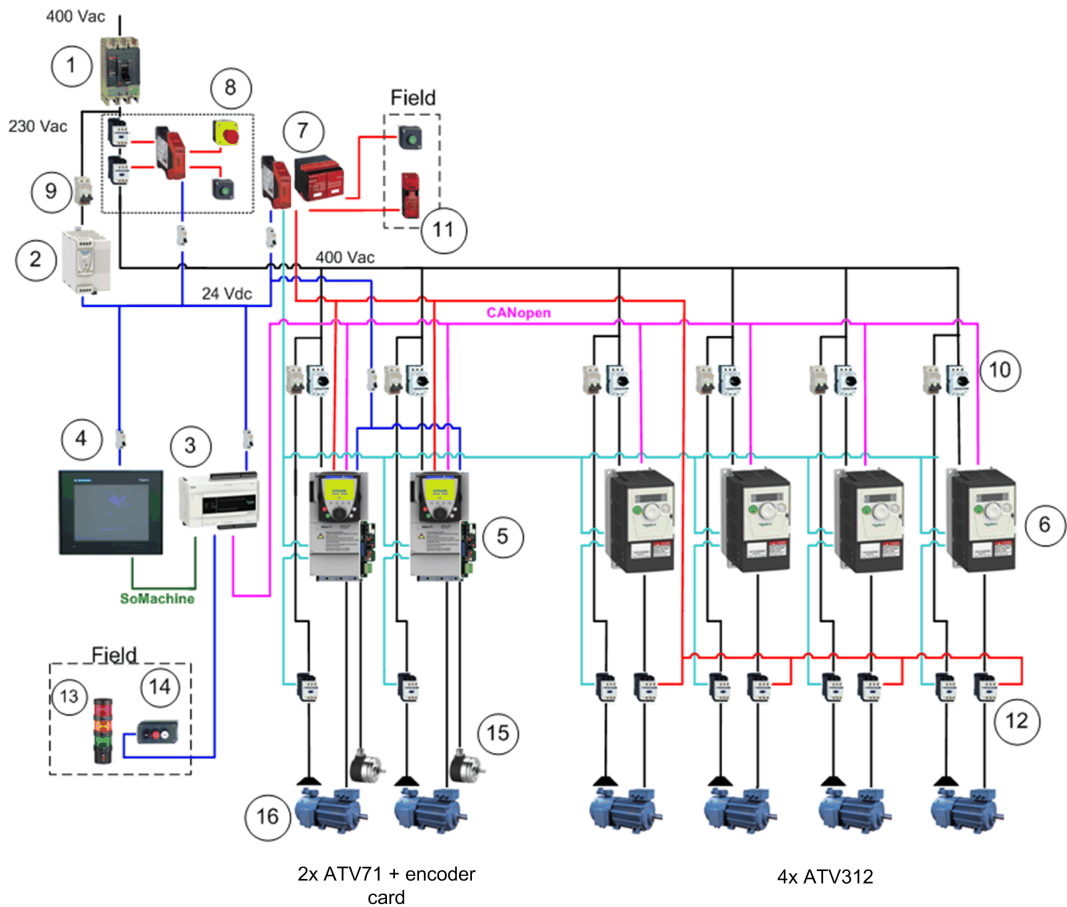


Hoisting Optimized CANopen Architecture With M238



1	Compact NSX main switch	9	Multi9 circuit breaker
2	Phaseo power supply ABL8	10	TeSys motor protection GV2L
3	Modicon M238 Logic controller	11	Preventa door guard XCS
4	Magelis graphic HMI XBTGT	12	TeSysD contactor LC1D
5	Altivar drive ATV71 + encoder card	13	Harmony tower light XVBC
6	Altivar drive ATV312	14	Harmony pushbuttons enclosure XALD
7	Preventa safety module XPS	15	Absolute encoder XCC
8	Harmony e-stop enclosure XALK	16	AC-motor

Optimized CANopen With M238 Architecture



1	Compact NSX main switch	9	Multi9 circuit breaker
2	Phaseo power supply ABL8	10	TeSys motor protection GV2L
3	Modicon M238 Logic controller	11	Preventa door guard XCS
4	Magelis graphic HMI XBTGT	12	TeSysD contactor LC1D
5	Altivar drive ATV71 + encoder card	13	Harmony tower light XVBC
6	Altivar drive ATV312	14	Harmony pushbuttons enclosure XALD
7	Preventa safety module XPS	15	Absolute encoder XCC
8	Harmony e-stop enclosure XALK	16	AC-motor

Part II

Advanced Positioning

Chapter 3

Advanced Positioning: Position Control of Single Axis Driven by a Variable Speed Drive

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	Functional Overview	50
3.2	Architecture	52
3.3	Function Block Description	54
3.4	Pin Description	58
3.5	Structured Parameters	71
3.6	Quick Reference Guide	93

Section 3.1

Functional Overview

Functional Overview

Why Use the AdvancedPositioning Function Block?

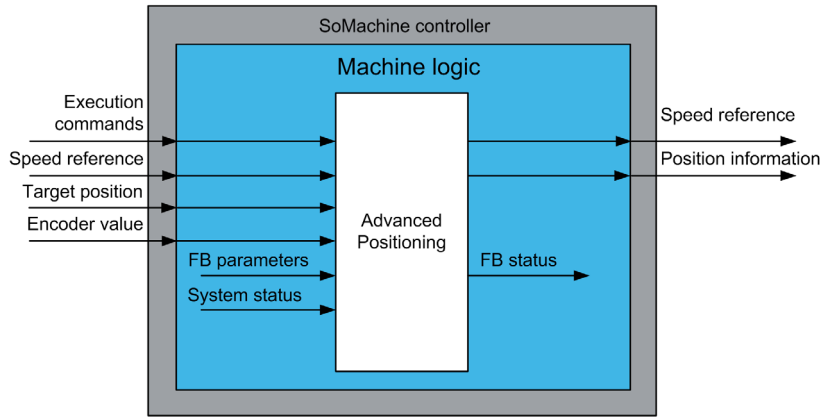
Although high performance positioning applications are the domain of servo drives with synchronous motors, asynchronous motors with variable speed drives offer performance that is sufficient for many applications. The function block extends the functionality of the variable speed drive by adding a position control function.

This function block is intended to have significant influence on the physical movement of its controlled axis and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

 WARNING
UNINTENDED EQUIPMENT OPERATION
Validate all function block input values before and while the function block is enabled.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Solution with the AdvancedPositioning Function Block

The AdvancedPositioning FB provides a solution for position control of a single axis driven by a variable speed drive.

Functional View

Section 3.2

Architecture

Architecture

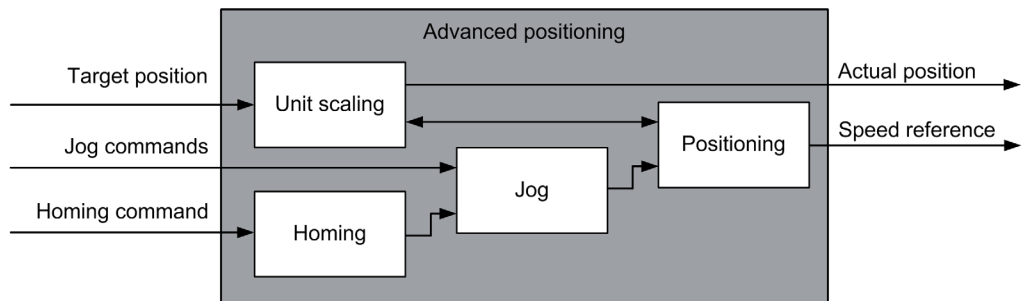
System Requirements and Environment

For details concerning the system requirements, refer to the chapter System Requirements (*see page 25*).

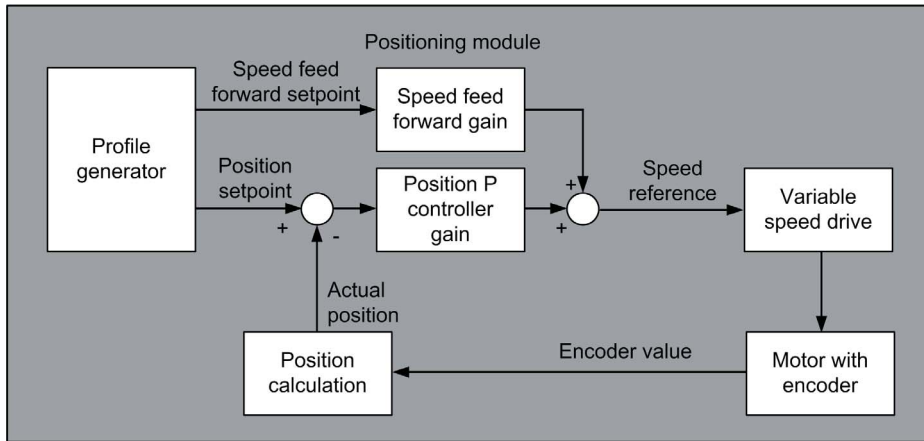
The `AdvancedPositioning` FB can run in a standard environment which is described inside the chapter Crane Environment and Architecture (*see page 37*).

Data Flow Overview

Internal structure of the `AdvancedPositioning` FB



Internal structure of the positioning module



Section 3.3

Function Block Description

AdvancedPositioning Function Block

Pin Diagram



Functional Description

The function block (FB) calculates a speed reference for variable speed drive. It contains a trapezoid speed profile generator and a position controller. Both ramping and speed reference calculations are handled by the FB. The variable speed drive must be configured to follow the speed reference given by the FB precisely. The FB supports absolute and relative movements.

Positioning is started by rising edge on `i_xPosExe` input.

Absolute movement:

- Movement to a pre-determined, fixed position of the axis.
- Supports change of the target position during execution (to prevent loss of position).

Relative movement:

- Relative change of axis position by a given distance.
- Does not support change of travel distance during execution (to prevent loss of position).

Modulo Axis

The FB supports positioning of modulo axis.

This feature allows the following:

- Positioning of the axis in one direction without risk of overflow of internal position value. This is useful, for example, for positioning of conveyors.
- Positioning of rotational axis in positive, negative and shortest distance directions.
- Absolute and relative positioning within one modulo range or across multiple modulo ranges.

Position Controller

Position controller modifies the speed reference given to the drive in order to decrease the difference between the target position and the actual position. It corrects position difference both during movement and while the axis is at a standstill.

Position controller is active when the

- variable speed drive is in RUN state,
- system is ready, and
- the FB is not in an alarm state.

Position controller is not active, when the

- FB was stopped using quick stop input,
- FB is in an alarm state, or
- the previous positioning task is not able to reach the target position.

Unit Scaling

Unit scaling allows scaling of the internal position value to user units. The absolute position of an axis is internally stored as a 32 bit integer value. For detailed description of unit scaling, refer to the description of the `i_stEncPara` structure.

Position Feedback

An encoder or another device dedicated to position measurement is necessary for correct functioning of the FB. It supports both incremental and absolute encoders.

The FB works internally with a 32 bit position value. If the resolution of the position measurement device interface is less than 32 bit (16 bit for the internal encoder interface of Altivar 71, usually 25 bit for absolute CANopen encoders), the FB compensates for overflow of the encoder interface value up to the range of the internal 32 bit position.

WARNING

UNINTENDED EQUIPMENT OPERATION

Always verify that the value of the encoder position increases in a positive direction and decreases in a negative direction before putting the machine or process into operational service.

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

Homing

The FB supports various homing methods. It uses positive and negative limit switches or an optional reference position limit switch to define a home position.

For detailed description of homing methods, refer to the description of the `i_stHmngPara` structure. Homing is performed automatically (without movement of the axis) after the first download and after a change between linear and modulo mode.

Jog Movement

Jog movement is designed to manually change a position of the axis. If configured (`i_stJogPara.diJogDist>0`), the axis performs a defined distance movement at a defined speed after a rising edge on a Jog command input (for example, one motor revolution). After this movement, the FB waits for a defined period (`i_stJogPara.wJogDelay`) and then, if the Jog command is still present, starts a continuous movement until the Jog command is set to FALSE or a limit switch is reached.

Limits of Movement

The FB supports limit switch signals for the definition of an operating area. It supports Normally Closed (N/C) limit switches. It does not support impulse limit switches, i.e., when the axis enters the area of the limit switch and beyond, the signal of the limit switch must remain FALSE.

Do not use positioning of linear axis without position limit switches.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use impulse or normally open contact switches in association with the function blocks.
- Only use Normally Closed contact switches with the function blocks.

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

NOTE: Limit switches are ignored in modulo mode (*see page 55*).

Control of Variable Speed Drive

The FB generates a signed speed reference profile for variable speed drive. Setting the drive to RUN state must be handled by the user application outside of the AdvancedPositioning FB.

Motor Control Mode of Used Drive

The positioning should be preferably used with drives running in closed loop motor control mode for better performance. However, it is still possible to position with a drive running in open loop motor control mode. For example, when the internal encoder interface of Altivar 71 is not used and a CANopen encoder is used for positioning, the positioning FB will still work, but the performance will be lower, since the drive running in open loop does not follow its speed reference as accurately as in closed loop.

Section 3.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	59
Output Pin Description	66

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
i_xPosMode	BOOL	Selects positioning mode. TRUE: Relative movement FALSE: Absolute movement Change of positioning mode is only possible when the axis is at a standstill ($q_wStat.2=FALSE$).
i_xPosExe	BOOL	Movement command (rising edge) Refer to detailed description below.
i_xJogFwd	BOOL	Jog forward command Refer to detailed description below.
i_xJogRev	BOOL	Jog reverse command. Refer to detailed description below.
i_xHmngExe	BOOL	Homing command Refer to detailed description below.
i_xQuickStop	BOOL	Quick stop command This input must be TRUE for the axis to operate. When the input is set to FALSE during axis movement, the movement is stopped using the pre-defined quick stop ramp ($i_stMotPara.diRampDecQuickStop$). A rising edge on reset input (i_xRst) is necessary to restore an operating state after performing a quick stop.
i_xHalt	BOOL	Halt command This input must be TRUE for the axis to operate. When set to FALSE, it temporarily interrupts the active movement. The axis decelerates to zero speed using the standard deceleration ramp. The axis resumes the operation after setting this input back to TRUE.
i_xLsFwd	BOOL	Normally Closed (N/C) forward limit switch signal. Refer to detailed description below.
i_xLsRev	BOOL	Normally Closed (N/C) reverse limit switch signal. Refer to detailed description below.
i_xRef	BOOL	Normally Open (N/O) reference switch signal. Refer to detailed description below.
i_xDrvRun	BOOL	Drive RUN state Refer to detailed description below.

Input	Data Type	Description
i_xSysRdy	BOOL	System Ready state Refer to detailed description below.
i_wPosSpdRef	WORD	Speed reference for positioning. Range: 0...6000 Default value: 0 Scaling/Unit: RPM Refer to detailed description below.
i_wJogSpdRef	WORD	Speed reference for jog movement. Range: 0...6000 Default value: 0 Scaling/Unit: RPM Refer to detailed description below.
i_wHmngSpdRef	WORD	Speed reference for homing. Range: 0...6000 Default value: 0 Scaling/Unit: RPM Refer to detailed description below.
i_diPosTarg	DINT	Target position Range: -2147483648...+2147483647 Scaling/Unit: user units Refer to detailed description below.
i_diEncVal	DINT	Encoder value input Range: -2147483648...+2147483647 Scaling/Unit: INC Refer to detailed description below.
i_stMotPara	stMotPara	Structure containing the motor parameters. Refer for detailed description to Structured Parameter (see page 72).
i_stPosPara	stPosPara	Structure containing the positioning parameters. Refer for detailed description to Structured Parameter (see page 73).
i_stJogPara	stJogPara	Structure containing the jog parameters. Refer for detailed description to Structured Parameter (see page 77).
i_stHmngPara	stHmngPara	Structure containing the homing parameters. Refer for detailed description to Structured Parameter (see page 78).
i_stEncPara	stEncPara	Structure containing the encoder parameters. Refer for detailed description to Structured Parameter (see page 84).

Input	Data Type	Description
i_xRst	BOOL	Resets detected alarms on rising edge, provided the cause of the alarm has been remedied. TRUE: Active FALSE: Inactive The input can also be used to stop a movement currently in progress and it is required after a quick stop.

i_xEn

When TRUE, this input activates the Advanced positioning function.

When FALSE, the FB enters a fallback state. In fallback state, speed reference outputs, status and alarm outputs are reset to zero (FALSE). Information about current position is retained. Any movement of the axis while the FB is disabled can lead to position information loss.

When i_xEn is set to FALSE while the axis is in motion, the currently active movement will continue until it is finished before entering disabled/fallback status.

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use the i_xEn input to attempt to stop an active movement.

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

i_xPosExe

A rising edge on this input executes a positioning task if the following conditions are fulfilled:

- FB enabled
- no alarm
- drive in RUN state (i_xDrvRun=TRUE)
- machine is ready to start (i_xSysRdy=TRUE)

Execution of positioning has a higher priority than an execution of jog or a homing movement if multiple executions are performed at the same time. If a homing or jog movement has already started, it must first finish the movement before the positioning task may start.

i_xJogFwd

A rising edge on this input starts a jog movement if the following conditions are fulfilled:

- drive in RUN state
- system is ready
- no alarm
- no movement in progress

If configured using `i_stJogPara.diJogDist` and `i_stJogPara.wJogDelay`, the axis performs a defined distance movement on a rising edge of `i_xJogFwd`. If `i_xJogFwd` is still TRUE after finishing the defined distance movement and elapsing of the time defined in `i_stJogPara.wJogDelay`, the axis enters into a continuous movement mode. When one or both of `i_stJogPara.diJogDist` and `i_stJogPara.wJogDelay` equal zero, continuous movement mode is started immediately.

When the axis is in continuous movement mode, setting the `i_xJogFwd` to FALSE stops the movement. The axis can be also stopped using Halt, Quick stop and Reset commands.

NOTE: If forward and reverse jog movements are requested at the same time, both commands are ignored.

Jog command has higher priority than homing command, but lower priority than positioning command if multiple commands are given at the same time.

i_xJogRev

A rising edge on this input starts a jog movement if the following conditions are fulfilled:

- drive in RUN state
- system is ready
- no alarm
- no movement in progress

If configured using `i_stJogPara.diJogDist` and `i_stJogPara.wJogDelay`, the axis performs a defined distance movement on rising edge of `i_xJogRev`. If `i_xJogRev` is still TRUE after finishing the defined distance movement and elapsing of the time defined in `i_stJogPara.wJogDelay`, the axis enters a continuous movement mode. When one or both of `i_stJogPara.diJogDist` and `i_stJogPara.wJogDelay` equal zero, continuous movement mode is started immediately.

When the axis is in continuous movement mode, setting the `i_xJogRev` to FALSE stops the movement. The axis can be also stopped using Halt, Quick stop and Reset commands.

NOTE: If forward and reverse jog movements are requested at the same time, both commands are ignored.

Jog command has higher priority than homing command, but lower priority than positioning command if multiple commands are given at the same time.

i_xHmngExe

A rising edge on this input starts homing operation if the following conditions are fulfilled:

- drive in RUN state
- system is ready
- no alarm
- no movement in progress

As an exception, the homing method 35 (*see page 78*) (immediate homing) does not require the drive to be in RUN state since homing method 35 does not require axis movement for successful homing.

i_xLsFwd

The input must be TRUE to allow movement in forward direction. The function block does not support impulse limit switches. If the limit switch is reached in relative positioning mode, a rising edge on the `i_xRst` input is necessary to cancel active motion task before starting another movement.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use impulse or normally open contact switches in association with the function blocks.
- Only use Normally Closed contact switches with the function blocks.

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

NOTE: Limit switch function is not active in modulo mode.

i_xLsRev

The input must be TRUE to allow movement in reverse direction. The function block does not support impulse limit switches. If the limit switch is reached in relative positioning mode, a rising edge on the `i_xRst` input is necessary to cancel active motion task before starting another movement.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use impulse or normally open contact switches in association with the function blocks.
- Only use Normally Closed contact switches with the function blocks.

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

NOTE: Limit switch function is not active in modulo mode.

i_xRef

This input defines the normally open (N/O) reference switch signal used in homing methods 23...30. Usage of this input is not mandatory when homing methods 23...30 are not used. For details, refer to the description of the structure (*see page 78*) `stHmngPara`.

i_xDrvRun

This input gives the FB information about the RUN status of the drive.

TRUE authorizes the start of the movement and must stay TRUE through the duration of the whole movement.

FALSE during the movement triggers an alarm state and stops the movement using Quick stop. For Altivar 71, 32 and 312, and other drives complying to CiA402/Drivecom standard, the information about RUN state is stored in bit 2 of the status word.

i_xSysRdy

This input gives information about status of relevant parts of the machine. The most important information is if the signal on the encoder value input `i_diEncVal` is valid.

The input `i_xSysRdy` must be TRUE to start the movement and must stay TRUE through the duration of the whole movement.

When this input is set to FALSE during movement, the FB enters an alarm state and stops the movement using Quick stop.

When the internal encoder interface of Altivar 71 is used, set the signal to TRUE when

- the drive is in operational state (while controlling the drive through CANopen), or
- the internal communication between ATV IMC and Altivar 71 is working (while controlling the drive through backplane bus).

NOTE: To best determine the communication status between the ATV IMC and the Altivar drive (while controlling the drive through backplane bus), use the `bError` output of MANDATORY_AT_EACH_CYCLE program.

When an external encoder is used (for example, absolute encoder on CANopen), this input needs to be TRUE only when the encoder is in an operational state and providing a position value.

i_wPosSpdRef

The speed reference is accepted at rising edge of `i_xPosExe` input. When there is a movement in progress, the change of `i_wPosSpdRef` is taken into account only at a rising edge of `i_xPosExe` input. This is possible only for absolute positioning (additive movement/position blending).

i_wJogSpdRef

The speed reference is accepted immediately even if it changes during movement.

i_wHmngSpdRef

The speed reference is used for reference/limit switch search. When the reference/limit switch is found, the axis continues with movement to homing edge at 1/4 of `i_wHmngSpdRef`.

When the speed is too high it would lead to overshooting of homing position or decreased accuracy of homing. Therefore, the FB detects this case and triggers a homing alarm.

i_diPosTarg

In absolute positioning mode (`i_xPosMode=FALSE`), this input defines the target position in user units.

In relative positioning mode (`i_xPosMode=TRUE`), this input defines length of relative movement from actual planned position.

NOTE: A planned position does not necessarily equal the actual position. It is normal that after stopping of the axis there is a certain difference between the planned position and the actual position. This is acceptable as long as the difference is within a defined deadband. However, these slight position differences could cause position loss when performing repetitive relative movements. The FB accounts for the discrepancies by calculating the relative movements based on planned position instead of actual position. For more information, refer to the description of `xMoveInDsbl` parameter (*see page 86*) inside `i_stEncPara` input structure.

i_diEncVal

This value can be taken either from the encoder interface of Altivar 71 with encoder card or an alternative source (CANopen absolute encoder or other position sensor).

WARNING

UNINTENDED EQUIPMENT OPERATION

Always verify that the value of the encoder position increases in a positive direction and decreases in a negative direction before putting the machine or process into operational service.

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

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	This output mirrors the value of i_xEn input variable. TRUE: The function block is enabled. FALSE: The function block is disabled. When the input i_xEn turns FALSE during movement of the axis, the FB enters the disabled state once the movement has finished.
q_iSpdRef	INT	Signed output speed reference. Range: -32768...+32767 Scaling/Unit: RPM Refer to detailed description below this table.
q_iSpdRefFull	INT	High resolution signed speed reference output. Range: -32768...+32767 Refer to detailed description below this table.
q_diPosActlInc	DINT	Actual position of the axis in increments. This is the position the FB works with internally. The value is a 32 bit integer value. Overflow of the internal position causes the FB to enter an alarm state. Range: -2147483648...+2147483647 Scaling/Unit: inc
q_diPosActlUsr	DINT	Contains actual position of the user axis in user units. It is calculated from the internal position using parameters (<i>see page 84</i>) defined inside the encoder configuration structure i_stEncPara. Range: -2147483648...+2147483647 Scaling/Unit: user units
q_diPosPlanInc	DINT	Planned position. Range: -2147483648...+2147483647 Scaling/Unit: inc Refer to detailed description below this table.
q_diPosDiffInc	DINT	Difference between planned and actual position. Range: -2147483648...+2147483647 Scaling/Unit: inc Refer to detailed description below this table.
q_diDistToTargInc	DINT	Distance to the target position in positioning modes in encoder increments. Range: -2147483648...+2147483647 Scaling/Unit: inc
q_diSpdActl	DINT	Actual speed of the axis. This value is calculated using the encoder feedback. Range: -2147483648...+2147483647 Scaling/Unit: inc/s

Output	Data Type	Description
q_diSpdPlan	DINT	Contains the speed reference calculated by the internal speed/position profile generator without added action of the position controller. Range: -2147483648...+2147483647 Scaling/Unit: RPM
q_xInPos	BOOL	Informs about the axis being in the planned position. This information is also available in the status word of the FB (q_dwStat).
q_dwStat	DWORD	This output is the status register. Refer to Status Notifications (see page 68). Range: 0...65535
q_xAlrm	BOOL	This output is TRUE when an alarm is detected. TRUE: Alarm detected. FALSE: No alarm is detected. In case of an alarm the FB performs a controlled stop, using the Quick stop ramp, and does not allow any further movement until the cause of the alarm has been remedied.
q_dwAlrmId	DWORD	This output is the detected alarm register. Refer to Notifications (see page 69). Range: 0...65535

q_iSpdRef

This output contains the signed speed reference for a variable speed drive. Using a negatively signed speed reference allows you to move the motor in reverse direction with forward command on the drive. This method allows for a simple change of direction while the command to the drive is controlled by user application outside of the FB.

For detailed information, refer to the Quick Reference Guide ([see page 93](#)).

q_iSpdRefFull

This output contains the signed speed reference for a variable speed drive in high resolution. The maximum and minimum values correspond to maximum and minimum speed reference/frequency configured in the FB. The output is intended for usage with a variable speed drive in high resolution frequency reference mode.

Altivar variable speed drives can be switched to high resolution frequency reference mode using bit 9 of extended command word (CMI).

For detailed information, refer to the documentation of your variable speed drive.

q_diPosPlanInc

This output contains the planned position of the axis in encoder increments. Ideally, this value should be equal to `q_diPosActlInc`. In reality, there is almost always deviation between these values. The deviation can increase during periods of acceleration and deceleration, and may trigger an alarm. If the deviation becomes too great, you need to review the configuration of the application/machine.

q_diPosDiffInc

This output contains the difference between the planned and the actual position in encoder increments. The lower the value, the greater is the accuracy of positioning. When this value exceeds the maximum allowed position difference defined in `i_stPosPara.diPosDiffMax`, a “position following” alarm is triggered and the axis stops using Quick stop.

q_dwStat

This output contains the status of the FB.

Bit Position	Status Description Represented by Bit Position
Bit 0	Axis ready for positioning task. Drive is in RUN state, encoder data is valid, axis position initialized, FB is not in alarm state, FB is not in Quick stop state.
Bit 1	In position. Axis is within in-position window defined by <code>i_stPosPara.diInPosWdow</code> from the target position.
Bit 2	Detected error on positioning. Target position not reached (actual position is not within the window defined by <code>i_stPosPara.diInPosWdow</code>) within defined time <code>i_stPosPara.wPosFailTime</code> .
Bit 3	Axis in motion.
Bit 4	Positioning task in progress.
Bit 5	Acknowledgement of execution and homing command. Jog command is not acknowledged.
Bit 6	Quick stop is active. The input <code>i_xQuickStop</code> is FALSE or was FALSE previously. A rising edge on the <code>i_xRst</code> input is required to reset the Quick stop state.
Bit 7	Halt is active. The input <code>i_xHalt</code> is FALSE.
Bit 8	Homing in progress.
Bit 9	Homing finished.
Bit 10	Homing done.
Bit 11	Unsuccessful homing.

Bit Position	Status Description Represented by Bit Position
Bit 12	Homing speed too high.
Bit 13	Homing method not supported.
Bit 14	Jog movement in progress.
Bit 15	Jog movement finished.
Bit 16	Long move mode. The distance to target is higher than 2147483647 internal increments.
Bit 17	User unit configuration causes overflow of internal target position.
Bit 18	Actual position cannot be converted to user units.
Bit 19	User unit configuration causes overflow of internal jog distance.
Bit 20	User unit configuration causes overflow of internal clearing distance.
Bit 21	User unit configuration causes overflow of internal homing reference position.
Bit 22	User unit configuration causes overflow of internal position of positive software stop switch.
Bit 23	User unit configuration causes overflow of internal position of negative software stop switch.
Bit 24	Axis reached either hardware or software limit switch.
Bit 25	Modulo mode.
Bit 26	Modulo direction not supported, supported directions are 0,1 and 2.
Bit 27	User unit configuration causes overflow of internal modulo range.
Bit 28...31	Reserved

q_dwAlrmId

This output identifies the detected alarm(s) indicated by the FB. A value of zero indicates no alarms detected.

Bit Position	Status Description Represented by Bit Position
Bit 0	The distance between planned and actual position exceeds preset threshold (<code>i_stPosPara.diPosDiffMax</code>).
Bit 1	System not ready or drive not running. Falling edge of either <code>i_xDrvRun</code> or <code>i_xSysRdy</code> detected during axis movement.
Bit 2	Overflow of internal 32 bit absolute position.
Bit 3	Two consecutive overflows of encoder interface detected. This indicates a possible loss of position. It can happen if the speed of the axis is too high, resolution of encoder interface is too low and execution period of positioning task is too long.
Bit 4	<code>i_wPosSpdRef > 6000</code>
Bit 5	<code>i_stMotPara.diRampAcc < 1</code>
Bit 6	<code>i_stMotPara.diRampDec < 1</code>

Bit Position	Status Description Represented by Bit Position
Bit 7	i_stMotPara.diRampDecQuickStop<1
Bit 8	i_stPosPara.diPosDiffMax<0
Bit 9	i_stPosPara.diInPosWdow<0
Bit 10	i_stPosPara.rKpPos<0
Bit 11	i_stPosPara.rKpFeedFwd<0
Bit 12	i_stMotPara.wMotFreqNom=0 or i_stMotPara.wMotFreqNom>3000
Bit 13	i_stMotPara.wMotFreqMax=0 or i_stMotPara.wMotFreqMax>3000
Bit 14	i_stMotPara.wMotSpdNom=0 or i_stMotPara.wMotSpdNom>6000
Bit 15	i_stJogPara.wJogSpdRef>6000
Bit 16	i_stJogPara.diJogDist<0
Bit 17	i_wHmngSpdRef>6000
Bit 18	i_stHmngPara.diHmngDistOut<0
Bit 19	i_stEncPara.wIntfResol<1 or i_stEncPara.wintfResol>32
Bit 20	i_stEncPara.diPulsPerRevo<1
Bit 21	i_stEncPara.diScalNumer<1
Bit 22	i_stEncPara.diScalDenom<1
Bit 23	i_stEncPara.diModRng<0 and i_stEncPara.xModulo=TRUE
Bit 24...31	Reserved.

Section 3.5

Structured Parameters

What Is in This Section?

This section contains the following topics:

Topic	Page
Structured Parameter - stMotPara	72
Structured Parameter - stPosPara	73
Structured Parameter - stJogPara	77
Structured Parameter - stHmngPara	78
Structured Parameter - stEncPara	84

Structured Parameter - stMotPara

Structure: stMotPara

Structure of parameters related to the motor configuration

Parameter	Data Type	Description
diRampAcc	DINT	Acceleration ramp. This parameter is common to Positioning, Jog and Homing modes. Increasing value of <code>i_stMotPara.diRampAcc</code> makes the axis accelerate faster. Decreasing it makes the axis accelerate slower. Range: 1...2147483647 Default value: 100 Scaling/Unit: RPM/s
diRampDec	DINT	Deceleration ramp. This parameter is common to Positioning, Jog and Homing modes. Increasing value of <code>i_stMotPara.diRampDec</code> makes the axis decelerate faster. Decreasing it makes the axis decelerate slower. Range: 1...2147483647 Default value: 100 Scaling/Unit: RPM/s
diRampDecQuickStop	DINT	This parameter defines deceleration of the axis for Quick stop. Increasing value of <code>i_stMotPara.diRampDecQuickStop</code> makes the axis decelerate faster. The input <code>i_diRampQuickStop</code> deceleration is used in case of an alarm of the FB. The value should be higher than <code>i_stMotPara.diRampDec</code> . Range: 0...2147483647 Default value: 100 Scaling/Unit: RPM/s
wMotFreqNom	WORD	Nominal motor frequency. Range: 0...3000 Default value: 500 Scaling/Unit: 0.1 Hz (at 50 Hz: <code>i_stMotPara.wMotFreqNom=500</code>)
wMotFreqMax	WORD	Maximum motor frequency used for movement for the axis. Range: 0...3000 Default value: 500 Scaling/Unit: 0.1 Hz (50 Hz: <code>i_stMotPara.wMotFreqMax=500</code>) This value must not exceed the high speed configured inside the drive.
wMotSpdNom	WORD	Nominal motor speed. Assign the value of nominal speed of the motor from the motor nameplate to this input. Range: 0...6000 Default value: 1500 Scaling/Unit: RPM

Structured Parameter - stPosPara

Structure: stPosPara

Structure of parameters related to positioning

Parameter	Data Type	Description
diPosDiffMax	DINT	Maximum allowed difference between planned and actual position. Range: 0...2147483647 Default value: 0 Scaling/Unit: inc Refer to detailed description below this table.
diInPosWdow	DINT	Size of in-position window. Range: 0...2147483647 Default value: 50 Scaling/Unit: inc Refer to detailed description below this table.
winPosTime	WORD	Time inside in-position window to consider positioning finished. Range: 0...65535 Default value: 200 Scaling/Unit: ms Refer to detailed description below this table.
wPosFailTime	WORD	Time for positioning failure detection. Range: 0...65535 Default value: 2000 Scaling/Unit: ms Refer to detailed description below this table.
rKpPos	REAL	Proportional gain of the position controller. Range: 0...3.4028e+38 Default value: 5 Refer to detailed description below this table.
rKpFeedFwd	REAL	Feed forward gain of the position controller. Range: 0...3.4028e+38 Default value: 0 Refer to detailed description below this table.
diLsPosFwdSw	DINT	Position of positive software limit switch. Range: -2147483648...+2147483647 Default value: 0 (limit switch inactive) Scaling/Unit: user units Refer to detailed description below this table.
diLsPosRevSw	DINT	Position of negative software limit switch. Range: -2147483648...+2147483647 Default value: 0 (limit switch inactive) Scaling/Unit: user units Refer to detailed description below this table.

diPosDiffMax

The parameter is defined in encoder pulses. Drive running in closed loop motor control mode will allow for lower value of `i_stPosPara.diPosDiffMax` because it can follow the target speed more accurately. When the drive is running in open loop motor control mode, it is necessary to increase the value to help preventing alarms.

diInPosWdow

This parameter defines the required accuracy of positioning.

Positioning is considered successfully finished when the difference between the actual and target position of the axis is lower than `i_stPosPara.diInPosWdow` for the time defined by `i_stPosPara.wInPosTime`. The parameter `i_stPosPara.diInPosWdow` is defined in encoder pulses.

Excessively low value will cause reports of detected error on positioning.

Zero value disables the in-position window function. Target position is considered reached immediately after profile generator reaches the target position.

winPosTime

Positioning is considered successfully finished when the axis is inside `i_stPosPara.diInPosWdow` for at least the time defined by `i_stPosPara.wInPosTime`.

wPosFailTime

This parameter detects unsuccessfully positioning. The time starts running when the internal position/speed profile generator reaches the target position. If the positioning does not finish successfully within this time, an alarm is raised.

The time must be long enough to allow for finishing of the positioning under normal conditions.

rKpPos

A higher value of this input will proportionally increase the output of the internal position controller. The parameter influences the reactivity of the axis and also the accuracy of positioning.

Start the commissioning with a low value (for example 5) and slowly increase to find the optimal setting. If the response of the axis is too slow and the position deviation is too great, the value of `rKpPos` is too low. If the axis starts to oscillate, the value is too high.

The value of proportional gain is internally compensated for encoder resolution (pulses per revolution). It is not necessary to change this value if an encoder with different resolution is used.

The value of applicable proportional gain depends also on the accuracy (lag) of position measurement and accuracy of the speed loop of used variable speed drive.

To allow setting considerably higher value of `i_stPosPara.rKpPos` and achieving a higher level of accuracy, proceed as follows:

- Run the positioning in a task without short execution period.
- Transfer information about axis position via backplane communication of Altivar 71 or a high speed CANopen connection.
- Configure the variable speed drive closed loop motor control mode.

Changes of the parameter value are taken into account immediately.

rKpFeedFwd

This parameter defines the direct influence of internally generated speed profile on speed reference for the drive. Higher value will make the axis more reactive. Too high value will cause oscillations of the axis.

Setting of this value is optional. The default value is 0. Find an optimal setting of `i_stPosPara.rKpPos` before increasing `i_stPosPara.rKpFeedFwd`.

Changes of the parameter value are taken into account immediately.

Excessive value of proportional gain of position controller and feed forward gain of the position controller may lead to oscillations of controlled axis.

WARNING

UNINTENDED EQUIPMENT OPERATION

Always start with low values of position controller gains and increase them gradually.

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

diLsPosFwdSw

This parameter defines the position of optional software limit switch in positive direction. The software limit switch function stops movement of the axis when actual position of the axis is higher than position of the software limit switch.

NOTE: This function is active in Positioning and Jog modes. It is not active in Homing mode.

The software limit switch function is active when homing has been done (`q_wStat.9` is TRUE) and `i_stPosPara.diLsPosFwdSw` does not equal 0.

Software limit switch function is not active in Modulo mode.

diLsPosRevSw

This parameter defines the position of optional software limit switch in negative direction. The software limit switch function stops movement of the axis when actual position of the axis is lower than position of the software limit switch.

NOTE: This function is active in Positioning and Jog modes. It is not active in Homing mode.

The software limit switch function is active when homing has been done (`q_wStat.9` is TRUE) and `i_stPosPara.diLsPosRevSw` does not equal 0.

Software limit switch function is not active in Modulo mode.

Structured Parameter - stJogPara

Structure: stJogPara

Structure of parameters related to jog movement.

Parameter	Data Type	Description
diJogDist	DINT	<p>Defines the distance of a movement after rising edge of Jog command. This parameter is used to perform jog movement by a certain distance.</p> <p>Setting <code>i_stJogPara.diJogDist</code> to 0 disables this option. The axis in this case enters immediately the continuous movement mode.</p> <p>Range: 0...2147483647 Default value: 0 Scaling/Unit: user units</p>
wJogDelay	WORD	<p>Configures delay between finishing the defined distance jog movement and start of continuous jog movement.</p> <p>Setting <code>i_stJogPara.wJogDelay</code> to 0 disables this option. The axis in this case enters immediately the continuous movement mode.</p> <p>Range: 0...65535 Default value: 0 Scaling/Unit: ms</p>

Structured Parameter - stHmngPara

Structure: stHmngPara

Structure of parameters related to jog movement.

Parameter	Data Type	Description
wHmngMeth	WORD	Selection of homing method. Range: 17, 18, 23...30, 35 Default value: 35 Refer to detailed description below this table.
diHmngDistOut	DINT	Sets the distance of clearance movement after homing on rising edge on reference/limit switch. Range: 0...2147483647 Default value: 0 Scaling/Unit: user units
diHmngPosRef	DINT	The FB uses this value as a position of the axis for homing. The value is written to the internal position of the axis at the moment of homing (rising edge on reference/limit switch). Range: -2147483648...+2147483647 Default value: 0 Scaling/Unit: user units

wHmngMeth

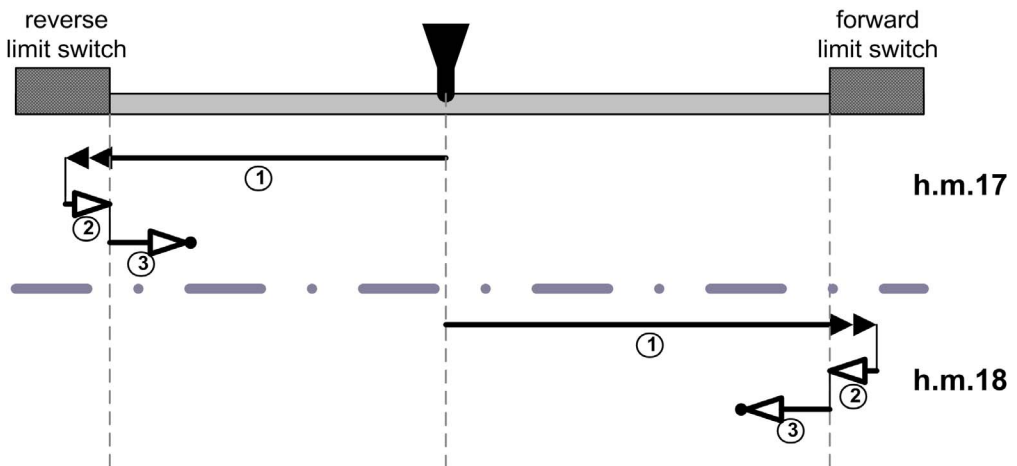
The FB implements homing methods using forward and reverse limit switches (homing method 17,18), methods using reference position switch (homing method 23...30) and immediate homing at current position (homing method 35).

Homing method overview:

Homing Method	Description
17	Homing on reverse limit switch.
18	Homing on forward limit switch.
23	Homing on reference position switch on falling edge of reference switch signal in reverse direction with clearance movement in reverse direction.
24	Homing on reference position switch on falling edge of reference switch signal in reverse direction with clearance movement in forward direction.
25	Homing on reference position switch on falling edge of reference switch signal in forward direction with clearance movement in reverse direction.
26	Homing on reference position switch on falling edge of reference switch signal in forward direction with clearance movement in forward direction.
27	Homing on reference position switch on falling edge of reference switch signal in forward direction with clearance movement in forward direction.

Homing Method	Description
28	Homing on reference position switch on falling edge of reference switch signal in forward direction with clearance movement in reverse direction.
29	Homing on reference position switch on falling edge of reference switch signal in reverse direction with clearance movement in forward direction.
30	Homing on reference position switch on falling edge of reference switch signal in reverse direction with clearance movement in reverse direction.
35	Immediate homing.

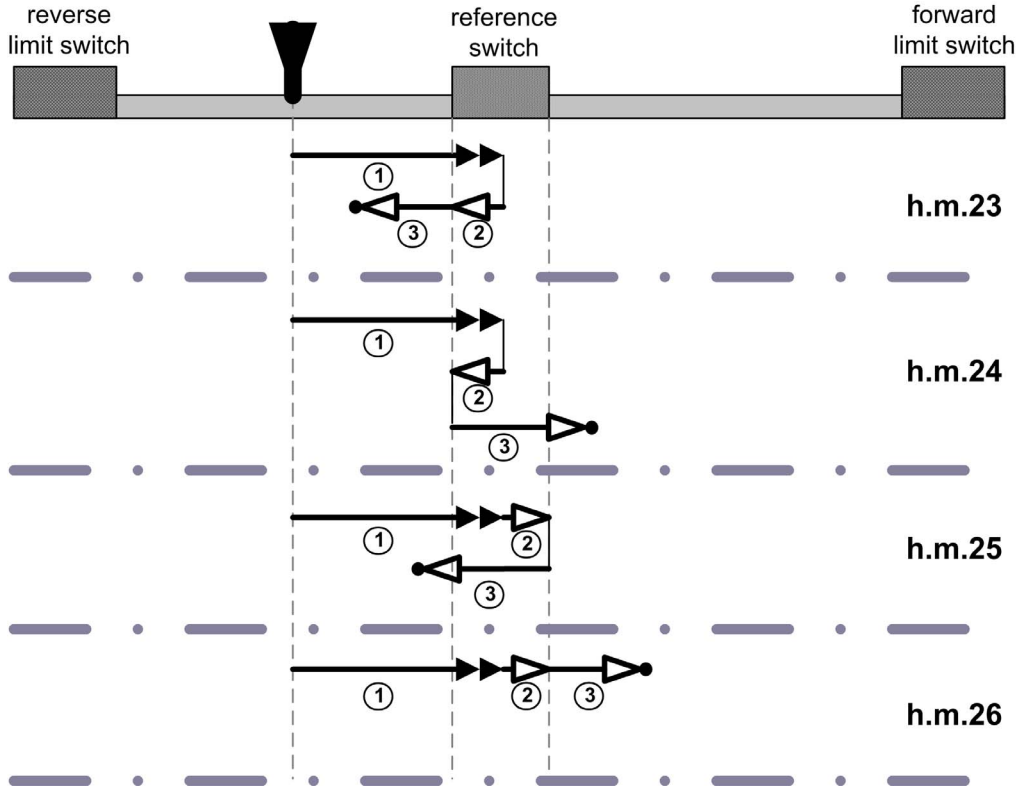
Homing methods (h.m.) 17 and 18



- 1 Movement towards limit switch at speed defined by `i_wHmngSpdRef`.
 - 2 Movement to homing edge at 1/4 of homing speed. The homing is performed at the rising edge of limit switch signal.
 - 3 Clearance movement at 1/4 of homing speed reference. The clearance distance is defined in `i_stHmngPara.diHmngDistOut`.
- Position of the axis after successful homing.

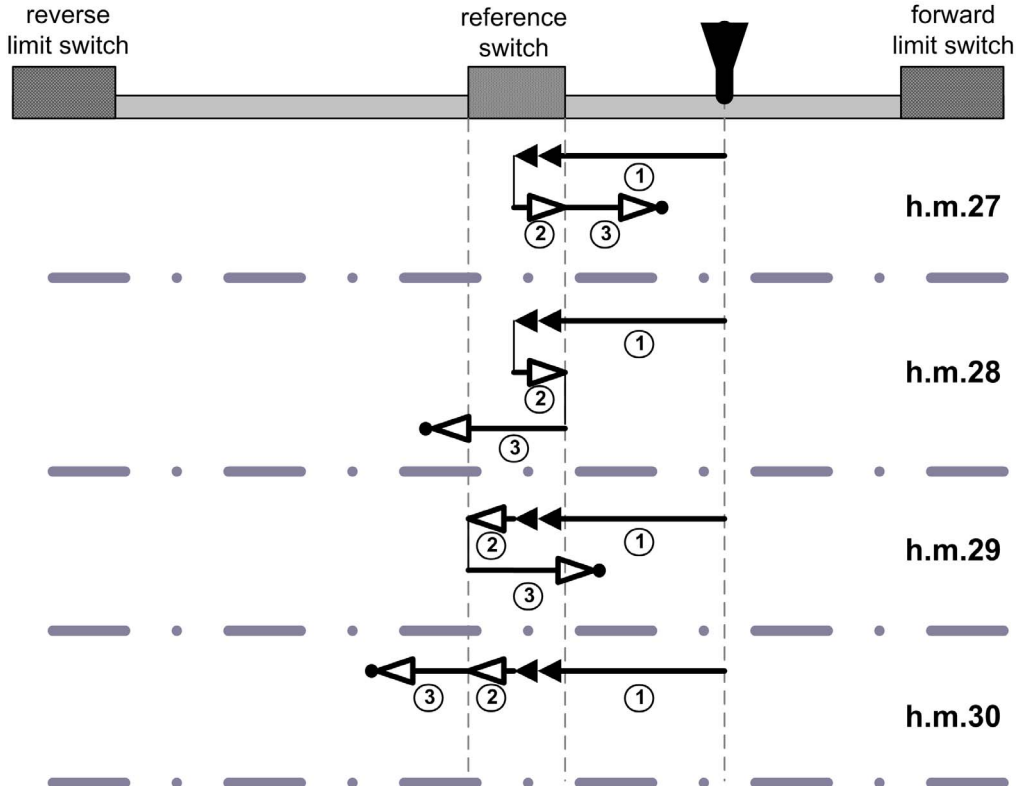
NOTE: Homing methods 17 and 18 are using limit switches and therefore are applicable only on linear axis. They are not supported in Modulo mode.

Homing methods (h.m.) 23...26



- 1 Movement towards reference switch at speed defined by `i_wHmngSpdRef`.
 - 2 Movement to homing edge at 1/4 of homing speed. The homing is performed at the rising edge of reference switch signal.
 - 3 Clearance movement at 1/4 of homing speed reference. The clearance distance is defined in `i_stHmngPara.diHmngDistOut`.
- Position of the axis after successful homing.

Homing methods (h.m.) 27...30



- 1 Movement towards reference switch at speed defined by $i_wHmngSpdRef$.
 - 2 Movement to homing edge at 1/4 of homing speed. The homing is performed at the rising edge of reference switch signal.
 - 3 Clearance movement at 1/4 of homing speed reference. The clearance distance is defined in $i_stHmngPara.diHmngDistOut$.
- Position of the axis after successful homing.

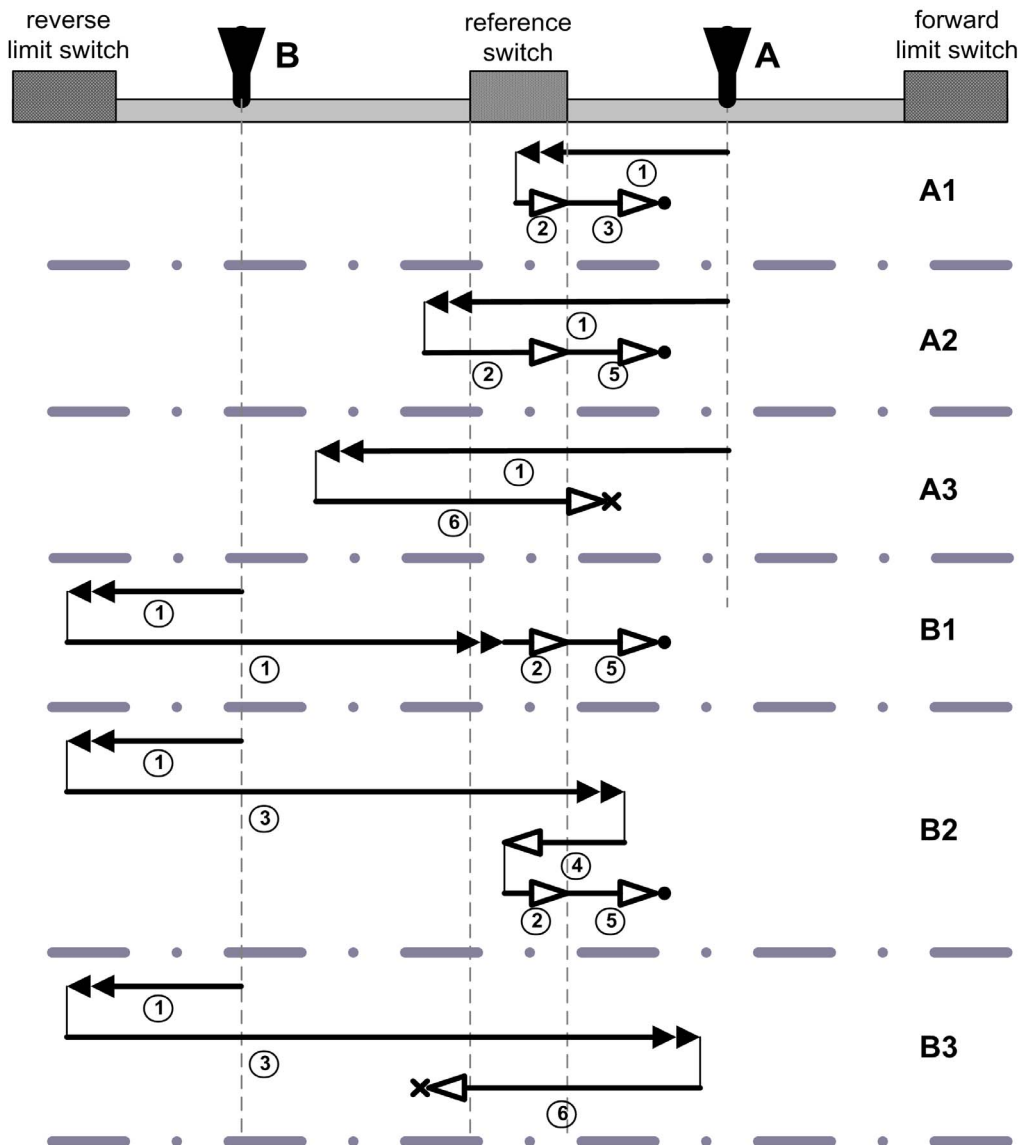
The following figure shows various kinds of possible behavior of an axis during homing with homing method 27.

The examples under A) show possible behavior when homing starts between the reference switch and the forward limit switch. Examples under B) describe behavior for homing with axis starting between the reference switch and reverse limit switch.

Behavior of the axis may differ for various values of homing speed $i_wHmngSpdRef$, various breadths of reference switch area and various execution times of the task calling `AdvancedPositioning FB`.

Similar behavior can be expected from homing methods using a reference position switch (homing method 23...30).

Possible behavior during homing for methods 23...30:



1 Movement towards reference switch at speed defined by `i_wHmngSpdRef`.

- 2 Movement to homing edge at 1/4 of homing speed. The homing is performed at the rising edge of reference switch signal.
- 3 Excessive speed (`i_wHmngSpdRef`) causing overshoot of the reference switch.
- 4 Return movement after overshoot at 1/4 of homing speed reference.
- 5 Clearance movement at 1/4 of homing speed reference. The clearance distance is defined in `i_stHmngPara.diHmngDistOut`.
- 6 The movement at reduced speed 1/4 of homing speed is still too fast and causes an overshoot of reference switch position.
 - Position of the axis after successful homing.
 - x Position of the axis after Unsuccessful homing.

Structured Parameter - `stEncPara`

Structure: `stEncPara`

Structure of parameters related to encoder interface.

Parameter	Data Type	Description
<code>wIntfResol</code>	WORD	Bit resolution of encoder interface. Range: 1...32 Default value: 16 Refer to detailed description below this table.
<code>diPulsPerRevo</code>	DINT	Defines the number of encoder increments/pulses per revolution of the motor. To change this parameter is only possible when <code>i_xDrvRun = FALSE</code> . Range: 1...2147483647 Default value: 1024
<code>xEncType</code>	BOOL	Encoder type selection. TRUE: Absolute encoder FALSE: Incremental encoder Refer to detailed description below this table.
<code>xPosIncSave</code>	BOOL	Save calculated absolute position when used with incremental encoder. TRUE: Saved FALSE: Not saved Refer to detailed description below this table.
<code>xMoveInDsbl</code>	BOOL	Movement in disabled state. TRUE: Movement allowed FALSE: Movement not allowed Refer to detailed description below this table.
<code>diScalNumer</code>	DINT	Numerator for position scaling. Range: 1...2147483647 Default value: 1 Refer to detailed description below this table.
<code>diScalDenom</code>	DINT	Denominator for position scaling. Range: 1...2147483647 Default value: 1024 Refer to detailed description below this table.
<code>xModulo</code>	BOOL	Selection between linear and modulo axis. TRUE: Modulo axis FALSE: Linear axis Refer to detailed description below this table.

Parameter	Data Type	Description
diModRng	DINT	Range of modulo axis. Range: 0...2147483647 Default value: 3600 Scaling/Unit: user units Refer to detailed description below this table.
wModDir	WORD	Direction of modulo movement. Range: 0...2 Default value: 0 Refer to detailed description below this table.
xModMultiRng	BOOL	Multi-range modulo movement. Default state: FALSE Refer to detailed description below this table.

wIntfResol

This parameter defines the bit resolution of used encoder interface:

16 bit resolution for the internal encoder interface of Altivar 71

25 bit resolution for the interface of absolute CANopen encoders of the XCC range

32 bit resolution for interfaces of integer type (-2147483648...2147483647)

NOTE: 32 bit interfaces of unsigned type (0 to 4294967296) are not supported.

Change of these parameter is possible only when `i_xDrvRun = FALSE`.

xEncType

FALSE selects an incremental encoder. Incremental encoder in this context means, that after power cycle of the machine, the value of encoder position on the input `i_diEncVal` is reset to 0.

TRUE selects an absolute encoder which retains its position value through a power cycle.

This information is important for correct function of the FB which calculates the internal position of the axis based on information from the encoder interface.

WARNING

UNINTENDED EQUIPMENT OPERATION

Remove the power of the incremental encoder interface every time the controller is switched off.

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

For example, when using the internal encoder interface of Altivar 71, it is necessary to switch the drive off and let the encoder value of the drive reset to zero every time the controller is restarted.

xPosIncSave

This parameter is only taken into account if an incremental encoder is used (`i_stEncPara.xEncType=FALSE`).

It configures whether the FB retains the internal absolute position during a power cycle or not. When `i_stEncPara.xPosIncSave` is `FALSE`, the internal absolute position is not retained and you need to perform homing after every start of the machine.

NOTE: If the drive remains under power while power is removed from the controller, the incremental encoder interface within the drive will retain its value. Once the controller is restarted, and if the parameter `i_stEncPara.xEncType=TRUE`, you can use the incremental encoder interface as a pseudo-absolute encoder.

In this case, the FB does not store the value of encoder interface when the controller is powered down. It stores only the information about number of overflows of the encoder interface of the drive and establishes the internal absolute position using the position information from incremental encoder interface of the drive. If the drive had been reset (either through power cycled or product reset) and the encoder position is reset (equal to zero), you must perform a new homing.

xMoveInDsbl

If the axis can move while the drive or FB is disabled or while the machine is powered off, set this parameter to `TRUE`. This parameter makes the planned position of the FB follow its actual position while the drive is not in `RUN` state. This helps preventing excessive acceleration applied to the axis.

If the axis position is fixed by a brake in disabled state or if the axis cannot move on its own, set this input to `FALSE`.

diScalNumer / diScalDenom

These parameters are the numerator and denominator values used for position value scaling. These values are used to calculate the scaling factor. The scaling factor defines the relationship between the number of motor revolutions and the corresponding distance in user units.

$$\text{Scaling factor} = \frac{i_stEncPara.diScalNumer}{i_stEncPara.diScalDenom} \left[\frac{\text{motor revolutions}}{\text{change of position in user units}} \right]$$

For example, when the axis movement by 10 mm corresponds to 7 motor revolutions, set

- `i_stEncPara.diScalNumer=7` and
- `i_stEncPara.diScalDenom=10`.

This will configure the user unit to 1 mm.

If a higher resolution is needed, set the `i_stEncPara.diScalDenom=100`. This will configure the user unit to 0.1 mm

$$\text{Scaling factor} = \frac{i_stEncPara.diScalNumer}{i_stEncPara.diScalDenom} = \frac{7}{100}$$

Similarly, the FB may be configured for positioning in angular units.

When one revolution of the machine axis (gearbox output shaft) corresponds to 15.5 revolutions of the motor, and the required resolution is 0.1° , then set the

- `i_stEncPara.diScalNumer=155` and
- `i_stEncPara.diScalDenom=36000`.

Numerator and denominator that are not relatively prime can be divided by their greatest common divisor.

In this case it is 5. That results in:

$$\text{Scaling factor} = \frac{i_stEncPara.diScalNumer}{i_stEncPara.diScalDenom} = \frac{15.5}{3600} = \frac{155}{36000} = \frac{31}{7200}$$

You can change these parameters only when `i_xDrvRun = FALSE`. When the axis is in Modulo mode, homing using homing method 35 is automatically performed on change of `i_stEncPara.diScalNumer` or `i_stEncPara.diScalDenom`.

xModulo

With this parameter you can select between positioning of linear (FALSE) and rotary (TRUE) axis. In Modulo mode is the position of the axis defined on a range between 0 and `i_stEncPara.diModRng`.

Modulo mode is very useful for positioning of axes turning in one direction because it prevents uncontrolled overflow of internal position.

Limit switch function (both, hardware and software) is disabled in Modulo mode.

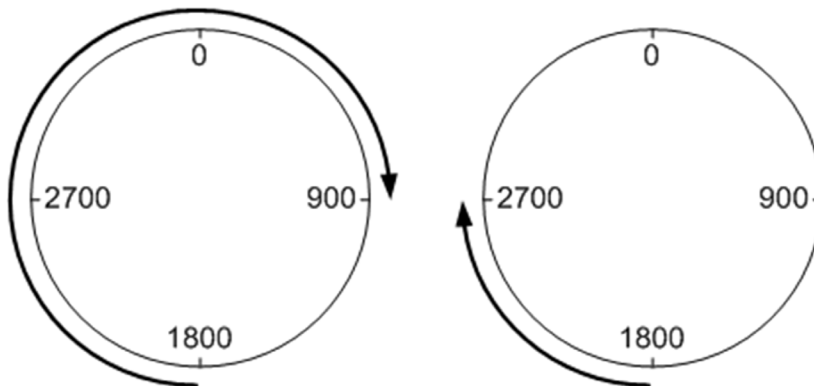
Modulo movement is supported in both, absolute and relative modes, depending on setting of input `i_xPosMode`.

Following images show movement with

- `i_diPosTarg = 900` and
- `wModDir = 1`.

The left image shows positioning in absolute mode, the right image positioning in relative mode. You can change this parameter only when `i_xDrvRun = FALSE`. When the axis is in Modulo mode, homing using homing method 35 is automatically performed on change of `i_stEncPara.xModulo`.

Position in modulo movement

**diModRng**

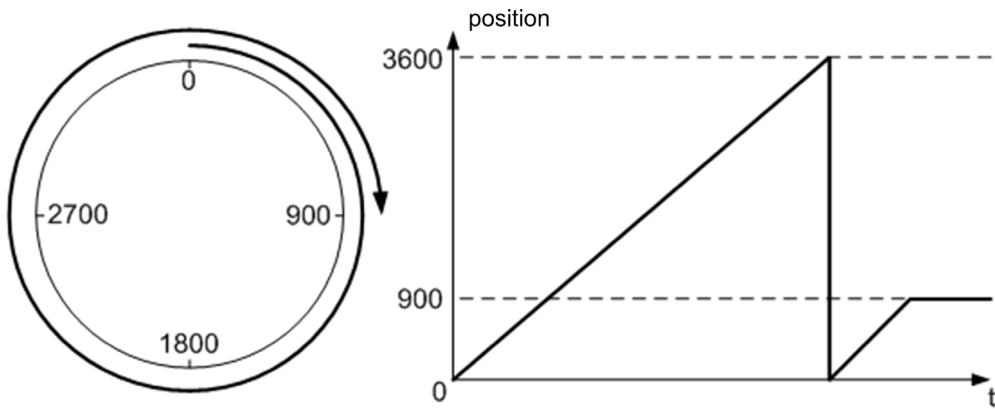
The value must be higher than 0 when `i_stEncPara.xModulo = TRUE`.

For example, the modulo range can be 3600 for a modulo axis with 360° rotation range. This gives the axis a resolution of 0.1°.

Following figure depicts change of position in time of modulo movement with `i_stEncPara.diModRng = 3600`.

You can change this parameter only when `i_xDrvRun = FALSE`. Homing using homing method 35 is automatically performed on change of `i_stEncPara.diModRng`.

Position in modulo movement

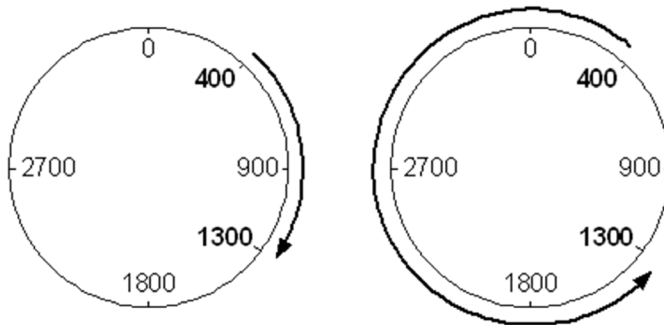
**wModDir**

This parameter defines direction of rotation of a modulo axis. Supported values are:

- 0 shortest distance
- 1 positive direction (forward)
- 2 negative direction (reverse)

The left figure shows movement in modulo directions 0 and 1, the right one movement in direction 2.

Modulo direction

**xModMultiRng**

With this parameter you can select between positioning within one revolution of the modulo axis (`i_stEncPara.xModMultiRng = FALSE`) and positioning over multiple modulo ranges (`i_stEncPara.xModMultiRng = TRUE`).

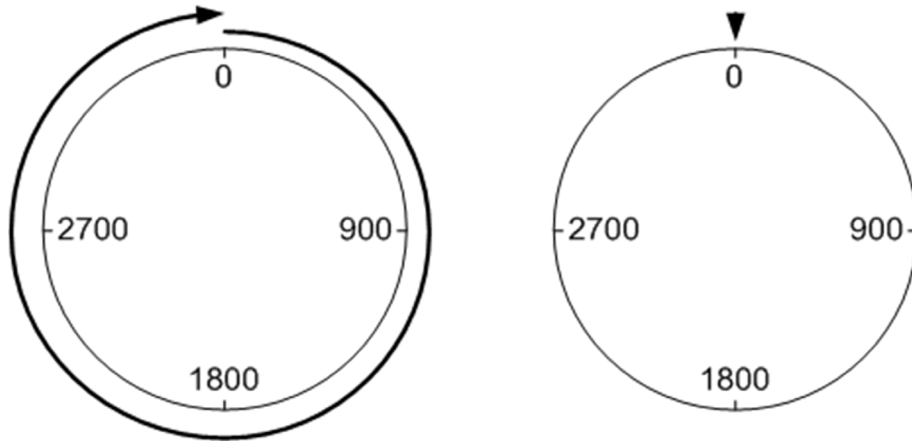
It can be used to perform positioning tasks ending at the same position as the starting position (whole-number multiples of modulo range) or tasks exceeding modulo range.

The following example describes a positioning task with

- target position `i_diPosTarg = 3600`,
- on modulo range `i_stEncPara.diModRng = 3600`,
- with `wModDir = 1`,
- starting in position 0.

The diagram on the left side shows behavior in multiple-range mode and the one on the right behavior in the single range mode. In single range mode is the target position equal to the actual position and therefore the axis does not move.

Moving by whole-number multiples of modulo range

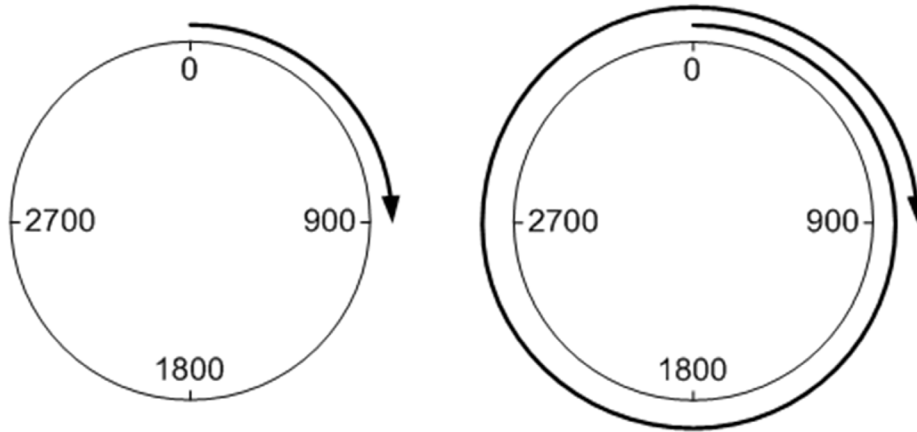


Following figures depict a positioning task

- starting in position 0,
- with target position $i_diPosTarg = 4500$,
- on modulo range $i_stEncPara.diModRng = 3600$,
- with $wModDir = 1$.

The left figure shows positioning within one modulo range and the right one positioning over multiple modulo ranges.

Single and multi range mode



Multiple range mode is supported only in modulo directions 1 (positive) and 2 (negative). Mode 0 (shortest distance) is not supported.

Single and multiple-range mode positioning tasks are supported in both, absolute and relative positioning modes.

Section 3.6

Quick Reference Guide

What Is in This Section?

This section contains the following topics:

Topic	Page
Function Block Visualization	94
Instantiation and Usage Example	95
Commissioning Procedure	97
Troubleshooting	100

Function Block Visualization

Visualization

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

AdvancedPositioning			
Instance:%s			
Status	Alarm		
00	Axis ready	00	Following error
01	In position	01	Not ready / Not in run
02	Positioning failed	02	32bit position overflow
03	In motion	03	Consecutive Overflow
04	Positioning active	04	Speed Reference
05	Execution acknowledgement	05	Acceleration ramp
06	Quick stop	06	Deceleration ramp
07	Halt	07	Quick stop ramp
08	Homing active	08	Maximum position difference
09	Homing finished	09	In position window
10	Homing done	10	Proportional gain
11	Homing failed	11	Feed-forward gain
12	Homing speed high	12	Nominal frequency
13	Homing method not supported	13	Maximum frequency
14	Jog active	14	Nominal speed
15	Jog done	15	Jog speed reference
16	Long move	16	Jog distance
17	Target position invalid	17	Homing speed reference
18	Actual position value invalid	18	Homing distance
19	Jog distance invalid	19	Resolution of encoder interface
20	Homing clearance invalid	20	Pulses per revolution
21	Homing reference invalid	21	Scaling numerator
22	LsFwd position invalid	22	Scaling denominator
23	LsRev position invalid	23	Modulo range
24	Limit switch active		
25	Modulo mode		
26	Modulo direction not supported		
27	Modulo range invalid		

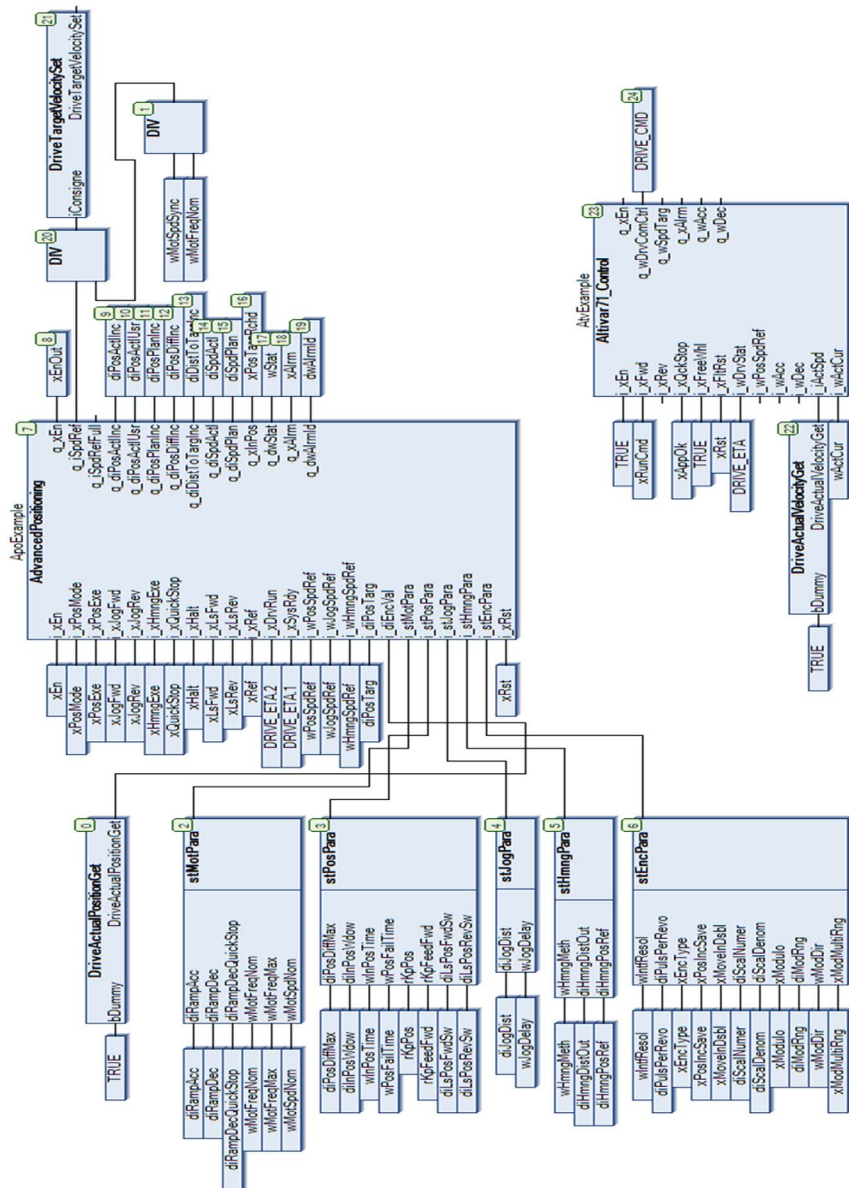
The visualization of the `AdvancedPositioning` FB displays status and alarm words of the FB. Its purpose is to help with the commissioning of the FB.

It is available inside the library and can be selected in the configuration of frame visualization object under `SE_HOIST.Visu_AdvancedPositioning`.

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the `AdvancedPositioning` function block via internal communication of Altivar 71 using the internal encoder interface. The program will differ slightly if external encoder or communication via CANopen is used.



Commissioning Procedure

Commissioning Procedure Necessary for Using the `AdvancedPositioning` Function Block

Commissioning involves the configuration of the equipment that controls your crane. You should be aware of all the hazards involved in such an operation, and to take any and all needed or otherwise required precautions to assure the safety of equipment and personnel in and around the crane.

WARNING

UNINTENDED EQUIPMENT OPERATION

Verify that all equipment is secured to prevent harm to personnel and equipment damage.

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

1. Set the drive to factory settings. In drive menu application functions disable, preset speeds. Set command and reference channel to controller card or CANopen depending on the preferred communication channel.
2. Configure the Altivar variable speed drive (VSD) with correct motor control parameters.
3. Configure the following parameters in the settings menu of Altivar 71:
 - K speed loop filter (SFC) = 100
 - IR compensation (UFR) = 100
 - Low speed (LSP) = 0 (or as close to 0 as possible)
 - Slip compensation (SLP) = 100
 - Motor control mode (CTT) depending on machine configuration (FVC is preferred).

If a different drive is used, set the applicable parameters.
4. Set the acceleration (ACC) and deceleration (DEC) ramp time on the Altivar variable speed drive to a minimum value (0.1 or 0.01 s).
5. Instantiate the FB.
6. Make sure that limit switch signals are connected to the controller and the signals given to the FB.
7. Connect `i_xDrvRun` and `i_xSysRdy` inputs.
8. Configure the parameters inside the encoder configuration structure (`i_stEncPara`).

9. Connect the speed reference output to input communication channel of the drive (typically CANopen or internal bus of Altivar 71 when used with ATV IMC). The value is signed and therefore the drive always gets only forward direction command. This simplifies the command control of the drive, since the command control is performed by user application and not by the AdvancedPositioning FB.

See the FB instantiation example ([see page 95](#)) for visual description of the program.

NOTE: When using the internal communication between Altivar 71 and ATV IMC to transfer the speed reference to Altivar (function **DriveTargetVelocitySet()**), the speed reference needs to be converted to an integer value with 0.1 Hz resolution.

For example 1500 RPM on 4 pole motor will result in 500 (50 Hz / 0.1).

NOTICE

POOR PERFORMANCE OR IRREGULAR PERFORMANCE OF EQUIPMENT

Verify that all speed references are properly converted with a 0.1 Hz resolution.

Failure to follow these instructions can result in equipment damage.

10. Recommended start-up parameters for commissioning/identification. Start with softer parameters and gradually increase the performance. The suitable setting differs from machine to machine.

Following values are supposed to give a rough idea. Only a subset of available parameters is mentioned in this section.

- `i_stMotPara.diRampAcc`, `i_stMotPara.diRampDec`, `i_stMotPara.diRampDecQuickStop` – start with a low value (for example, 200 RPM/s).
- `i_stPosPara.diPosDiffMax` depends on used encoder, use value corresponding to acceptable position deviation (for example, several motor revolutions).
- `i_stPosPara.diInPosWdow` - for starters use a value corresponding to ~1/10 of motor revolution.
- `i_wInPosTime` - ~200 ms
- `i_stPosPara.wPosFailTime` - ~1000 ms
- `i_stPosPara.rKpPos` - start with a low value (5...10) and slowly increase. If the axis starts to oscillate, the value is too high.
- `i_stPosPara.rKpFeedFwd` - The default value is 0. Increase the value slowly to find the optimal setting. Find an optimal setting of `i_stPosPara.rKpPos` first.

11. Finish your application, build it and download it to the controller.

12. Start the controller. The axis must be *In position* after the first start of the controller. If it is in *Positioning failed* state or if there is a following error alarm detected it suggests that the `i_xSysRdy` input was set to TRUE before the actual encoder value was present on the `i_diEncVal` input.

 WARNING
--

UNINTENDED EQUIPMENT OPERATION

Verify that the actual encoder value is present before the <code>i_xSysRdy</code> input is set to TRUE.

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

13. Test the behavior of the FB in applicable modes.

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The axis oscillates during motion.	Proportional gain or feed forward gain of position controller are too high.	Stop the axis using reset or quick stop input. The axis will stop after the time defined in <code>i_stPosPara.wPosFailTime</code> elapses. Decrease value of proportional and/or feed forward gain.

Part III

Anti-Crab

Overview

This part explains the functionality and implementation of `AntiCrab` function block.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	AntiCrab: Correction of Bridge Movement to Solve Wear of Rail and Wheels	103
5	AntiCrab_2: Correction of Bridge Movement to Solve Wear of Rail and Wheels	125

Chapter 4

AntiCrab: Correction of Bridge Movement to Solve Wear of Rail and Wheels

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AntiCrab_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
4.1	Functional and Machine Overview	104
4.2	Architecture	109
4.3	Function Block Description	113
4.4	Pin Description	114
4.5	Troubleshooting	123

Section 4.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	105
Machine Overview	108

Functional Overview

Functional Description

The Anti-crab function is designed to correct the skew and the drift of the bridge in industrial cranes.

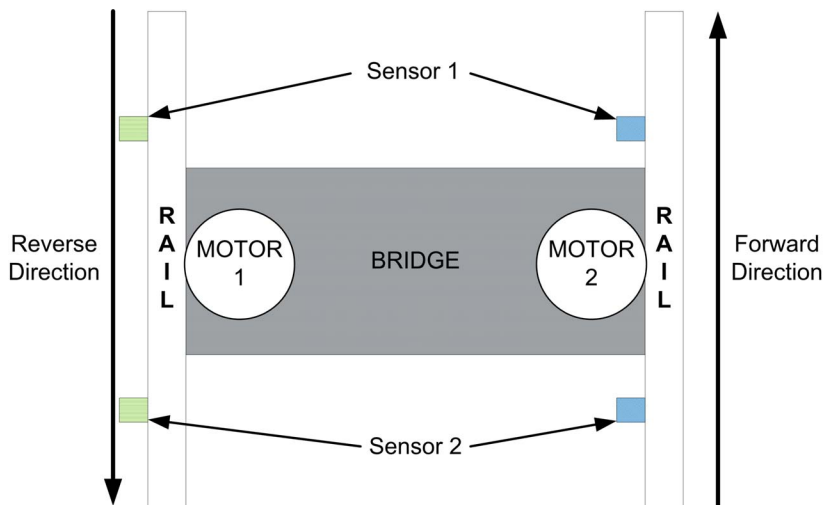
NOTE: The Anti-crab solution is frequently referred to as the Anti-skew solution. The Anti-crab solution is not compatible with the `AntiSwayOpenLoop` function block on the same axis.

The Anti-crab solution uses 2 analog inductive sensors fixed onto either end of the bogie of the crane. These analog sensor outputs are wired to the analog input channels of the SoMachine controller. Alternatively, any other analog input of the crane system can be used.


The 2 sensors measure the distance of the bogie to the rail. If both sensors measure the same distance, the bridge is parallel to the rail. If the sensors measure different values, the bridge is no longer parallel to the rail. This is called skewing. The function calculates different speeds for the 2 motors (one faster, one slower) to bring the bridge back into a proper parallel position.


Besides skewing, the bridge drifts on rails to one side or another with both sensors measuring increasing (or decreasing) values over a period of time. In this case, the function also calculates different motor speed offsets to realign the bridge.

The principle is illustrated in the following figure.



Legend for the graphic:

Symbol	Meaning
	Sensor mounting position 1

Symbol	Meaning
	Sensor mounting position 2

Why Use the AntiCrab Function Block?

The usage of the AntiCrab function block helps to prevent:

- Excessive wear and tear on wheels and stress to wheel flanges.
- Horizontal forces at right angles to the rail that can result in abnormal stress to the crane, runway beams, and building structure.
- Differing diameters of crane wheels caused by wear, which subsequently leads the crane to skew, especially when polyurethane wheels are used on trackless industrial cranes.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

Solution with the AntiCrab Function Block

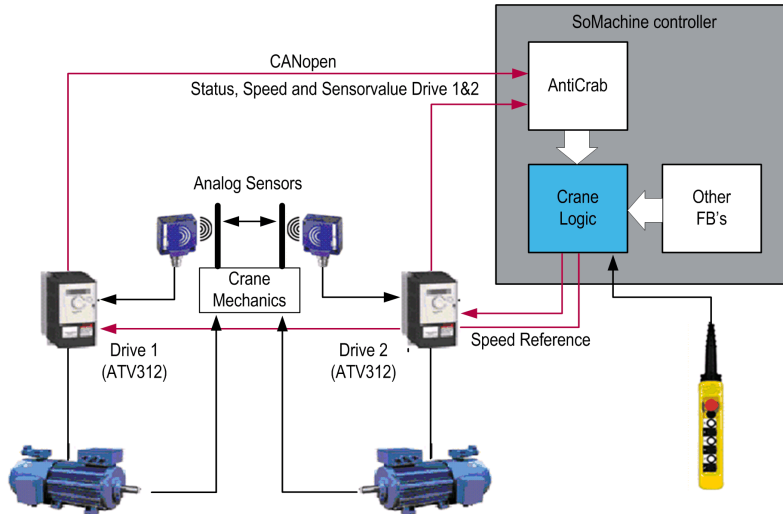
Analog sensors measure the displacement of the bridge from the rail. These signals are used to take corrective measures so that the bridge remains parallel to the rail.

Design and Realization Constraints and Assumptions

The design and realization constraints and assumptions are as follows:

- Manual override of sensor average occurs if any one of the input `i_wSenAvge1` or `i_wSenAvge2` values are greater than zero. If both values are zero, then manual override operation is disabled.
- Drift and skew are corrected simultaneously.

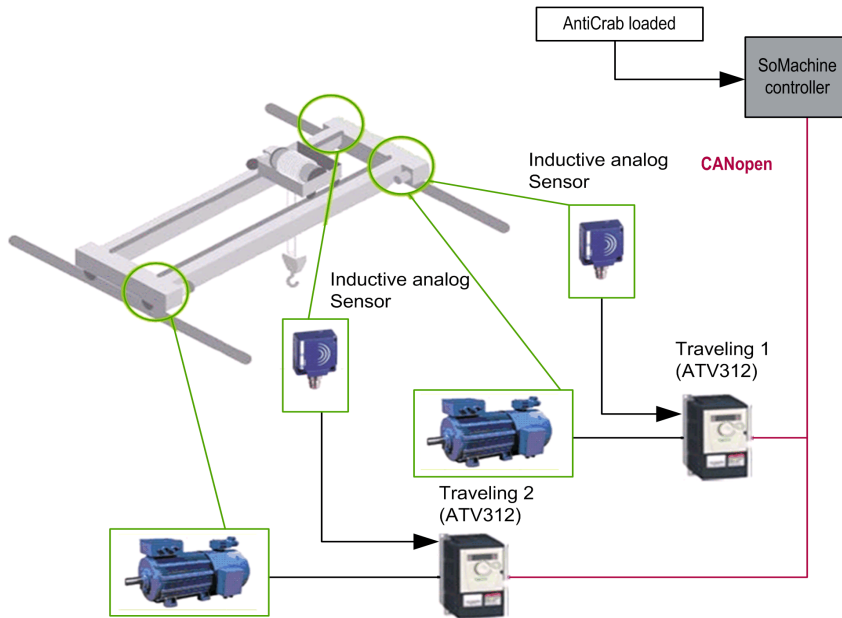
Functional View



Machine Overview

Machine View

The following figure is the Machine View of Anti-crab in an industrial crane using two motors to move the bridge.



NOTE: The function Anti-crab works with ATV312 or ATV71. However, both drives used in the application must be of the same model type. The graphic shows a pair of ATV312 drives, an alternative option would be to use a pair of ATV71 drives.

Section 4.2

Architecture

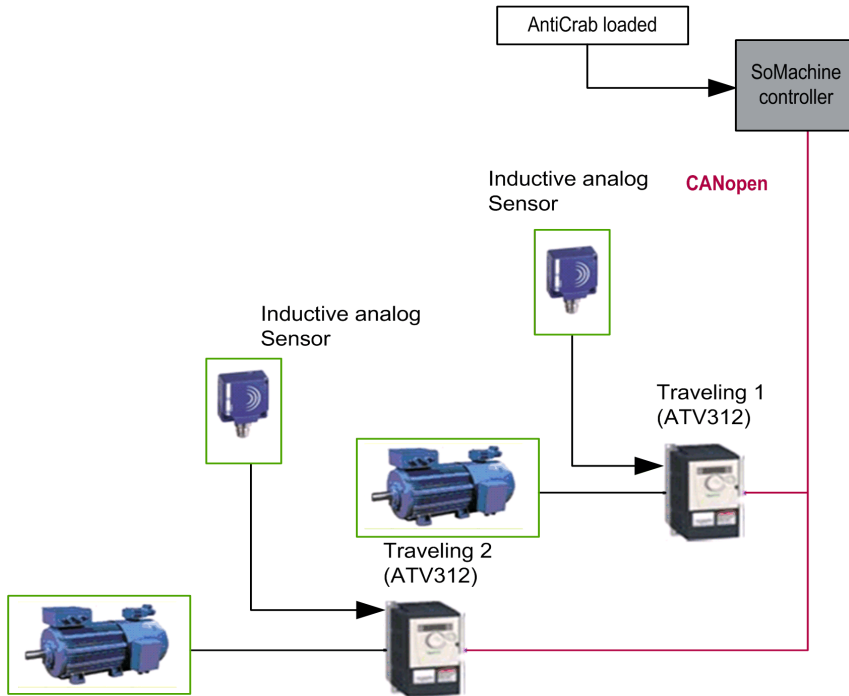
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	110
Software Architecture	111

Hardware Architecture

Hardware Architecture Overview

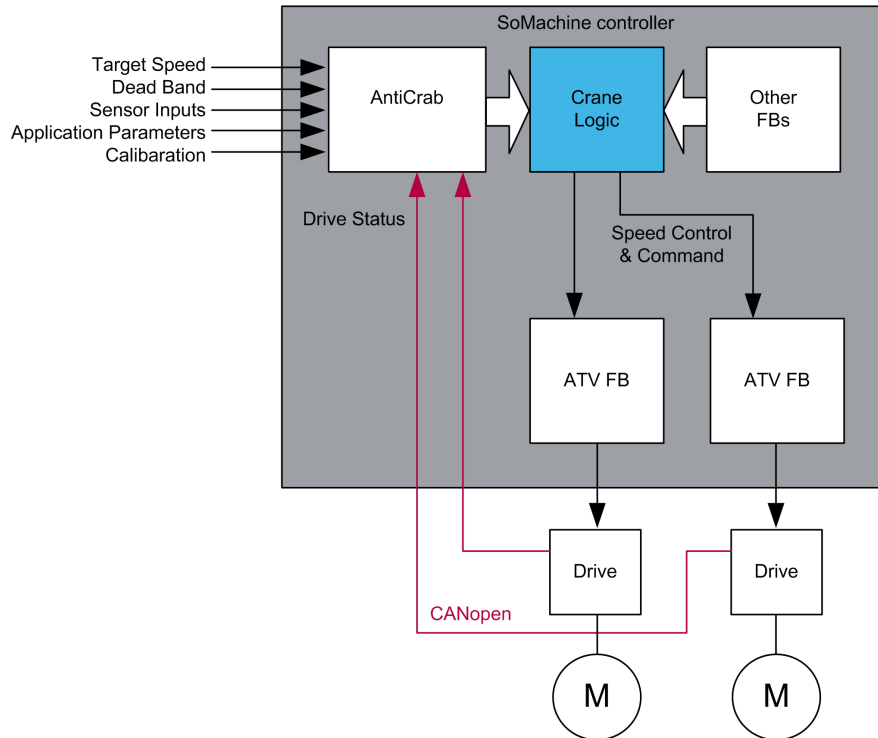


A configuration with 2 drives and 2 motors is the most commonly used configuration. It uses 1 drive and 1 motor for each of the bogies in the system.

NOTE: The function Anti-crab works with ATV312 or ATV71. However, both drives used in the application must be of the same model type. The graphic shows a pair of ATV312 drives, an alternative option would be to use a pair of ATV71 drives.

Software Architecture

Data Flow Overview



Skew and Drift Sub-Function

This sub-function takes the sensor inputs and the configuration of the sensors into account to calculate the current skew and the drift.

Speed Correction Sub-Function

This sub-function calculates the new speeds for the drives to correct the skew and drift. The correction depends on the current speed and user adjustable parameters (Acceleration, Deceleration and Speed Reference value). New acceleration or deceleration value are written to the drive in each cycle to make sure both drives reach their target speed at the same time.

Calibration Sub-Function

If `i_xCalb` is TRUE, the function block is storing the actual sensor values at every cycle.

To find the middle point of the sensor-range, the function is checking,

- if the actual sensor value is bigger than the biggest stored value and
- if the actual sensor value is smaller than the smallest stored value.

The function replace the values if they are bigger or smaller.

Therefore over time, the stored min./max. range is reflecting the real behavior of the machine. And the midpoint calculation is the middle between the absolute minimum and the absolute maximum of the machine.

NOTE: You should either use the manual override of the sensors (`i_wSenAvge1`, `i_wSenAvge2`), or enable calibration by setting `i_xCalb` to TRUE and leave it TRUE.

The calibration is intended to stay TRUE and it will get more accurate the longer it is running.

NOTE: If `i_xCalb` is FALSE and input `i_wSenAvge1` > 0 and `i_wSenAvge2` > 0 then offset is calculated as below:

`Offset1:= 12000 - i_wSenAvge1.`

`Offset2:= 12000 - i_wSenAvge2.`

If `i_xCalb` is TRUE (*calculation of middle points*) then offset is calculated as below:

`rCalbSenAvg1:= (wSen1Min + ((wSen1Max - wSen1Min) / 2))`

`rCalbSenAvg2:= (wSen2Min + ((wSen2Max - wSen2Min) / 2))`

`rOffsetSen1:= (12000.0 - rCalbSenAvg1);`

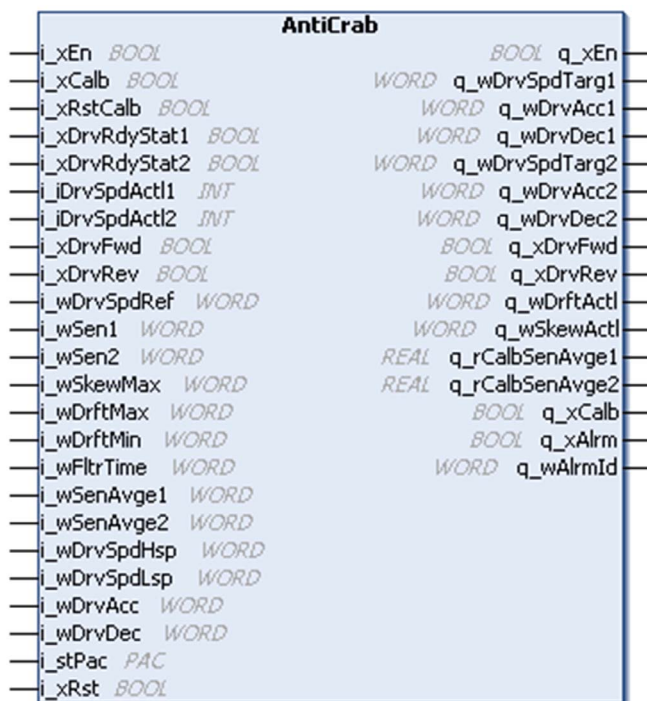
`rOffsetSen2:= (12000.0 - rCalbSenAvg2).`

Section 4.3

Function Block Description

AntiCrab Function Block

Pin Diagram



Function Block Description

The AntiCrab function block does the following:

- Skew Correction
- Drift Correction
- Calibration of the functions in case of improper sensor alignment

Section 4.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	115
Output Pin Description	121

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (see page 117) of i_xEn.
i_xCalb	BOOL	Used to activate the calibration. Calibration is started on rising edge of this input, refer to detailed description (see page 118) of i_xCalb.
i_xRstCalb	BOOL	Resets calibrated values of automatic calibration to default. Max: 4000 Min: 20000 The normal value can be 4000...20000 (4...20 mA), and the input (i_xRstCalb) reset the calibration values invert (min: 20000; max: 4000) to start a new calibration.
i_xDrvRdyStat1	BOOL	TRUE: Drive ready FALSE: Drive not ready This input must be connected to CANopen status of the drive 1, Ready state / status word bit 1.
i_xDrvRdyStat2	BOOL	TRUE: Drive ready FALSE: Drive not ready This input must be connected to CANopen status of the drive 2, Ready state / status word bit 1.
i_iDrvSpdAct11	INT	Actual speed input from drive 1 Scaling/Unit: 1 RPM
i_iDrvSpdAct12	INT	Actual speed input from drive 2 Scaling/Unit: 1 RPM
i_xDrvFwd	BOOL	Command input from the pendant control for forward movement TRUE: Forward FALSE: Not forward
i_xDrvRev	BOOL	Command input from the pendant control for movement TRUE: Reverse FALSE: Not reverse
i_wDrvSpdRef	WORD	Gives the target speed of the bridge. All the skew and drift corrections are based on this target speed. Range: 0...6000

Input	Data Type	Description
i_wSen1	WORD	Sensor 1 input signal, refer to detailed description (see page 119) of i_wSen1, i_wSen2. Range: 4000...20000
i_wSen2	WORD	Sensor 2 input signal, refer to detailed description (see page 119) of i_wSen1, i_wSen2. Range: 4000...20000
i_wSkewMax	WORD	Maximum allowed skew, difference of sensors, refer to detailed description (see page 119) of i_wSkewMax. Range: 0...16000
i_wDrftMax	WORD	Upper drift threshold, refer to detailed description (see page 120) of i_wDrftMax. Range: 0...20000
i_wDrftMin	WORD	Lower drift threshold, refer to detailed description (see page 120) of i_wDrftMin. Range: 0...20000
i_wFltrTime	WORD	Timer to delay the alarm bits1, 2, 3. Once a drift or skew value is detected, a timer starts to filter out momentary misalignment. Input is the duration of this filter. Range: 0..65535
i_wSenAvge1	WORD	Manual override of middle point of sensor 1, refer to detailed description (see page 120) of i_wSenAvge1, i_wSenAvge2. Range: 4000...20000 If 0, then values from calibration is used.
i_wSenAvge2	WORD	Manual override of middle point of sensor 2, refer to detailed description (see page 120) of i_wSenAvge1, i_wSenAvge2. Range: 4000...20000 If 0, then values from calibration is used.
i_wDrvSpdHsp	WORD	Maximum speed of the drives (HSP) converted into RPM. Cannot correct speed above HSP. In this case, the second drive is decelerated using the remaining value of the correction minus the difference between actual speed and HSP of the first drive. This value must be identical to the drive setting. Range: 0...6000

Input	Data Type	Description
i_wDrvSpdLsp	WORD	Minimum speed of the drives (LSP) converted into RPM. Speed cannot be below LSP. In this case, the second drive is accelerated using the remaining value of the correction minus the difference between actual speed and LSP of the first drive. This value must be identical to the drive setting. Range: 0...6000
i_wDrvAcc	WORD	Acceleration time for both drives from zero to nominal speed. It is the base for the calculated correction ramp. Corrected acceleration time is always greater than this value. Range: 1...9999 Target drive must be set to 0.1 accuracy
i_wDrvDec	WORD	Deceleration time for both drives from nominal speed to zero. It is the base for the calculated correction ramp. Corrected deceleration time is always greater than this value. Range: 1...9999 Target drive must be set to 0.1 accuracy
i_stPAC	STRUCT PAC Refer below for details	Represents a structure of 2 values for the parameterization of the Drift and Skew controllers. Range for Skew: 0...3 Range for Drift: 0...20
i_xRst	BOOL	Reset input to clear all alarms. All detected alarms are reset on a rising edge. TRUE: Active FALSE: Inactive

Description of PAC Structure

Parameter	Data Type	Description
rKpDrft	REAL	This is the proportional gain of the drift controller Range: 0...100 % Factory setting: 1.5 %
rKpSkew	REAL	This is the proportional gain of the skew controller Range: 0...100 % Factory setting: 0.25 %

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `AntiCrab` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_wDrvFwd</code>	WORD	<code>i_wDrvFwd</code>
<code>q_wDrvRev</code>	WORD	<code>i_wDrvRev</code>
<code>q_wDrvAcc1</code>	WORD	<code>i_wDrvAcc</code>
<code>q_wDrvAcc2</code>	WORD	<code>i_wDrvAcc</code>
<code>q_wDrvDec1</code>	WORD	<code>i_wDrvDec</code>
<code>q_wDrvDec2</code>	WORD	<code>i_wDrvDec</code>
<code>q_wDrvSpdTarg1</code>	WORD	<code>i_wDrvSpdRef</code>
<code>q_wDrvSpdTarg2</code>	WORD	<code>i_wDrvSpdRef</code>
<code>q_xAlrm</code>	BOOL	FALSE
<code>q_wAlrmId</code>	WORD	0

Outputs that are not listed will retain their current values.

`i_xCalb`

Calibration is needed if the sensors do not give the same value when the bridge is aligned properly. Calibration can also take into account that the rails are uneven over the entire movement. Calibration samples the input values of both sensors and stores the absolute minimums and absolute maximums of all values during active calibration. The respective middle points are always calculated following this formula:

$$\text{Middle point} = \text{Minimum} + ((\text{Maximum} - \text{Minimum}) / 2)$$

For more detailed information about the calibration process, refer to the Calibration Sub-Function (*see page 112*) description.

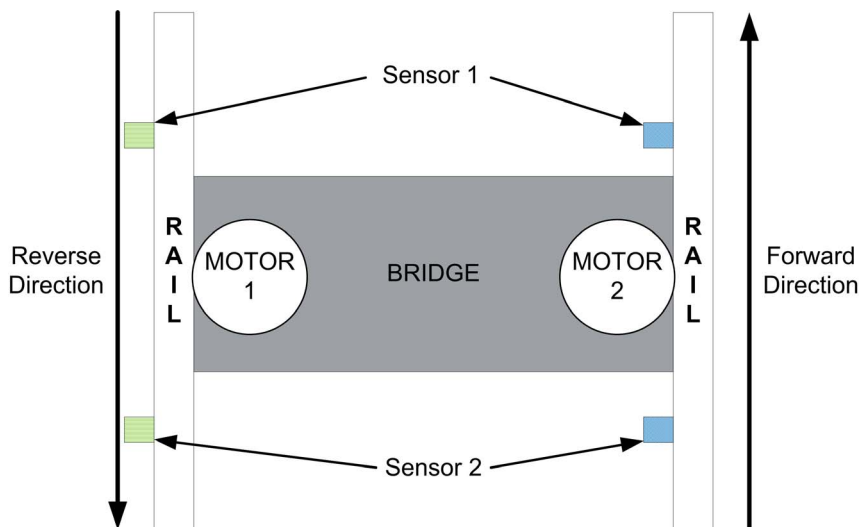
`i_xDrvRdyStat1`, `i_xDrvRdyStat2`

The input need to be logical connected with the CANopen status of the controlled drive. This could be found in the CANopen status word bit 1 / Ready state.



i_wSen1, i_wSen2

These are the sensor inputs for measuring the distance between the bogie and the rail. These values have to be scaled by the Scale Input function block to a 4000 to 20000 range (representing 4...20 mA) before using them with the `AntiCrab` function block. For example, if a 0...10 V sensor is used, the input to the function needs to be scaled to represent a range of 4000... 20000. The sensors have to be mounted in a way that they show values of 12000 for a properly aligned bridge. As seen in the diagram below, depending on the mounting position of the sensor set, they need to be connected to these inputs according to the picture (only 2 sensors are required).

The following figure represents the 2 sensor mounting positions for `AntiCrab` function block.



Legend for the graphic:

Symbol	Meaning
	Sensor mounting position 1
	Sensor mounting position 2

i_wSkewMax

This is the maximum allowed difference in sensor readings (skew). If $((\text{sensor 1} + \text{offset 1}) - (\text{sensor 2} + \text{offset 2})) > \text{SkewMax}$ an alarm is generated, bit 2 of `q_wAlrmId` is set high, the speed output to the drives is 0, and the crane stops.

i_wDrftMax

This is the maximum allowed sensor averaged value of sensor 1 and sensor 2. This is the maximum distance the bridge is allowed to drift away from the rail. If $((\text{sensor 1} + \text{offset 1}) + (\text{sensor 2} + \text{offset 2}))/2 > \text{DrftMax}$, an alarm is generated, bit 0 of `q_wAlrmId` is set high, the speed output to the drives is 0, and the crane stops.

i_wDrftMin

This is the minimum allowed sensor averaged value of sensor 1 and sensor 2. This is the minimum distance the bridge is allowed to drift towards the rail. If $(\text{sensor 1} + \text{offset 1} + \text{sensor 2} + \text{offset 2})/2 < \text{DrftMin}$, an alarm is generated, bit 1 of `q_wAlrmId` is set to high, the speed output to the drives is 0, and the crane stops.

i_wSenAvge1, i_wSenAvge2

These values are used to manually override the calculated middle points of both sensors during calibration. The default of these inputs is 0. If any of the values is non-zero, manual override becomes active and the input value is used as the middle point reference for all internal calculations. Once reset to 0, the calibration value is used again.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_wDrvSpdTarg1	WORD	Drive 1 target speed Range: 0...6000
q_wDrvAcc1	WORD	Drive 1 acceleration Range: 1...9999
q_wDrvDec1	WORD	Drive 1 deceleration Range: 1...9999
q_wDrvSpdTarg2	WORD	Drive 2 target speed Range: 0...6000
q_wDrvAcc2	WORD	Drive 2 acceleration Range: 1...9999
q_wDrvDec2	WORD	Drive 2 deceleration Range: 1...9999
q_xDrvFwd	BOOL	Forward command to the drives
q_xDrvRev	BOOL	Reverse command to the drives
q_wDrftAct1	WORD	Output indicating the actual amount of drift which is the result of: $12000 - ((\text{sensor1} + \text{offset1}) + (\text{sensor2} + \text{offset2})) / 2$ if drift correction (q_wDrftAct1) is active. Number of units from the middle point
q_wSkewAct1	WORD	Output of the actual amount of skew. It is the actual result of: $((\text{sensor 1} + \text{offset 1}) - (\text{sensor 2} + \text{offset 2}))$ if skew correction (q_wSkewAct1) is active. Range: 0...16000 Number of units from the middle point
q_rCalbSenAvge1	REAL	Sensor 1 calibrated value, refer to detailed description (see page 122) of q_rCalbSenAvge1, q_rCalbSenAvge2. Range: 4000...20000
q_rCalbSenAvge2	REAL	Sensor 2 calibrated value Range: 4000...20000
q_xCalb	BOOL	Status of calibration TRUE: Calibration done FALSE: Calibration not done

Output	Data Type	Description
q_xAlrm	BOOL	Displays the detected alarm status of the function block. TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification. Range: 0...255

q_rCalbSenAvge1, q_rCalbSenAvge2

This is the output of the actual active middle points of sensor 1 and sensor 2 either calculated by the calibration process or manually overridden by the `i_wSenAvge1`, `i_wSenAvge2` inputs. From these values, the offsets are calculated:

Sensor average1 = 12000 ± offset1. Sensor average2 = 12000 ± offset2

NOTE: On detection of Alarm in Calibration state, the `q_rCalbSenAvge1` and `q_rCalbSenAvge2` output are set to 0.

Notifications

Bit Number	Description
0	Maximum drift value, drift is above upper range.
1	Minimum drift value, drift is below of lower range.
2	Maximum skew value, skew is out of range.
3	<code>i_wDrvSpdLsp</code> is bigger or equal to <code>i_wDrvSpdHsp</code> .
4	Manual override is incomplete, only one of the inputs (<code>i_wSenAvge1</code> or <code>i_wSenAvge2</code>) has a value higher than zero.
5	No middle points present. That means no calibration is done or zero (0) value is active for <code>i_wSenAvge1</code> or <code>i_wSenAvge2</code> .
6	<code>i_wDrftMax</code> is smaller or equal than <code>i_wDrftMin</code> .
7	Drive 1 not ready or Drive 2 not ready.

Section 4.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The drives start, but then, they suddenly stops	An alarm is detected	Check the value of <code>q_wAlrmId</code> and configure the dead bands and other ranges accordingly
Drift or Skew correction is active, but the bridge does not realign itself	Physical sensor configuration is wrong	Check if the configuration/positioning of sensors is done according to the graphic representing the 2 sensor mounting positions for AntiCrab function block. Refer graphic below <code>i_wSen1</code> , <code>i_wSen2</code> for 2 different possibilities to place the sensor couple (<i>see page 119</i>).
Drift or Skew correction is active, but the bridge does not realign itself	Gain is not properly set	Check the correction limit and the factors <code>rKpDrft</code> and <code>rKpSkew</code> . Standard setting for the drift is a value of 1.5; skew is by default 0.25. It is recommended that changes of this value will be done in delta increments of 0.05 or less.
One of the drives is in event state or lost connection to the controller and the other one continues to run.	Function block does not have information about event of the drive connected or bus supervision is not configured.	Connect the information about status of the CANopen node to the corresponding <code>i_xDrvRdyStat</code> input together with information about the ready status of the drive. Input in logical AND with information about Ready status of the drive. Make sure that the bus supervision is configured.
The actual motor speed does not correspond to speed reference.	Preset speeds configuration interferes with speed reference from controller.	Disable preset speeds in the application functions setting of the drive.

Issue	Cause	Solution
The actual motor speed does not correspond to speed reference.	Limit switch configuration interferes with speed reference from controller. Preset speeds in application function is enabled, or wrong motor data.	Disable limit switch function in application functions setting of the drive.
The actual motor speed or read actual speed do not correspond to speed reference.	PDO mapping is inconsistent.	Make sure that both speed reference and actual speed values have the same unit (Hz or RPM).
It is difficult to find correct combination of skew and drift controller gains.	Gains must be set in correct order because the controllers are cascaded. Drift controller gives setpoint to skew controller.	Set the skew controller gain while gain of drift controller is zero. When the correct setting has been found, proceed to set the drift controller.

Chapter 5

AntiCrab_2: Correction of Bridge Movement to Solve Wear of Rail and Wheels

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
5.1	Functional and Machine Overview	126
5.2	Architecture	129
5.3	Function Block Description	132
5.4	Pin Description	137
5.5	Quick Reference Guide	148

Section 5.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	127
Machine Overview	128

Functional Overview

Why Use the AntiCrab_2 Function Block?

The usage of the AntiCrab_2 function block helps to prevent:

- Excessive wear and tear on wheels and stress to wheel flanges.
- Horizontal forces at right angles to the rail that can result in abnormal stress to the crane, runway beams, and building structure.

It is achieved by reducing the skew and drift of the bridge in industrial cranes.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

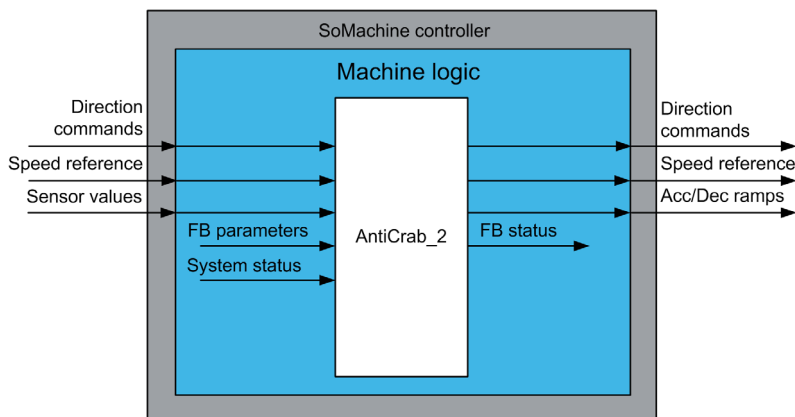
Validate all function block input values before and while the function block is enabled.

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

Solution with the AntiCrab_2 Function Block

Analog sensors measure the displacement of the bridge from the rail. These signals are used to take corrective measures so that the bridge remains parallel to the rail.

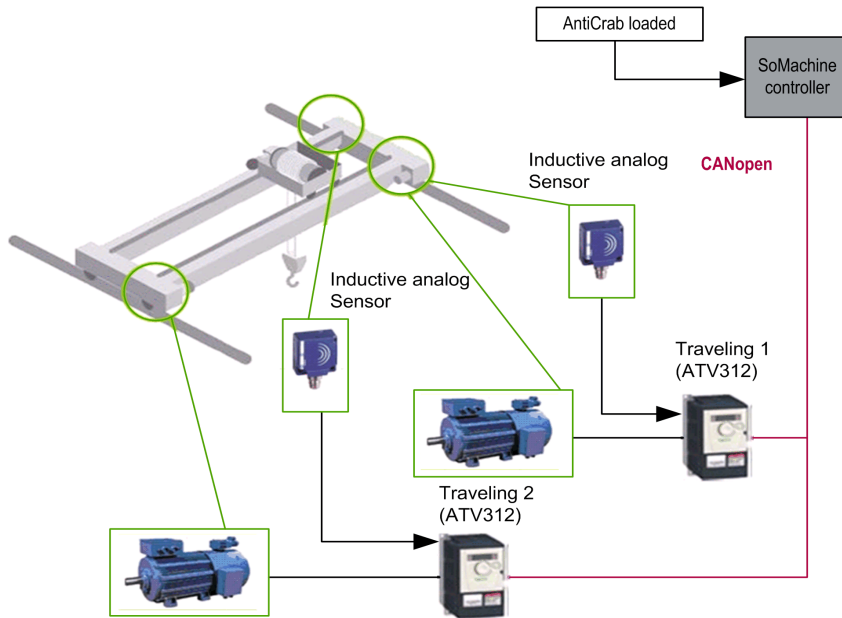
Functional View



Machine Overview

Machine View

The following figure is the Machine View of Anti-crab in an industrial crane using two motors to move the bridge.



NOTE: The function Anti-crab works with ATV312, ATV32 or ATV71. However, both drives used in the application must be of the same model type. The graphic shows a pair of ATV312 drives, an alternative option would be to use a pair of ATV71 drives.

Section 5.2

Architecture

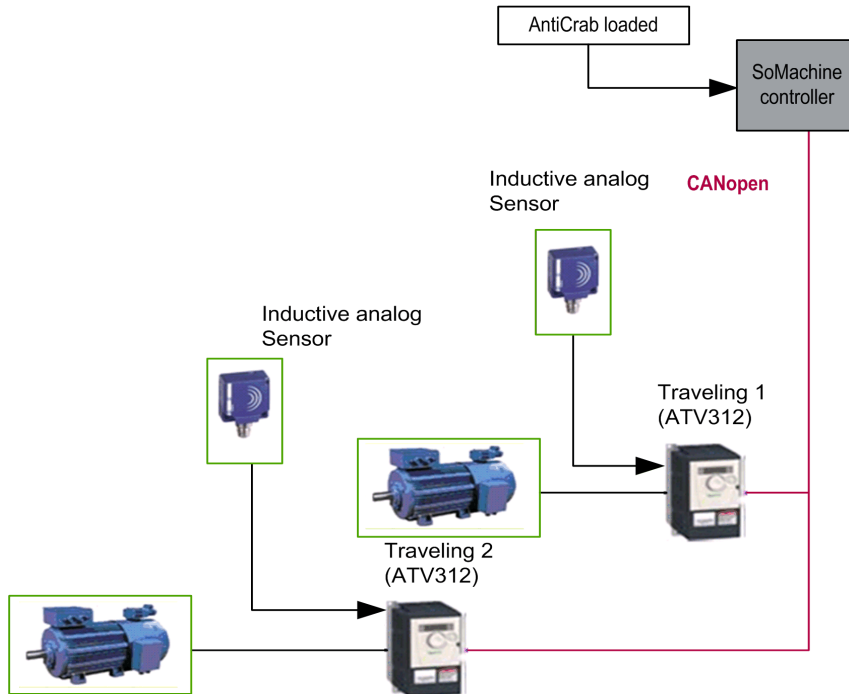
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	130
Software Architecture	131

Hardware Architecture

Hardware Architecture Overview



A configuration with 2 drives and 2 motors is the most commonly used configuration. It uses 1 drive and 1 motor for each of the bogies in the system.

NOTE: The function Anti-crab works with ATV312, ATV32 or ATV71. However, both drives used in the application must be of the same model type. The graphic shows a pair of ATV312 drives, an alternative option would be to use a pair of ATV71 drives.

⚠ WARNING

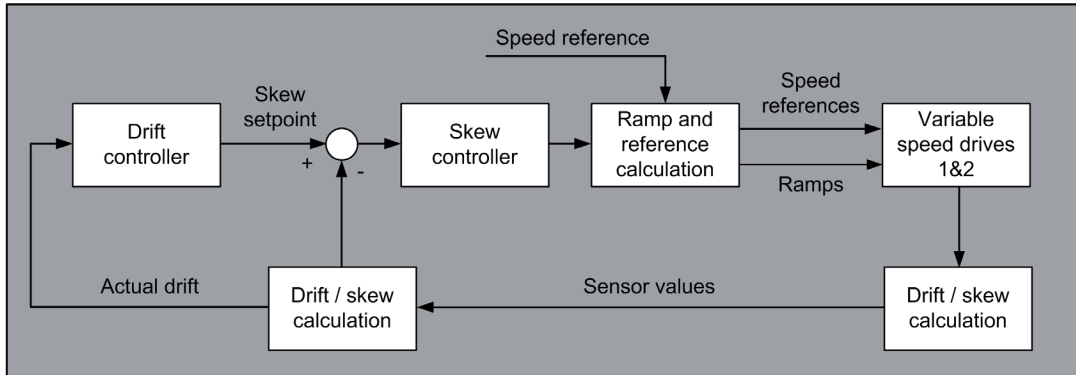
UNINTENDED EQUIPMENT OPERATION

- Equip the bridge in industrial cranes with mechanical safeguards protecting the crane against de-railing and other damage.
- Do not use the FB without the proper mechanical safeguard integrated into the crane application.

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

Software Architecture

Data Flow Overview

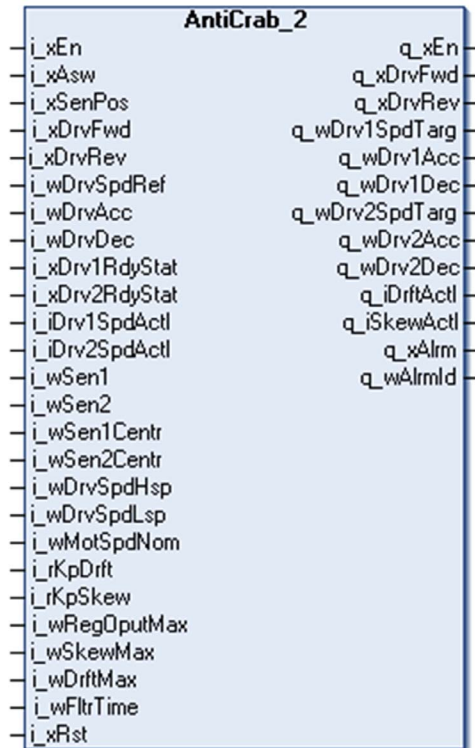


Section 5.3

Function Block Description

AntiCrab_2 Function Block

Pin Diagram



Functional Description

The Anti-crab function is designed to correct the skew and the drift of the bridge in industrial cranes.

Anti-crab function uses 2 analog sensors to measure distance from a crane runway rail or another surface parallel to the rail. Both sensors are placed on one side of one of the bogies. They can be placed in any of the positions corresponding to a) through d) in the following figure. The FB has to be aware of placement of the sensors to operate correctly. The information is given to the FB via `i_xSenPos` input.

`i_xSenPos` must be

- FALSE if sensors are mounted in positions a) or c) on the left side of the rail in forward direction, and
- TRUE if they are in positions b) or d) on the right side of the rail in forward direction.

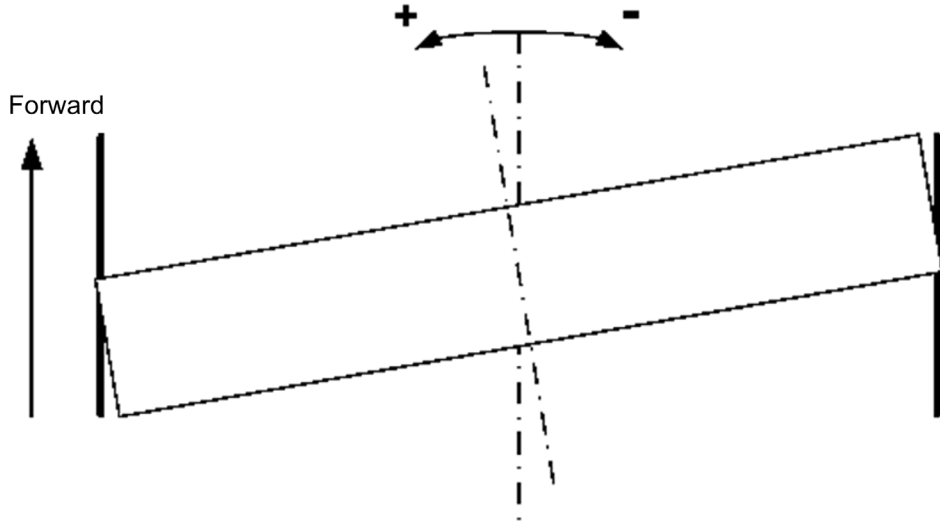
The principle is illustrated in the following figure.



Skew and Drift

Skew is a misalignment of the transversal axis of the crane bridge and the longitudinal axis of the crane runway. The crane has no skew when these axes are parallel.

Bridge in industrial cranes in a positive skew situation:



The actual skew is calculated from the reading of the 2 sensors and the center positions of the sensors. Center positions are input values that help to compensate for an imperfect mechanical alignment of the sensors. The values are configured during commissioning.

The formula for calculation of actual skew:

$$ActualSkew = i_wSen1 - i_wSen1Cen tr + 12000 - (i_wSen2 - i_wSen2Cen tr + 12000)$$

when $i_xSenPos = FALSE$ and

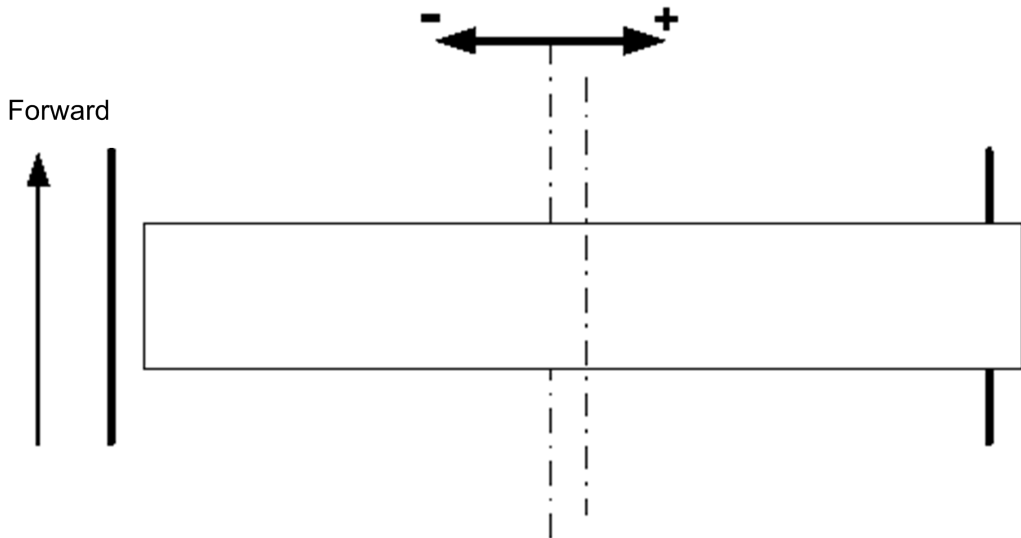
$$ActualSkew = i_wSen2 - i_wSen2Cen tr + 12000 - (i_wSen1 - i_wSen1Cen tr + 12000)$$

when $i_xSenPos = TRUE$

The actual value of skew is available at the $q_iSkewAct1$ output of the FB.

Drift is a lateral misalignment of the transversal axis of the bridge in industrial cranes and the longitudinal axis of the crane runway. There is no drift if the 2 axes intersect in the middle of the bridge in industrial cranes.

Bridge in industrial cranes in a positive drift situation:



The actual drift is calculated from the reading of the 2 sensors and the center positions of the sensors. Center positions are input values that help to compensate for an imperfect mechanical alignment of the sensors. The values are configured during commissioning.

The formula for calculation of actual drift:

$$ActualDrift = 12000 - \frac{i_wSen1 - i_wSen1Cen\ tr + 12000 + i_wSen2 - i_wSen2Cen\ tr + 12000}{2}$$

when $i_xSenPos = FALSE$ and

$$ActualDrift = 12000 - \frac{i_wSen1 - i_wSen1Cen\ tr + 12000 + i_wSen2 - i_wSen2Cen\ tr + 12000}{2}$$

when $i_xSenPos = TRUE$

The actual value of drift is available at the $q_iDrftAct1$ output of the FB.

Signs of the skew and drift do not depend on whether the sensors are mounted on the right or left side of the rail. The FB must get correct information about sensors placement via the $i_xSenPos$ input.

Skew and drift of the bridge in industrial cranes are present concurrently. When a bridge is skewed, it tends to drift in the direction of the skew while moving. This dependency is used by the correction algorithm.

Drift and Skew Correction

The function block contains a proportional controller cascade. The outer loop controls drift of the bridge by setting a setpoint for the skew controller in the inner loop. Using this algorithm, it is possible to compensate for the drift and skew of the bridge at the same time. Refer to the schematic graphical interpretation of the control loop in the Data Flow Overview (*see page 131*).

Combination of Anti-Sway and Anti-Crab Function

The AntiCrab_2 FB supports concurrent usage of Anti-sway and Anti-crab functions on the same axis. When the AntiCrab_2 FB is being used without Anti-Sway, ramping of the speed reference is handled by variable speed drives. In combination with Anti-sway are the ramps handled by the FBs and the variable speed drives get a target speed profile from the application.

The AntiCrab_2 FB can be switched between operation with and without Anti-sway using the input `i_xAsw`.

The AntiSwayOpenLoop_2 FB does not require any special configuration for usage in combination with AntiCrab_2.

An example of implementation is available in the Quick Commissioning (*see page 153*) section.

When using the AntiCrab_2 FB without AntiSwayOpenLoop_2 FB in an application where the AntiSwayOpenLoop_2 FB is present but disabled, the direction commands for the AntiCrab_2 FB must come directly from operators' commands and not from the direction command outputs of the AntiSwayOpenLoop_2 FB.

AntiSwayOpenLoop_2 FB keeps generating a linear speed profile even when the Anti-sway function is disabled and the active direction command is TRUE for the whole duration of that linear profile generation.

NOTE: Using the direction command of the AntiSwayOpenLoop_2 function block when the function block is disabled will result in a considerable delay of stopping performance of the bridge in the industrial crane.

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use direction command provided by the AntiSwayOpenLoop_2 function block when the AntiSwayOpenLoop_2 is disabled.

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

When using the combination of AntiCrab_2 and AntiSwayOpenLoop_2 FBs, set the ramp of the used variable speed drives steep enough not to interfere with the speed profile generated in the application. The FB sets its ramp outputs automatically to a shortest possible value when the `i_xAsw` input is TRUE.

Section 5.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	138
Output Pin Description	145

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (see page 140) of i_xEn.
i_xAsw	BOOL	Switching between standalone and combination with Anti-sway. TRUE: With Anti-sway FALSE: Standalone Refer to detailed description (see page 140) of i_xAsw.
i_xSenPos	BOOL	Position of sensors TRUE: Sensors on the right side. FALSE: Sensors on the left side. Refer to detailed description (see page 141) of i_xSenPos.
i_xDrvFwd	BOOL	Command input for forward movement of the bridge in industrial cranes. TRUE: Forward FALSE: Not forward
i_xDrvRev	BOOL	Command input for reverse movement of the bridge in industrial cranes. TRUE: Reverse FALSE: Not reverse
i_wDrvSpdRef	WORD	Speed reference for movement of the bridge. The value is used as a base for calculated output speed references to both drives. Scaling/Unit: RPM
i_wDrvAcc	WORD	Acceleration time Scaling/Unit: 0.1 s Refer to detailed description (see page 142) of i_wDrvAcc.
i_wDrvDec	WORD	Deceleration time Scaling/Unit: 0.1 s Refer to detailed description (see page 142) of i_wDrvDec.
i_xDrv1RdyStat	BOOL	Drive 1 ready bit from the status word Refer to detailed description (see page 142) of i_xDrv1RdyStat.
i_xDrv2RdyStat	BOOL	Drive 2 ready bit from the status word Refer to detailed description (see page 142) of i_xDrv2RdyStat.
i_iDrv1SpdAct1	INT	Actual speed of drive 1. Scaling/Unit: RPM
i_iDrv2SpdAct1	INT	Actual speed of drive 2. Scaling/Unit: RPM

Input	Data Type	Description
i_wSen1	WORD	Scaled value of sensor 1. Range: 4000...20000 Refer to detailed description (see page 143) of i_wSen1.
i_wSen2	WORD	Scaled value of sensor 2. Range: 4000...20000 Refer to detailed description (see page 143) of i_wSen2.
i_wSen1Centr	WORD	Value of sensor 1 in centered position Range: 4000...20000 Refer to detailed description (see page 143) of i_wSen1Centr.
i_wSen2Centr	WORD	Value of sensor 2 in centered position Range: 4000...20000 Refer to detailed description (see page 143) of i_wSen2Centr.
i_wDrvSpdHsp	WORD	Maximum speed Range: 0...6000 Scaling/Unit: RPM Refer to detailed description (see page 143) of i_wDrvSpdHsp.
i_wDrvSpdLsp	WORD	Minimum speed Range: 0...i_wDrvSpdHsp-1 Scaling/Unit: RPM Refer to detailed description (see page 143) of i_wDrvSpdLsp.
i_wMotSpdNom	WORD	Nominal motor speed Range: 0...65535 Default value: 0 Scaling/Unit: RPM Refer to detailed description (see page 143) of i_wMotSpdNom.
i_rKpDrft	REAL	Proportional gain of the drift controller Range: 0.0...20.0 Default value: 0.6 Refer to detailed description (see page 144) of i_rKpDrft.
i_rKpSkew	REAL	Proportional gain of the skew controller Range: 0.0...3.0 Default value: 0.015 Refer to detailed description (see page 144) of i_rKpSkew.
i_wRegOputMax	WORD	Maximum allowed output of controller cascade - maximum difference of speed references. Range: 0...65535 Default value: 0 Scaling/Unit: RPM Refer to detailed description (see page 144) of i_wRegOputMax.
i_wSkewMax	WORD	Maximum allowed skew Range: 0...65535 Default value: 16000 Refer to detailed description (see page 144) of i_wSkewMax.

Input	Data Type	Description
i_wDrftMax	WORD	Maximum allowed drift Range: 0..65535 Default value: 16000 Refer to detailed description (see page 144) of i_wDrftMax.
i_wFltrTime	WORD	Time for filtering out maximum drift and skew alarms caused by peaks in sensor readings. Range: 0...200 Default value: 10 Scaling/Unit: 0.1 s
i_xRst	BOOL	Resets detected alarms on rising edge, provided the cause of the alarm has been removed. TRUE: Active FALSE: Inactive

i_xEn

When TRUE, this input enables function of AntiCrab_2 FB. When FALSE, the FB enters a fallback state. In fallback state, the FB channels the target speeds, ramps and direction commands to outputs for both drives.

i_xAsw

When TRUE, the FB is in a mode for concurrent operation of Anti-crab and Anti-sway. When FALSE, the FB is in a standalone operation mode.

In standalone mode is the ramping of target speed performed by variable speed drives, the AntiCrab_2 FB writes target speeds and ramp values to both drives via communication in every cycle.

In combined operation of Anti-Crab and Anti-sway is the ramping of speed reference performed by the `AntiCrab_2` and `AntiSwayOpenLoop_2` FBs. The ramp parameters of both bridge travel drives must be set to a low value. The FB sets its ramp outputs automatically to a shortest possible value when the `i_xAsw` input is TRUE. The drives then follow directly the speed reference calculated by the application.

The FB does not prevent switching modes while the drives are running. Be sure that the value of `i_xAsw` is not modified while the drives are running.

NOTE: Modifying the `i_xAsw` parameter of the `AntiSwayOpenLoop_2` function block while the drive(s) are running will result in a considerable delay of stopping performance of the bridge in the industrial crane.

⚠ WARNING

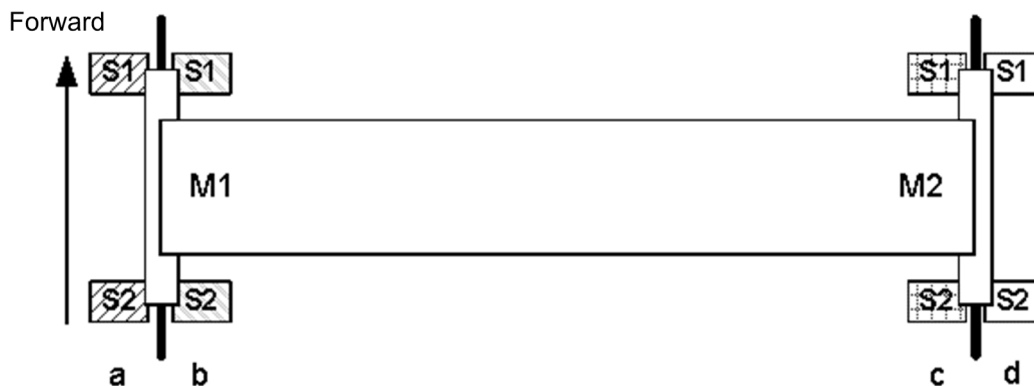
UNINTENDED EQUIPMENT OPERATION

Do not modify the `i_xAsw` parameter while the drive is running.

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

`i_xSenPos`

Selects between 2 possible positions of sensors. When FALSE, the FB is configured for sensors on the left side of the rail in direction of movement, positions a) and c) in the following figure. And when TRUE, the FB is configured for sensors on the right side of the rail, positions b) and d) in the following figure:



i_wDrvAcc

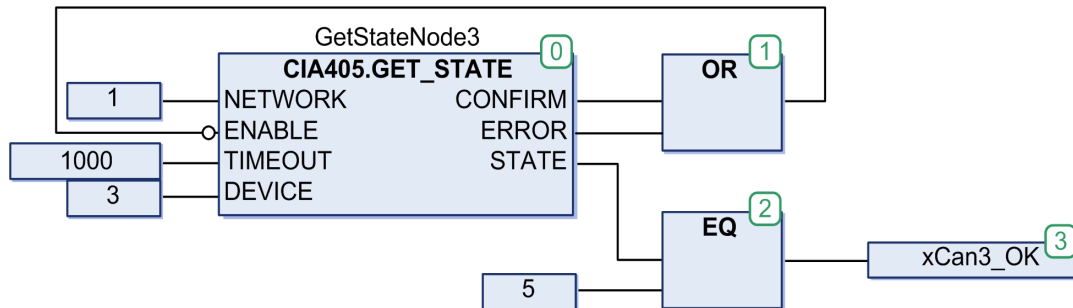
Requested acceleration ramp time of bridge travel motors. The value is used as a base for calculated output acceleration ramp times to both drives. When the FB is used in combination with Anti-sway, this ramp time is used for separate ramping of Anti-crab correction.

i_wDrvDec

Requested deceleration ramp time of bridge travel motors. The value is used as a base for calculated output deceleration ramp times to both drives. When the FB is used in combination with Anti-sway, this ramp time is used for separate ramping of Anti-crab correction.

I_xDrv1RdyStat, I_xDrv2RdyStat

Inputs of ready status of both drives. The information can be obtained from bit 1 of status word of Altivar drives. This information has to be interlocked (logical AND) with an information that the drive is communicating on CANopen and is in operational state.



If the communication with any of the 2 drives is interrupted, the FB has to get the information in order to stop the movement.

WARNING
<p>UNINTENDED EQUIPMENT OPERATION</p> <p>Ensure that, in your application, the status of CANopen communication of both drives is interlocked (logical AND) with both drives status (drives are ready and CANopen is in an operational state).</p> <p>Failure to follow these instructions can result in death, serious injury, or equipment damage.</p>

The FB needs to have information about status of both drives. If one of the drives is not ready, the FB does not authorize start of the movement. If one of the ready bits turns FALSE during movement, both drives are stopped.

i_wSen1, i_wSen2

Scaled values of sensor outputs. The FB expects values scaled between 4000 and 20000. Corresponding to an usual 4...20 mA output of distance sensor. 4 mA being the position of sensor being closest to the rail, and 20 mA the position farthest from the rail.

The `ScaleInput` FB can be used to scale sensor values. See the Function Block Instantiation Example (*see page 151*).

i_wSen1Centr, i_wSen2Centr

These parameters correspond to values of sensors 1 and 2 when the crane is centered on the runway and both wheels of the bogie where the sensors are mounted are in the middle of the rail. In an ideal state both of the sensors should output a value that results in a value of 12000 after scaling.

However, it is not always possible eliminate deviation in the alignment of the sensors.

Using these inputs, it is possible to tell the FB that the centered position of either or both sensors deviate from the ideal.

The internal skew and drift controllers use these values as a regulation setpoint.

If the value of `i_wSen1Centr / i_wSen2Centr` is zero, the FB takes 12000 as the centered position of the related sensor, assuming there is no correction necessary.

i_wDrvSpdHsp

The parameter contains the maximum allowed speed the FB is allowed to write to target speed outputs. The value must not exceed the value defined as the high speed in the variable speed drives.

i_wDrvSpdLsp

The parameter contains the minimum allowed speed the FB is allowed to write to target speed outputs. The value must not be lower than the value defined as the low speed in the variable speed drives. If the mechanics and motor configuration allow it, use `i_wDrvSpdLsp = 0`

i_wMotSpdNom

The parameter contains the nominal speed of the applied motors. This parameter is used for ramping of skew and drift correction when the FB is used together with Anti-sway. In standalone operation it is not necessary to configure this parameter.

The synchronous value of the motor speed is optimal for working with this FB. For example, for 4-pole motor @ 50 Hz, use 1500 RPM. Using the speed from the motor nameplate which is including the motor slip is possible, but not necessarily optimal.

i_rKpDrft

The parameter contains the proportional gain of the drift controller. Optimal value depends on the crane mechanics. Set this value after the `i_rKpSkew` has been configured. Setting this input to 0 disables the drift correction, but leaves the skew correction active.

For more information on configuration of controller gains, check the Commissioning Procedure (*see page 153*) section.

i_rKpSkew

The parameter contains the proportional gain of the skew controller. Optimal value depends on the crane mechanics. Set this value first before `i_rKpDrft`. Setting this input to 0 disables both skew and drift correction.

For more information on configuration of controller gains, check the Commissioning Procedure (*see page 153*) section.

i_wRegOputMax

The parameter defines the maximum output of the drift and skew controller. It corresponds to a maximum allowed difference in target speeds for drives 1 and 2. When left to 0, the output of the controller is limited only by the difference of `i_wDrvSpdHsp` and `i_wDrvSpdLsp`. The input can help reducing a stress to the runway and building structure in case of problems with the mechanics of the crane or sensor signals.

i_wSkewMax

The parameter contains the maximum allowed value of the bridge skew. If the actual value of skew exceeds this value for longer than defined by the `i_wFltrTime`, an alarm is raised. Setting this value to 0 disables the excessive skew detection.

i_wDrftMax

The parameter contains the maximum allowed value of the bridge drift. If the actual value of drift exceeds this value for longer than defined by the `i_wFltrTime`, an alarm is raised. Setting this value to 0 disables the excessive drift detection.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (see page 146) of q_xEn.
q_xDrvFwd	BOOL	Forward command to the drives
q_xDrvRev	BOOL	Reverse command to the drives
q_wDrv1SpdTarg	WORD	Drive 1 target speed Range: 0...6000 Scaling/Unit: RPM
q_wDrv1Acc	WORD	Drive 1 acceleration time Range: 0...999 Scaling/Unit: 0.1 s Refer to detailed description (see page 146) of q_wDrv1Acc.
q_wDrv1Dec	WORD	Drive 1 deceleration time Range: 0...999 Scaling/Unit: 0.1 s Refer to detailed description (see page 146) of q_wDrv1Dec.
q_wDrv2SpdTarg	WORD	Drive 2 target speed Range: 0...6000 Scaling/Unit: RPM
q_wDrv2Acc	WORD	Drive 2 acceleration time Range: 0...999 Scaling/Unit: 0.1 s Refer to detailed description (see page 146) of q_wDrv2Acc.
q_wDrv2Dec	WORD	Drive 2 deceleration time Range: 0...999 Scaling/Unit: 0.1 s Refer to detailed description (see page 146) of q_wDrv2Dec.
q_iDrftAct1	INT	Value of the actual drift. The value is signed and gives information about position of the bridge with respect to its centered position. Range: -12000...+12000
q_iSkewAct1	INT	Value of the actual skew. The value is signed and gives information about relative angle of the bridge with respect to its straight heading. Range: -16000...+16000

Output	Data Type	Description
q_xAlrm	BOOL	Displays the detected alarm status of the function block. In case of an alarm the FB removes run commands from both drives and does not allow any further movement until the cause of the alarm has been removed. TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification. Refer to notifications (<i>see page 147</i>).

q_xEn

Enable status of the FB. While the FB is enabled, it modifies target speeds and ramp times of both drives based on actual sensor readings. In disabled state it channels the input commands, speed reference and ramp times to its outputs. Alarm outputs are set to zero (FALSE).

Transition between disabled and enabled state of the FB and vice versa while the crane is running is not restricted but it is not recommended.

q_wDrv1Acc, q_wDrv2Acc

Acceleration times for drives 1 and 2. The acceleration times are calculated along with the target speeds in order to optimize the skew and drift control. These values must be written to acceleration time parameters of the drives.

When the FB is used together with Anti-sway, both outputs are written to 1 and the ramping is handled by the FBs.

q_wDrv1Dec, q_wDrv2Dec

Deceleration times for drives 1 and 2. The deceleration times are calculated along with the target speeds in order to optimize the skew and drift control. These values must be written to deceleration time parameters of the drives.

When the FB is used together with Anti-sway, both outputs are written to 1 and the ramping is handled by the FBs.

Notifications

Bit Number	Description
0	Actual drift exceeds the maximum drift threshold.
1	Actual skew exceeds the maximum skew threshold.
2	Input configuration, $i_wDrvSpdLsp \geq i_wDrvSpdHsp$.
3	One or both of the drives are not ready and a command to move the crane was given or a drive ready signal lost during movement.
4	$i_wMotSpdNom = 0$ while the FB is configured to work in cooperation with Anti-sway ($i_xAsw = TRUE$).

Section 5.5

Quick Reference Guide

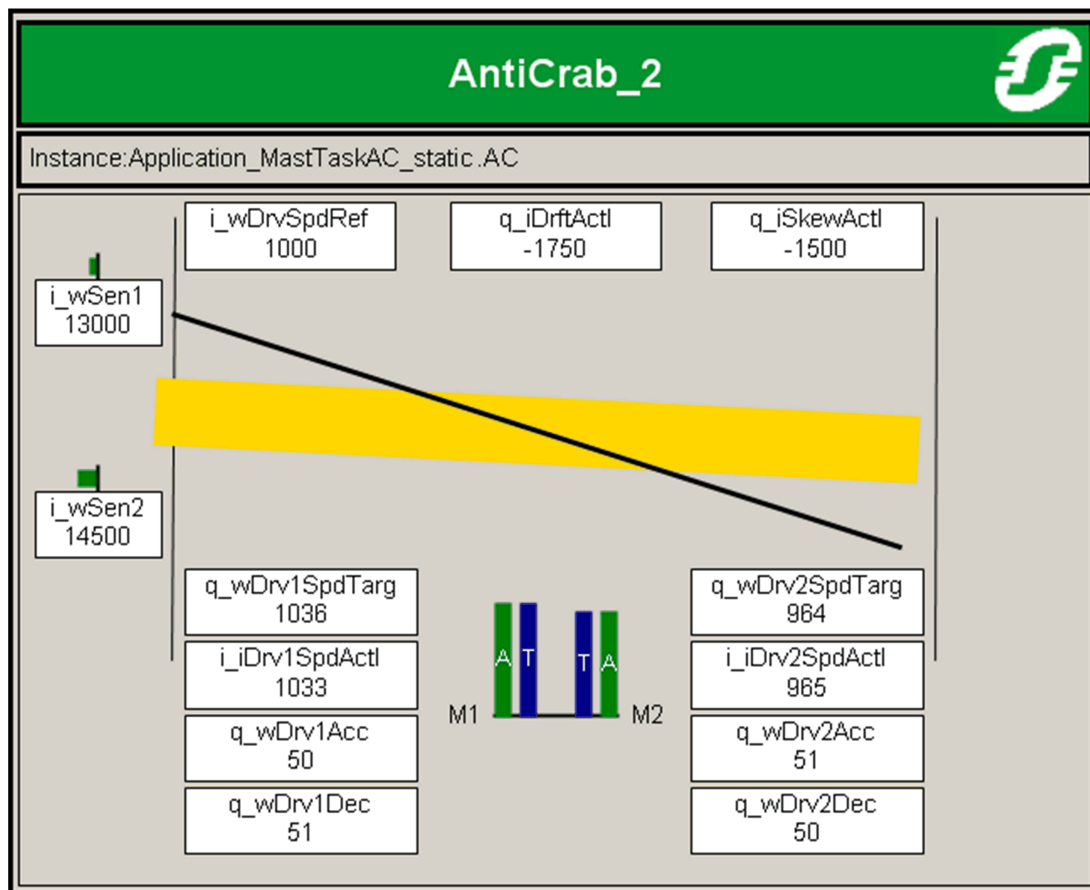
What Is in This Section?

This section contains the following topics:

Topic	Page
Visualization	149
Instantiation and Usage Example	151
Commissioning Procedure	153
Troubleshooting	155

Visualization

Overview



The visualization is designed for assistance with commissioning of Anti-crab function.

It dynamically shows the following:

- actual position of the bridge in industrial cranes (yellow rectangle)
- target position of the bridge in industrial cranes as a result of controller output (black line)
- target speeds for motors 1 and 2 (blue rectangles with white T)
- actual speeds of motors 1 and 2 (green rectangles with white A)

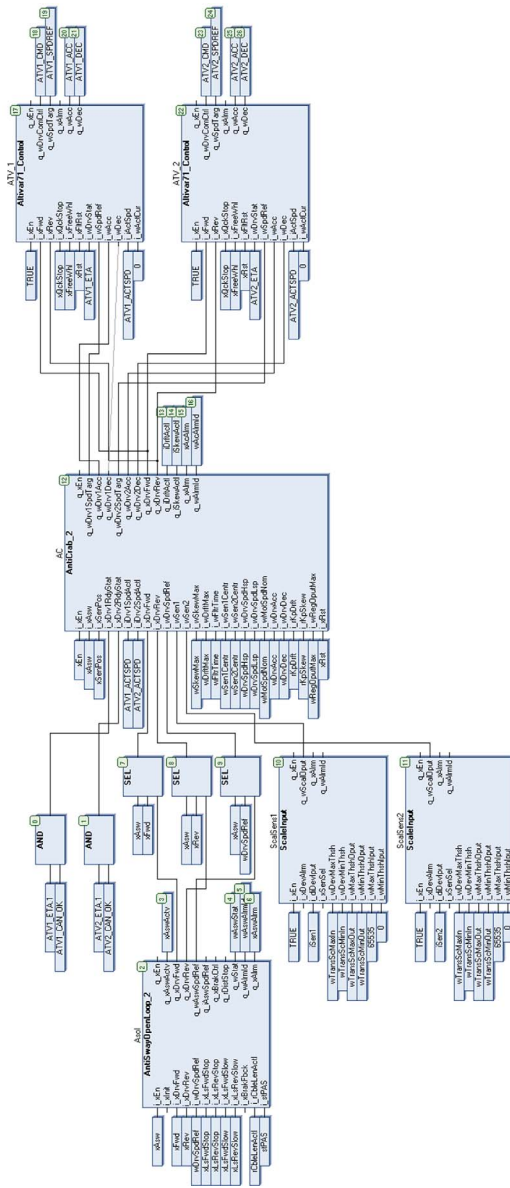
- actual displacement of sensors 1 and 2 compensated with center positions `i_wSen1Centr` and `i_wSen2Centr` (horizontal green rectangles)
- actual values of sensor positions, skew, drift, target and actual speeds and ramp times

It is available inside the library and can be selected in the configuration of frame visualization object under `SEAD_HOIST.Visu_AntiCrab_2`.

Instantiation and Usage Example

Instantiation and Usage Example

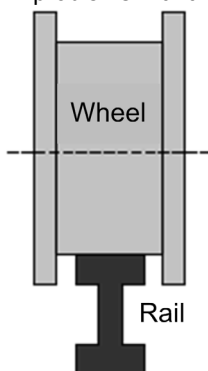
This figure shows an instantiation example of the `AntiCrab_2` function block:



Commissioning Procedure

Commissioning Procedure of the AntiCrab_2 Function Block

1. Even if the AntiCrab_2 FB is going to be used together with AntiSwayOpenLoop_2, first configured them independently. Steps 2 to 12 describe the standalone configuration of AntiCrab_2 FB.
2. Set the drives to factory settings. In drive menu application functions disable, preset speeds. Set command and reference channel to CANopen.
3. Configure the Altivar variable speed drives (VSDs) with correct motor control parameters.
4. In the settings menu of Altivar 71 enter the following:
 - Low speed = 0 (or as close to 0 as possible)
 - Slip compensation = 100 for more rigid control, lower value for softer control
 - Motor control mode depending on machine configuration (SVC-V is preferred).
 - If a different drive is used, set the applicable parameters.
5. Verify that the CANopen communication is functioning correctly between both drives.
6. Instantiate the FB and integrate it in your application.
7. Add scaling functions in order to scale the sensor output values.
8. Move the bridge in industrial cranes in manual mode in order to center the wheels of the bogie with attached sensors on the rail. The crane may be moved by controlling the drives independently, moving sides of the crane at slow speed.
The crane may be also centered on the rail by moving it against the buffer at the end of the runway.
When both wheels are centered on the rail, note down the scaled values of both sensors. Write these values as center positions on inputs `i_wSen1Centr` and `i_wSen2Centr`.
Once the wheels are centered, verify also the other side of the crane. The wheels on the other side of the bridge should be centered as well. If they are too far from the center, it could signal problems with the geometry of the crane or crane runway.



9. Set both drift and skew controllers gain to 0 and move the crane without Anti-crab to get familiar with its behavior.

- 10.** Leave the drift control gain at 0. Increase the value of proportional gain of skew controller gradually while moving the crane and trace the value of actual skew and actual speeds. Find an optimum value of skew controller. The skew should be reduced to a low value without causing oscillations of the motor speeds on both sides of the bridge.
Watch the drift of the bridge in industrial cranes during the movement. If the bridge tends to drift to the left in one direction of movement and to the right in the other direction of movement, it is possible that the center positions are not set correctly and the skew controller does not keep the bridge in industrial cranes straight. In this case, modify the `i_wSen1Centr` and `i_wSen2Centr` values.
- 11.** Keep the setting of the skew controller and start to increase the gain of the drift controller. Find an optimum value of the drift controller.
- 12.** Watch the torque of both drives during the tuning. The purpose of Anti-crab is not keeping the crane dead centered at all cost. It should prevent the wheels from grinding the rails. It is necessary to find an optimal compromise between centering of the bridge and balance of the torques.
- 13.** When the independent configuration has successfully been done, the outputs of the `AntiSwayOpenLoop_2` FB may be connected to inputs of the `AntiCrab_2` FB.

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The motor speeds oscillate during movement.	Gains of skew and drift controllers are too high.	Optimize the controller gains.
The torques of the drives are too different, one of the motors pulls the crane while the other one is braking.	The gains of the skew and drift controllers are not set correctly or the sensor center positions are not aligned. The geometry of the crane causes the difference in torques.	Verify that the alignment of the sensors and the torque value of both drives. If the difference is caused by the geometry, lower the gains of controller gains to get the torque difference to an acceptable level.

Part IV

Anti-Sway

Chapter 6

AntiSwayOpenLoop: Correction of Sway in Horizontal Movements without Sensor

WARNING

UNCOMPENSATED INITIAL SWAY

- Do not start the movement until the load is at rest and motionless.
- Do not engage the Anti-sway function until all initial load movement has ceased.

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

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Functional and Machine Overview	160
6.2	Architecture	165
6.3	Function Block Description - Speed Reference	169
6.4	Function Block Description - Cable Length	186
6.5	Function Block Description - AntiSwayOpenLoop	214
6.6	Function Block Description - AntiSwayOpenLoop_2	231
6.7	Troubleshooting	254

Section 6.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	161
Machine Overview	164

Functional Overview

Functional Description

The Anti-sway function is designed for industrial cranes to correct the sway of the bridge or trolley movement.

NOTE: The Anti-sway solution is not compatible with the `AntiCrab` function block on the same axis.

The Anti-sway function is implemented and designed to be used to control ATV31/312 and ATV71 drives.

The following table shows the 3 main features provided by the solution:

Main Features	Description
Estimator	Load sway estimation is based on an adaptive model using setting parameters. (For example: Linear speed reference, acceleration, deceleration...) and the cable length.
Algorithm	Adaptive continuous algorithm provides Anti-sway correction to the operator command including dual axis Anti-sway operator assistance (For example: including simultaneous trolley, bridge crane and hoisting movements).
Working area control	Activation and suppression of the Anti-sway assistance.

Why Use the Anti-Sway Function?

The main goal of this function is to assist the operator in correcting the load sway of the crane.

During trolley or translation movement, the load which is suspended tends to sway. The swaying may cause damage to the load or surrounding structures. The swaying can additionally increase the time required by the operator to keep the load in the correct position when setting it down. Only experienced operators are able to control the load properly.


WARNING

UNCOMPENSATED WIND SPEED AND DIRECTION

Do not use the `AntiSwayOpenLoop` or `AntiSwayOpenLoop_2` function block to control equipment that is in an outdoor environment, nor any environment subject to high velocity air movement sufficient to sway the load.

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

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

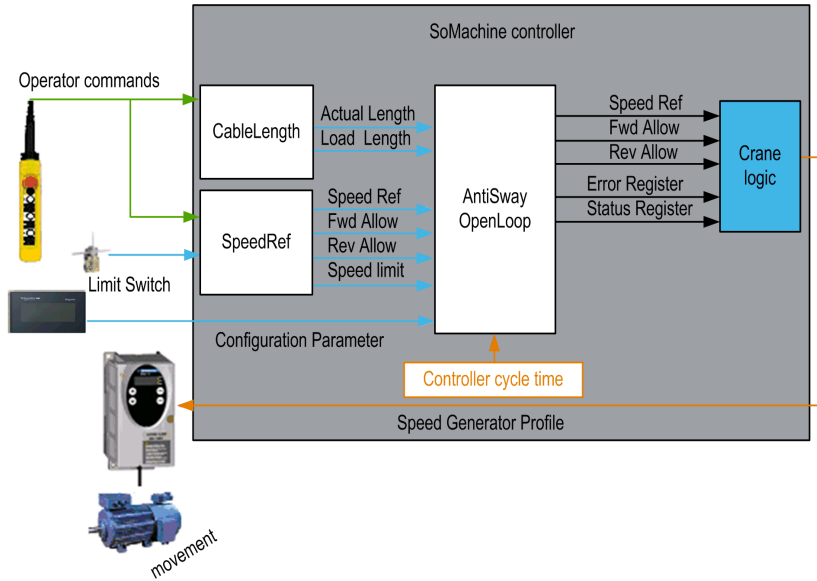
 WARNING
<p>UNINTENDED EQUIPMENT OPERATION</p> <p>Validate all function block input values before and while the function block is enabled.</p> <p>Failure to follow these instructions can result in death, serious injury, or equipment damage.</p>

Solution with the Anti-Sway Function

The following table shows the main advantages of this solution:

Main Features	Description
Cost-effectiveness	Cost effective when compared to a conventional control system based on relays and conductors.
Easy to use	There are only a few parameter settings and no additional sensors.
Higher productivity	The function offers the possibility to run the crane up to +25% of normal crane operating speed which assists in providing equipment protection.
Increase the machine life cycle	Less sway results in less mechanical shock and stress on the crane mechanism and structure.

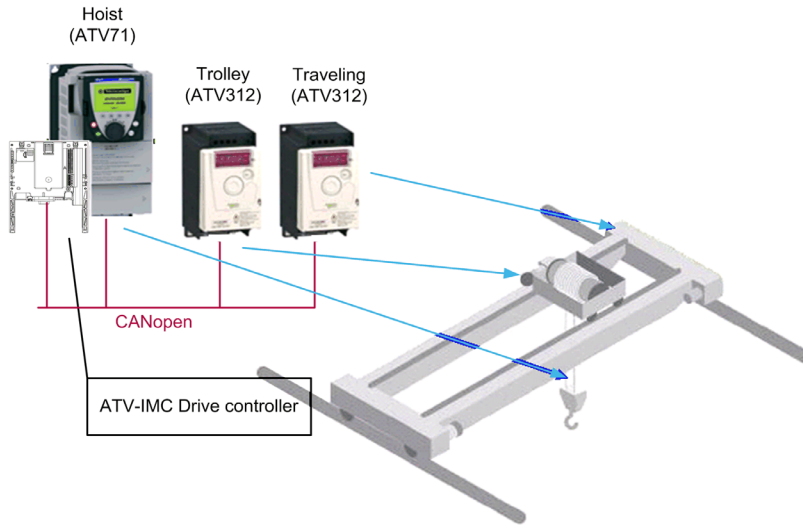
Functional View



Machine Overview

Machine View

The following figure is the machine view of Anti-sway in an overhead traveling crane.



Section 6.2

Architecture

What Is in This Section?

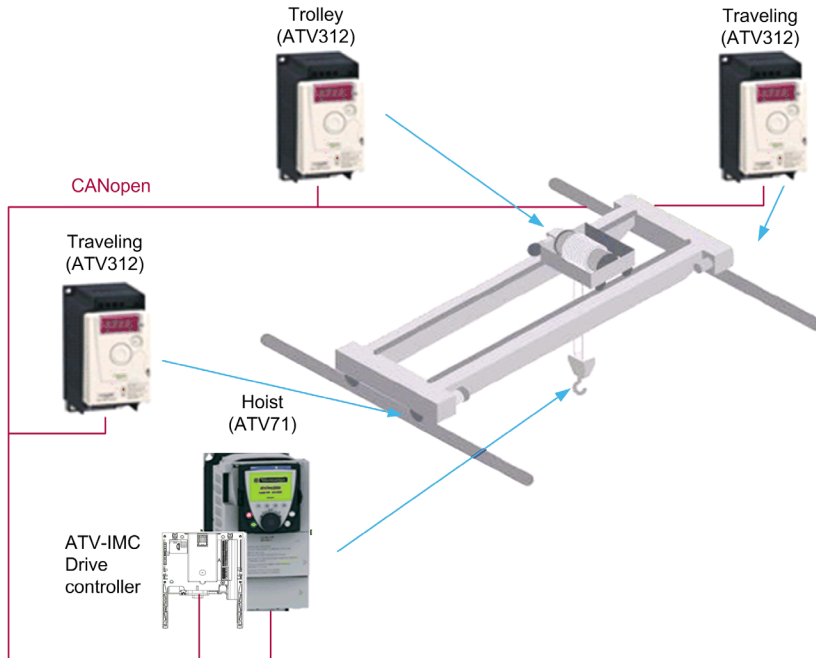
This section contains the following topics:

Topic	Page
Hardware Architecture	166
Software Architecture	167

Hardware Architecture

Hardware Architecture Overview

The following figure displays the hardware architecture of the Anti-sway function with an SoMachine controller.



Software Architecture

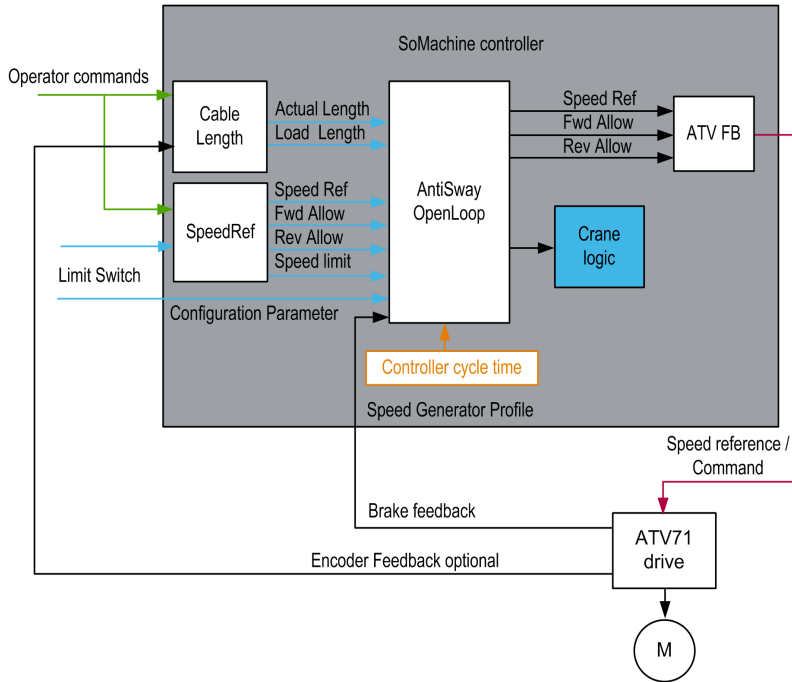
Software Architecture Overview

The Anti-sway function consists of several individual function blocks fulfilling different sub-functions. These 3 main functions must be selected and configured:

- Anti-sway
- Cable length
- Speed reference

AntiSway	Function Block	Description
Anti-sway	AntiSwayOpenLoop AntiSwayOpenLoop_2	These function blocks give the speed profile for the trolley or bridge movement, in order to correct the sway.
Cable length	CableLength_2pos	This function block provides the cable length with 2 positions on a switch selector and an input for the load length.
	CableLength_3pos	This function block provides the cable length with 3 positions on a screw selector and an input for the load length.
	CableLength_Enc CableLength_Enc_2	These function blocks provide the cable length using an encoder and an input for the load length.
Speed reference	SpeedRef_2	This function block gives the speed reference (2 speeds) and the run order for the AntiSwayOpenLoop function block. This function block controls the working area using a limit switch.
	SpeedRef_4	This function block gives the speed reference (4 speeds) and the run order for the AntiSwayOpenLoop function block. This function block controls the working area using a limit switch.
	SpeedRef_A1	This function block gives the speed reference (analog value) and the run order for the AntiSwayOpenLoop function block. This function block controls the working area using a limit switch.

Data Flow Overview



Section 6.3

Function Block Description - Speed Reference

What Is in This Section?

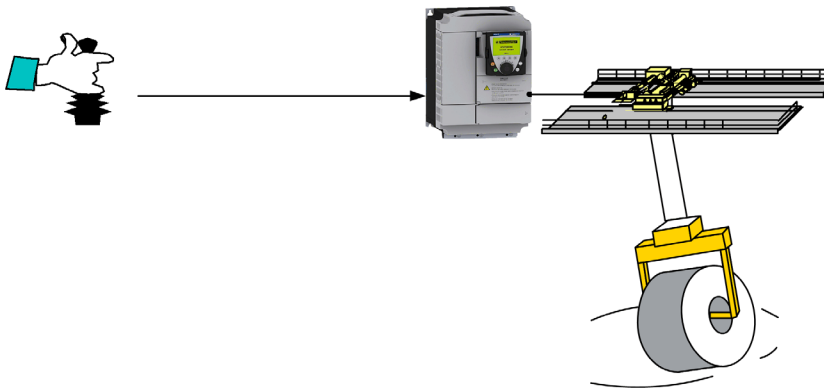
This section contains the following topics:

Topic	Page
Speed Reference - Overview	170
SpeedRef_2 Function Block	171
Input Pin Description - SpeedRef_2	172
SpeedRef_4 Function Block	175
Input Pin Description - SpeedRef_4	176
SpeedRef_AI Function Block	179
Input Pin Description - SpeedRef_AI	180
Output Pin Description - SpeedRef_2, SpeedRef_4 and SpeedRef_AI	183

Speed Reference - Overview

Why Use the Speed Reference Function?

The following figure represents the speed reference overview of Anti-sway function.



The Speed reference function is designed to provide:

- the speed reference according to the command selected.
- the proper speed reference in accordance with the working area by taking the limit switches into account.
- a parameter to set all the linear speed references.
- a register to manage the status of any detected alarms.
- three solutions for managing:
 - 2 speed reference
 - 4 speed reference
 - analog reference

SpeedRef_2 Function Block

Pin Diagram



Function Block Description

The SpeedRef_2 function block gives the speed reference (2 speeds) and the RUN command for the AntiSwayOpenLoop function block. This block manages the working area defined by the limit switches.

Input Pin Description - SpeedRef_2

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_rDrvSpdHigh	REAL	This input is the maximum linear speed reference for the crane movement. Range: 0..4 m/s Accuracy: 0.001 m/s
i_rDrvSpdLow	REAL	This input is the minimum linear speed reference for the crane movement. Range: 0..4 m/s Accuracy: 0.001 m/s
i_rDrvSpdSlowFwd	REAL	This input is the linear speed reference when the sensor slow down forward is reached. Range: 0..4 m/s Accuracy: 0.001 m/s
i_rDrvSpdSlowRev	REAL	This input is the linear speed reference when the sensor slow down reverse is reached. Range: 0..4 m/s Accuracy: 0.001 m/s
i_xDrvFwdSlow	BOOL	This input is for the forward slow down sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvRevSlow	BOOL	This input is for the reverse slow down sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvFwdStop	BOOL	This input is for the stop forward sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvRevStop	BOOL	This input is for the stop reverse sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached

Input	Data Type	Description
i_xDrvFwd	BOOL	This input is for the forward direction coming from the operator command. TRUE: Command FALSE: No action
i_xDrvRev	BOOL	This input is for the reverse direction coming from the operator command. TRUE: Command FALSE: No action
i_xSpdHighSel	BOOL	This input is for the selection of the linear high speed reference coming from the operator command. TRUE: Command FALSE: No action
i_xSpdLowSel	BOOL	This input is for the selection of linear low speed reference coming from the operator command. TRUE: Command FALSE: No action

Speed Selection

i_xSpdLowSel	i_xSpdHighSel	q_rDrvSpd
0	0	= 0
1	0	i_rDrvSpdLow
0	1	i_rDrvSpdHigh
1	1	i_rDrvSpdLow

i_xDrvFwdSlow	i_xDrvRevSlow	q_rDrvSpd
0	0	sensor alarm
1	0	i_rDrvSpdSlowRev if reverse direction
0	1	i_rDrvSpdSlowFwd if forward direction
1	1	i_rDrvSpdHigh or i_rDrvSpdLow

Run Command Selection

i_xDrvFwdStop	i_xDrvRevStop	q_rDrvSpd
0	0	= 0
1	0	Fast Stop must be managed externally
0	1	Fast Stop must be managed externally
1	1	Depending on the input selection and the slow down sensor.

i_xDrvFwd	i_xDrvRev	q_xDrvFwd & q_xDrvRev
0	0	No direction and $q_rDrvSpd = 0$ $q_xDrvFwd = FALSE$ $q_xDrvRev = FALSE$
1	0	$q_xDrvFwd = TRUE$
0	1	$q_xDrvRev = TRUE$
1	1	Keep the first command present

SpeedRef_4 Function Block

Pin Diagram



Function Block Description

The SpeedRef_4 function block gives the speed reference (4 speeds) and the RUN command for the AntiSwayOpenLoop function block. This block manages the working area defined by the limit switches.

Input Pin Description - SpeedRef_4

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_rDrvSpdHigh	REAL	This input is the maximum linear speed reference for the crane movement. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdPre2	REAL	This input is the linear preset2 speed reference for the crane movement. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdPre1	REAL	This input is the linear preset1 speed reference for the crane movement. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdLow	REAL	This input is the minimum linear speed reference for the crane movement. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdSlowFwd	REAL	This input is the linear speed reference when the sensor slow down forward is reached. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdSlowRev	REAL	This input is the linear speed reference when the sensor slow down reverse is reached. Range: 0...4 m/s Accuracy: 0.001 m/s
i_xDrvFwdSlow	BOOL	This input is for the forward slow down sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvRevSlow	BOOL	This input is for the reverse slow down sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvFwdStop	BOOL	This input is for the stop forward sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached

Input	Data Type	Description
i_xDrvRevStop	BOOL	This input is for the stop reverse sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvFwd	BOOL	This input is for the forward direction command coming from the operator command. TRUE: Command FALSE: No action
i_xDrvRev	BOOL	This input is for the reverse direction command coming from the operator command. TRUE: Command FALSE: No action
i_xSpdHighSel	BOOL	This input is for the selection of the linear high speed reference coming from the operator command. TRUE: Command FALSE: No action
i_xSpdRef2Sel	BOOL	This input is for the selection of linear preset speed 2 reference coming from the operator command. TRUE: Command FALSE: No action
i_xSpdRef1Sel	BOOL	This input is for the selection of linear preset speed 1 reference coming from the operator command. TRUE: Command FALSE: No action
i_xSpdLowSel	BOOL	This input is for the selection of linear low speed reference coming from the operator command. TRUE: Command FALSE: No action

Speed Selection

i_xSpdLowSel	i_xSpdRef1Sel	i_xSpdRef2Sel	i_xSpdHighSel	q_rDrvSpd
1	X	X	X	i_rDrvSpdLow
0	1	X	X	i_rDrvSpdPre1
0	0	1	X	i_rDrvSpdPre2
0	0	0	1	i_rDrvSpdHigh
0	0	0	0	= 0

i_xDrvFwdSlow	i_xDrvRevSlow	q_rDrvSpd
0	0	= 0
1	0	i_rDrvSpdSlowRev if reverse direction
0	1	i_rDrvSpdSlowFwd if forward direction
1	1	i_rDrvSpdHigh or i_rDrvSpdPre1 or i_rDrvSpdPre2 or i_rDrvSpdLow

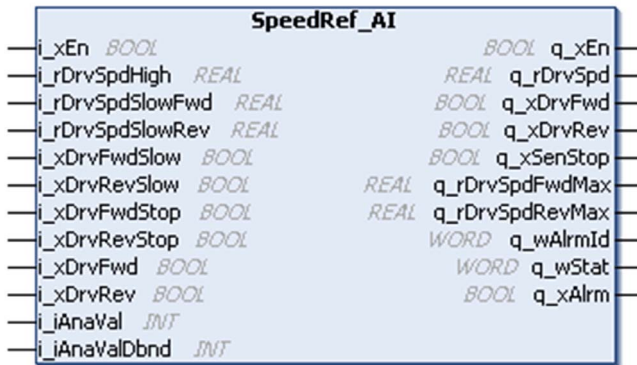
Run Command Selection

i_xDrvFwdStop	i_xDrvRevStop	q_rDrvSpd
0	0	= 0
1	0	Fast Stop must be managed externally.
0	1	Fast Stop must be managed externally.
1	1	Depending on the input selection and the slow down sensor.

i_xDrvFwd	i_xDrvRev	q_xDrvFwd & q_xDrvRev
0	0	No direction and q_rDrvSpd = 0 q_xDrvFwd = FALSE q_xDrvRev = FALSE
1	0	q_xDrvFwd = TRUE
0	1	q_xDrvRev = TRUE
1	1	Keep the first command present.

SpeedRef_AI Function Block

Pin Diagram



Function Block Description

The SpeedRef_AI function block gives the speed reference (Analog input) and the RUN command for the AntiSwayOpenLoop function block. This block manages the working area defined by the limit switches.

Input Pin Description - SpeedRef_AI

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_rDrvSpdHigh	REAL	This input is the maximum linear speed reference for the crane movement. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdSlowFwd	REAL	This input is the linear speed reference when the sensor slow down forward is reached. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdSlowRev	REAL	This input is the linear speed reference when the sensor slow down reverse is reached. Range: 0...4 m/s Accuracy: 0.001 m/s
i_xDrvFwdSlow	BOOL	This input is for the forward slow down sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvRevSlow	BOOL	This input is for the reverse slow down sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvFwdStop	BOOL	This input is for the forward stop sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvRevStop	BOOL	This input is for the reverse stop sensor. A normally closed latching sensor must be used. TRUE: Position not reached FALSE: Position reached
i_xDrvFwd	BOOL	This input is for the forward direction coming from the operator command. TRUE: Command FALSE: No action

Input	Data Type	Description
i_xDrvRev	BOOL	This input is for the reverse direction coming from the operator command. TRUE: Command FALSE: No action
i_iAnaVal	INT	This input is for the analog input for the speed coming from the operator command. Range: 0...8192 (refer the note below for resolution values)
i_iAnaValDbnd	INT	This input is the value for the deadband. The analog input has to be above this value to be taken into account. (Refer the below graph in <i>Deadband View</i> .) Range: 0...8192

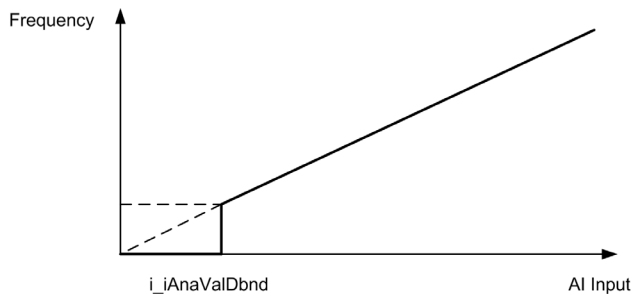
i_iAnaVal

NOTE: The resolution range for the input i_iAnaVal changes depending on the maximum speed value:

- 8192 = resolution of i_rDrvSpdHigh when the slow down sensor is not reached.
- 8192 = resolution of i_rDrvSpdSlowFwd or i_rDrvSpdSlowRev while within the slow down area.

Deadband View

The graph below shows the deadband view with the analog input above the i_iAnaValDbnd value.



Speed Selection

i_xDrvFwdSlow	i_xDrvRevSlow	q_rDrvSpd
0	0	= 0
1	0	i_rDrvSpdSlowRev or i_iAnaVal if reverse direction
0	1	i_rDrvSpdSlowFwd or i_iAnaVal if forward direction
1	1	i_iAnaVal

NOTE: $q_rDrvSpd = i_iAnaVal$ and NOT $i_rDrvSpdSlowRev$ only when the value $i_iAnaVal < i_rDrvSpdSlowRev$.

The same case applies for $i_rDrvSpdSlowFwd$ and $i_iAnaVal$.

Run Command Selection

i_xDrvFwdStop	i_xDrvRevStop	q_rDrvSpd
0	0	= 0
1	0	Fast Stop must be managed externally.
0	1	Fast Stop must be managed externally.
1	1	i_rDrvSpdHigh or i_rDrvSpdLow or i_rDrvSpdSlowRev if reverse direction or i_rDrvSpdSlowFwd if forward direction

i_xDrvFwd	i_xDrvRev	q_xDrvFwd & q_xDrvRev
0	0	No direction AND $q_rDrvSpd = 0$ $q_xDrvFwd = FALSE$ $q_xDrvRev = FALSE$
1	0	$q_xDrvFwd = TRUE$
0	1	$q_xDrvRev = TRUE$
1	1	Keep the first direction present.

Output Pin Description - SpeedRef_2, SpeedRef_4 and SpeedRef_A1

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	This output mirror the i_xEn input variable. TRUE: Enables output FALSE: Disables output
q_rDrvSpd	REAL	This output is the speed reference for AntiSwayOpenLoop function block. Range: 0...4 m/s Accuracy: 0.001 m/s
q_xDrvFwd	BOOL	This output is the forward direction command for the AntiSwayOpenLoop function block.
q_xDrvRev	BOOL	This output is the reverse direction command for the AntiSwayOpenLoop function block.
q_xSenStop	BOOL	This output is the stop command when a stop sensor is reach for the AntiSwayOpenLoop function block.
q_rDrvSpdFwdMax	REAL	This output is the maximum linear speed for the forward direction. Range: 0...4 m/s Accuracy: 0.001 m/s
q_rDrvSpdRevMax	REAL	This output is the maximum linear speed for the reverse direction. Range: 0...4 m/s Accuracy: 0.001 m/s
q_wAlrmId	WORD	This output is the detected alarm register. Refer Notifications (<i>see page 184</i>). Range: 0...127
q_wStat	WORD	This output is the status register. Refer Status Notifications (<i>see page 184</i>). Range: 0...16383
q_xAlrm	BOOL	This output is the alarm bit when an alarm is detected. TRUE: Detected alarm. FALSE: No alarm is detected.

Notifications

q_wAlrmId Value	Alarm Register	Condition
Bit 0	Alarm consistency data	(i_rDrvSpdHigh > 4.0) or (i_rDrvSpdSlowFwd < 0.0) or (i_rDrvSpdSlowFwd > 4.0) or (i_rDrvSpdSlowRev < 0.0) or (i_rDrvSpdSlowRev > 4.0)
Bit 1	Stop sensor	(NOT i_xDrvFwdStop) AND (NOT i_xDrvRevStop)
Bit 2	Slow down sensor	(NOT i_xDrvFwdSlow) AND (NOT i_xDrvRevSlow)
Bit 3	Slow sensor FW	(NOT i_xDrvFwdStop) AND i_xDrvFwdSlow
Bit 4	Slow sensor RV	(NOT i_xDrvRevStop) AND i_xDrvRevSlow
Bit 5	Slow sensor FW and Stop RV	(NOT i_xDrvFwdStop) AND (NOT i_xDrvRevSlow)
Bit 6	Slow sensor RV Stop FW	(NOT i_xDrvFwdSlow) AND (NOT i_xDrvRevStop)

Status Notifications

q_wStat Value	Status Register	Condition
Bit 0	Alarm detected	q_wAlrmId <> 0
Bit 1	Not used	
Bit 2	High speed	q_rDrvSpd = i_rDrvSpdHigh
Bit 3	Forward direction	q_xDrvFwd
Bit 4	Reverse direction	q_xDrvRev
Bit 5	Slow down forward	(NOT i_xDrvFwdSlow) AND q_xDrvFwd
Bit 6	Slow down reverse	(NOT i_xDrvRevSlow) AND q_xDrvRev
Bit 7	Slow down sensor forward	NOT i_xDrvFwdSlow
Bit 8	Slow down sensor reverse	NOT i_xDrvRevSlow
Bit 9	Stop sensor forward	NOT i_xDrvFwdStop
Bit 10	Stop sensor reverse	NOT i_xDrvRevStop
Bit 11	Not used	

q_wStat Value	Status Register	Condition
Bit 12	Not used	
Bit 13	Speed Reference within the Dead Band	$i_iAnaVal < i_iAnaValDbnd$

Section 6.4

Function Block Description - Cable Length

Notice

The function block `CableLength_Enc` has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `CableLength_Enc_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

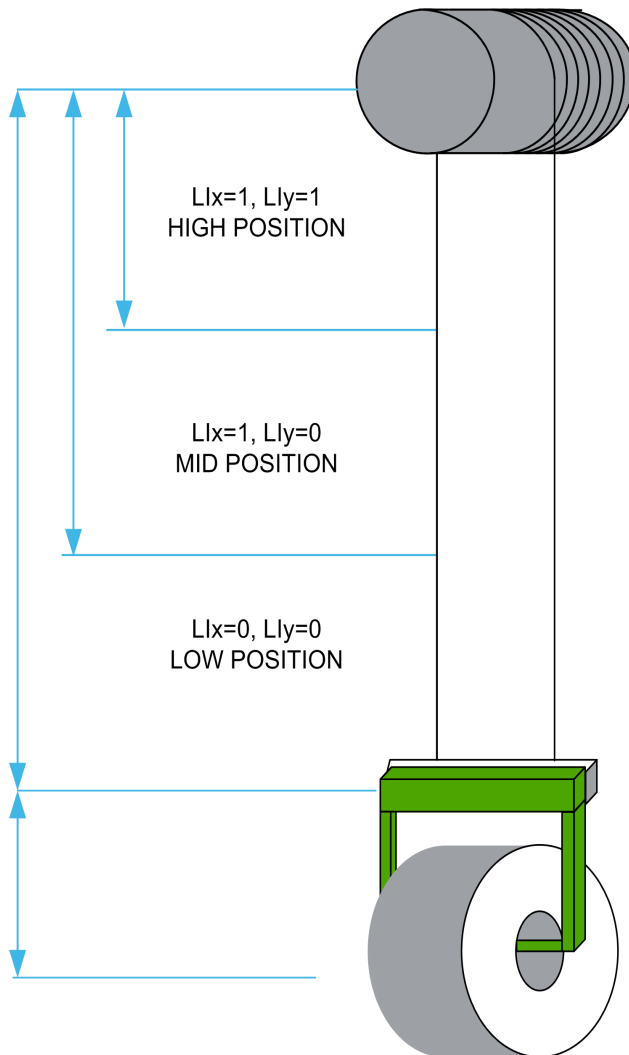
This section contains the following topics:

Topic	Page
Cable Length - Overview	187
<code>CableLength_2Pos</code> Function Block	189
Input Pin Description - <code>CableLength_2Pos</code>	190
Output Pin Description - <code>CableLength_2Pos</code>	192
<code>CableLength_3Pos</code> Function Block	194
Input Pin Description - <code>CableLength_3Pos</code>	195
Output Pin Description - <code>CableLength_3Pos</code>	197
<code>CableLength_Enc</code> Function Block	199
Input Pin Description - <code>CableLength_Enc</code>	200
Output Pin Description - <code>CableLength_Enc</code>	202
Calibration Procedure for the <code>CableLength_Enc</code> Function Block	205
<code>CableLength_Enc_2</code> Function Block	207
Input Pin Description - <code>CableLength_Enc_2</code>	209
Output Pin Description - <code>CableLength_Enc_2</code>	212
Instantiation and Usage Example	213

Cable Length - Overview

Why Use the Cable Length Function?

The following figure represents the cable length overview of Anti-sway function.



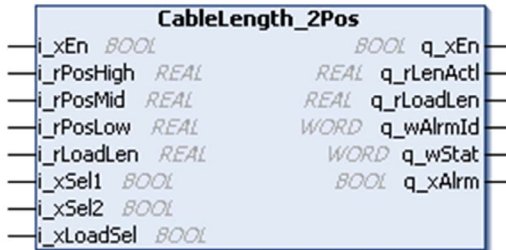
The Cable length function is designed to provide:

- the actual cable length.
- the possibility to add the load length.

- the possibility to perform a calibration when the encoder method is used.
- a register to manage the status of any detected alarms.
- three solutions for managing the cable length.
 - automatic solution with screw selector
 - semi-automatic solution with switch selector
 - automatic solution with encoder

CableLength_2Pos Function Block

Pin Diagram



Function Block Description

This function block gives the actual cable length using a screw selector and the length of the load using a switch selector. This block allows to set 3 different length positions. Refer to the Cable Length overview diagram (*see page 187*). This method is recommended for simple cranes with a maximum operating height of 40 m (131.2 ft) and the area between the 2 zones does not exceed 2 m (6.56 ft).

This function block requires metric input units.

WARNING

UNINTENDED EQUIPMENT OPERATION

You must convert imperial units to metric units in meters before being used as inputs for this function (1 meter equals 3.28 feet).

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

Example:

Low position = 10 m

Mid position = 8 m

High position = 6 m (less angle effect)

Best performance is reached with Low position = 10 m.

Maximum value for Low position (maximum cable length) is 40 m.

Input Pin Description - CableLength_2Pos

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_rPosHigh	REAL	This input is the cable length for the high position. Range: 0...40 m Accuracy: 0.01 m
i_rPosMid	REAL	This input is the cable length for the middle position. Range: 0...40 m Accuracy: 0.01 m
i_rPosLow	REAL	This input is the cable length for the low position. Range: 0...40 m Accuracy: 0.01 m
i_rLoadLen	REAL	This input is the load length value. Range: 0...10 m Accuracy: 0.01 m
i_xSel1	BOOL	This input is the cable length selector position 1. TRUE: Low position FALSE: Medium or high position
i_xSel2	BOOL	This input is the cable length selector position 2. TRUE: Medium position FALSE: High position; depends on i_xSel1
i_xLoadSel	BOOL	This input is used to add the load length to the total cable length calculation. TRUE: Offset used FALSE: Offset not used

Selection Priority

i_xSel2	i_xSel1	q_rLenActl
0	0	i_rPosHigh
1	0	i_rPosMid
1	1	i_rPosLow

NOTE: The screw selection of $i_xSel2 = 0$ and $i_xSel1 = 1$ is invalid and has to be avoided. It will be interpreted as $i_rPosLow$.

Output Pin Description - CableLength_2Pos

Output Pin Description

Input	Data Type	Description
q_xEn	BOOL	This output mirrors the value of i_xEn input variable. TRUE: Enables the function block. FALSE: Disables the function block.
q_rLenAct1	REAL	This output is the actual cable length position. Range: 0...40 m Accuracy: 0.01 m
q_rLoadLen	REAL	This output is the actual load length value. Range: 0...10 m Accuracy: 0.01 m
q_wAlrmId	WORD	This output is the detected alarm register. Refer Notifications (<i>see page 197</i>). Range: 0...3
q_wStat	WORD	This output is the status register. Refer Status Notifications (<i>see page 198</i>). Range: 0...31
q_xAlrm	BOOL	This output is the alarm bit when an alarm is detected. TRUE: Alarm detected FALSE: No alarm is detected

Notifications

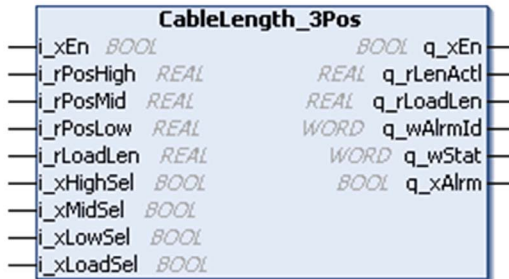
q_wAlrmId Value	Alarm Register	Condition
Bit 0	Alarm consistency data	$i_rPosHigh < 0.0$ or $i_rPosHigh \geq i_rPosMid$ or $i_rPosMid \geq i_rPosLow$ or $i_rPosLow > 40.0$ or $i_rLoadLen < 0.0$ or $i_rLoadLen > 10.0$
Bit 1	Sensor alarm detected	i_xSel1 AND NOT i_xSel2 . The screw selector configuration or operation is not correct.

Status Notifications

q_wStat Value	Status Register	Condition
Bit 0	Alarm detected	$q_wAlrmId \neq 0$
Bit 1	Length high	$\text{NOT } i_xSel1 \text{ AND NOT } i_xSel2.$
Bit 2	Length middle	$\text{NOT } i_xSel1 \text{ AND } i_xSel2$
Bit 3	Length low	$(i_xSel1 \text{ AND } i_xSel2) \text{ OR } (i_xSel1 \text{ AND NOT } i_xSel2)$
Bit 4	Load present	$i_xLoadSel$

CableLength_3Pos Function Block

Pin Diagram



Function Block Description

This function block gives the actual cable length using the switch selector and the length of the load using the switch selector. This block gives the possibility of setting three different length areas. This method is recommended for simple cranes with a maximum operating height of 40 m (131.2 ft) and the area between the 2 zones does not exceed 2 m (6.56 ft).

This function block requires metric input units.

WARNING

UNINTENDED EQUIPMENT OPERATION

You must convert imperial units to metric units in meters before being used as inputs for this function (1 meter equals 3.28 feet).

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

Example:

Low position = 10 m

Mid position = 8 m

High position = 6 m (less angle effect)

Best performance is reached with Low position = 10 m.

Maximum value for Low position (maximum cable length) is 40 m.

Input Pin Description - CableLength_3Pos

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_rPosHigh	REAL	This input is the cable length for the high position. Range: 0...40 m Accuracy: 0.01 m
i_rPosMid	REAL	This input is the cable length for the middle position. Range: 0...40 m Accuracy: 0.01 m
i_rPosLow	REAL	This input is the cable length for the low position. Range: 0...40 m Accuracy: 0.01 m
i_rLoadLen	REAL	This input is the load length value. Range: 0...10 m Accuracy: 0.01 m
i_xHighSel	BOOL	This input is the cable length selection position high. TRUE: Selected FALSE: Not selected
i_xMidSel	BOOL	This input is the cable length selection position middle depending on i_xHighSel. TRUE: Medium position selected FALSE: Medium position deselected
i_xLowSel	BOOL	This input is the cable length selection position low depending on i_xHighSel and i_xMidSel. TRUE: Low position selected FALSE: Low position deselected
i_xLoadSel	BOOL	This input is used to add the load length to the total cable length calculation. TRUE: Offset used FALSE: Offset not used

Selection Priority

i_xHighSel	i_xMidSel	i_xLowSel	q_rLenActl
1	0	0	i_rPosHigh
0	1	0	i_rPosMid
0	0	1	i_rPosLow

Output Pin Description - CableLength_3Pos

Output Pin Description

Input	Data Type	Description
q_xEn	BOOL	This output mirrors the value of i_xEn input variable. TRUE: Enables the function block. FALSE: Disables the function block.
q_rLenActl	REAL	This output is the actual cable length position. Range: 0...40 m Accuracy: 0.01 m
q_rLoadLen	REAL	This output is the actual load length value. Range: 0...10 m Accuracy: 0.01 m
q_wAlrmId	WORD	This output is the detected alarm register. Refer Notifications (see page 197). Range: 0...1
q_wStat	WORD	This output is the status register. Refer Status Notifications (see page 198). Range: 0...31
q_xAlrm	BOOL	This output is the alarm bit when an alarm is detected. TRUE: Alarm detected FALSE: No alarm is detected

Notifications

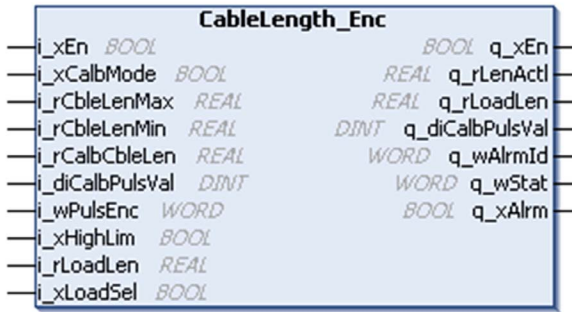
q_wAlrmId Value	Alarm Register	Condition
Bit 0	Alarm consistency data	i_rPosHigh < 0.0 or i_rPosHigh ≥ i_rPosMid or i_rPosMid ≥ i_rPosLow or i_rPosLow > 40.0 or i_rLoadLen < 0.0 or i_rLoadLen > 10.0

Status Notifications

q_wStat Value	Status Register	Condition
Bit 0	Alarm detected	$q_wAlrmId \neq 0$
Bit 1	Length high	$\text{NOT } i_xLowSel \text{ AND NOT } i_xMidSel \text{ AND } i_xHighSel$
Bit 2	Length middle	$\text{NOT } i_xLowSel \text{ AND } i_xMidSel$
Bit 3	Length low	$i_xLowSel \text{ OR } (\text{NOT } i_xLowSel \text{ AND NOT } i_xMidSel \text{ AND NOT } i_xHighSel)$
Bit 4	Load present	$i_xLoadSel$

CableLength_Enc Function Block

Pin Diagram



Function Block Description

This function block gives the actual cable length using an encoder and the length of the load using a switch. Three different length areas are possible and it is also possible to perform a calibration for the encoder resolution. The use of the CableLength_Enc function block is recommended for cranes with a maximum operating height of 40 m.

This function block requires metric input units.

WARNING

UNINTENDED EQUIPMENT OPERATION

You must convert imperial units to metric units in meters before being used as inputs for this function (1 meter equals 3.28 feet).

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

NOTE: CableLength_Enc is an automatic function that does not require input from the operator.

Input Pin Description - CableLength_Enc

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_xCalbMode	BOOL	This input is the calibration mode. TRUE (rising edge): The calibration mode is selected. FALSE (falling edge): The calibration mode is finished and the value from Encoder Calibration parameter i_rCalbCbleLen is taken into account.
i_rCbleLenMax	REAL	This input is the cable maximum length for the bottom position. Range: 0...40 m Factory setting: 5.00 m Accuracy: 0.01 m
i_rCbleLenMin	REAL	This input is the cable minimum length for the top position. Range: 0...40 m Factory setting: 0.00 m Accuracy: 0.01 m
i_rCalbCbleLen	REAL	This input is the value of the cable length at the end of the calibration procedure. Only the length of travel is taken into account, not the complete length. Range: 0...40 m Factory setting: 5.0 m Only the traveled length (that is not the complete length) is taken into account. The value must be retentive.
i_diCalbPulsVal	DINT	This input is the cable length selection position high. Range: -2147483648...2147483647 Used typically when it is the same crane.
i_wPulsEnc	WORD	This input is for the actual cable length value. The ATV71 (encoder pulse register) parameter, PUC is used to determine this value. Range: 0 to 65535
i_xHighLim	BOOL	The high limit sensor input is used for the cable length position and during calibration. Refer also to detailed description (<i>see page 201</i>) of i_xHighLim. TRUE: Position not reached FALSE: Position reached
i_rLoadLen	REAL	This input is the load length value. Range: 0...10 m Accuracy: 0.01 m

Input	Data Type	Description
i_xLoadSel	BOOL	This input is used to add the load length to the total cable length calculation. TRUE: Offset used FALSE: Offset not used

i_xHighLim

In this configuration, a Normally Close contact is mandatory at the top position. The Homing function is used to establish the cable length and to allow for stretching.

On top position = FALSE; else = TRUE.

Output Pin Description - CableLength_Enc

Output Pin Description

Input	Data Type	Description
q_xEn	BOOL	This output is equal to i_xEn input variable. TRUE: Enabled FALSE: Disabled
q_rLenAct1	REAL	This output is the actual cable length position. Range: 0...40 m Accuracy: 0.01 m
q_rLoadLen	REAL	This output is the actual load length value. Range: 0...10 m Accuracy: 0.01 m
q_diCalbPulsVal	DINT	This is the pulse value corresponding to the cable length distance at the end of the calibration Range: -2147483648...2147483647
q_wAlrmId	WORD	This output is the detected alarm register. Refer Notifications (see page 202). Range: 0...31
q_wStat	WORD	This output is the status register. Refer Status Notifications (see page 203). Range: 0...16383
q_xAlrm	BOOL	This output is the alarm bit when an alarm is detected. TRUE: Alarm detected FALSE: No alarm detected

Notifications

q_wAlrmId Value	Alarm Register	Condition
Bit 0	Alarm consistency data	$i_rCbleLenMin \geq i_rCbleLenMax$ OR $i_rCbleLenMin < 0.0$ OR $i_rCbleLenMax > 40.0$ OR $i_rLoadLen < 0.0$ OR $i_rLoadLen > 10.0$ OR $q_rLenAct1 < i_rCbleLenMin$ OR $q_rLenAct1 > i_rCbleLenMax$ OR $i_rCalbCbleLen \leq i_rCbleLenMin$ OR $i_rCalbCbleLen > i_rCbleLenMax - i_rCbleLenMin$
Bit 1	Not used	-

q_wAlrmId Value	Alarm Register	Condition
Bit 2	Did not pass calibration	Calibration Done AND (*Check that the calibration has counted pulses*) Calib Count pulse = 0 OR (*Check if the calibration meter was change*) i_rCalbCbleLen <> Saved length OR i_rCalbCbleLen < 0
Bit 3	Cable Min greater than Cable Max	$i_rCbleLenMin \geq i_rCbleLenMax$
Bit 4	Overflow PUC	Overflow of the pulse accumulation register. This detected alarm can be reset only by restarting calibration. The encoder resolution is too high and must be changed by using the PID parameter (Encoder pulse divisor).
Bit 5	Calibration did not pass with restitute pulse value	$i_diCalbPulsVal \neq 0$ AND $i_rCalbCbleLen = 0$ OR $i_rCalbCbleLen \leq i_rCbleLenMin$

Status Notifications

q_wStat Value	Status Register	Condition
Bit 0	Alarm detected	$q_wAlrmId \neq 0$
Bit 1	Not used	-
Bit 2	Not used	-
Bit 3	Length low	-
Bit 4	Load present	$i_xLoadSel$
Bit 5	Calibration not done	Calibration mode AND top sensor not reached.
Bit 6	Calibration In progress	Calibration started (pulses accumulating) AND $i_xCalbMode = TRUE$
Bit 7	Calibration done	Calibration mode exit ($i_xCalbMode$ fall to 0)
Bit 8	Alarm PUC does not change	Calibration done AND Calibration Count pulses = 0
Bit 9	In Calibration mode	Calibration not done
Bit 10	Running mode	Calibration has finished
Bit 11	Overflow of the PUC accumulation register	Overflow of the accumulated pulses. This detected alarm can only be reset by restarting the calibration. The encoder resolution is too high and must be changed by using the PID parameter (Encoder pulse divisor).
Bit 12	Up movement	The load is being raised. If no movement occurs, the last direction is maintained.

q_wStat Value	Status Register	Condition
Bit 13	Down movement	The load is being lowered. If no movement occurs, the last direction is maintained.

Calibration Procedure for the CableLength_Enc Function Block

Procedure Example

Example

Parameter	Description
i_xEn	LI51 (ATV IMC)
i_xCalbMode	LI52 (ATV IMC)
i_rCbleLenMax	40.0 m (131.23 ft)
i_rCbleLenMin	2.00 m (6.56 ft)
i_rCalbCbleLen	0.00 m
i_wPulsEnc	The PUC parameter register of the encoder ATV71 drive is used.
i_xHighLim	LI54 (ATV IMC)
i_rLoadLen	0.00 m
i_xLoadSel	FALSE

Procedure

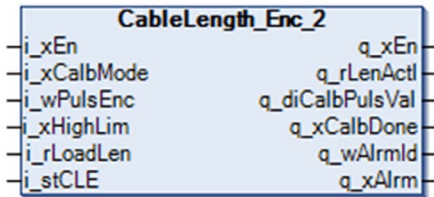
Steps	Description
1	i_xEn = TRUE
2	i_xCalbMode = TRUE a. Signal calibration not done changes to TRUE q_wStat. bit5 b. Signal calibration mode changes to TRUE q_wStat. bit9
3	Run forward until reaching the i_xHighLim signal. The signal changes to FALSE. This is Up movement.
4	Run reverse and leave i_xHighLim signal. The output signals change in this way. this is Down movement. a. i_xHighLim changes to TRUE b. Calibration in process changes to TRUE q_wStat. bit6 c. Calibration mode remains TRUE value q_wStat. bit9 d. Down movement changes to TRUE q_wStat. bit13 e. Calibration not done changes to FALSE q_wStat. bit5
5	Stop the reverse command and introduce the value into i_rCalbCbleLen. (The Anti-sway correction is better if the precision of the value is better. The accuracy is 0.01 m. The distance can be measured with a laser telemeter).

Steps	Description
6	Change <code>i_xCalbMode = FALSE</code> a. Calibration in process changes to FALSE <code>q_wStat. bit6</code> b. Calibration mode changes to FALSE <code>q_wStat. bit9</code> c. Calibration done changes to TRUE <code>q_wStat. bit7</code> d. Running mode changes to TRUE <code>q_wStat. bit10</code>

NOTE: Cable length movements are not recognized if the `CableLength_Enc` function block has not been calibrated. For example, start of a movement when the slave on CANopen is not ready. A new calibration procedure must be performed.

CableLength_Enc_2 Function Block

Pin Diagram

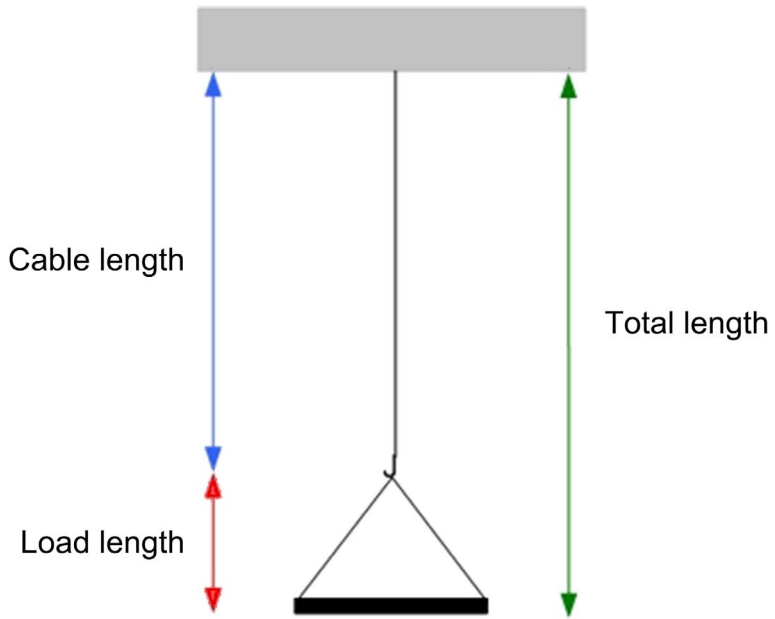


Function Block Description

This function block calculates the actual cable length using an encoder. It is possible to perform a calibration with any given encoder resolutions. The function block does not require any inputs from the operator after the calibration has taken place. Additionally it is possible to add the length of the carried load to the calculated length of the cable to achieve higher precision of the Anti-sway function block.

This function block is needed to provide the accurate length of the hoist cable to the Anti-sway function block. The performance of Anti-sway depends on the accuracy of that value.

Length diagram of CableLengthEnc_2 function block:



Input Pin Description - CableLength_Enc_2

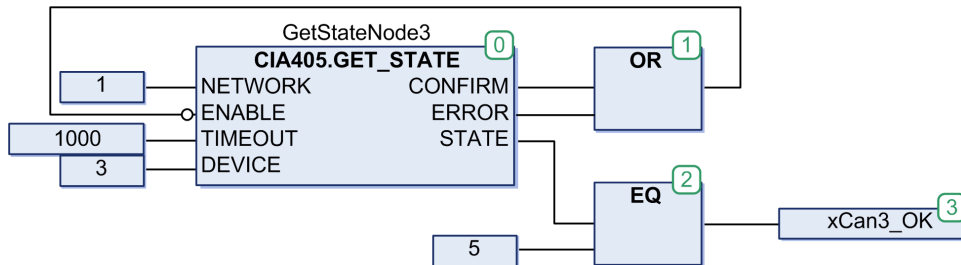
Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (<i>see page 209</i>) below this table.
i_xCalbMode	BOOL	This input enables the calibration. TRUE: Calibration active. FALSE: Calibration inactive. Refer to detailed description (<i>see page 210</i>) below this table.
i_wPulsEnc	WORD	This is the encoder pulses input from the drive (PUC). Range: 0 to 65535
i_xHighLim	BOOL	This is the forward stop limit switch input of the hoisting axis. NOTE: It is mandatory to use a limit switch which is normally closed (NC contact). TRUE: Limit inactive FALSE: Limit active
i_rLoadLen	REAL	This input is the input for an additional load length if needed. Range: 0...100 Scaling/Unit: 1 m
i_stCLE	CLE	Refer to Sub-structure description (<i>see page 211</i>).

i_xEn

NOTE: If the information of the limit switch input `i_xHighLim` is transferred via CANopen, we strongly recommend connecting the status of the CANopen bus to this `i_xEn` input, because during the boot-up phase of the controller, the CANopen bus will get operational later than the program of the controller. This could lead to a wrong cable length output of the function block.

The following figure shows an example of obtaining the CANopen network status of the hoisting drive, assuming that the drive has been assigned a Node ID of 3:



By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state. As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that the status of CANopen communication is used to enable the function block.

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

The fallback state of `CableLengthEnc_2` is given in the table below:

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_rLenAct1</code>	REAL	0
<code>q_diCalbPulsVal</code>	DINT	no change
<code>q_xCalbDone</code>	BOOL	no change
<code>q_wAlrmId</code>	WORD	0
<code>q_xAlrm</code>	BOOL	FALSE

`i_xCalbMode`

This is the input to enter into calibration mode of the FB. The calibration is conducted as followed:

1. It is mandatory to use a limit switch which is normally closed (NC contact).
2. Lift the hoist up to its highest limit, so the `i_xHighLim` is getting FALSE.
3. Set the `i_xCalbMode` input to TRUE.
4. Lower the hoist down to the ground.

5. Enter the distance from the top position down to the ground into the `i_stCLE.rCalbCbleLen` input, scaling is 1.0 meter.
6. Set the `i_xCalbMode` input to FALSE.

Sub-Structure Description of `i_stCLE`

A structure of the data type `CLE` containing configuration parameters.

For example, a declaration of structure variable with initial values:

```
VAR
structure_instance: data_type := (element1 := XX, element2 := YY);
END_VAR
```

Structure Parameter	Data Type	Description
<code>rCbleLenMin</code>	REAL	Remaining cable length at forward stop. When the hoist was stopped by the upper limit switch (forward stop) there is normally a distance left between the hook and the point where the cable is attached to the drum, the remaining cable length which cannot be under-run. This is the minimum cable length. Range: 0.5...10 Scaling/Unit: 1 m
<code>rCalbCbleLen</code>	REAL	Length of the calibration run. This is the length driven during calibration, usually the length in between the upper limit switch of the hoist and the hook touching the ground, the maximum length of the crane. It is used to determine the ratio of length versus pulses and only entered once during calibration. Range: 0...100 Scaling/Unit: 1 m
<code>diCalbPulsVal</code>	DINT	Number of pulses to <code>rCalbCbleLen</code> for manual override of calibration. In a series of cranes where the calibration results are known, this is the manual override to omit the calibration procedure. It determines the number of pulses relating to the entered value of pulses at <code>rCalbCbleLen</code> . Range: 1...2147483647 Scaling/Unit: 1 pulse

Output Pin Description - CableLength_Enc_2

Output Pin Description

Input	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled FALSE: Function block disabled
q_rLenAct1	REAL	This output is the actual hoisting cable length. It is mandatory to conduct a calibration or use the manual override possibility once to ensure the accuracy of this output. Range: 0.5...100 Scaling/Unit: 1 m
q_diCalbPulsVal	DINT	Pulse count corresponding to the distance covered during calibration movement.
q_xCalbDone	BOOL	Status of calibration. TRUE: Valid calibration present. FALSE: Calibration not done.
q_wAlrmId	WORD	This output is the detected alarm register. Refer to Notifications (<i>see page 212</i>). Range: 0...127
q_xAlrm	BOOL	This output is the alarm bit when an alarm is detected. TRUE: Alarm detected FALSE: No alarm detected

Notifications

The alarm ID when q_xAlrm is TRUE. The ID is indicated by a bit set in this output.

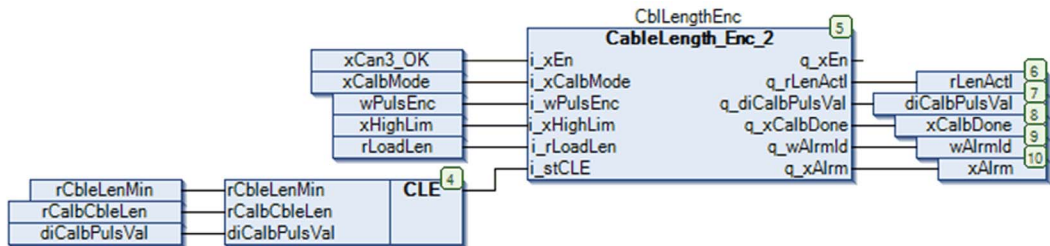
The bit values have the following meaning:

Alarm Bit q_wAlrmId	Description
0	i_stCLE.rCbleLenMin is smaller than 0.5
1	i_stCLE.rCalbCbleLen is smaller than i_stCLE.rCbleLenMin
2	i_rLoadLen is smaller than 0
3	i_stCLE.diCalbPulsVal is smaller than 0

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the CableLength_Enc_2 function block:



Section 6.5

Function Block Description - AntiSwayOpenLoop

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AntiSwayOpenLoop_2`. The new function block is not pin-compatible with the obsolete function block.

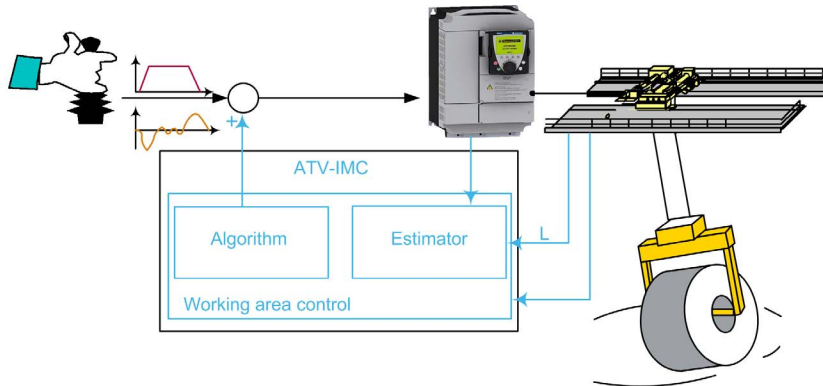
What Is in This Section?

This section contains the following topics:

Topic	Page
Anti-Sway - Overview	215
<code>AntiSwayOpenLoop</code> Function Block	216
Input Pin Description - <code>AntiSwayOpenLoop</code>	217
Structure Parameter - <code>AntiSwayOpenLoop</code>	219
Output Pin Description - <code>AntiSwayOpenLoop</code>	222
<code>AntiSwayOpenLoop</code> Main Parameters	224

Anti-Sway - Overview

Why Use the Anti-Sway Function?



The Anti-sway solution is designed to:

- Provide the speed profile reference to correct the sway.
- Provide innovative highly efficient Anti-sway control without any additional sensors on the crane.
- Estimate the load sway based on an adaptive model using setting parameters (Linear speed reference, acceleration deceleration and so on) and the cable length.
- Provide Anti-sway correction to the operator command using an adaptive algorithm.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

AntiSwayOpenLoop Function Block

Pin Diagram



Function Block Description

This function block gives the speed profile for the drive to correct the sway.

Input Pin Description - AntiSwayOpenLoop

Input Pin Description

Input	Data Type	Description
i_xInit	BOOL	This function initializes the internal calculation for the Anti-sway correction. It returns the function to the factory settings (internal variable = 0). The initialization is performed on the rising edge. This function is used after each drive detects the alarm and after each movement which was not requested by the block (Drive detected alarm, Quick Stop, Forced local, Communication alarm...) TRUE: Initialization FALSE: No action
i_xAswEn	BOOL	This input activates the Anti-sway correction. Anti-sway activation occurs when the drive is stopped (Anti-sway speed profile). If the drive is running, the first Anti-sway activation occurs once the drive is stopped. Anti-sway function can be deactivated regardless of the drive status (linear speed profile). TRUE: Enable AntiSwayOpenLoop FALSE: Disable AntiSwayOpenLoop, linear ramp
i_rDrvSpd	REAL	This input is the linear speed reference coming from the speed reference function block. The value must be an absolute value. Range: 0..4 m/s Accuracy: 0.001 m/s
i_xDrvFwd	BOOL	This input is the forward direction command coming from the speed reference function block.
i_xDrvRev	BOOL	This input is the reverse direction command coming from the speed reference function block.
i_xFbBrk	BOOL	This is the feedback brake status. A TRUE value indicates the brake is released. The feedback is needed to start the movement without the brake engaged. TRUE: Brake released FALSE: Brake engaged

Input	Data Type	Description
i_rDrvSpdFwdMax	REAL	This is the maximum linear speed for the forward direction coming from the speed reference function block. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rDrvSpdRevMax	REAL	This is the maximum linear speed for the reverse direction coming from the speed reference function block. Range: 0...4 m/s Accuracy: 0.001 m/s
i_rCbleLenAct1	REAL	This is the actual cable length coming from the cable length function block. Range: 0...40 m Accuracy: 0.01 m
i_rLoadLen	REAL	This is the length of the load coming from the cable length function block. Range: 0...10 m Accuracy: 0.01 m
i_strAswConf	STRUCT	This parameter is a structure for the referenced configuration parameters (see page 219).

Structure Parameter - AntiSwayOpenLoop

i_strAswConf

This parameter is a structure for the following configuration parameters. The parameters inside the structure are normally changed only during the commissioning phase.

NOTE: The syntax for structure declaration is as follows:

TYPE <Structurename>:

STRUCT

<Declaration of identifier Variables 1>

<Declaration of identifier Variables n>

END_STRUCT

END_TYPE

Parameter	Data Type	Description
rFreqMax	REAL	This parameter is the maximum frequency for the movement. Range: 50...200 Hz Factory setting: 50.0 Hz Accuracy: 0.1 Hz
rSpdLinMax	REAL	This parameter is the maximum linear speed for the maximum frequency movement. Range: 0...4 m/s Factory setting: 1.0 m/s Accuracy: 0.001 m/s
rAcc	REAL	This parameter is the drive acceleration time when the AntiSwayOpenLoop function block is used. Range: 0.5...10 s Factory setting: 4.0 s Accuracy: 0.1 s
rDec	REAL	This parameter is the drive deceleration time when the AntiSwayOpenLoop function block is used. Range: 0.5...10 s Factory setting: 4.0 s Accuracy: 0.1 s
rCoefFrct	REAL	This is the friction coefficient during the movement. Range: 0.0...1.0 Recommendation: 0.8 Bridge/Translation and 0.2 for Trolley

Parameter	Data Type	Description
rAswCorr	REAL	The Anti-sway correction is for smooth action. Refer also to detailed description (<i>see page 220</i>) of rAswCorr. Range: 0.0...100.0% Factory setting: 100.0 % Use this setting in steps of 1%. 100% correction is very fast.
rAswCorrRamp	REAL	This command is used to adapt the reaction time of the AntiSwayOpenLoop. Refer also to detailed description (<i>see page 221</i>) of rAswCorrRamp. Range: 0.0...100.0% Factory setting: 80.0% of acceleration and deceleration value
rAswJogSpd	REAL	This is the speed activation for the Anti-sway correction; (% of maximum linear speed). Typically, for small movement the Anti-sway function is not necessary Range: 0.0...100.0% Factory setting: 25.0% of maximum linear speed
rAswSpdEnd	REAL	The speed of end movement for stopping the Anti-sway correction; (% of maximum linear speed) Range: 0.0...5.0% Accuracy: 0.6% of maximum linear speed
rAswTimeEnd	REAL	The time to stay at rAswSpdEnd for end movement. Range: 100.0...2000.0 ms Factory setting: 800.0 ms
rSmplRate	REAL	This is the controller scan time. The cycle time is used as a periodic cycle (Function block Cycle Time Set). A cycle time of 40 ms is recommended. Range: 30.0...100.0 ms Factory setting: 40.0 ms The cycle time must be set to the periodic cycle.

rAswCorr

The Anti-sway correction is for smooth action. 100% correction is very fast and normally used when the crane has no occupied operator cabin. The setting should be applied in steps of 1%. When the crane has an operator the setting must start at 80% (to reduce the vibration and the stress for the machine).

rAswCorrRamp

This command is used to adapt the reaction time of the `AntiSwayOpenLoop` during a movement when `AntiSwayOpenLoop` is active (% of acceleration and deceleration value). A setting of 100% speed is used for the acceleration and the deceleration. When the setting is 50% only half of the acceleration and deceleration value is used. **Example:**

```
rAswCorrRamp = 50%
```

```
rAcc= 3 s
```

```
rDec = 3 s
```

The ramp applied on the crane are Acceleration = 1.5 s and

Deceleration = 1.5 s

Output Pin Description - AntiSwayOpenLoop

Output Pin Description

Output	Data Type	Description
q_xAswEn	BOOL	This output is the status for the Anti-sway correction. TRUE: Correction in progress FALSE: No correction
q_xAswMove	BOOL	This output is the movement status from the function block. TRUE: Movement in progress FALSE: No movement
q_rAswDrvSpdtarg	REAL	This output is the speed reference for the drive. Range: -rFreqMax...+rFreqMax in Hz
q_xAswDrvFwd	BOOL	This output is the forward command for the drive managed by the function block and not by the operator. TRUE: Forward command FALSE: No command
q_xAswDrvRev	BOOL	This output is the reverse command for the drive managed by the function block and not by the operator. TRUE: Forward command FALSE: No command
q_wAlrmId	WORD	This output is the detected alarm register. Refer alarm register values in Notifications (<i>see page 223</i>). Range: 0...15
q_wStat	WORD	This output is the status register. Refer status register values in Status Notifications (<i>see page 223</i>). Range: 0...255
q_xAlrm	BOOL	This output is the alarm bit when an alarm is detected. TRUE: Alarm detected FALSE: No alarm detected

Notifications

q_wAlrmId Value	Alarm Register
Bit 0	Inconsistent inputs
Bit 1	Inconsistent structure inputs
Bit 2	Tried to initialize the AntiSwayOpenLoop, but the AntiSwayOpenLoop speed profile is currently active ==> output of the AntiSwayOpenLoop
Bit 3	Tried to activate AntiSwayOpenLoop, but the AntiSwayOpenLoop speed profile is currently active ==> output of the AntiSwayOpenLoop

Status Notifications

q_wStat Value	Status Register
Bit 0	Copy of output q_xAlrm
Bit 1	AntiSwayOpenLoop output reference is stabilized (1% of the linear speed)
Bit 2	Forward limitation active
Bit 3	Reverse limitation active
Bit 4	Copy of output q_xAswEn
Bit 5	Copy of output q_xAswMove
Bit 6	Copy of output q_xAswDrvFwd
Bit 7	Copy of output q_xAswDrvRev

AntiSwayOpenLoop Main Parameters

Overview

The following main parameters are described:

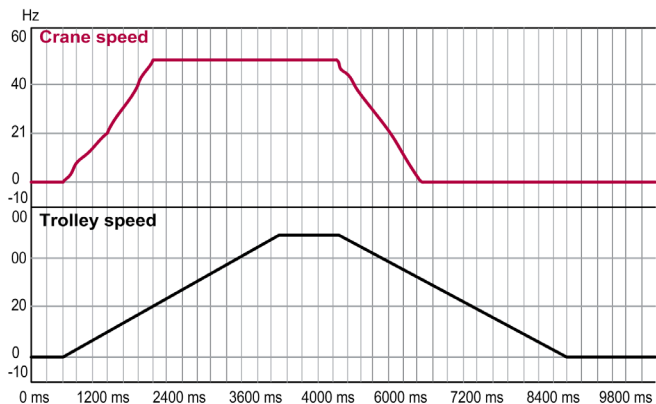
- Speed Profile (*see page 224*)
- Scan Time Effect (*see page 226*)
- Slow Down Sensor (*see page 227*)
- Stop Sensor (*see page 227*)
- Start and Stop the Anti-Sway Correction (*see page 228*)
- Speed End and Time Delay (*see page 228*) (`rAswSpdEnd` and `rAswTimeEnd`)
- Speed Activation Anti-Sway (*see page 228*) (`rAswJogSpd`)
- ATV71 Configuration (*see page 229*)
- ATV31/312 Configuration (*see page 230*)

Speed Profile

When the Anti-sway function is not activated, a specific non linear speed profile is followed. This profile is set up in real time by the `AntiSwayOpenLoop` function block, according to the following main parameters:

- Acceleration and deceleration
- Set point speeds
- Start and stop commands

The following figure represents the speed profile when the Anti-sway function is not activated.

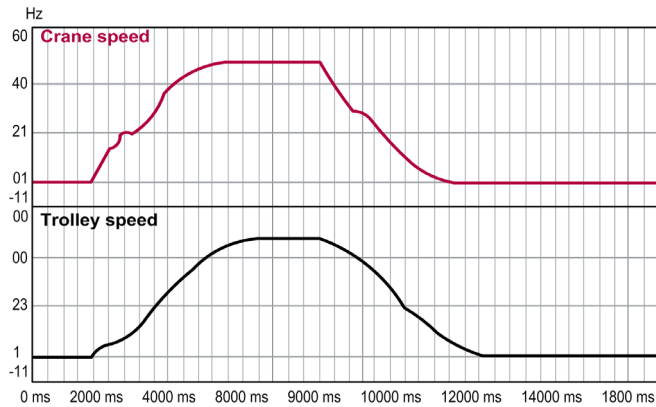


When the Anti-sway function is activated a specific non linear profile is followed. This profile is set up in real time by the `AntiSwayOpenLoop` according to the following main parameters:

- Acceleration and deceleration
- Set point speeds

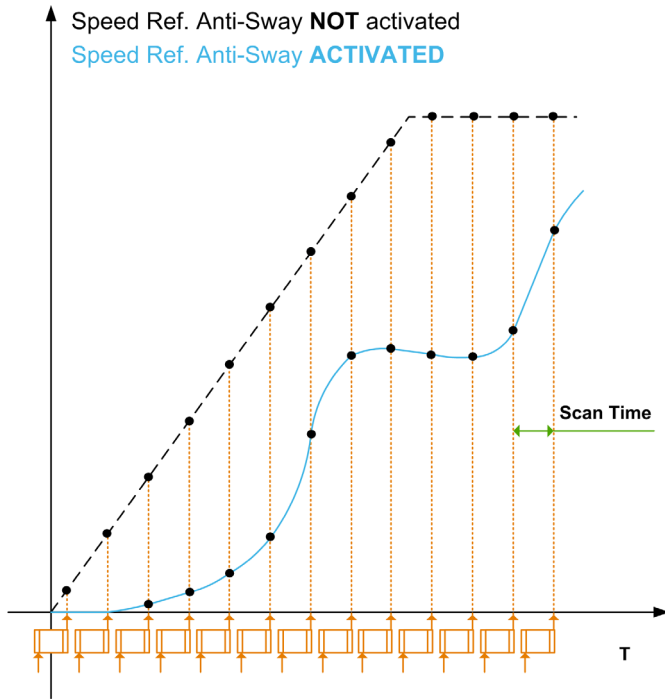
- Start down and limit switches
- Instantaneous length of (cable + load)

The following figure represents the speed profile when the Anti-sway function is activated.



Scan Time Effect

The graph in the below figure shows the effects of scan time.

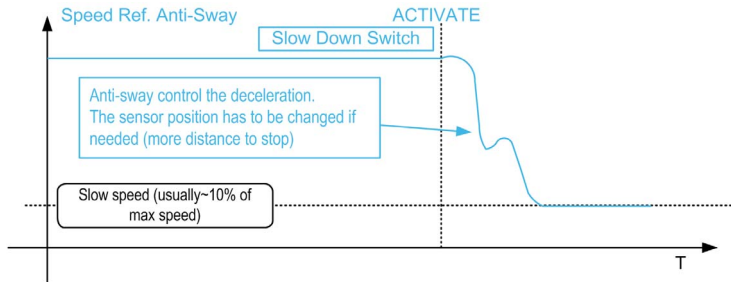


The ramp profile does not have enough value to correct the sway if the cycle time is too high. The value cannot exceed 100 ms for the parameter `rSmp1Rate`.

The cycle time must be measured during implementation. Validate that the setting has the correct value.

Slow Down Sensor

The following figure represents the graph depicting the reaction on the Anti-sway slow down sensor.



When the Slow down sensor switch is open, the speed reference is equal to $i_rDrvSpdFwdMax$ or $i_rDrvSpdRevMax$ depending on the **RUN** command.

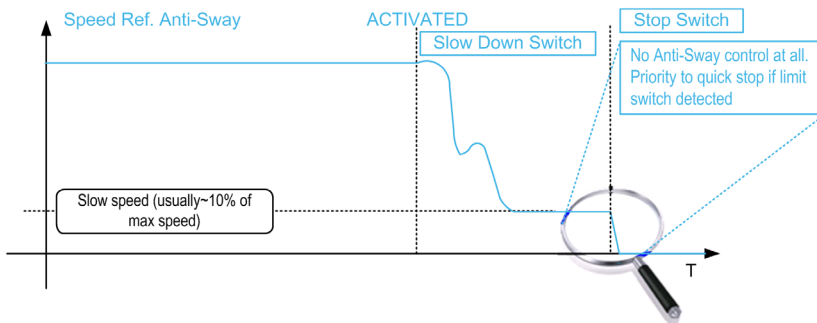
Forward Order $\rightarrow i_rDrvSpdFwdMax$

Reverse Order $\rightarrow i_rDrvSpdRevMax$

NOTE: In some cases, it may be necessary to change the position of the sensor due to the Anti-sway correction.

Stop Sensor

The following figure represents the graph depicting the reaction on the Anti-sway stop sensor.

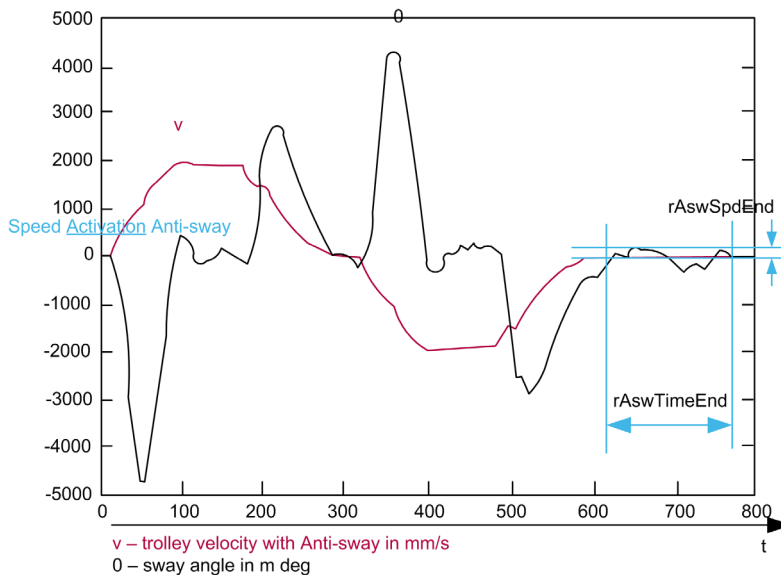


When the Stop sensor is open, the drive must stop as fast as possible.

NOTE: The stop command must be run outside the function block.

Start and Stop the Anti-Sway Correction

The following figure represents the graph of Anti-sway speed activation/deactivation.



Speed End and Time Delay ($rAswSpdEnd$ and $rAswTimeEnd$)

These inputs pins are used in cases of automatic movement on the bridge crane (movement commanded by external controller). On each axis, if the speed is lower than $rAswSpdEnd$ during a time higher or equal to $rAswTimeEnd$, the drive is stopped with a stop ramp.

These parameters allow the definition of a compromise between the precision of the stop position and the lead time for the complete stop. The 2 values may have different priorities depending on the application. (For example: A smoother movement if there is an operator cabin or a more rigid trolley to save time).

The Anti-sway activity indicator is active until the completion of movement on both axis.

Speed Activation Anti-Sway ($rAswJogSpd$)

When the operators need accurate positioning during bridge or trolley movement, they have the option of making short movements with a low speed set point. This parameter allows them to deactivate the Anti-sway assistant correction automatically.

If the speed is lower than $rAswJogSpd$ on each axis, the Anti-sway assistant has no effect and generates linear ramps.

ATV71 Configuration

The following table contains typical ATV71 configuration settings:

Menu	Submenu	Parameter	Value
[SIMPLY START] (SIM-)	-	[Macro configuration] (CFG)	[M.handling] (HdG) (Refer note1 below)
[COMMAND] (CtL-)	-	[Profile] (CHCF)	[Not separ.] (SIM)
[COMMAND] (CtL-)	-	[Ref.1 channel] (Fr1)	[CANopen] (CAn)
[SETTINGS] (SEt-)	-	[Acceleration] (ACC)	[0.1] (0.1)
[SETTINGS] (SEt-)	-	[Deceleration] (dEC)	[0.1] (0.1)
[SETTINGS] (SEt-)	-	[Low speed] (LSP)	[0] (0)
[APPLICATION FUNCT.] (FUu-)	[BRAKE LOGIC CONTROL] (bLC-)	[Brake assignment] (bLC)	[R2] (r2)
[APPLICATION FUNCT.] (FUu-)	[BRAKE LOGIC CONTROL] (bLC-)	[Movement type] (bst)	[Traveling] (HOR)
[APPLICATION FUNCT.] (FUu-)	[BRAKE LOGIC CONTROL] (bLC-)	[Brake Release time] (brt)	[0] (0)
[APPLICATION FUNCT.] (FUu-)	[BRAKE LOGIC CONTROL] (bLC-)	[Brake Engage time] (bet)	[0] (0)
[APPLICATION FUNCT.] (FUu-)	[PRESET SPEEDS] (PSS-)	[2 preset speeds] (PS2)	[No] (no)
[APPLICATION FUNCT.] (FUu-)	[PRESET SPEEDS] (PSS-)	[4 preset speeds] (PS4)	[No] (no)
[APPLICATION FUNCT.] (FUu-)	[PRESET SPEEDS] (PSS-)	[8 preset speeds] (PS8)	[No] (no)
[APPLICATION FUNCT.] (FUu-)	[PRESET SPEEDS] (PSS-)	[16 preset speeds] (PS16)	[No] (no)
[FAULT MANAGEMENT] (FLt-)	[COM. FAULT MANAGEMENT] (CLL-)	[CANopen fault mgt] (COL)	[Freewheel] (YES)

ATV31/312 Configuration

The following table contains typical ATV31/312 configuration settings:

Menu	Submenu	Parameter	Value
Control menu] (CtL-)	–	Mixed mode (CHCF)	Combined (SIM)
Control menu] (CtL-)	–	Configuration reference (Fr1)	Reference from CANopen (CAN)
Settings menu (SEt-)	–	Acceleration ramp time (ACC)	(0 . 1)
Settings menu (SEt-)	–	Deceleration ramp time (dEC)	(0 . 1)
Settings menu (SEt-)	–	Low speed (LSP)	(0)
Application function menu (FU _n -)	Brake control (bLc-)	Brake control configuration (bLC)	(R2)
Application function menu (FU _n -)	Preset speeds (PSS-)	2 preset speeds (PS2)	(n0)
Application function menu (FU _n -)	Preset speeds (PSS-)	4 preset speeds (PS4)	(n0)
Application function menu (FU _n -)	Preset speeds (PSS-)	8 preset speeds (PS8)	(n0)
Application function menu (FU _n -)	Preset speeds (PSS-)	16 preset speeds (PS16)	(n0)
Fault menu (FLt-)	–	[CANopen fault management] (COL)	Freewheel stop (YES)

When the forced local mode setting is used

Menu	Submenu	Parameter	Value
[Control menu] (CtL-)	–	Function access level (LAC)	Access to advanced function (L3)
Application function menu (FU _n -)	Brake control (bLc-)	Brake release time (brt)	(0)
Application function menu (FU _n -)	Brake control (bLc-)	Brake engage time (bEt)	(0)

Section 6.6

Function Block Description - AntiSwayOpenLoop_2

What Is in This Section?

This section contains the following topics:

Topic	Page
AntiSwayOpenLoop_2 - Overview	232
AntiSwayOpenLoop_2 Function Block	234
Input Pin Description - AntiSwayOpenLoop_2	235
Structure Parameter - AntiSwayOpenLoop_2	240
Output Pin Description - AntiSwayOpenLoop_2	245
Instantiation and Usage Example - AntiSwayOpenLoop_2	248
Commissioning Procedure	249

AntiSwayOpenLoop_2 - Overview

Why Use the AntiSwayOpenLoop_2 Function Block?

The AntiSwayOpenLoop_2 function block helps to prevent sway of suspended loads on industrial cranes by calculating an optimal speed profile for horizontal movement of the trolley and/or the bridge. It is suitable for both, manually operated and automatic cranes.

Substantial reduction of load sway considerably increases the comfort of crane operation. It also shortens the time for crane operations and increases its productivity.

When used on an automatic crane the FB co-operates with the positioning system to arrive at a target position with significantly limited residual sway in the load.

This FB is an optimization of the AntiSwayOpenLoop FB. There are several modifications to solve unit issues with the existing library and modify the reaction time of the non-sway movement.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

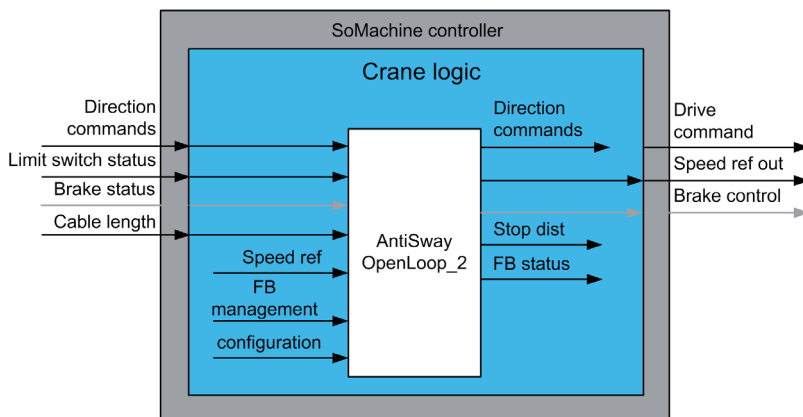
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

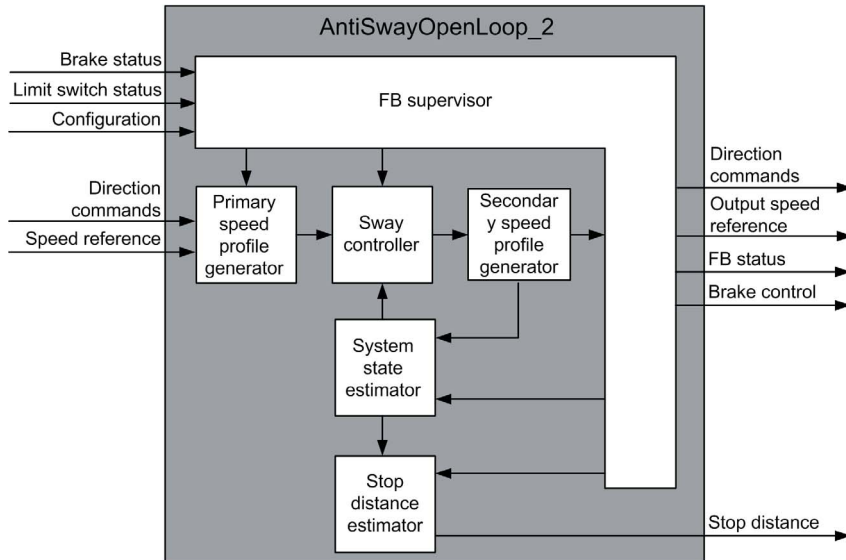
Validate all function block input values before and while the function block is enabled.

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

Functional View

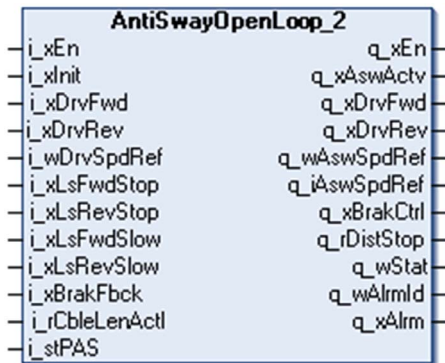


Data Flow Overview



AntiSwayOpenLoop_2 Function Block

Pin Diagram



Function Block Description

Horizontal movement of a crane trolley or bridge induces sway on a suspended load. The AntiSwayOpenLoop_2 function block helps to reduce this induced sway caused by horizontal movement.

The AntiSwayOpenLoop_2 function block is designed for the following:

- Calculate a speed reference profile to correct for sway induced by horizontal movement of the crane.
- Increase the comfort of crane operation.
- Provide an innovative highly efficient Anti-sway control without additional sensors.
- Provide information about the stop distance for use in automatic positioning cranes.

Input Pin Description - AntiSwayOpenLoop_2

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	Anti-sway activation. TRUE: Enables Anti-sway FALSE: Disables Anti-sway, linear ramp Refer to detailed description below this table.
i_xInit	BOOL	This function initializes the internal calculation for the Anti-sway correction. It returns the function to the factory settings (internal variable = 0). The initialization is performed on the rising edge. This function is used after each drive detects the alarm and after each movement which was not requested by the block (Drive detected alarm, Quick Stop, Forced local, Communication alarm...). Initialization of the FB is only possible after the currently active Anti-sway profile is finished. TRUE: Initialization FALSE: No action
i_xDrvFwd	BOOL	Forward command. In FB enabled state starts generation of Anti-sway profile. In FB disabled state starts generation of linear speed profile.
i_xDrvRev	BOOL	Reverse command. In FB enabled state starts generation of Anti-sway profile. In FB disabled state starts generation of linear speed profile.
i_wDrvSpdRef	WORD	Drive speed reference input. Range: 0...6000 Scaling/Unit: RPM (alternatively 0.1 Hz, the unit must be the same as for i_stPAS.wDrvSpdRefMax)
i_xLsFwdStop	BOOL	Stop movement of the trolley/bridge in a forward direction. Movement is stopped using a linear ramp defined in i_stPAS.wDecEmgy with no Anti-sway profile. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE. NOTE: This input is processed in both, the enabled and disabled state of the FB.

Input	Data Type	Description
i_xLsRevStop	BOOL	<p>Stop movement of the trolley/bridge in a reverse direction. Movement is stopped using a linear ramp defined in <code>i_stPAS.wDecEmgy</code> with no Anti-sway profile.</p> <p>NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.</p> <p>NOTE: This input is processed in both, the enabled and disabled state of the FB.</p>
i_xLsFwdSlow	BOOL	<p>Slow-down forward position.</p> <p>NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.</p> <p>NOTE: This input is processed only in enabled state of the FB.</p> <p>Refer to detailed description below this table.</p>
i_xLsRevSlow	BOOL	<p>Slow-down reverse position.</p> <p>NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.</p> <p>NOTE: This input is processed only in enabled state of the FB.</p> <p>Refer to detailed description below this table.</p>
i_xBrakFbck	BOOL	<p>Brake feedback. TRUE: Brake released FALSE: Brake engaged</p> <p>NOTE: This input is processed in both, the enabled and disabled state of the FB.</p> <p>NOTE: If this input is not used, <code>i_stPAS.wBrakDly</code> parameter must be set to TRUE.</p> <p>Refer to detailed description below this table.</p>
i_rCbleLenAct1	REAL	<p>Actual cable length. Range: 0.5...60 Scaling/Unit: 1 m Refer to detailed description below this table.</p>
i_stPAS	PAS	<p>Configuration parameter, set during commissioning phase. Refer to Sub-Structured Description (<i>see page 240</i>).</p>

i_xEn

This input activates the Anti-sway function.

Activation is possible only when the output speed of the Anti-sway FB equals zero. If the output speed is higher than zero, the Anti-sway function is only activated once it reaches zero.

The Anti-sway function can be de-activated regardless of the actual speed reference value.

If the FB is disabled it generates a speed profile using linear acceleration and deceleration ramps defined by `wAswAccDsb1` and `wAswDecDsb1`. This profile can be used to control the motor speed without changing the setting of acceleration and deceleration ramps in the drive (e.g. when there is no more place in rPDOs to write the parameters).

However it is preferable to use the ramping of the drive, if it is possible.

NOTE: It is not possible to activate Anti-sway during the movement because the calculation of the pendulum position is only active when the Anti-sway function is enabled. If a movement with activated Anti-sway function starts while the load sways, this initial sway is not suppressed by the function.

WARNING

UNCOMPENSATED INITIAL SWAY

- Do not start the movement until the load is at rest and motionless.
- Do not engage the Anti-sway function until all initial load movement has ceased.

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

i_xLsFwdSlow

This signal is used to assist the slow down at the slow down limit switch in a forward direction. This input does not modify the speed reference. It sets shorter ramps in order to slow the movement down in a shorter distance while still using the Anti-sway function. The speed reference required in the slow area must be given to the input `i_wDrvSpdRef` from an external source (e.g. `SpeedSelect` function block).

i_xLsRevSlow

This signal is used to assist the slow down at the slow down limit switch in the reverse direction. This input does not modify the speed reference. It sets shorter ramps in order to slow the movement down in a shorter distance while still using the Anti-sway function. The speed reference required in the slow area must be given to the input `i_wDrvSpdRef` from an external source (e.g. `SpeedSelect` function block).

i_xBrakFbck

This is the feedback brake status. A TRUE value indicates the brake is released. The feedback is needed to start the movement without the brake engaged.

If there is a command to run present (`i_xDrvFwd` or `i_xDrvRev`) and other conditions to start the movement are fulfilled, the corresponding direction output command (`q_xDrvFwd` or `q_xDrvRev`) is set to TRUE immediately. (in order to switch the drive to RUN state) Ramp generation starts after `i_xBrakFbck` becomes TRUE.

The input is important for smoothness of movement. The internal ramp generator starts the ramp only after this input is set to TRUE. If it is TRUE all the time, ramping begins before the motor is fluxed and brake is open, which in combination with a 0.1 s ramp on the drive results in a steep ramp at the beginning of movement. This is especially important for motors with a longer fluxing time.

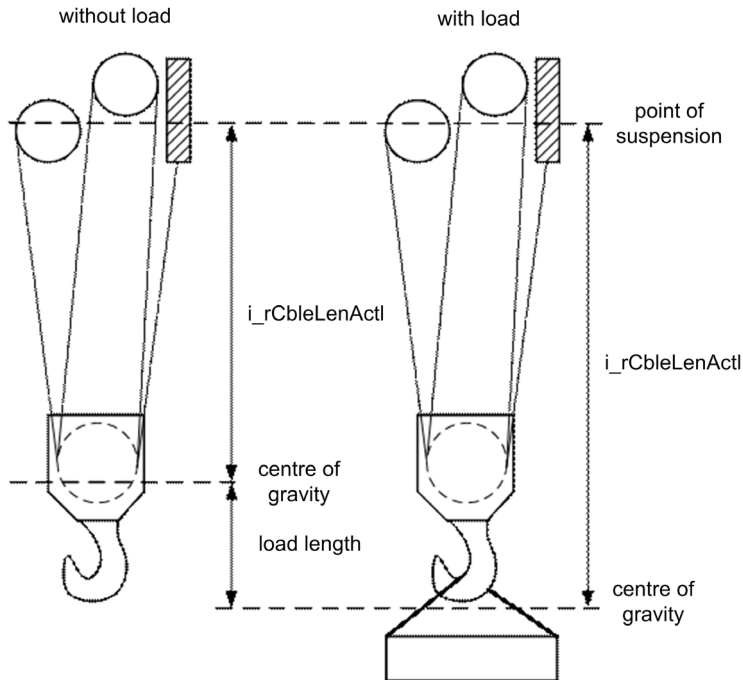
Using the contact of the brake contactor is acceptable but not optimal, especially for brakes which take a long time to open. (If the brake needs 200 ms to release and the task time is 50 ms, the ramp runs for 4 cycles before the brake releases).

Alternatively, if there is no information on the state of the brake, see `i_stPAS.wBrakDly` parameter.

i_rCbleLenActl

This is the actual cable length from the point where the rope is attached to the drum (or the average position for complex systems) to the approximate center of gravity of the system (the system consists of the hook, load and also rope, but the rope can usually be neglected) in meters (coming from the `CableLength` function block or similar). The load length must be added to the cable length beforehand.

Correct setting of `i_rCbleLenActl` input:



Structure Parameter - AntiSwayOpenLoop_2

i_stPAS

A structure of the data type PAS containing configuration parameters.

For example, a declaration of structure variable with initial values:

```
VAR
```

```
struct_instance: data_type := (element1 := XX, ement2 := YY);
```

```
END_VAR
```

Parameter	Data Type	Description
wDrvSpdRefMax	WORD	Maximum speed reference for the drive. Range: 1..6000 (default is 1500) Scaling/Unit: RPM (alternatively 0.1 Hz, the unit must be the same as i_wDrvSpdRef)
rSpdLinMax	REAL	Maximum linear speed corresponding to maximum speed reference. Range: 0.0001...5 (default is 1) Scaling/Unit: m/s (accuracy 0.001 m/s)
wAswAccDsbl	WORD	Acceleration ramp time between zero and i_stPAS.wDrvSpdRefMax for operation without Anti-sway. When the Anti-sway is disabled, the system accelerates using a linear ramp. Range: 5...300 (default is 50) Scaling/Unit: 0.1 s
wAswDecDsbl	WORD	Deceleration ramp time between i_stPAS.wDrvSpdRefMax and zero for operation without Anti-sway. When the Anti-sway is disabled, the system accelerates using a linear ramp. Range: 5...300 (default is 50) Scaling/Unit: 0.1 s
wAswAccStrt	WORD	Acceleration ramp time between zero and i_stPAS.wDrvSpdRefMax for operation with Anti-sway enabled, before the Anti-sway action starts (while the speed reference is under i_stPAS.rAswSpdStrt). The FB uses this parameter to calculate a ramp for smooth transfer between pre-Anti-sway and Anti-sway speed profile. Range: wAswRampLim...300 (default is 100) Scaling/Unit: 0.1 s

Parameter	Data Type	Description
wAswDecStrt	WORD	Deceleration ramp time between <code>i_stPAS.wDrvSpdRefMax</code> and zero for operation with Anti-sway enabled, before the Anti-sway action starts (while the speed reference is under <code>i_stPAS.rAswSpdStrt</code>). The FB uses this parameter to calculate a deceleration ramp analogic to pre-Anti-sway acceleration ramp. Range: <code>wAswRampLim</code> ...300 (default is 100) Scaling/Unit: 0.1 s
wAswRampLim	WORD	Maximum allowed acceleration for Anti-sway. Range: 5...300 (default is 20) Scaling/Unit: 0.1 s Refer to detailed description below this table.
wDrvDecEmgy	WORD	Emergency deceleration ramp time. This sets a short emergency ramp, used in case of an emergency stop without the Anti-sway function. This ramp is based on the stop limit switch and in the case that the feedback signal from a brake goes to FALSE during a movement. Range: 1...100 (default is 5) Scaling/Unit: 0.1 s
wBrakDly	WORD	Brake open delay. Range: 0...5000 (default is 0) Scaling/Unit: ms NOTE: This function is active in both, enabled and disabled state of the FB. Refer to detailed description below this table.
rCoefFrct	REAL	Friction coefficient. Range: 0...1 (default is 0.2) Refer to detailed description below this table.
rAswSpdStrt	REAL	Speed threshold for activation of Anti-sway function. Range: 0...100 (default is 25% of maximum speed) Scaling/Unit: 1% Refer to detailed description below this table.
rAswSpdEnd	REAL	Speed threshold for stopping the movement with active Anti-sway function. Anti-sway movement is finished when the output speed reference is below <code>i_stPAS.rAswSpdEnd</code> for longer than <code>i_stPAS.wAswTimeEnd</code> . Range: 0...20 (default is 1% of maximum linear speed) Scaling/Unit: 1%

Parameter	Data Type	Description
wAswTimeEnd	WORD	Time of speed below rAswSpdEnd before deactivation of Anti-sway. This defines how long the output speed reference must remain below i_stPAS.rAswSpdEnd before finishing the movement. Range: 0...5000 (default is 200) Scaling/Unit: 1 ms
wSmplRate	WORD	Controller cycle time. This is the period of execution of the Anti-sway FB. Too short a cycle time results in loss of information, because of over-sampling. Too long a cycle time results in loss of information due to under-sampling. Both cases degrade the Anti-sway performance. NOTE: A sampling rate between 40 and 100 ms is recommended. NOTE: AntiSwayOpenLoop_2 FB must run with a fixed execution period to work correctly. Range: 30...200 (default is 50, corresponding task must be periodic) Scaling/Unit: 1 ms
wCalcDistTime	WORD	Time reserved for stop distance calculation. Range: 0...wSmplRate (default is 0) Scaling/Unit: 1 ms Refer to detailed description below this table.
xOptimRampDecEn	BOOL	Enables additional stop ramp optimization. Refer to detailed description below this table.

NOTE: Unlike the acceleration and deceleration time parameters on the Altivar that are between zero and nominal speed, the ramp time values of Anti-sway FB are calculated between zero and maximum speed. Therefore the ramp times need to be set accordingly when the maximum speed does not equal the nominal speed of the drive to maintain the required ramp slope.

wAswRampLim

This parameter defines the steepest ramp the Anti-sway function is allowed to use in order to correct the sway. Lower values of this parameter makes the correction more aggressive and allow shortening of the acceleration and deceleration phase which improves load handling for the operator (it is easier to estimate a 2 m stop distance than a 4 m and therefore it is possible to arrive to a target position with much higher precision).

The disadvantage is that the stress of the crane structure is increased.

It is essential to find the right compromise during commissioning.

`q_wStat` bit 6 indicates whether an additional stop distance optimization is possible with the current combination of `i_stPAS.wAswRampLim` and `i_rCbleLenAct1`. If it is not possible, the `i_stPAS.wAswRampLim` value can be lowered in order to allow for the optimization.

Stop distance optimization is beneficial mainly with longer cable lengths that approach or exceed 10 m. For shorter cable lengths, the effect is not that significant and the normal stopping distances are acceptable without optimization.

wBrakDly

This is an alternative to the feedback signal from a brake (`i_xBrakFbck`). If the brake feedback signal is not available and the `i_stPAS.wBrakDly` value is not zero, an internal timer is used to enable the speed profile generation instead.

If there is a command to run present (`i_xDrvFwd` or `i_xDrvRev`) and other conditions to start the movement are fulfilled, the corresponding direction output command (`q_xDrvFwd` or `q_xDrvRev`) is set to TRUE immediately. (in order to switch the drive to RUN state) Ramp generation starts after the time specified by `i_stPAS.wBrakDly` has elapsed.

rCoeffFrct

This is the friction coefficient during the movement which is used to adjust the mathematical model to be closer to the behavior of the real crane. A high value for the friction coefficient results in higher damping of the calculated sway in the mathematical model. Use a low value for a hoist with a simple pulley block and a higher value for complex pulley blocks.

Example: set 0.1 for a single rope hoist without much friction, 0.9 for a hoist with a very complex pulley block, 12 thick ropes and a high friction.

rAswSpdStrt

This is the speed threshold for activation of the Anti-sway function (given in % of maximum speed).

When set to zero, Anti-sway is active from the beginning of a movement. As it corrects the sway continuously, this results in a less predictable behavior during short movements.

An alternative is to set this threshold higher than the lowest speed of the crane. This allows for short, accurate movements without Anti-sway.

wCalcDistTime

This defines how much time the FB is allowed to spend on recursive calculation of the required stop distance for one cycle. A zero value disables the calculation completely.

The calculation can run over several controller cycles. A low value for `i_stPAS.wCalcDistTime` saves processing time, but increases the number of controller cycles needed to finish the calculation. This decreases accuracy in the result especially during acceleration and deceleration.

A high value of `i_stPAS.wCalcDistTime` speeds up the calculation but requires more processing time. This must be taken in account especially when more than one `AntiSwayOpenLoop_2` function blocks are executed simultaneously on a slow controller.

NOTE: The function is designed for use with automatic cranes. On manually controlled cranes it can be left disabled to save processing time.

xOptimRampDecEn

This parameter enables additional stop distance optimization. It allows a considerable shortening of a stop distance. Performance of this function depends on the maximum allowed steepness of acceleration and deceleration ramp (see description of `i_stPAS.wAswRampLim`). Sharper acceleration and deceleration ramps results in shorter stop distances and less time but at the cost of possibly higher mechanical stress to the crane structure.

Output Pin Description - AntiSwayOpenLoop_2

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the Anti-sway function. FALSE: Disables the Anti-sway function.
q_xAswActv	BOOL	Anti-sway correction active. TRUE: Anti-sway active FALSE: Anti-sway inactive
q_xDrvFwd	BOOL	Forward command. Connect this output to the forward command input of the Altivar_Control FB. TRUE: Forward FALSE: Not forward
q_xDrvRev	BOOL	Reverse command. Connect this output to the reverse command input of the Altivar_Control FB. TRUE: Reverse FALSE: Not reverse
q_wAswSpdRef	REAL	Unsigned speed reference output. Use this output to control the drive via function blocks from the Altivar Library (ATV). Range: 0...stPAS.wDrvSpdRefMax Scaling/Unit: 1 RPM
q_iAswSpdRef	REAL	Signed speed reference. Range: -stPAS.wDrvSpdRefMax ...stPAS.wDrvSpdRefMax Scaling/Unit: 1 RPM
q_xBrakCtrl	BOOL	Brake control. TRUE: Open brake FALSE: Close brake Refer to detailed description below this table.
q_rDistStop	REAL	Calculated stop distance. Range: -3.4e+38...3.4e+38 Scaling/Unit: m Refer to detailed description below this table.
q_wStat	WORD	Status register. Range: 0...512 Refer to detailed description below this table.
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (<i>see page 247</i>). Range: 0...15

Output	Data Type	Description
q_xAlrm	BOOL	Detected alarm bit. TRUE: Alarm detected FALSE: No alarm detected NOTE: Alarm reset automatically after the cause of the alarm is removed.

q_xBrakCtrl

This is a brake control output. Normally the brake is controlled by the brake logic of the drive. If this is so, leave this output unconnected. If an accurate Anti-sway control is also required for short movements, it is possible to let the Anti-sway FB control the brake directly. The output goes TRUE as soon as a command in the forward/reverse direction is given, the output is set to FALSE when the movement finishes.

NOTE: When brake logic is used on an ATV31, ATV312 or ATV71 running in open loop motor control, the drive closes the brake when the actual speed drops to/below the brake engage frequency. Since Anti-sway has to lower the speed to zero on short movements before applying the sway correction, closing of the brake interferes with the Anti-sway action.

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use parameter Brake engage delay (TBE) inside the Brake logic control menu of the Altivar drive.

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

Setting of the Brake engage delay to value higher than 0 may cause the drive to lock up in DCB state during reversing of direction of movement. If you need to delay closing of the brake, use q_xBrakCtrl instead.

q_rDistStop

This is the calculated distance necessary to stop the movement using the Anti-sway function. When this function is enabled, in setting i_stPAS.wCalcDistTime to a value higher than zero, the function block continuously calculates the necessary stopping distance. The calculation is spread over several controller cycles.

This function is used on automatic cranes to determine the point where the stopping ramp has to start in order to arrive at the target position.

q_wStat

Status Bit	Description
0	When TRUE, the FB is in alarm state due to a detected error.
1	When TRUE, Anti-sway output reference is stabilized (1% of the linear speed).
2	When TRUE, FB is enabled.
3	When TRUE, Anti-sway function is active.
4	Status of output <code>q_xDrvFwd</code> .
5	Status of output <code>q_xDrvRev</code> .
6	When TRUE, additional stop distance optimization is available with actual setting of <code>i_stPAS.wAswRampLim</code> .

Notifications

This output indicates what alarm has been reported.

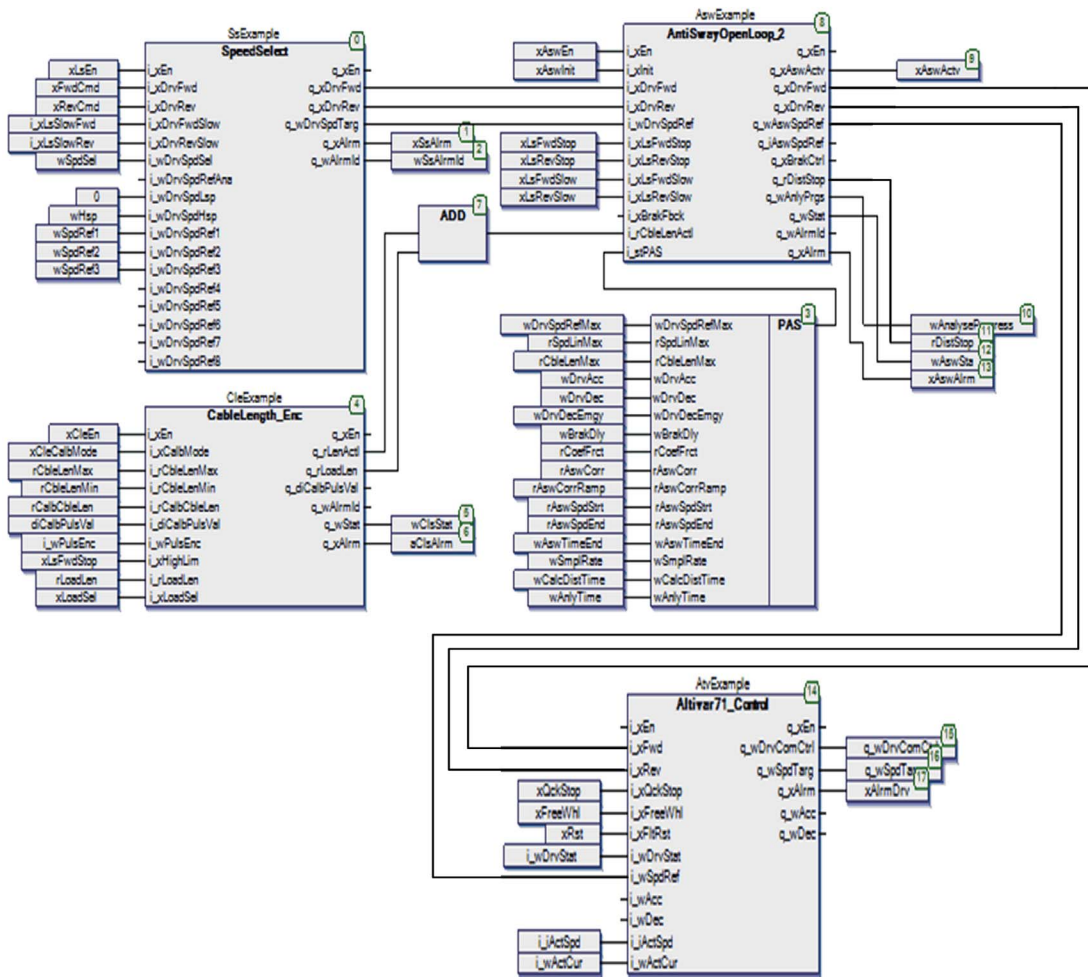
Alarm resets automatically after the cause of the alarm is removed.

Alarm Bit <code>q_wAlrmId</code>	Description
0	One or more of the FB inputs is/are out of range.
1	One or more of the input structure elements is/are out of range.
2	Tried to initialize the Anti-sway, but the Anti-sway speed profile is currently active -> output of the Anti-sway.
3	Command to enable Anti-sway given while previous profile has not finished yet.

Instantiation and Usage Example - AntiSwayOpenLoop_2

Instantiation and Usage Example

This figure shows an instantiation example of the AntiSwayOpenLoop_2 function block:



Commissioning Procedure

Pre-requisites

1. The drives must be set to follow the speed reference accurately. When possible, use the vector control. Set slip compensation to 100%.
Set the acceleration and deceleration parameters in the drive to 0.1 s.
2. A compromise setting must often be made when setting the system for usage with Anti-sway. Anti-sway requires a smooth speed profile, especially during the stopping phase, to achieve the highest possible quality of sway control. Because most motors used on trolleys and bridges are run in an open loop, the engineer commissioning the crane must decide whether to set the LSP and brake engage frequency to the value of motor's slip or to 0 Hz.
Setting the LSP to 0 Hz improves smoothness of the movement but increases the risk of losing control over the axis when the motor cannot apply sufficient torque on low or zero speed. This must be kept in mind during the commissioning.
3. If a high precision Anti-sway function is required, set the LSP to 0 Hz to prevent LSP from interfering with the Anti-sway profile.
4. Configure the behavior of the brake. The brake logic of the drive can interfere with Anti-sway movements when the speed profile reaches the brake engage frequency before the end of the movement. This can happen during short movements or when the acceleration of the crane movement is higher than the acceleration of the pendulum (the cable is long and acceleration time of trolley/bridge is short).
In this case disable the brake logic and let the Anti-sway FB control the brake directly using the `q_xBrakCtrl` output.

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use parameter Brake engage delay (TBE) inside the Brake logic control menu of the Altivar drive.

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

Setting of the Brake engage delay to value higher than 0 may cause the drive to lock up in DCB state during reversing of direction of movement. If you need to delay closing of the brake, use `q_xBrakCtrl` instead.

5. If there is a risk of the load moving when the trolley/bridge is at or close to zero speed, use DC braking to hold the motor in position.

6. Calculate or measure linear speed of movement for each axis. Make sure that you receive a correct value of cable length as explained for the `i_rCbleLenAct1` description (a laser range finder is a very useful tool when commissioning the Anti-sway FB). A correct value of load length and linear speed are essential for the precision of the Anti-sway function especially when moving at high speeds and accelerating/decelerating on steep ramps.

Commissioning Procedure of the `AntiSwayOpenLoop_2` Function Block

The `AntiSwayOpenLoop_2` FB contains an open loop control algorithm. Therefore it does not require feedback from the drive. However, when the drive enters an error state during a movement, the Anti-sway control must be stopped and the function block must be reinitialized.

This section describes the configuration of the FB using the `i_stPAS` input structure and the influence of these parameters on the behavior of the crane. Only specific parameters which require extended explanation in addition to the basic description of `i_stPAS` are mentioned here.

1. `i_stPAS.wAswAccDdbl`, `i_stPAS.wAswDecDdbl`. The FB uses these parameters as an acceleration and deceleration ramp when the Anti-sway function is disabled. Set them to values suitable for operation without Anti-sway.
2. `i_stPAS.wAswAccStrt`, `i_stPAS.wAswDecStrt`. The optimal setting is usually a long ramp. It helps to protect the load from sway during short movements below `i_stPAS.rAswSpdStrt` when Anti-sway is enabled.

Start with a value of 10 s.

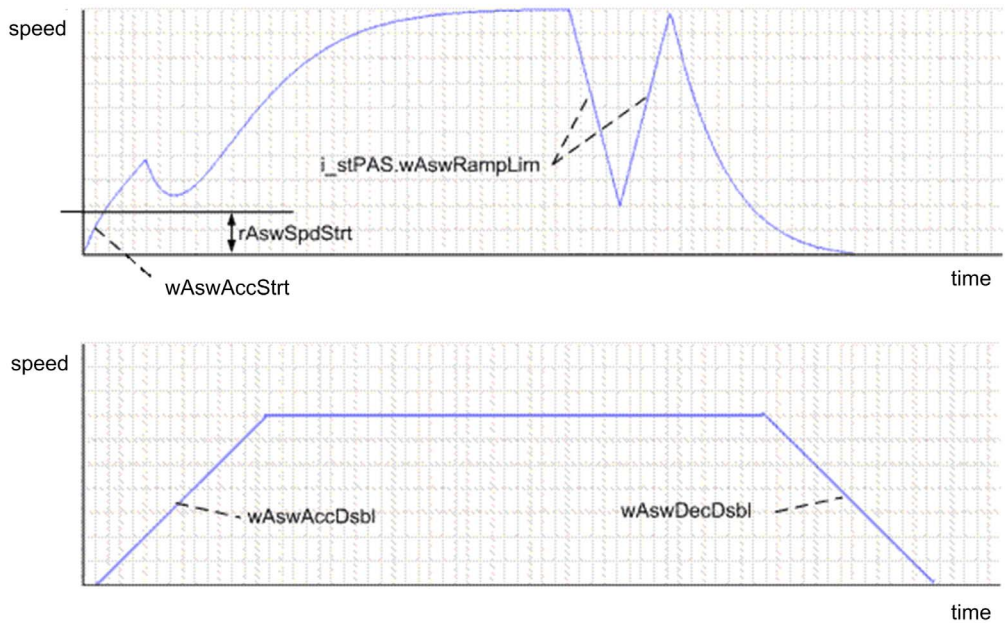
Default value:

```
i_stPAS.wAswAccStrt:=100  
i_stPAS.wAswDecStrt:=100
```

3. `i_stPAS.wAswRampLim`. Defines the steepest ramp the Anti-sway function is allowed to use in order to correct the sway. Low values for this parameter makes the correction more aggressive and allows shortening of the acceleration and deceleration phase, which improves the load handling for the operator (it is easier to estimate a 2 m stop distance than a 4 m and therefore it is possible to arrive at the target position more precisely). The disadvantage is that the stress on the crane structure increases. It is essential to find the right compromise during commissioning.

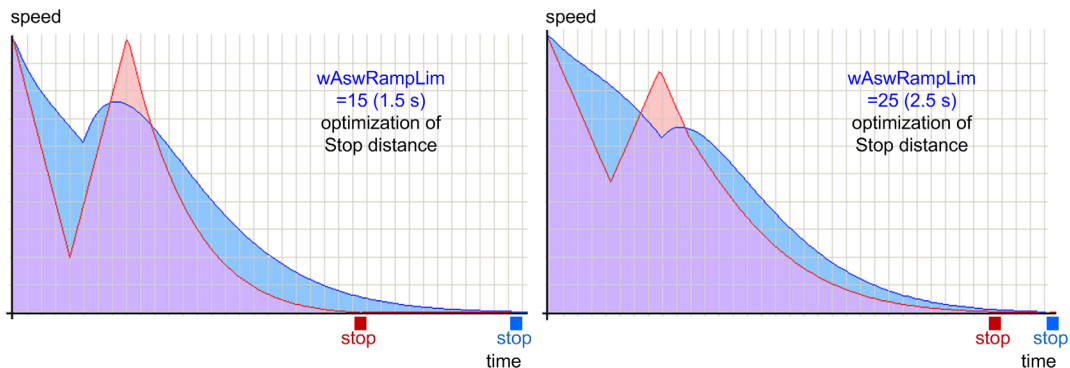
`q_wStat`. Bit 6 gives information on whether an additional stop distance optimization is possible with the current combination of `i_stPAS.wAswRampLim` and `i_rCbleLenAct1`. If it is not possible, the `i_stPAS.wAswRampLim` value can be lowered in order to allow for the optimization.

The stop distance optimization is beneficial mainly with longer cable lengths close to or over 10 m. For shorter cable lengths the effect is not that significant and the stopping distances are acceptable without optimization.



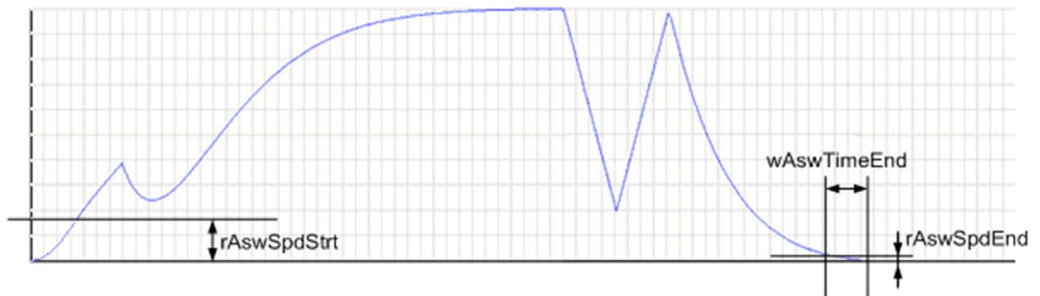
4. $i_stPAS.wDrvDecEmgy$. The value of this parameter must be small enough to stop the crane safely when a stop limit switch is reached or a contact signal from a brake is lost. (Default value: 5)
5. $i_stPAS.xOptimRampDecEn$. Enables additional stopping distance optimization. It allows a considerable shortening of the stopping distance. The performance of this function depends on maximum allowed steepness of the acceleration and deceleration ramp (see description of $i_stPAS.wAswRampLim$). A steeper acceleration and deceleration ramp results in shorter stop distance and time but at the cost of higher mechanical stress on the crane structure.

The following figures show an optimized (red) and non-optimized (blue, rounder form) Anti-sway speed profile for a crane with 8 m cable length moving at 1 m/s with various acceleration and deceleration ramps:



6. `i_stPAS.rAswSpdStrt`. Defines the speed threshold at which Anti-sway starts to correct the sway (% of maximum speed). Micro movements are often a concern when commissioning an Anti-sway function. The majority of operators request Anti-sway to be disabled during short movements. If this is the case, set the `i_stPAS.rAswSpdStrt` threshold above the micro/jog speed.
7. `i_stPAS.wAswAccStrt`, `i_stPAS.wAswDecStrt`. It is also better if the ramps at low speed are longer than usual to minimize the sway and increase the positioning precision. (see description of step 2 `i_stPAS.wAswAccStrt`, `i_stPAS.wAswDecStrt`).
8. `i_stPAS.rAswSpdEnd`. Defines the speed threshold to stop the Anti-sway correction at the end of a movement (% of maximum speed). Setting this value too high stops the movement earlier but decreases the performance. Setting it too low increases the stopping time. (Default value: 1)
9. `i_stPAS.wAswTimeEnd`. Defines how long must be the speed output reference below `i_stPAS.rAswSpdEnd` to consider the movement finished.

Description of $i_stPAS.rAswSpdStrt$, $i_stPAS.rAswSpdEnd$ and $i_stPAS.wAswTimeEnd$ parameters:



10. Be aware that an Anti-sway speed profile is generally longer (in both time and distance) than a linear acceleration speed profile. Therefore you must have in place limit switches on the crane adjusted to allow safe stopping/slowdown with all possible speeds and cable lengths.
11. As a safeguard, the application should check the difference between speed reference and actual speed and stop the movement if the difference is too great. This is not implemented in the FB since it is application dependent.

Section 6.7

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The actual motor speed does not correspond to speed reference.	Preset speeds configuration interferes with speed reference from controller.	Disable preset speeds in application functions setting of the drive.
The actual motor speed does not correspond to speed reference.	Limit switch configuration interferes with speed reference from controller.	Disable limit switch function in application functions setting of the drive.
The actual motor speed or read actual speed do not correspond to speed reference.	PDO mapping is inconsistent.	Make sure that both speed reference and actual speed values have the same unit (Hz or RPM).
α_xAlrm of the FB is TRUE.	One or more of the input parameters are out of range.	Verify that the input parameters are within the specified range.
FB generates a correct speed profile, but the actual motor speed lags behind the speed reference, quality of Anti-sway is low.	Acceleration and deceleration ramps inside the drive are not set to 0.1 s.	Set the ramps in drive to 0.1 s.
Speed profile is generated and followed by the motor, but the quality of Anti-sway is low.	Some of the input parameters are incorrect.	Check the cable length and maximum linear speed parameters.

Part V

Diagnostic Coverage

Chapter 7

Diagnostic Coverage: Increase Diagnostic Coverage of Used Control System

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	Functional Overview	258
7.2	Architecture	260
7.3	Function Block Description	262
7.4	Pin Description	263
7.5	Quick Reference Guide	271

Section 7.1

Functional Overview

Functional Overview

Why Use the DiagnosticCoverage Function Block?

The function block compares application signature and firmware version with a configured value. It also watches executions of sub-programs, actual cycle time of a cyclic task and provides interface for cross-checking of 2 controllers. Internally it tests integrity of variable memory and correctness of boolean and floating point operations.

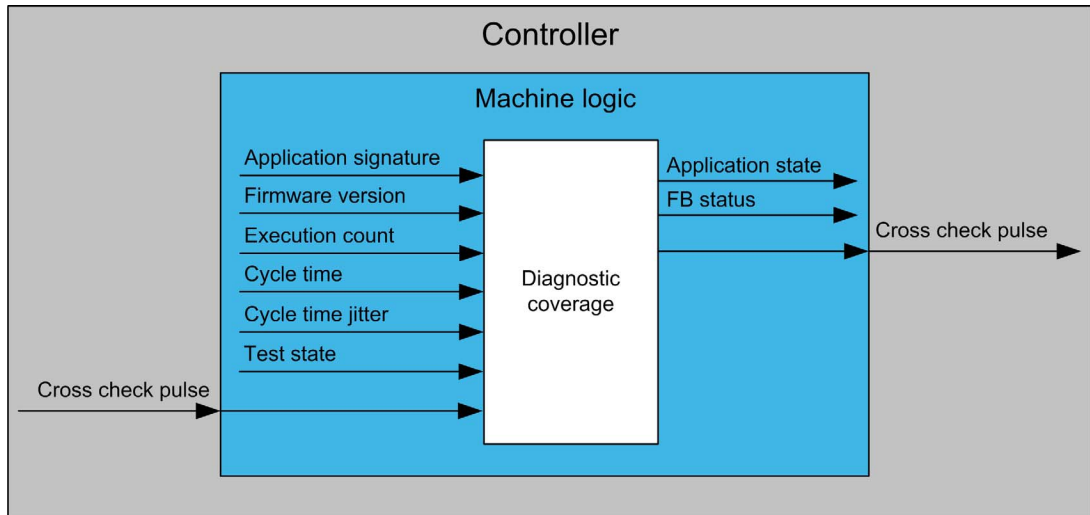
This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

 WARNING
UNINTENDED EQUIPMENT OPERATION
Validate all function block input values before and while the function block is enabled.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Solution with the DiagnosticCoverage Function Block

The DiagnosticCoverage FB helps increase the diagnostic coverage of the applied control system.

Functional View



Section 7.2

Architecture

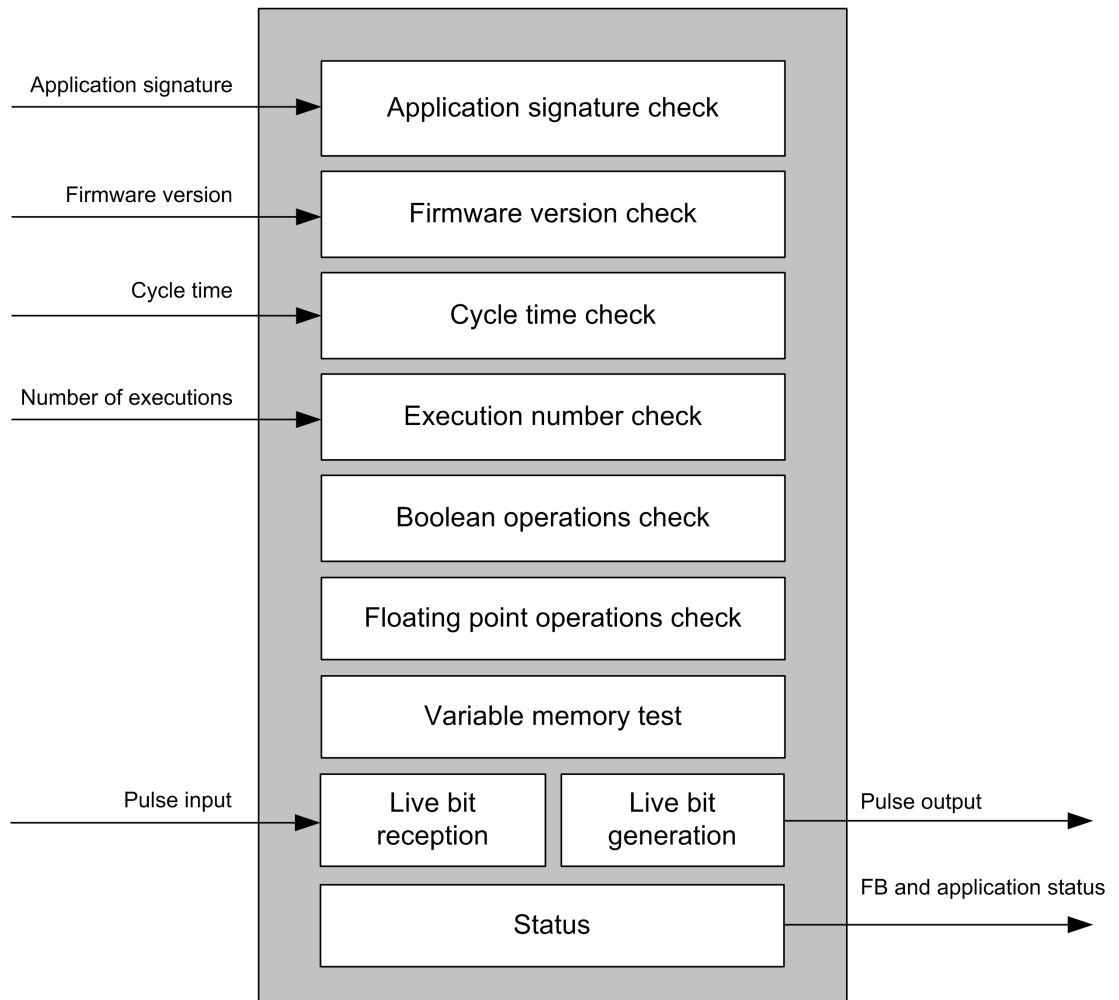
Architecture

System Requirements

For any details of the system requirements, refer to the chapter System Requirements (*see page 25*).

Data Flow

Internal structure of the DiagnosticCoverage FB

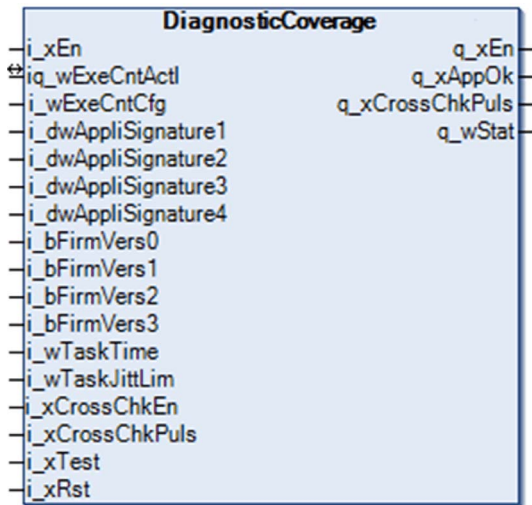


Section 7.3

Function Block Description

DiagnosticCoverage Function Block

Pin Diagram



Section 7.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input/Output Pin Description	264
Input Pin Description	265
Output Pin Description	269

Input/Output Pin Description

Input/Output Pin Description

Input/Output	Data Type	Description
iq_wExeCntAct1	WORD	This pin is used for testing the number of executions of sub-programs within one cycle of a main cyclic task. Connect a global variable to this input. The global variable must be incremented by one within each called sub-program. The number of sub-programs must correspond to the value of i_wExeCntCfg. The DiagnosticCoverage FB may be executed at any position in the program. Range: 0...65535

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below.
i_wExeCntCfg	WORD	Number of program executions Range: 0...65535 Refer to detailed description below.
i_dwAppliSignature1	DWORD	Application signature 1 Range: 0...4294967295 Refer to detailed description below.
i_dwAppliSignature2	DWORD	Application signature 2 Range: 0...4294967295 Refer to detailed description below.
i_dwAppliSignature3	DWORD	Application signature 3 Range: 0...4294967295 Refer to detailed description below.
i_dwAppliSignature4	DWORD	Application signature 4 Range: 0...4294967295 Refer to detailed description below.
i_bFirmVers0	BYTE	Firmware version 0 Range: 0...255 Refer to detailed description below.
i_bFirmVers1	BYTE	Firmware version 1 Range: 0...255 Refer to detailed description below.
i_bFirmVers2	BYTE	Firmware version 2 Range: 0...255 Refer to detailed description below.
i_bFirmVers3	BYTE	Firmware version 3 Range: 0...255 Refer to detailed description below.
i_wTaskTime	WORD	Task execution time. Enter the value of task time in milliseconds corresponding to the value of the cyclic task in which the instance of Diagnostic coverage is executed (the master task). Range: 0...65535 Scaling/Unit: ms

Input	Data Type	Description
i_wTaskJittLim	WORD	Limit of task execution jitter. Range: 0...65535 Scaling/Unit: ms Refer to detailed description below.
i_xCrossChkEn	BOOL	Enables cross checking. Refer to detailed description below.
i_xCrossChkPuls	BOOL	Cross checking pulse input. Refer to detailed description below.
i_xTest	BOOL	Test state input. Refer to detailed description below.
i_xRst	BOOL	Resets status of the FB on rising edge and will reset the q_xAppOK to TRUE if the conditions are fulfilled.

I_xEn

When TRUE, enables the FB. When FALSE, the FB enters a fallback state:

q_xEn	FALSE
q_xAppOK	FALSE
q_xCrossCheckPuls	FALSE
q_wStat	0

I_wExeCntCfg

This input contains a value corresponding to the number of execution of all programs and sub-programs in the main cyclic task. The FB monitors the number of executions on the pin iq_wExeCntAct1. If the number of executions (increments of the value connected to iq_wExeCntAct1) is not equal to i_wExeCntCfg, the q_xAppOk output is set to FALSE.

i_dwAppliSignature1...4

These inputs are used for testing of application signature. Each application has a unique application signature consisting of four 32 bit DWORD values. Connect retained DWORD variables to these inputs and after downloading the application, write values found in PLC_R.i_dwAppliSignature1..4 to these retained variables.

NOTE: Do not write hard-coded values to the application; each change of the application changes also its signature.

Location of application signature information

Watch 1	
Expression	
[-]	MyController.Application.PLC_R
[-]	i_dwAppliSignature1
[-]	i_dwAppliSignature2
[-]	i_dwAppliSignature3
[-]	i_dwAppliSignature4

i_bFirmVers0...3

These inputs are used for checking of controller firmware version. The firmware version consists of 4 byte values. The actual firmware version can be found in the PLC_R structure in an array of BYTE.PLC_R.i_bFirmVers[0..3].

Location of controller firmware version information

Watch 1	
Expression	
[-]	MyController.Application.PLC_R
[-]	i_byFirmVersion
[-]	i_byFirmVersion[0]
[-]	i_byFirmVersion[1]
[-]	i_byFirmVersion[2]
[-]	i_byFirmVersion[3]

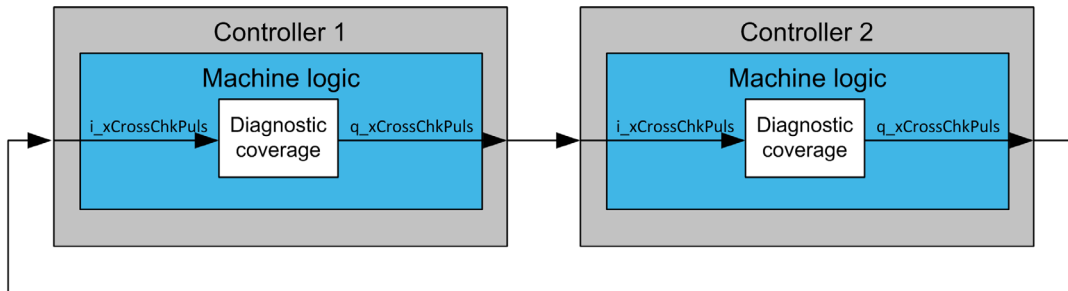
i_wTaskJittLim

This input contains the maximum allowed value of execution time jitter. If the cyclic task execution time exceeds the sum of i_wTaskTime and i_wTaskJittLim, q_xAppOk output is set to FALSE. The protection only checks if the actual execution period is longer than configured, it will not react if the actual cycle time is shorter than configured.

i_xCrossChkEn

TRUE on this input enables cross checking function for mutual checking of 2 controllers. The FB generates a pulse train with 400 ms period and 50% duty cycle at q_xCrossChkPuls and expects a pulse train on the input i_xCrossChkPuls. The output of the FB should be connected to a physical (solid state) output of the controller and the input of the FB to an input of the controller. Two controllers running an instance of the FB with mutually connected inputs and outputs are necessary for cross checking.

Connection of controllers for cross-checking

**i_xCrossChkPuls**

Input for the pulse train coming from a second controller. The function is enabled when `i_xCrossChkEn` is TRUE and gets activated after the first rising edge on this input. Activation of the function on the first rising edge prevents false alarms when one of the controllers involved in cross checking is switched on with a delay.

Before activation is the `q_xAppOk` FALSE. Once activated the `q_xAppOk` becomes TRUE and reverts to FALSE state only after 500 ms without incoming pulses.

i_xTest

TRUE on this input bypasses checking of application signature, firmware version information, execution count, cycle time test and mutual cross checking of 2 controllers. Boolean operation test and floating point operation test remain active. It allows for a simple way of keeping the `q_xAppOk` TRUE during commissioning period.

The input keeps its effect for 5 hours, if it stays TRUE for more than 5 hours, it is ignored.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	This output mirrors the value of i_xEn input variable. TRUE: Enables the function block. FALSE: Disables the function block.
q_xAppOk	BOOL	Application OK flag. The output is TRUE when all the checks performed by the FB are successful. This input must be interlocked by a logical AND with all parts of a program influencing physical movement of any axis on the machine. It can be for example connected to freewheel or Quick stop inputs of variable speed drive device blocks.
q_xCrossChkPuls	BOOL	Output for a pulse train to be physically connected to a second controller. It generates a pulse train with 400 ms period and 50% duty cycle.
q_wStat	WORD	Status of the application. Range: 0..65535 Refer to q_wStat below.

q_wStat

This output contains the status of the FB.

Bit Position	Status Description Represented by Bit Position
Bit 0	i_dwAppliSignature1 is not entered correctly.
Bit 1	i_dwAppliSignature2 is not entered correctly.
Bit 2	i_dwAppliSignature3 is not entered correctly.
Bit 3	i_dwAppliSignature4 is not entered correctly.
Bit 4	i_bFirmVers0 is not entered correctly.
Bit 5	i_bFirmVers1 is not entered correctly.
Bit 6	i_bFirmVers2 is not entered correctly.
Bit 7	i_bFirmVers3 is not entered correctly.
Bit 8	Unsuccessful bit rotation test. Check the controller.
Bit 9	Unsuccessful floating point operation test. Check the controller.
Bit 10	Unsuccessful volatile memory test. Check the controller.
Bit 11	Unsuccessful retained memory test. Check the controller.

Bit Position	Status Description Represented by Bit Position
Bit 12	Number of executions of sub-programs per cycle does not correspond to <code>i_wExeCntCfg</code> .
Bit 13	Execution period length exceeded defined threshold. The execution time was longer than <code>i_wTaskTime + i_wTaskJittLim</code> .
Bit 14	Unsuccessfully cross checking with a second controller. Live pulse signal lost.

Section 7.5

Quick Reference Guide

What Is in This Section?

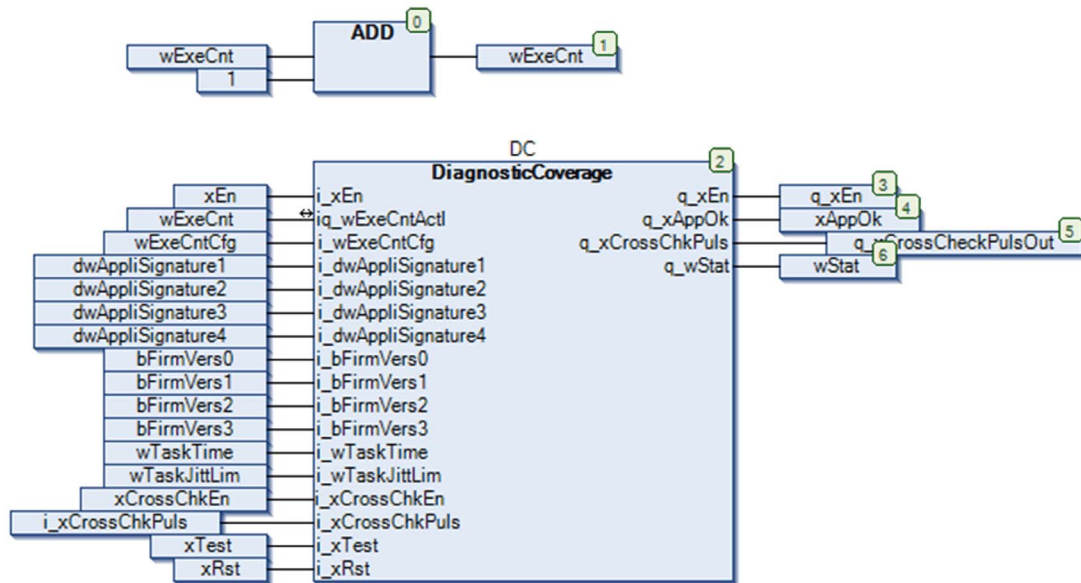
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	272
Commissioning Procedure	273
Troubleshooting	274

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the DiagnosticCoverage function block:



Commissioning Procedure

Commissioning Procedure of the DiagnosticCoverage Function Block

1. Instantiate the FB in a program called in a main cyclic task and enable it.
2. Create a global variable, connect it to the input/output pin `iq_wExeCntAct1` and increment it by one in every called program and sub-program. Then connect a value corresponding to the number of increments to the input `i_wExeCntCfg`.
3. Create 4 retained variables and connect them to the inputs `i_dwAppliSignature1` to `i_dwAppliSignature4`. The values must be in variables since it is not possible to hard-code them in the application.
4. Connect variables or hard-coded values with the used controller firmware version to the inputs `i_bFirmVers0` to `i_bFirmVers3`.
5. Connect a variable or a hard-coded value corresponding to actual setting of task cycle to the input `i_wTaskTime` and the allowed value of maximum allowed jitter to `i_wTaskJittLim`.
6. Verify that the watchdog of cyclic task is enabled in the controller configuration.
7. If cross-checking of 2 controllers is required
 - set `i_xCrossChkEn` to TRUE,
 - assign `i_xCrossChkPuls` to a digital input and
 - `q_xCrossChkPuls` to a solid state output.

Then do the same on the other controller and connect them both input to output.

8. If variable speed drives are not used, then interlock (logical AND) the output `q_xAppOk` with free wheel or Quick stop inputs of device functions of variable speed drives or direction commands.
9. Download the application, read the application signature from **PLC_R** structure and write the values of 4 DWORD variables to the retained variables in your application.
10. Power-cycle the controller or use the `i_xRst` input.
11. The `q_xAppOk` output should be TRUE at this point. If it is not, then verify the `q_wStat` output to find out why it is not TRUE.
12. During commissioning you can set the input `i_xTest` to TRUE to bypass checking of application signature, firmware version information, execution count, cycle time test and mutual cross checking of 2 controllers. TRUE on this input will suppress the checking for 5 hours.

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The output <code>q_xAppOk</code> is FALSE.	One or more of the checks are unsuccessful.	Verify <code>q_wStat</code> for more information.

Part VI

Electronic Potentiometer

Chapter 8

ElectronicPotentiometer: Set Speed and Direction of a Movement

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
8.1	Functional Overview	278
8.2	Architecture	279
8.3	Function Block Description	282
8.4	Pin Description	286
8.5	Quick Reference Guide	292

Section 8.1

Functional Overview

Functional Overview

Functional Description

The `ElectronicPotentiometer` FB offers an effective approach to setting an arbitrary analog speed reference and direction of movement of a variable speed drive using 3 or 4 digital inputs.

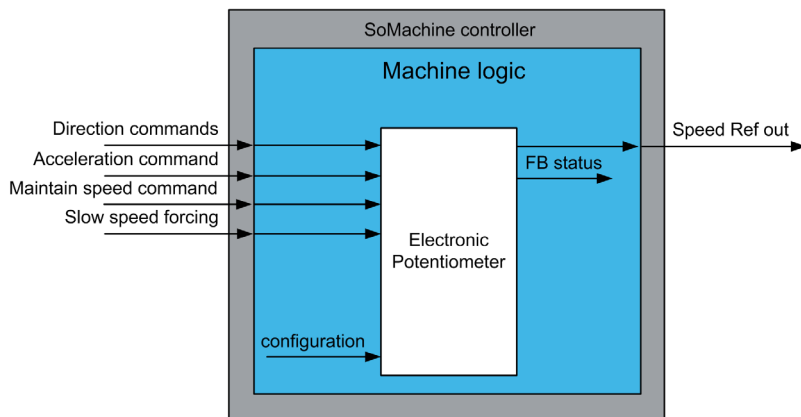
Why Use the `ElectronicPotentiometer` Function Block?

This FB replaces the built-in motor potentiometer function of ATV71, 31 or 312 when you want to set the speed reference of the drive by a controller application.

Solution with the `ElectronicPotentiometer` Function Block

The `ElectronicPotentiometer` function block allows an operator of a machine to set speed and direction of a movement.

Functional View



Section 8.2

Architecture

What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	280
Software Architecture	281

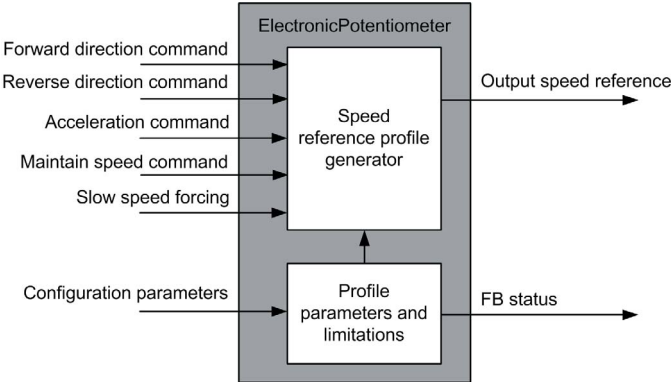
Hardware Architecture

Hardware Architecture Overview

The FB can run in a standard environment, see section Hardware Architecture (*see page 45*).

Software Architecture

Data Flow Overview

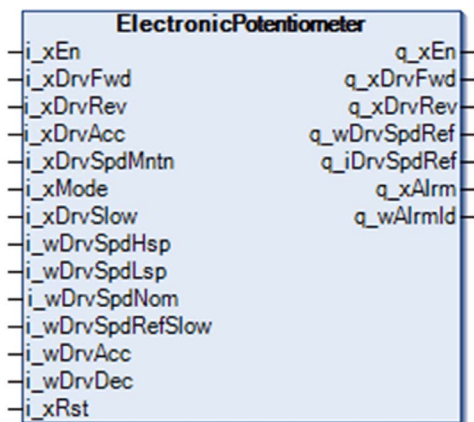


Section 8.3

Function Block Description

ElectronicPotentiometer Function Block

Pin Diagram



Profile Generation

The FB generates an output speed profile based on configuration parameters and control inputs.

Direction command in either forward `i_xDrvFwd` or reverse `i_xDrvRev` direction causes the output speed reference to ramp to low speed `i_wDrvSpdLsp`. Low speed can equal zero for drives running in full vector control. If the drive is not running in a full vector control, the value of `i_wDrvSpdLsp` should be set to the same value as the LSP parameter inside the drive.

There are 2 modes of operation supported by the FB:

- three-wire control (uses 3 inputs `i_xDrvFwd`, `i_xDrvRev` and `i_xDrvAcc`)
- four-wire control (uses the 3 inputs plus `i_xDrvSpdMntn` in addition)

In **three-wire control mode**

- `i_xMode` is FALSE
- The output speed reference increases when `i_xDrvAcc` is TRUE and the direction command in actual direction is also TRUE.

- The increase is limited by `i_wDrvSpdHsp`, and if `i_xDrvSlow` is TRUE, then it is limited by `i_wDrvSpdRefSlow`.
- The purpose of `i_xDrvSlow` is to decrease the output speed reference independently of operator commands if a slow-down limit switch is activated.
- When the `i_xDrvAcc` input returns to FALSE state but the active direction command stays TRUE, the actual value of output speed reference is held.
- When `i_wDrvSpdHsp` changes during a profile generation and becomes lower than the output speed reference, the output speed reference is ramped down.
- If the active direction command input is set to FALSE, the output speed reference gradually ramps down to zero speed using the ramp defined by `i_wDrvDec`.

In four-wire control mode

- `i_xMode` is TRUE
- The output speed reference increases when `i_xDrvAcc`, `i_xDrvSpdMntn` and the direction command in actual direction are all TRUE.
- The increase is limited by `i_wDrvSpdHsp`, and if `i_xDrvSlow` is TRUE, then it is limited by `i_wDrvSpdRefSlow`.
- The purpose of `i_xDrvSlow` is to decrease the output speed reference independently of operator commands, if a slow-down limit switch is activated.
- When the `i_xDrvAcc` input returns to FALSE state but the active direction and `i_xDrvSpdMntn` command stay TRUE, the actual value of output speed reference is held.
- When `i_wDrvSpdHsp` changes during a profile generation and becomes lower than the output speed reference, the output speed reference is ramped down.
- If the active direction command input is set to FALSE, the output speed reference gradually ramps down to zero speed using the ramp defined by `i_wDrvDec`.
- If the `i_xDrvSpdMntn` is set to FALSE, but the active direction is still TRUE, the output speed reference gradually ramps down to `i_wDrvSpdLsp` using the ramp defined by `i_wDrvDec`.

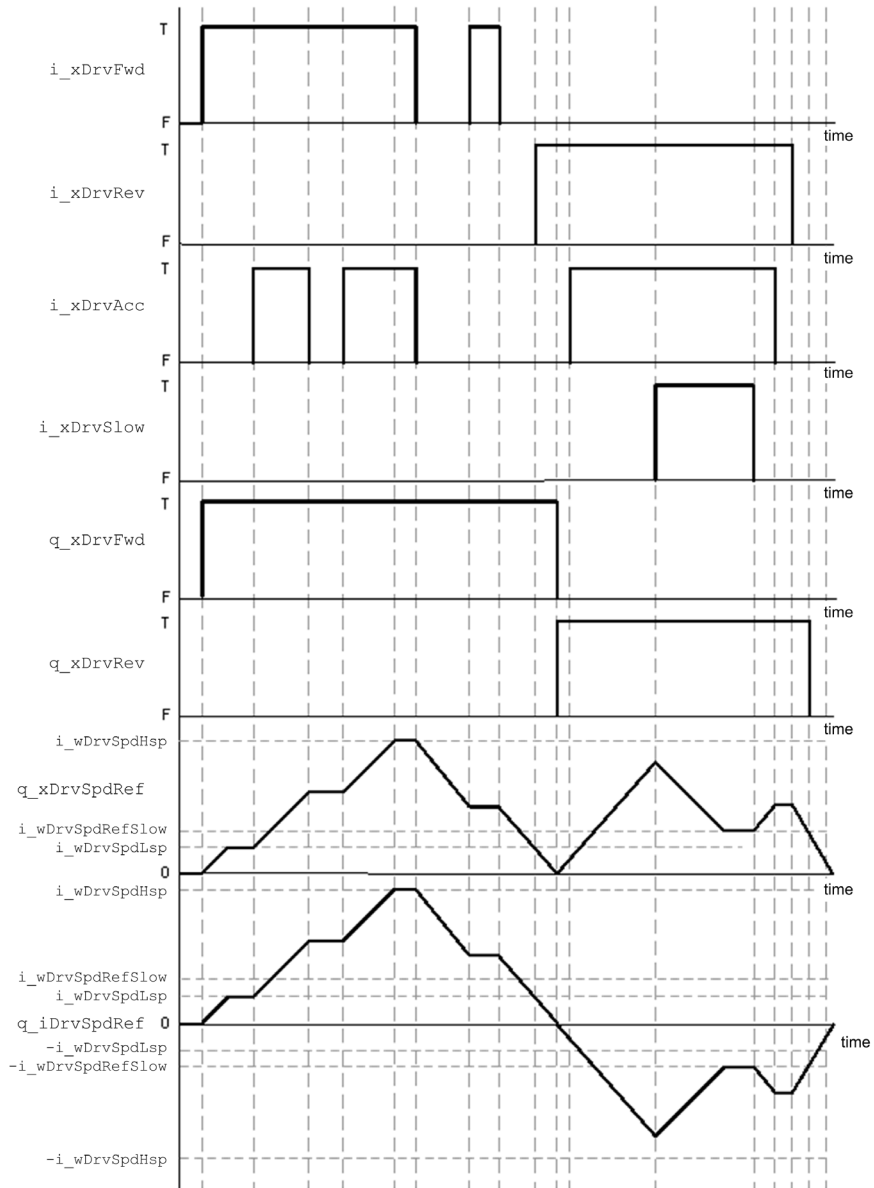
Both modes

There are 2 speed output reference outputs.

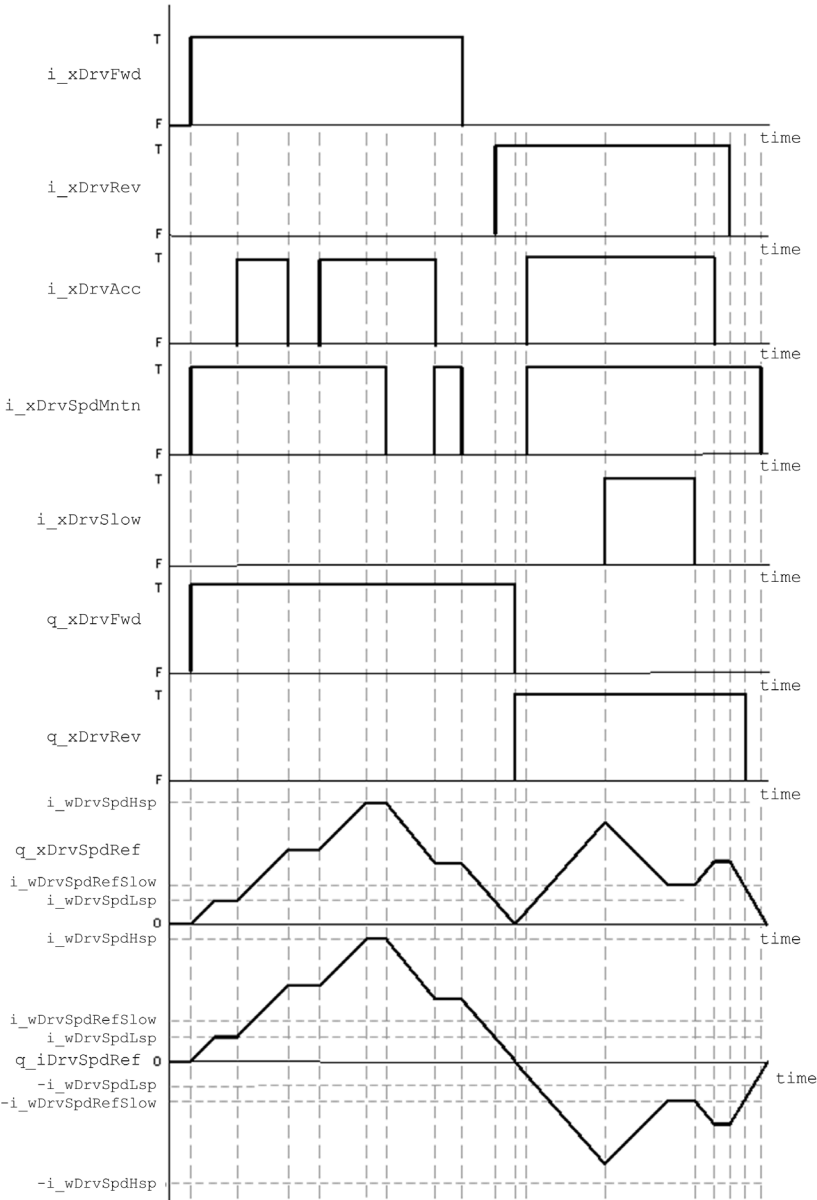
- The unsigned `q_wDrvSpdRef` is positive for both directions.
- The signed `q_iDrvSpdRef` is positive in forward and negative in reverse direction.

NOTE: Use the unsigned `q_wDrvSpdRef` output when working with `Altivar71_Control`, `Altivar32_Control` and `Altivar31_Control` device FBs.

Timing Diagram for Three-Wire Mode



Timing Diagram for Four-Wire Mode



Section 8.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	287
Output Pin Description	290

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	This input activates the Electronic potentiometer function. When the function is deactivated, the output $q_wDrvSpdRef$ is set to zero and alarm reporting is suppressed. TRUE: Enables the function block. FALSE: Disables the function block.
i_xDrvFwd	BOOL	Drive forward run command. TRUE: Forward FALSE: Not forward Refer to detailed description below this table.
i_xDrvRev	BOOL	Drive reverse run command. TRUE: Reverse FALSE: Not reverse Refer to detailed description below this table.
i_xDrvAcc	BOOL	Acceleration command. When this input is TRUE together with $i_xDrvFwd$ or $i_xDrvRev$, the output $q_wDrvSpdRef$ increases towards $i_wDrvSpdHsp$. When the input is FALSE, the output $q_wDrvSpdRef$ is held as constant long as one of the direction commands, either $i_xDrvFwd$ or $i_xDevRev$, is TRUE.
i_xDrvSpdMntn	BOOL	Command to maintain current speed. Refer to detailed description below this table.
i_xMode	BOOL	Mode selection: TRUE: Four-wire mode FALSE: Three-wire mode
i_xDrvSlow	BOOL	Command to slow down. TRUE: Slow speed forced FALSE: Speed limited by Hsp Refer to detailed description below this table.
i_wDrvSpdHsp	WORD	High speed value. Is the maximum value the $q_wDrvSpdRef$ can reach. Range: 0..6000 Scaling/Unit: RPM
i_wDrvSpdLsp	WORD	Low speed value. Range: 0.. $i_wDrvSpdHsp$ Scaling/Unit: RPM

Input	Data Type	Description
i_wDrvSpdNom	WORD	Nominal speed of the used motor. It is used for calculation of acceleration and deceleration ramp. Range: 0...6000 Scaling/Unit: RPM
i_wDrvSpdRefSlow	WORD	Defines the speed reference when i_xDrvSlow is TRUE. It is supposed to serve as a reduced speed when the controlled axis enters a slow-down area. Range: i_wDrvSpdLsp...i_wDrvSpdHsp Scaling/Unit: RPM
i_wDrvAcc	WORD	Acceleration time. Range: 0...9999 Scaling/Unit: 0.1 s Refer to detailed description below this table.
i_wDrvDec	WORD	Deceleration time. Range: 0...9999 Scaling/Unit: 0.1 s Refer to detailed description below this table.
i_xRst	BOOL	Alarm reset. Rising edge on this input resets a detected alarm in case the cause of the alarm has been removed.

i_xDrvFwd

When the output q_wDrvSpdRef is zero and i_xDrvFwd is set TRUE, the q_wDrvSpdRef ramps up to the value of i_wDrvSpdLsp.

When the speed reference in forward direction is higher than i_wDrvSpdLsp, i_xDrvFwd is TRUE and i_xDrvAcc is FALSE, q_wDrvSpdRef is held.

When i_xDrvFwd is FALSE, q_wDrvSpdRef decreases towards zero using deceleration ramp defined by i_wDrvDec.

See the timing diagram for three-wire mode (*see page 284*) and for four-wire mode (*see page 285*) for a visual depiction of the function.

i_xDrvRev

When the output q_wDrvSpdRef is zero and i_xDrvRev is set TRUE, the q_wDrvSpdRef ramps up to the value of i_wDrvSpdLsp.

When the absolute value of speed reference in the reverse direction is higher than i_wDrvSpdLsp, i_xDrvRev is TRUE and i_xDrvAcc is FALSE, q_wDrvSpdRef is held.

When i_xDrvRev is FALSE, q_wDrvSpdRef decreases towards zero using deceleration ramp defined by i_wDrvDec.

See the timing diagram for three-wire mode (*see page 284*) and for four-wire mode (*see page 285*) for a visual depiction of the function.

i_xDrvSpdMntn

Command to maintain the actual speed reference. This input is used in four-wire mode. In three-wire mode it does not have any effect.

If this input is TRUE together with `i_xDrvAcc` and a direction command, the FB is ramping up the speed reference in requested direction.

If this input is TRUE but `i_xDrvAcc` is FALSE and direction command in actual direction of movement is present, the FB keeps the speed reference.

If this input is FALSE and direction command in actual direction of movement is present, the FB ramps the speed reference towards `i_wDrvSpdLsp`.

i_xDrvSlow

If this input is TRUE, the FB forces a slow speed reference defined by `i_wDrvSpdRefSlow`.

If the current value of `q_wDrvSpdRef` is higher than that, the output speed reference gets ramped down using a deceleration ramp defined by `i_wDrvDec`.

This input can be used with a slow-down signal from a limit switch.

i_wDrvAcc

This input sets the acceleration ramp time between zero and nominal motor speed defined by `i_wDrvSpdNom`. The resolution of this parameter is 0.1 s. The acceleration ramp time should be equal to or longer than the ramp configured in the Altivar variable speed drive.

i_wDrvDec

This input sets the deceleration ramp time between nominal motor speed defined by `i_wDrvSpdNom` and zero. The resolution of this parameter is 0.1 s. The deceleration ramp time should be equal to or longer than the ramp configured in the Altivar variable speed drive.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_xDrvFwd	BOOL	Forward command output. TRUE: Forward command FALSE: No command Refer to detailed description below this table.
q_xDrvRev	BOOL	Reverse command output. TRUE: Reverse command FALSE: No command Refer to detailed description below this table.
q_wDrvSpdRef	WORD	Unsigned output speed reference. Range: 0...6000 Scaling/Unit: RPM
q_iDrvSpdRef	INT	Signed output speed reference. Range: -6000...+6000 Scaling/Unit: RPM
q_xAlrm	BOOL	TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Alarm identification Range: 0...63 Refer to Notifications (<i>see page 291</i>).

q_xDrvFwd / q_xDrvRev

Usage of this output is optional. It is not necessary if a command to the drive is handled by other FB (HoistPositionSync, AntiCrab) or if the deceleration time parameter set in the drive is higher or equal to `i_wDrvDec`.

Notifications

Identification of the alarm, if `q_xAlrm` is TRUE.

Alarm Bit <code>q_wAlrmId</code>	Description
0	<code>i_wDrvSpdHsp</code> is out of range.
1	<code>i_wDrvSpdLsp</code> is higher than <code>i_wDrvSpdHsp</code> .
2	<code>i_wDrvSpdNom</code> is out of range.
3	<code>i_wDrvSpdRefSlow</code> is higher than <code>i_wDrvSpdHsp</code> or lower than <code>i_wDrvSpdLsp</code> .
4	<code>i_wDrvAcc</code> is out of range.
5	<code>i_wDrvDec</code> is out of range.

Section 8.5

Quick Reference Guide

What Is in This Section?

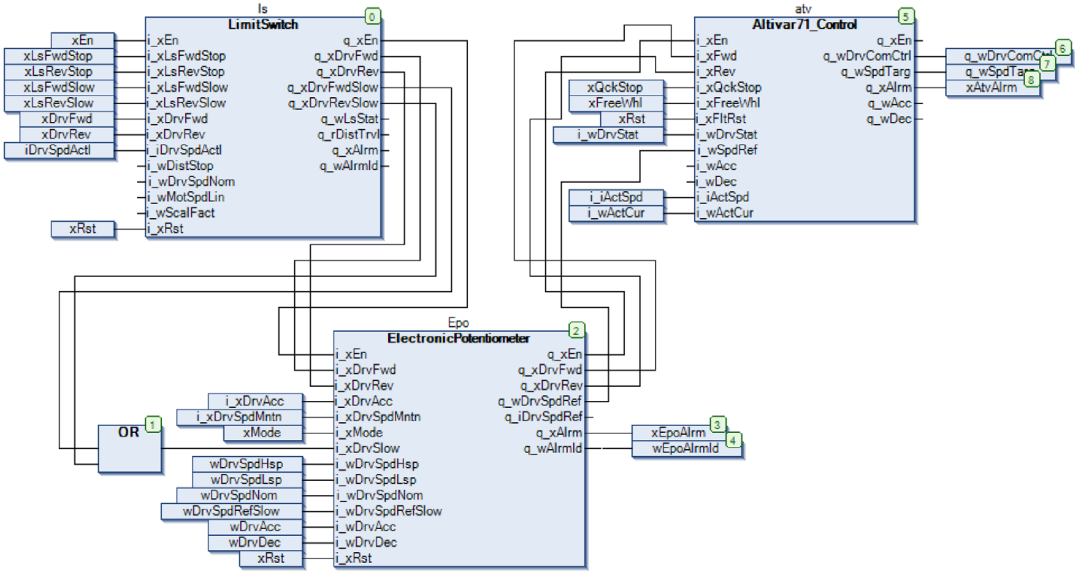
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	293
Commissioning Procedure	294
Troubleshooting	295

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the ElectronicPotentiometer function block:



Commissioning Procedure

Commissioning Procedure of the `ElectronicPotentiometer` Function Block

1. Instantiate the FB.
2. Connect the direction and acceleration commands (and speed maintain command in four-wire connection).
3. Connect the `i_xDrvSlow` input to a signal from a limit switch. It is possible to leave this input unconnected if a slow-down limit switch is not used in the system.
4. Nominal speed value given to the FB should match the nominal speed of the drive (compensated for slip if slip compensation is used)
For example, 1500 RPM for 2 pole pair motor, 3000 RPM for 1 pole pair motor.
5. `i_wDrvSpdHsp`, `i_wDrvSpdLsp`, `i_wDrvAcc`, `i_wDrvDec`, should match their counterparts in the setting of Altivar.
6. `q_xDrvFwd` and `q_xDrvRev` can be used to control the `Altivar71_Control`, `Altivar32_Control` or `Altivar31_Control` function block if a short ramp time is set in Altivar configuration parameters.
In this case must be the direction command present until the drive decelerates on a speed profile generated by `ElectronicPotentiometer` function block. If the direction command is removed and the deceleration time configured in Altivar is shorter than deceleration time parameter of `ElectronicPotentiometer` FB, Altivar stops using its internal ramp rather than the one generated by `ElectronicPotentiometer` FB.
7. There are 2 speed reference outputs on the function block. `q_wDrvSpdRef` is an unsigned output which should be used with `Altivar71_Control` or `Altivar31_Control` device function blocks and `q_iDrvSpdRef` that can be used wherever a signed speed reference value is needed.

Troubleshooting

Troubleshooting

Issue	Cause	Solution
q_xAlrm of the FB is TRUE.	One or more of the input parameters are out of range.	Check the input parameters.

Part VII

Grab Control

Chapter 9

GrabControl: Control of a Four Cable Grab

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	Functional Overview	300
9.2	Architecture	302
9.3	Function Block Description	304
9.4	Pin Description	309
9.5	Quick Reference Guide	325

Section 9.1

Functional Overview

Functional Overview

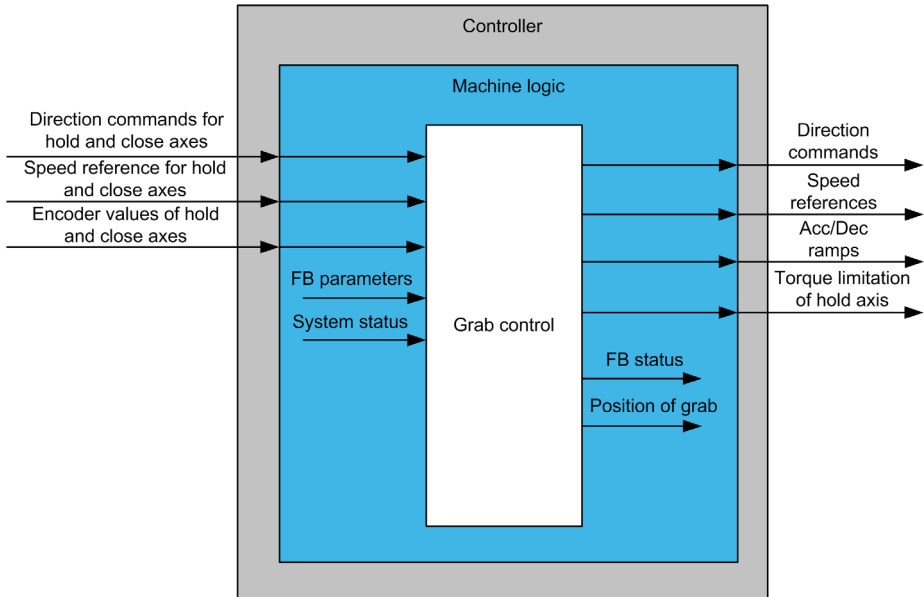
Why Use the GrabControl Function Block?

The function block helps to control hold and close axes of a four cable grab. It targets clamshell and spider grabs and contains an automatic function for closing on stack which simplifies the operation.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

 WARNING
UNINTENDED EQUIPMENT OPERATION
Validate all function block input values before and while the function block is enabled.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Functional View



Section 9.2

Architecture

Architecture

System Requirements

GrabControl requires two Altivar 71 variable speed drives. One of the drives controlling hold axis and the other one controlling the close axis.

Both hold and close axes must be equipped with encoders and both Altivar 71 drives must be configured to run in flux vector control motor control mode. The FB works with incremental encoders. The positions of both axes are stored in non-volatile memory.

The hold and close axes must have identical linear speeds and encoder resolutions.

Vertical movement must be limited in upwards direction by an emergency limit switch.

Environment

The following equipment are required to use GrabControl function block:

- SoMachine controller
- Two Altivar 71 variable speed drives
- Two encoder interface cards for Altivar 71
- Two incremental encoders, typically 1024 PPR (Pulses Per Revolution)
- CANopen cabling

The following equipment is typically used in conjunction with GrabControl function block:

- A two-axis joystick for control of close and hold axes.
NOTE: The function block supports analog and digital joysticks.
- Three buttons for resetting the calibration values and setting calibration of open and closed grab positions (can be done via HMI).
- Additional button(s) on the joystick for extended functions like scraping and rapid opening.
- Two buttons, selector or potentiometer for direct modification of hold torque during closing on stack. These buttons are useful when materials of various densities are transported.
- Optional strain gauge weighing system for enabling automatic closing on stack only when the grab is on the stack.

The hold motor is working in a torque limited mode during automatic closing on stack. If activation of the automatic closing on stack function must be suppressed while the grab is mid-air, the user application must take into account the enabling and disabling of the automatic closing on stack function. The situation often results in a detected over speed fault reported by the Altivar 71 drive.

NOTICE

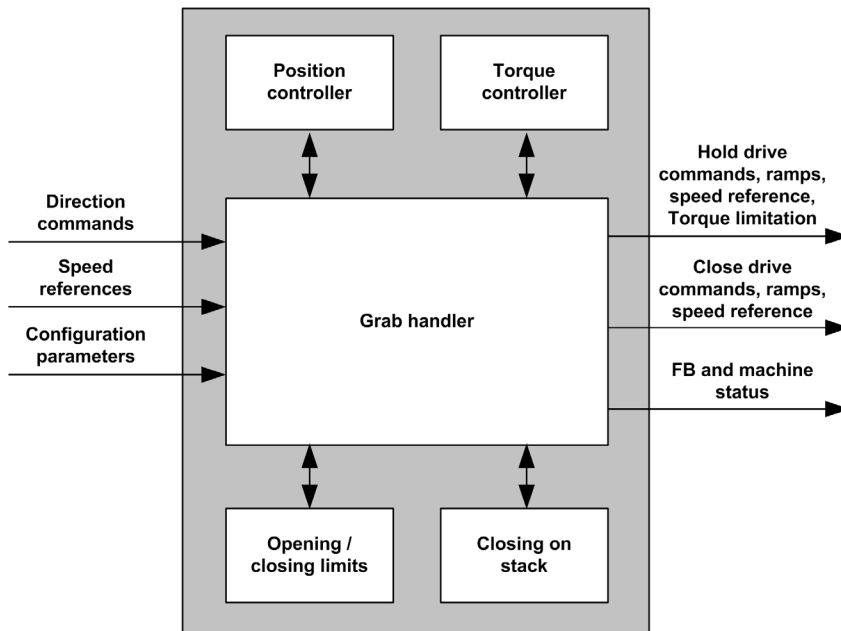
GRAB MECHANISM DAMAGE

Only use the automatic closing on stack function when the grab mechanism is in contact with the material to be moved.

Failure to follow these instructions can result in equipment damage.

Data Flow Overview

The following figure is the internal Structure of the GrabControl function block



Section 9.3

Function Block Description

GrabControl Function Block

Pin Diagram

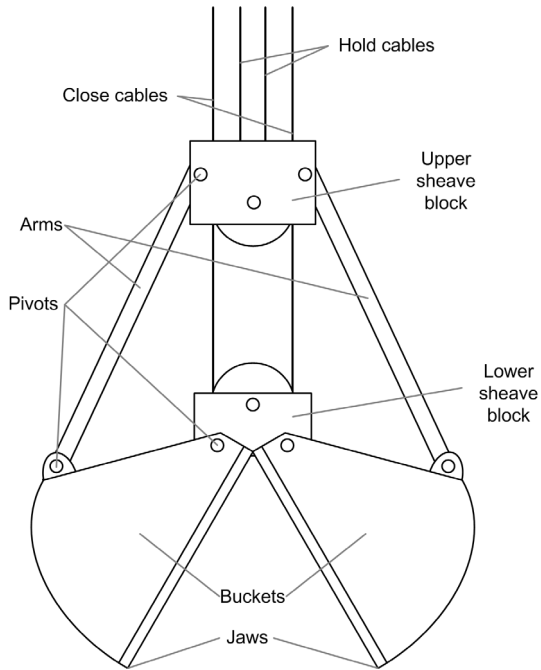
The pin diagram of function block GrabControl:



Targeted Cranes

The function block is suitable for control of four cable grabs. It supports both clamshell and spider grabs and can be used on bridge cranes, gantry cranes, luffing jib cranes and floating grab cranes.

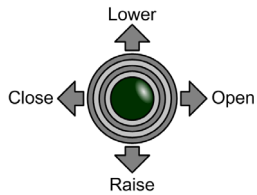
The schematic diagram of clamshell grab is as follows:



Control Interface

The function block is designed for use with a two-axis joystick. It requires one additional button on the joystick for optional functions of rapid opening and/or scraping.

Top-down view of a two-axis joystick :



Controls and command inputs for the function block:

Controls	Command Inputs
Raise	i_xDrvFwdHold
Lower	i_xDrvRevHold

Controls	Command Inputs
Close	i_xDrvFwdClose
Open	i_xDrvRevClose

Alternative control interface may be used as long as the FB receives control commands in the supported form.

Grab Position Calibration

Position of a grab is calculated from relative difference of hold and close axes and values recorded during position calibration. Additional sensors for detecting open and closed state of the grab are not necessary.

The calibration values are used for the soft limit switch function which is limiting opening and closing of the grab.

Perform calibration by moving the grab successively to closed and open positions and setting the corresponding inputs to TRUE. Perform calibration on rising edge of the signal.

You can do the calibration of closed and open positions in an arbitrary order.

Calibration values are stored in a non-volatile memory of the controller.

Do not repeat calibration unless at least one of the axes moves while the controller and/or drives are switched off, while the communication between controller and drives is not working or when there is a problem with encoder signals.

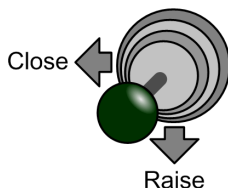
Re-calibrate the position if the mechanics influencing relative positions of the axes changes, for example, cables are replaced.

Reset calibration values using the calibration reset input. Calibrate again the closed and open positions after re-setting the calibration values.

Automatic Closing on Stack

This function automates the process of closing on stack. The purpose of this function is fast and soft closing of the grab followed by smooth transition to upwards movement of the closed and filled grab.

The following figure is the command to activate closing on stack:



It is activated by giving closing and raising commands simultaneously.

During closing, the close axis moves in forward direction closing the grab. The hold axis must provide enough torque to keep hold cables straight. It must also allow the grab to sink in the transported material in order to fill it properly. The amount of sinking can be influenced by configuration parameters.

The hold drive torque is limited during automatic closing on stack which can lead to hold drive losing the load.

The hold motor is working in a torque limited mode during automatic closing on stack. If activation of the automatic closing on stack function must be suppressed while the grab is mid-air, the user application must take into account the enabling and disabling of the automatic closing on stack function. The situation often results in a detected over speed fault reported by the Altivar 71 drive.

NOTICE

GRAB MECHANISM DAMAGE

Only use the automatic closing on stack function when the grab mechanism is in contact with the material to be moved.

Failure to follow these instructions can result in equipment damage.

Torque Sharing and Position Synchronization

When the grab is moving upwards or downwards without closing or opening, the hold and closed axes are synchronized.

When the grab is not closed, the FB synchronizes position of both axes in order to keep their actual position difference.

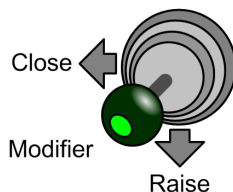
When the grab is closed, the FB synchronizes either the position of both axes or sets target speed of both drives/motors in order to share the load between the two axes. The FB allows selection between these two modes. Torque sharing is preferred in most cases.

Scraping

Scraping is a special case of closing on stack usable with clamshell grabs. Its purpose is to gather remaining material from a flat surface. The grab closes while the jaws stay in contact with the surface.

When the grab is fully closed, the function starts upwards movement of the grab.

The command to activate scraping is as follows:



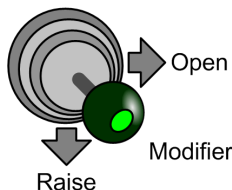
The function is optional and does not need to be configured and used if it is not needed.

An additional modifier button, preferably on the joystick, is needed. The same button may be also used for rapid opening function.

Rapid Opening

This function allows fast opening of the grab in order to release the load as quickly as possible. It is typically used when the grab is emptied while it is in a swing. The feature is usually used on pontoon cranes with a luffing jib.

The following figure is the command to activate rapid opening:



The function is optional and does not need to be configured and used if it is not needed.

An additional modifier button, preferably on the joystick, is needed. The same button can also be used for scraping function.

Hook Mode

Hook mode is an alternative to grab function. While in hook mode, the FB synchronizes the hold and close axes either based on actual relative positions of both axes or modifies speeds of both axes in order to share the load. The FB can be configured to do one or the other.

NOTE: Only the raise and lower commands are available in hook mode. Commands to close or open the grab are ignored.

Maintenance Mode

Use maintenance mode during changing of the cables and other maintenance tasks. Disabling the FB to activate the maintenance mode. In this mode the FB just channels the direction commands, speed references and ramp values through without changing them. Open and closed software limit switch positions are not active.

Section 9.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	310
Structured Variable Description	316
Output Pin Description	321

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	Enables and disables the function block TRUE: Enables the function block FALSE: Disables the function block
i_xDrvFwdHold	BOOL	Forward direction command for hold axis
i_xDrvRevHold	BOOL	Reverse direction command for hold axis
i_xDrvRdyStatHold	BOOL	Ready status of hold drive
i_wDrvSpdRefHold	WORD	Speed reference for hold axis Range: 0...6000 Scaling/Unit: RPM
i_iDrvSpdActlHold	INT	Actual speed of hold motor Range: -32768...32767 Scaling/Unit: RPM
i_iDrvTrqActlHold	INT	Actual torque read from hold drive Range: -32768...32767 Scaling/Unit: 0.1%
i_wEncPosHold	WORD	Encoder position of hold axis Range: 0... 65535 Scaling/Unit: INC
i_xDrvFwdClose	BOOL	Forward direction command for close axis
i_xDrvRevClose	BOOL	Reverse direction command for hold axis
i_xDrvRdyStatClose	BOOL	Ready status of close drive
i_wDrvSpdRefClose	WORD	Speed reference for close axis Range: 0...6000 Scaling/Unit: RPM
i_iDrvSpdActlClose	INT	Actual speed of close motor Range: -32768...32767 Scaling/Unit: RPM
i_iDrvTrqActlClose	INT	Actual torque read from close drive Range: -32768...32767 Scaling/Unit: 0.1%
i_wEncPosClose	WORD	Encoder position of close axis Range: 0...65535 Scaling/Unit: INC
i_xGrabCalbOpen	BOOL	Calibration of open position of the grab
i_xGrabCalbClosed	BOOL	Calibration of closed position of the grab
i_xGrabCalbRst	BOOL	Command to reset calibration values

Input	Data Type	Description
i_xClsOnStackEn	BOOL	Enables automatic closing on stack
i_xTrqShareEn	BOOL	Enables torque sharing TRUE: Enables the function FALSE: Disables the function
i_xScrapEn	BOOL	Enables scraping TRUE: Enables the function FALSE: Disables the function
i_xHookModeEn	BOOL	Enables hook mode TRUE: Enables the function FALSE: Disables the function
i_xRapidOpenEn	BOOL	Enables rapid opening TRUE: Enables the function FALSE: Disables the function
i_stPGC	PGC	Structure of parameters
i_xRst	BOOL	Alarm reset command

i_xEn

When TRUE, enables the function block. When FALSE, the FB enters a fallback state. In fallback state, the function block uses the configuration parameters, target speeds, ramps and direction commands to provide output for both drives. Torque limitation of hold drive is set to 300% (is disabled). Internal position calculation of both axes stays active.

Transition between disabled and enabled state of the FB and vice versa while the crane is running is not restricted but is not recommended.

Disabling the FB can be used as a switch to maintenance mode for changing of cables.

i_xDrvFwdHold

Command for forward (upwards) movement of the hold axis. When there is no command for movement of close axis present at the same time, the forward command for the hold axis is used to move both, hold and close axes synchronously.

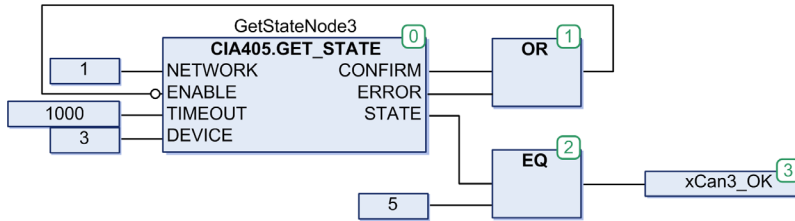
i_xDrvRevHold

Command for reverse (downwards) movement of the hold axis. When there is no command for movement of close axis present at the same time, the reverse command for the hold axis is used to move both, hold and close axes synchronously.

i_xDrvRdyStatHold, i_xDrvRdyStatClose

Inputs of ready status of both hold and close drives. The information can be obtained from bit 1 of status word of Altivar drives. This information has to be interlocked (logical AND) with an information that the drive is communicating on CANopen and is in operational state.

The following figure shows an example of CANopen status check for node3:



If the communication with any of the two drives is interrupted, the FB has to get the information in order to stop the movement.

Failing to interlock the CANopen communication status with the ready status of the drive may result in a dangerous situation when one of the motors stops because of communication interruption between the controller and drive. When the FB does not receive information about the loss of communication, the other motor stays in motion.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Interlock the CANopen communication status with the ready status of the drive.

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

The FB needs to have information about status of both drives. If one of the drives is not ready, the FB does not authorize start of the movement. If one of the ready bits turns FALSE during movement, both motors are stopped.

i_wDrvSpdRefHold, i_wDrvSpdRefClose

Speed reference for hold and close drives. [rpm]

i_iDrvSpdActlHold, i_iDrvSpdActlClose

Actual speed of hold and close motors read from the drives. [rpm]

i_iDrvTrqActlHold, i_iDrvTrqActlClose

Actual torque of hold and close motors read from the drives.

Use the “Motor torque scope” object (SOTR). CANopen index 2002:1B.

The value must be either scaled in the user application to the range -3000...3000 corresponding to -300...300% of nominal motor torque or internally in the GrabControl FB.

To scale the value in your application, divide it by 4.096 and convert it to integer.

To let it scale automatically within the `GrabControl` FB, connect the un-scaled values and set the `i_stPGC.xTrqScopeScale` parameter to `TRUE`.

The values are automatically filtered by a low-pass filter inside the function block. The filtered values are available at the outputs `q_iTrqFltrHold` and `q_iTrqFltrClose`.

i_wEncPosHold, i_wEncPosClose

Encoder positions of hold and close axes. The FB is designed to work with position information read out of Altivar 71 drives running in closed loop. Use the Encoder position object (PUC) in the Altivar 71, CANopen index 201 A:0C.

Encoder value must increase when motor is moving in positive direction.

i_xGrabCalbOpen

Rising edge on this input calibrates position difference corresponding to open state of the grab. You must perform calibration for the soft limit switch function to work. You can perform a new calibration without re-setting the previously calibrated value. If desired position cannot be reached because of previous calibration, you can reset the calibration values by rising edge on input `i_xGrabCalbRst`.

i_xGrabCalbClosed

Rising edge on this input calibrates position difference corresponding to closed state of the grab. Perform calibration for the soft limit switch function to work. You can perform a new calibration without re-setting the previously calibrated value. If desired position cannot be reached because of previous calibration, you can reset the calibration values by rising edge on input `i_xGrabCalbRst`.

i_xGrabCalbRst

Rising edge on this input resets calibration values for open and closed grab positions.

i_xClsOnStackEn

`TRUE` on this input enables automatic closing on stack. Activate this function by applying commands to close and raise the grab simultaneously. When the input is `FALSE`, automatic closing on stack is disabled and the commands to close and raise are acting on their respective axes.

The input must be `TRUE` to start the automatic closing on stack. It does not need to stay `TRUE` for the whole duration of closing.

You can set this input continuously to `TRUE`. Automatic closing on the stack is only to be used while the grab is on the material.

A load cell detecting that the grab is on the ground may be used to enable automatic closing on stack.

The hold motor is working in a torque limited mode during automatic closing on stack. If activation of the automatic closing on stack function must be suppressed while the grab is mid-air, the user application must take into account the enabling and disabling of the automatic closing on stack function. The situation often results in a detected over speed fault reported by the Altivar 71 drive.

NOTICE

GRAB MECHANISM DAMAGE

Only use the automatic closing on stack function when the grab mechanism is in contact with the material to be moved.

Failure to follow these instructions can result in equipment damage.

i_xTrqShareEn

TRUE on this input enables sharing of the load between hold and close axes when the grab is closed. When it is FALSE, the axes are position synchronized.

Use torque sharing and switch to position synchronization only when the torque values are too low or unstable for load sharing.

You can configure a torque offset in order to have more torque on the close motor to keep the filled grab closed. See parameter `i_stPGC.wTrqAddOnClose`.

NOTE: Switching between load sharing and position synchronization while the axes are moving is possible when the position with balanced load is similar to the target position of position synchronization.

Example: In hook mode, when the hook is empty and the torque of one or both axes is too low for effective load sharing, set `i_xTrqShareEn` to FALSE. And set it to TRUE once the torque increases.

In grab mode, you should set `i_xTrqShareEn` to TRUE.

i_xScrapEn

TRUE on this input enables the scraping function. Scraping is active only during automatic closing on stack. Ideally it is directly controlled by a modifier button on the joystick. It can share the button with the function of rapid opening. When the button gets released during scraping, the FB automatically changes to standard closing on stack.

i_xHookModeEn

TRUE on this input enables the hook mode. Commands to close drive are ignored in the hook mode. The axes operate either in load sharing or position synchronization mode, based on the state of `i_xTrqShareEn` input.

NOTE: Switching between hook mode and standard grab operation while the axes are running is possible but not recommended.

i_xRapidOpenEn

The function is active when commands to raise and open and the `i_xRapidOpenEn` are active at the same time. It enables a fast opening of the grab by moving hold and close axes in opposite directions.

i_xRst

Resets alarms on rising edge, if the cause of the alarm is no longer present.

Structured Variable Description

i_stPGC

The following table describes the parameter structure of the data type i_stPGC

Output	Data Type	Description
wDrvAcc	WORD	Acceleration ramp time Range: 1...999 Scaling/Unit: 0.1 s Default Value: 20
wDrvDec	WORD	Deceleration ramp time Range: 1...999 Scaling/Unit: 0.1 s Default Value: 20
wDrvSpdHsp	WORD	High Speed Range: 0...6000 Scaling/Unit: RPM Default Value: 1500
wEncPulsPerRevo	WORD	Encoder pulses per motor revolution Range: 0...65535 Scaling/Unit: INC Default Value: 1024
wDrvSpdNom	WORD	Nominal motor speed Range: 0...6000 Scaling/Unit: RPM Default Value: 1500
rKpPosSync	REAL	Proportional gain of position controller Range: 0...3 Default Value: 0.01
rKpTrqShr	REAL	Proportional gain of torque controller Range: 0...3 Default Value: 0.1
wTrqAddOnClose	WORD	Torque add on for close axis Range: 0...500 Scaling/Unit: 0.1% Default Value: 0
wLowTrqDecRamp	WORD	Short no-load deceleration Range: 1...999 Scaling/Unit: 0.1 s Default Value: 1
wLowTrqDecRampThsh	WORD	Torque threshold for no-load deceleration ramp Range: 0...3000 Scaling/Unit: 0.1% Default Value: 0

Output	Data Type	Description
wClsHoldTrqLim	WORD	Torque limit of hold axis for closing on stack Range: 0...3000 Scaling/Unit: 0.1% Default Value: 100
wClsHoldSpdRef	WORD	Speed of hold axis for closing on stack Range: 0...6000 Scaling/Unit: RPM Default Value: 100
wClsTrqCloseThsh	WORD	Torque threshold to consider grab closed Range: 0...3000 Scaling/Unit: 0.1% Default Value: 0
wClsTrqCloseTimeFltr	WORD	Time filter for closing on torque threshold Range: 0...65535 Scaling/Unit: ms Default Value: 0
wClsClosedPosOfst	WORD	Offset of closed position for closing on stack Range: 0...65535 Scaling/Unit: REV Default Value: 0
wScrapHoldTrqLim	WORD	Torque limit of hold axis for scraping Range: 0...3000 Scaling/Unit: 0.1% Default Value: 100
wBrakTimeDelay	WORD	Time for brake delay compensation Range: 0...65535 Scaling/Unit: ms Default Value: 0
xTrqScopeScale	BOOL	Selector for torque scaling Default Value: TRUE

wDrvAcc

Acceleration ramp used for hold and close axes. The value is used by the FB in position synchronization and load sharing as a base for output acceleration values $q_wDrvAccHold$ and $q_wDrvAccClose$.

wDrvDec

Deceleration ramp used for hold and close axes. The value is used by the FB in position synchronization and load sharing as a base for output deceleration values $q_wDrvDecHold$ and $q_wDrvDecClose$.

wDrvSpdHsp

High speed value used for hold and close axes. Defines the maximum speed reference that the FB is allowed to use for both axes.

wEncPulsPerRevo

Number of encoder pulses per motor revolution. Must be identical for both axes.

wDrvSpdNom

Nominal (synchronous, without slip) speed of the motor. Must be identical for both axes.

50 Hz:

- 1pole pair: 3000 RPM
- 2pole pairs: 1500 RPM
- 3pole pairs: 1000 RPM
- 4pole pairs: 750 RPM

60 Hz:

- 1pole pair: 3600 RPM
- 2pole pairs: 1800 RPM
- 3pole pairs: 1200 RPM
- 4pole pairs: 900 RPM

rKpPosSync

Proportional gain of position controller used during position synchronization. It defines the difference of speed references of hold and close axes based on their actual position deviation. The default value is 0.01. It is internally normalized by encoder pulses per motor revolution `i_stPGC.wEncPulsPerRevo` and does not need to be changed when encoder type changes.

Decrease the value if motor speeds or torques are oscillating during position synchronization. Increase the value if the correction is not sufficient to reduce the position deviation to an acceptable value.

rKpTrqShr

Proportional gain of torque controller used during load sharing. It defines the difference of speed references of hold and close axes based on the actual difference of torques of both motors. Default value is 0.1. Decrease the value if motor speeds or torques are oscillating during load sharing. Increase the value if the correction is not sufficient to keep the motor torques at similar values.

wTrqAddOnClose

A torque offset for close axis. The FB allows for a torque difference between close and hold axes during load sharing when the grab is closed. This difference with higher torque on the close axis helps to keep the filled grab closed and prevent spillage of material. The default value is 50 [0.1%]. This parameter does not have any influence in hook mode.

wLowTrqDecRamp

An alternative short deceleration ramp used in downwards direction when a grab is on the ground. The ramp reduces the rope slack. It is used when the difference of torques of hold and close is below `i_stPGC.wLowTrqDecRampThsh`.

wLowTrqDecRampThsh

Torque threshold for switching between normal and alternative short deceleration ramp. [0.1%]. The function block switches the deceleration ramp to a short ramp defined in `i_stPGC.wLowTrqDecRamp` when the difference between hold and close motor torques gets below the value defined by `i_stPGC.wLowTrqDecRampThsh`. You can disable the function by setting `i_stPGC.wLowTrqDecRampThsh` to zero. The function is active when hold and close are moving synchronously. It is not active when the grab is opening or closing during downwards movement.

wCIsHoldTrqLim

Torque limitation of hold motor during closing on stack. The value must be high enough to keep hold cables straight, but low enough to allow sinking of the grab into material during closing on stack. The usual value is between 10 and 15% of nominal torque. Default value: 100 [0.1%]

wCIsHoldSpdRef

Speed reference for hold drive during closing on stack. The speed reference should be approximately 10% of nominal motor speed.

wCIsTrqCloseThsh

Torque threshold of close motor for considering the grab closed. The threshold is an alternative to detecting closed state of the grab by relative position of hold and close axes. It is used when a full grab does not close completely because of bulky material. Typical example is a cactus (spider) grab used in scrap metal or waste handling. The right value of torque must be found during commissioning, ideally using a trace in SoMachine. [0.1%]

wCIsTrqCloseTimeFitr

Time filter used for detection of closed state of the grab based on torque. It defines for how long must the torque of close motor stay above `i_stPGC.wCIsTrqCloseThsh` to consider the grab closed. [ms]

wClsClosedPosOfst

Offset of closed position for closing on stack. This parameter allows exceeding the closed position during closing on stack. When controlling clamshell grabs, it is usually required to exceed the closed position during closing on stack to prevent spillage of material during lifting. Start with a zero value and increase it if the grab spills during lifting. Spider (orange peel) grabs do not close fully when they are filled with material and therefore it is not necessary to use this parameter. Set this value to zero when controlling a spider grab.

wScrapHoldTrqLim

Torque limitation of hold motor during scraping. The value is slightly higher than `i_stPGC.wClsHoldTrqLim`. The torque must be low enough for the jaws to stay in contact with the surface, but high enough to prevent the grab from scraping too hard. [0.1%]

wBrakTimeDelay

Estimation of the time brakes of hold and close axes need to open. This parameter is used to prevent one of the axes moving while the other one is opening its brake. [ms]

xTrqScopeScale

When TRUE, the values of actual torque are divided by 4.096 internally by the FB for scaling of the "Motor torque scope" object (SOTR). CANopen index 2002:1B to the standard -3000 to 3000 range with 0.1% resolution. If it is FALSE, input torque values must be scaled to this range outside of the FB.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	Enable status of the FB TRUE: FB enabled FALSE: FB disabled
q_diEncPosDiff	DINT	Actual position difference of both axes Range: -2147483648...2147483647 Scaling/Unit: INC
q_diEncPosDev	DINT	Position deviation from target during position synchronization Range: -2147483648...2147483647 Scaling/Unit: INC
q_xDrvFwdHold	BOOL	Forward command for hold drive
q_xDrvRevHold	BOOL	Reverse command for hold drive
q_wDrvSpdTargHold	WORD	Target speed for hold drive Range: 0...6000 Scaling/Unit: RPM
q_wDrvAccHold	WORD	Acceleration for hold drive Range: 1...999 Scaling/Unit: 0.1 s
q_wDrvDecHold	WORD	Deceleration for hold drive Range: 1...999 Scaling/Unit: 0.1 s
q_wDrvTrqLimHold	WORD	Torque limitation for hold drive Range: 0...3000 Scaling/Unit: 0.1%
q_xDrvFwdClose	BOOL	Forward command for close drive
q_xDrvRevClose	BOOL	Reverse command for close drive
q_wDrvSpdTargClose	WORD	Target speed for close drive Range: 0...6000 Scaling/Unit: RPM
q_wDrvAccClose	WORD	Acceleration for close drive Range: 1...999 Scaling/Unit: 0.1 s
q_wDrvDecClose	WORD	Deceleration for close drive Range: 1...999 Scaling/Unit: 0.1 s
q_xGrabOpen	BOOL	Grab open
q_xGrabClosed	BOOL	Grab closed

Output	Data Type	Description
q_wGrabPos	WORD	Position of grab Range: 0...100 Scaling/Unit: %
q_iTrqFltrHold	INT	Filtered torque of hold drive Range: -3000...3000 Scaling/Unit: 0.1%
q_iTrqFltrClose	INT	Filtered torque of close drive Range: -3000...3000 Scaling/Unit: 0.1%
q_wStat	WORD	Status word
q_xAlrm	BOOL	Alarm output
q_wAlrmId	WORD	Alarm identification

q_xEn

Enable status of the FB.

q_diEncPosDiff

Encoder position difference between hold and close axes.

q_diEncPosDev

Deviation of actual position difference between hold and close axes from the target position difference. This value is written only during synchronized movement and load sharing.

When position synchronization is active this value should be close to zero if the proportional position controller is configured correctly.

During torque sharing it shows the difference between closed position and the position corresponding to actual torque difference.

q_xDrvFwdHold, q_xDrvRevHold

Forward and reverse command for hold drive.

q_wDrvSpdTargHold

Target speed for hold drive.

q_wDrvAccHold, q_wDrvDecHold

Acceleration and deceleration ramps for hold drive.

q_wDrvTrqLimHold

Torque limitation for hold drive. Torque limitation is used during closing on stack. Torque limitation must be enabled in configuration of the hold drive. [0.1%]

q_xDrvFwdClose, q_xDrvRevClose

Forward and reverse command for close drive.

q_wDrvSpdTargClose

Target speed for close drive.

q_wDrvAccClose, q_wDrvDecClose

Acceleration and deceleration ramps for close drive.

q_xGrabOpen

TRUE when the grab is fully open. The open position must be calibrated in order for open position detection to work.

q_xGrabClosed

TRUE when the grab is fully closed or if it is considered closed based on torque of the close motor during closing on stack. The closed position must be calibrated in order for closed position detection to work.

q_wGrabPos

Actual position of the grab. The range of this output is 0 to 100%. 0% corresponds to a fully closed and 100% to a fully open grab. Both closed and open positions must be calibrated for this output to work.

q_iTrqFiltrHold, q_iTrqFiltrClose

Actual torque scope values of hold and close drives filtered by the FB. These values can be used for tracing during commissioning.

q_wStat

Status word of the FB:

Bit Position	State Description
bit 0	Grab fully open
bit 1	Grab fully closed
bit 2	Closed state detected based on torque value
bit 3	Calibration of open position done

Bit Position	State Description
bit 4	Calibration of closed position done
bit 5	Position synchronization active
bit 6	Load sharing active
bit 7	Closing on stack active

q_xAlrm

In case of an alarm the FB removes run commands from both drives and does not allow any further movement until the cause of the alarm has been cleared.

q_dwAlrmId

This output identifies the detected alarm.

Alarm	Alarm Description
bit 0	One or both of the drives are not ready and a command to move the grab was given.
bit 1	Input configuration, <code>i_st.PGC.wDrvSpdLsp >= i_st.PGC.wDrvSpdHsp</code>

Section 9.5

Quick Reference Guide

What Is in This Section?

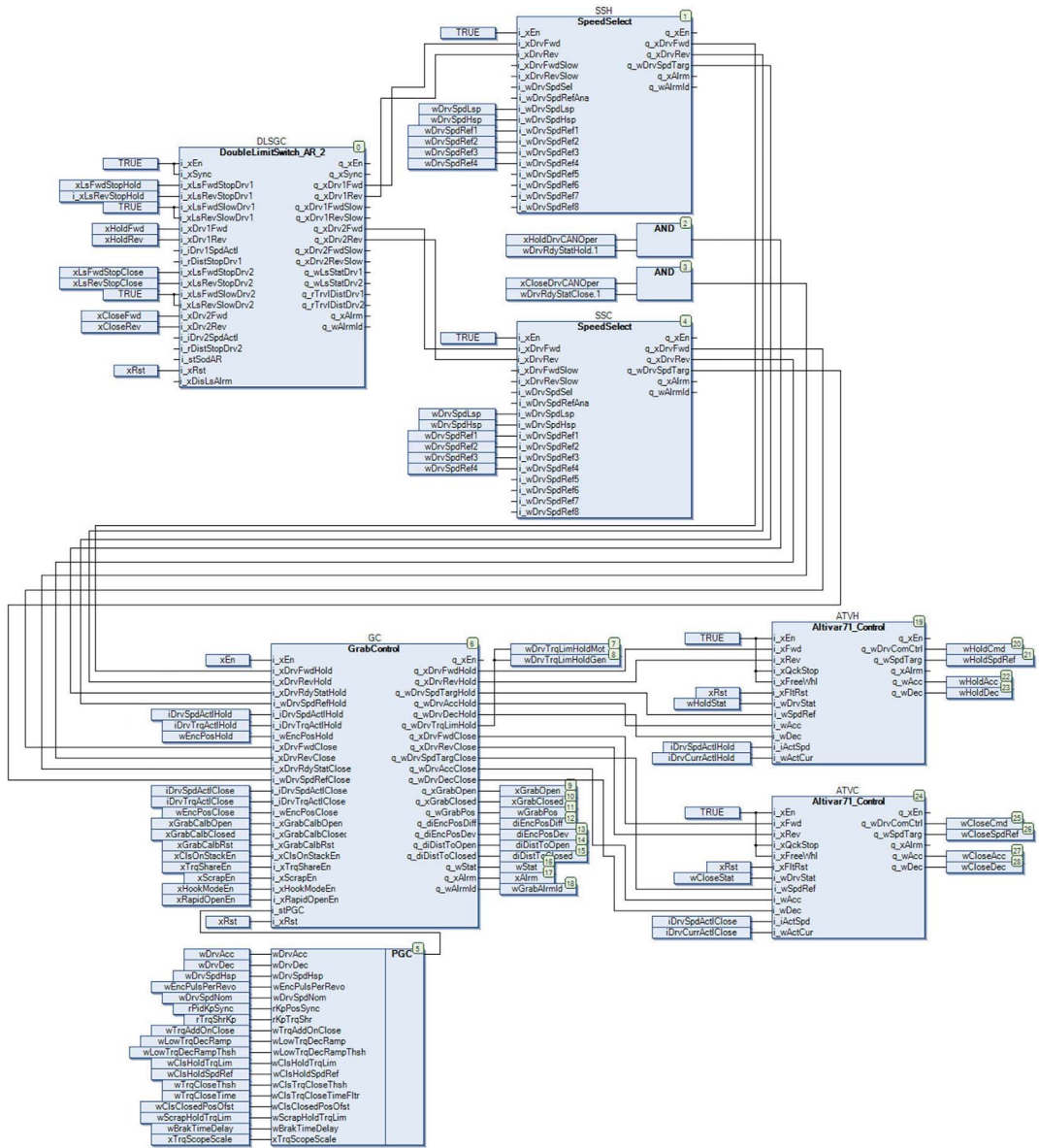
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	326
Commissioning Procedure	328
Troubleshooting	334

Instantiation and Usage Example

Instantiation and Usage Example

The implementation and usage example of `GrabControl` function block is as follows:



Commissioning Procedure

Commissioning of Hold and Close Drives

The following points apply to both drives unless stated otherwise.

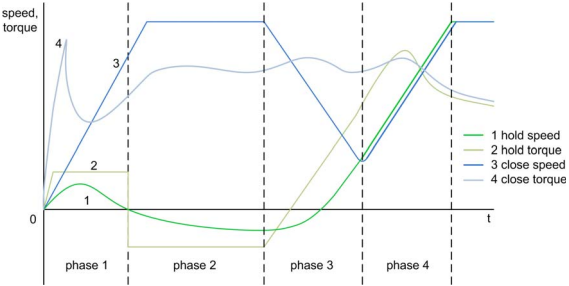
Step	Action
1	Set hold and close drives to factory settings.
2	Configure correct motor control parameters.
3	Configure brake logic control.
4	Perform motor tuning and encoder test.
5	Set the motor control mode to FVC.
6	In drive menu application functions disable preset speeds. Set command and reference channel to CANopen.
7	Configure CANopen addresses and bus speed, drive power cycle is needed after change of CANopen configuration.
8	In settings menu set the "K speed loop filter" to 100. This parameter influences how accurately the actual speed follows the target speed. The higher the value the more accurately Altivar 71 responds to changing target speed. If motor speeds or torque tend to oscillate, it is possible to lower the value at a cost of lowering precision of virtual limit switch function during opening and closing.
9	Enable torque limitation on the hold drive and set it to 0.1% resolution. The hold drive is torque limited during closing on stack.
10	Disable detection of load slipping in encoder fault configuration in fault management. If load slipping fault detection is enabled during closing on stack, hold drive reports a load slipping fault.

Commissioning of GrabControl Function Block

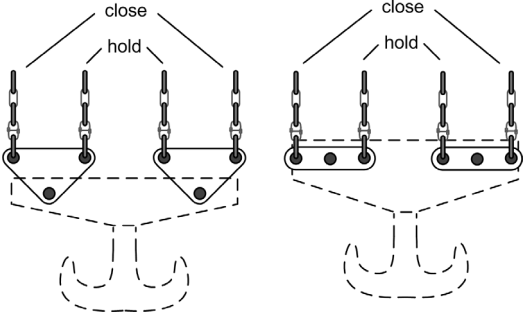
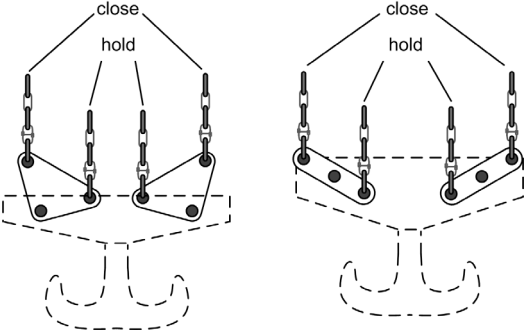
Step	Action
1	Without moving the motors verify that all commands and target speeds are being received correctly from the operator interface.
2	Add two Altivars 71 to CANopen configuration. Configure the PDOs: <ul style="list-style-type: none"> ● Receive PDOs <ul style="list-style-type: none"> Both drives: Command word, Target speed, Acceleration, Deceleration Hold drive only: Motor torque limitation, Generator torque limitation ● Transmit PDOs <ul style="list-style-type: none"> Both drives: Status word, Actual speed (Control effort), Encoder position value (PUC), Torque scope value, Motor current
3	Instantiate the GrabControl FB in your program and parameterize it with initial parameters.

Step	Action
4	If the torque scope values of both motors are connected to the FB directly, set the <code>i_stPGC.xTrqScopeScale</code> parameter to TRUE. Otherwise the torque values need to be scaled in the application by dividing it by 4.096 and the <code>i_stPGC.xTrqScopeScale</code> parameter must be set to FALSE.
5	Instantiate it in a program executed in a cyclic task. Preferably in the Master task. The same task should be a parent task for CANopen communication. Master task is the parent task by default.
6	Configure the execution period of the corresponding task to a value in the range between 20 and 50 ms.
7	Download the application, put it to run and verify status of CANopen communication.
8	Calibrate open and closed positions of the grab. Following two steps describing calibration of closed and open positions may be performed in an arbitrary order. Incorrect calibration may be reset by rising edge on <code>i_xGrabCalbRst</code> input.
9	Close the grab fully and calibrate the closed position by rising edge on <code>i_xGrabCalbClosed</code> . NOTE: When the grab is in good mechanical condition (no space between jaws when it softly closes), calibrate it in a state when the jaws are closed, but both hold and close cables are still fully tensioned. When the mechanics of the grab is not aligned perfectly, you may want to calibrate it in a little overclosed position (more load on close cables).
10	Open the grab fully and calibrate the open position by rising edge on <code>i_xGrabCalbOpen</code> .
11	Close the grab at slow speed and verify that it stops automatically when it reaches closed state. If it functions correctly, repeat this step at nominal speed. The grab should close softly without an impact of the jaws and jerk of the crane. If the grab was calibrated in a slightly over-closed position as described earlier, certain impact when the grab closes is acceptable and even desirable.
12	Repeat the test for open position. The grab must open softly. Both, hold and close cables should be straight without slack in open position.
13	Set the proportional gain for position synchronization <code>i_stPGC.rKpPosSync</code> . The default gain is 0.01 and the FB internally compensates for resolution of the encoder. Therefore it is not necessary to change the value when encoder with different resolution is used.
14	Test the behavior of position synchronization. Set the <code>i_xTrqShareEn</code> input to FALSE or move the grab to a non-closed position to activate position synchronization <code>q_wStat.5 = TRUE</code> . Move the grab up and down by giving command to the hold drive without giving command to the close drive. Take a trace of actual torque, speeds of both motors and the position deviation output <code>q_diEncPosDev</code> . When the position deviation is higher than required and if the motor speeds and torques do not oscillate, increase the value of <code>i_stPGC.rKpPosSync</code> . If motor speeds and torques are oscillating, decrease the value of <code>i_stPGC.rKpPosSync</code> .
15	Set the proportional gain for load sharing <code>i_stPGC.rKpTrqShr</code> . The default gain is 0.1.

Step	Action
16	<p>Test the behavior in load sharing. Close the grab and set the <code>i_xTrqShareEn</code> input to TRUE to activate load sharing <code>q_wStat.6 = TRUE</code>. Set the <code>i_stPGC.wTrqAddOnClose</code> parameter to 0. Move the grab up and down by giving command to the hold drive without giving command to the close drive. Take a trace of actual torque of both motors. Both torques should converge to the same value. When the torques do not converge fast enough, increase the value of <code>i_stPGC.rKpTrqShr</code>. If the torques are oscillating around a common average value, decrease the value of <code>i_stPGC.rKpTrqShr</code>.</p>
17	<p>Set a value of <code>i_stPGC.wTrqAddOnClose</code> to a value of ~50 (5%) if it is required to keep higher torque on the close motor in order to keep the grab closed. The value may be modified later during tests with filled grab. Take a trace to test that the torque of close motor is higher than the torque of the hold motor by defined value and that the motor torques are stable.</p>
18	<p>Configure a trace object in SoMachine containing actual speed, speed reference and actual torque of both motors and commands from the joystick. Add also bits from status word. (1,2,5,6,7) The trace is very useful during commissioning and for troubleshooting.</p>
19	<p>Configure the closing on stack function. Set the value of <code>i_stPGC.wClsHoldTrqLim</code> to ~100 (10%). The FB will limit torque of the hold drive to this value during closing on stack. The value must be high enough to keep the hold cables straight and support the grab in upright position, but must be low enough to allow sinking of the grab to the material during closing. Set the <code>i_stPGC.wClsHoldSpdRef</code> to ~10% of the nominal speed of the motor. The speed may be increased if slack of the hold cables is not compensated for fast enough and decreased if straightening of hold cables lifts the closing grab. Configure the <code>wClsClosedPosOfst</code> parameter to 0, it may be increased later to help keeping the filled grab closed during vertical movement. Test the automatic closing on stack by laying an open grab on the stack and giving and keeping the command to activate closing on stack. Speed of the close motor can be modified during closing on stack. Speed of the hold motor is given by <code>i_stPGC.wClsHoldSpdRef</code> and <code>i_stPGC.wClsHoldTrqLim</code>. The grab must close fully and when the command is still being held it must softly lift the closed grab up.</p>

Step	Action
20	<p data-bbox="351 199 1181 253">Following figure depicts speeds and torques of both drives and describes phases of the procedure:</p>  <ul style="list-style-type: none"> <li data-bbox="351 597 1249 704">● Phase 1 Close axis accelerates in forward direction, closing the grab Hold axis has also a positive speed reference, but is torque limited, which prevents it from lifting the whole grab. Therefore it only straightens the slack cables and then stops. <li data-bbox="351 708 1249 841">● Phase 2 Close axis continues to close the grab. Hold axis is pulled down by the weight of the grab because of its torque limitation. This allows for a proper filling of the grab. The amount of sinking of the grab is influenced by the parameter <code>i_stPGC.wClsHoldTrqLim</code>. <li data-bbox="351 844 1249 951">● Phase 3 The grab is approaching the closed position. Close axis decelerates and the torque limitation of hold axis is gradually lifted. The hold axis accelerates and reaches the same speed as the close axis in the moment when the grab is fully closed <li data-bbox="351 954 1249 1062">● Phase 4 Both axes move in load sharing or position synchronization depending on setting of the parameter <code>i_xTrqShareEn</code>. The torques are equalizing and lifting of the filled grab continues
21	<p data-bbox="351 1073 1249 1206">If the grab is spilling when it's lifted from the stack, increase the value of <code>i_stPGC.wClsClosedPosOfst</code> by 2 and repeat the closing on stack. Increase the value some more if necessary. The parameter causes a slight over-closing of the grab and helps to contain the material in a grab with less than optimal mechanical precision. When controlling a spider grab, leave this value at 0.</p> <p data-bbox="351 1209 1249 1344">When you are controlling a spider grab or any grabbing application where a grab might not close fully because of objects between the jaws, configure the <code>i_stPGC.wClsTrqCloseThsh</code> and <code>i_stPGC.wClsTrqCloseTimeFltr</code> parameters. Trace the torque value of close motor during opening and select an appropriate torque level for considering the grab closed. Configure a filter time to filter out torque peaks.</p>

Step	Action
22	Move the closed and filled grab upwards and downwards and trace actual torques of both motors. The torques should stabilize and the close torque should be higher than the hold torque by the value specified in <code>i_stPGC.wTrqAddOnClose</code> . If the torques do not converge fast enough, increase the value of <code>i_stPGC.rKpTrqShr</code> , if they oscillate, decrease it. If the grab spills its content, increase the value of <code>i_stPGC.wTrqAddOnClose</code> .
23	Configure quick deceleration ramp. It helps reducing rope slack by allowing fast stop ramp in downwards direction when the grab is laid on the stack. Measure the torque of hold drive during downwards movement with open grab and the difference it makes when grab lands on the ground/stack and the movement continues with slack cables. Set the <code>i_stPGC.wLowTrqDecRampThsh</code> parameter lower than absolute value of torque with grab hanging on the cables and moving down. Test that the short ramp is used only when the grab lies on the stack. This function is not active while the grab is opening or closing during movement.
24	If required, configure the optional scraping function. It allows scraping of rests of transported material from a hard flat surface. The function is similar to closing on stack, but it keeps contact with the surface until the grab is fully closed. The parameter <code>i_stPGC.wScrapHoldTrqLim</code> defines torque limitation of the hold drive during scraping. It defines how much force is applied on the surface. Set the value higher to apply less force on the surface during scraping. The value must be low enough to keep contact between jaws of the grab and the surface for the whole duration of scraping.
25	If required, enable and test the optional rapid opening function.
26	If the brakes on hold and close axes need a long time to open, it may lead to slack cables. This usually happens when a downward movement command is given right after the grab fully opened. The brake of the close motor is still open because it just finished the opening movement, but the brake of the hold drive is closed. Therefore the close drive starts to move immediately, but the hold drive waits until its brake opens. This delay causes more or less significant slack of close cables. The function block uses the parameter <code>i_stPGC.wBrakTimeDelay</code> to compensate for the brake delay. Set the value to the brake opening time in milliseconds.

Step	Action
27	<p data-bbox="351 203 1249 280">If required, test the hook mode. Hook mode allows usage of a hook on a four cable system. There are several hook types. Two of them are described here. The following figure is a hook type with two variants of hooks for four cable grabs:</p>  <p data-bbox="351 662 1238 711">The following figure is a hook type with two variants of hooks with position difference between hold and close axes:</p>  <p data-bbox="351 1110 1249 1188">The left hook uses a lever principle to distribute load depending on position difference between hold and close axes and the right one distributes the load evenly independent of the position difference.</p> <p data-bbox="351 1192 1249 1401">The left hook uses the shape of triangular connection blocks and the resulting variable lever arm length ratio to apply more load on close cables than on hold cables in the depicted position. The right hook applies the same load on both cables because the lever arm length ratio stays the same regardless of the position difference as long as both cables are straight. This makes the left hook well suited for operation in load sharing and the right one for position synchronization. When moving an unloaded hook, the torque may be too low for efficient load sharing. In this case it is possible to switch from position synchronization using the input <code>i_xTrqShareEn</code> when the torque value exceeds certain value.</p> <p data-bbox="351 1404 1249 1455">Switching between load sharing and positioning on the fly is possible as long as the synchronized positions are not far from positions in which the loads are in balance.</p>

Troubleshooting

Troubleshooting

Issue	Cause	Solution
Grab does not close fully when it is being closed in the air	Position calibration values invalid	Reset calibration values and perform a new calibration.
Grab does not slow down when approaching closed or open positions	Position calibration values not present or invalid	Reset calibration values and perform a new calibration.
Motor speeds and/or torques oscillate in synchronous movement	Gain of position controller too high	Lower the gain of position controller
Motor speeds and/or torques oscillate in load sharing	Gain of torque controller too high	Lower gain of the torque controller
Grab spills material	Grab is not fully closed	Increase the value of <code>i_stPGC.wTrqAddOnClose</code>

Part VIII

Hoisting Position Synchronization

Overview

This part explains the functionality and implementation of `HoistPositionSync` function block in the Hoisting Library.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
10	HoistPositionSync: Synchronization of Two Identical Axes for Hoist or Trolley Movement with Encoder	337
11	HoistPositionSync_2: Position Synchronization of Two Crane Axes	359

Chapter 10

HoistPositionSync: Synchronization of Two Identical Axes for Hoist or Trolley Movement with Encoder

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
10.1	Functional and Machine Overview	338
10.2	Architecture	342
10.3	Function Block Description	345
10.4	Pin Description	346
10.5	Troubleshooting	356

Section 10.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	339
Machine Overview	341

Functional Overview

Functional Description

The `HoistPositionSync` function block for industrial cranes is designed to synchronize the position of 2 axes for simultaneous handling of large objects. The block provides a relative synchronization. This means that the difference in position of 2 motors is kept constant in synchronized state and the absolute position of hoists is not taken in account.

Why Use the `HoistPositionSync` Function Block?

Synchronized hoists are needed to move long loads. The function block synchronizes 2 axes. The axes could be 2 hoist axes or 2 trolley axes.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

Solution with the `HoistPositionSync` Function Block

This function block uses information about position of both motors to synchronize them. The information from motor encoders is evaluated by encoder cards of ATV71 variable speed drives. This information is sent to the controller via CANopen and the difference in position between motors is used as a process error input of proportional controller algorithm within the function block (hereafter referred to as the P-controller). The output of the controller is then used to correct speed of slave or master drive.

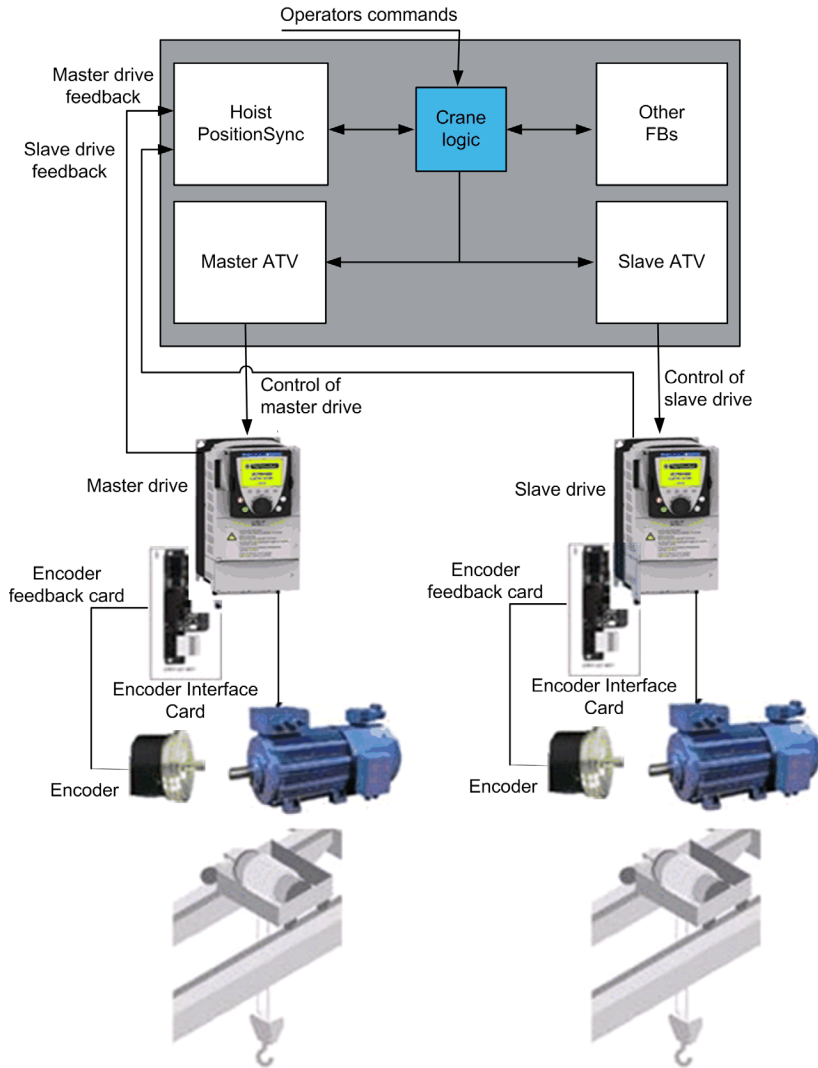
NOTE: The terms master and slave when referring to the drives are only an indication of the first and second drives in the function block. The terms are not a reference to communications, priority nor primacy.

If calculated speed of slave drive is lower than the set high speed or lower than the set low speed, speed of the master drive is corrected instead.

Design & Realization Assumptions

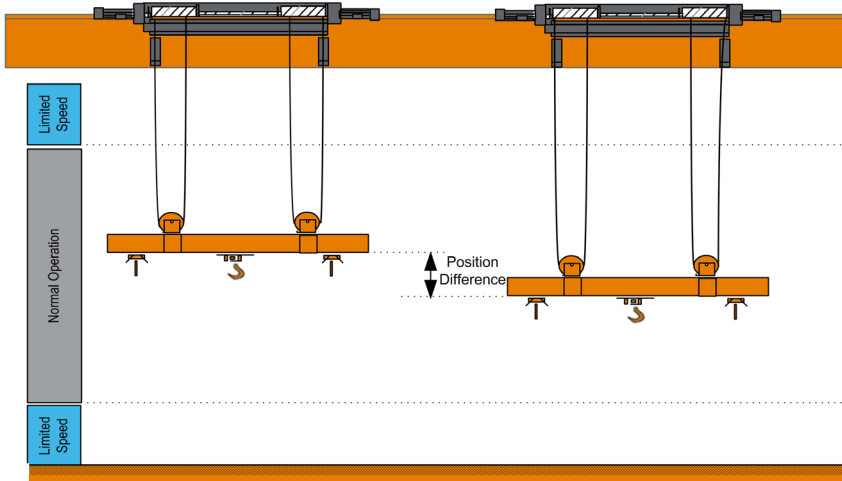
It is assumed that when the motor moves in the forward direction, the Hoist moves up and when the motor moves in reverse direction, the Hoist moves down.

Functional View



Machine Overview

Machine View



Section 10.2 Architecture

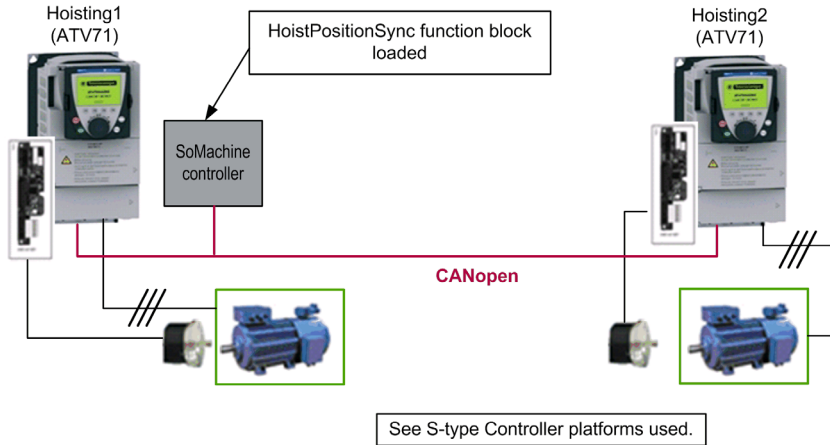
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	343
Software Architecture	344

Hardware Architecture

Hardware Architecture Overview



Software Architecture

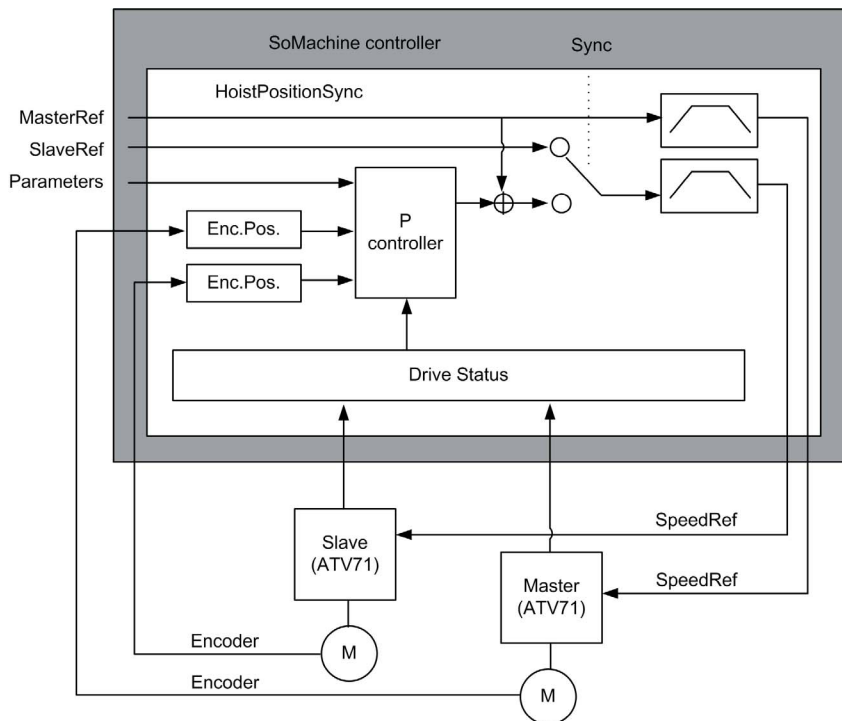
DataFlow Overview

The function block is used to synchronize 2 motors. It uses 2 drives:

- Master
- Slave

The motors are equipped with encoders for the position feedback. The function block uses these encoder feedbacks in synchronization mode. The function block uses these encoder feedbacks in synchronization mode. The encoder position difference is fed as process detected error to the P controller. The P controller generates manipulated value to correct the process error. The P controller generates manipulated value to correct the process error.

In non-synchronized mode, user can operate the drives independently.

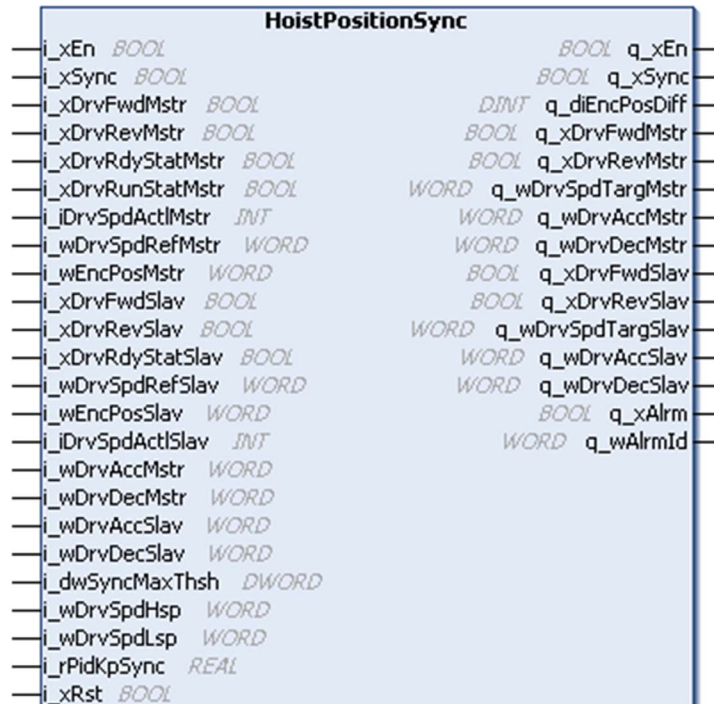


Section 10.3

Function Block Description

HoistPositionSync Function Block

Pin Diagram



Function Block Description

The following are the functions of HoistPositionSync function block:

- Used to synchronize the motion of 2 trolley motors or 2 hoist motors. This is helpful when a single large load is being moved by 2 motors.
- Achieves a synchronized motion of 2 motors.
- Controls both motors in synchronous and asynchronous mode.

Section 10.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	347
Acceleration and Deceleration Parameter	352
Output Pin Description	353

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	<p>TRUE: Enables the function block. FALSE: Disables the function block.</p> <p>The enable function enables or disables the function. This input can be connected to an HMI in case you want to disable the function at any time.</p> <p>NOTE: When the function block is disabled, (q_xEn) goes to FALSE, alarm outputs q_xAlrm and q_xAlrmId are set to zero and both master and slave drives operate independently with their appropriate commands.</p> <p>Refer to detailed description (<i>see page 350</i>) of i_xEn.</p>
i_xSync	BOOL	<p>TRUE: Enables the synchronization. FALSE: Disables the synchronization.</p> <p>NOTE: The function block can enter Synchronous or Asynchronous state only after both drives are stopped.</p> <p>Refer to detailed description (<i>see page 351</i>) of i_xSync.</p>
i_xDrvFwdMstr	BOOL	<p>TRUE: Forward FALSE: Not Forward</p> <p>This is the input for the forward commands of the master drive. In Synchronous mode, this input will control the slave drive as well.</p>
i_xDrvRevMstr	BOOL	<p>TRUE: Reverse FALSE: Not Reverse</p> <p>This is the input for the reverse commands of the master drive. In Synchronous mode, this input will control the slave drive as well</p>
i_xDrvRdyStatMstr	BOOL	<p>TRUE: Drive ready FALSE: Drive not ready</p> <p>This input must be connected to CANopen status of the master drive, Ready state / status word bit 1.</p> <p>This Boolean input normally comes from the status word of the drive through CANopen.</p> <p>Refer to detailed description (<i>see page 351</i>) of i_xDrvRdyStatMstr.</p>

Input	Data Type	Description
i_xDrvRunStatMstr	BOOL	TRUE: Drive running FALSE: Drive Not running This boolean input normally come from the status word of the drive through CANopen.
i_iDrvSpdActlMstr	INT	Range: -6000...6000 RPM This is the actual speed of the master drive received through CANopen.
i_wDrvSpdRefMstr	WORD	Range: 0...6000 RPM This is the target speed input of the master drive. In Synchronous mode, it is the target speed input of both drives.
i_wEncPosMstr	WORD	Range: 0...65535 count This is the position feedback of the encoder of the master drive.
i_xDrvFwdSlav	BOOL	TRUE: Forward FALSE: Not Forward This is the input for the forward command of the slave drive. This command is ignored in Synchronous mode.
i_xDrvRevSlav	BOOL	TRUE: Reverse FALSE: Not Reverse This is the input for the Reverse command of the slave drive. This command is ignored in Synchronous mode.
i_xDrvRdyStatSlav	BOOL	TRUE: Drive ready FALSE: Drive not ready This input must be connected to CANopen status of the slave drive, Ready state / status word bit 1. This input comes from the status word of the drive through CANopen. i_xDrvRdyStatSlav is TRUE when the drive is in Ready state. Refer to detailed description (<i>see page 351</i>) of i_xDrvRdyStatSlav.
i_wDrvSpdRefSlav	WORD	Range: 0...6000 RPM This is the target speed input of the slave drive.
i_wEncPosSlav	WORD	Range: 0...65535 count This is the position feedback of the encoder of the slave drive.
i_iDrvSpdActlSlav	INT	Range: -6000...6000 RPM This is the target speed input of the slave drive.

Input	Data Type	Description
i_wDrvAccMstr	WORD	Range: 1...9999 s Master drive acceleration. Scaling/Unit: 0.1 s For details, refer to the Acceleration and Deceleration Parameter (<i>see page 352</i>)
i_wDrvDecMstr	WORD	Range: 1...9999 s Scaling/Unit: 0.1 s Master drive deceleration Scaling/Unit: 0.1 s For details, refer to the Acceleration and Deceleration Parameter (<i>see page 352</i>)
i_wDrvAccSlav	WORD	Range: 1...9999 s Slave drive acceleration Scaling/Unit: 0.1 s For details, refer to the Acceleration and Deceleration Parameter (<i>see page 352</i>)
i_wDrvDecSlav	WORD	Range: 1...9999 s Slave drive deceleration Scaling/Unit: 0.1 s For details, refer to the Acceleration and Deceleration Parameter (<i>see page 352</i>)
i_dwSyncMaxThsh	DWORD	Range: 0...4294967295 count This is the maximum position difference allowed between the master and the slave encoders. If this value is exceeded, an alarm signal is generated.
i_wDrvSpdHsp	WORD	Range: 0...6000 RPM This is the maximum allowed speed for both drives (HSP). It also serves as a threshold for the correction calculation. If the correction value of the P control would set the speed above i_wDrvSpdHsp, the master drive will be corrected instead.
i_wDrvSpdLsp	WORD	Range: 0...6000 RPM This is the minimum allowed speed for both drives (LSP). It also serves as a threshold for the correction calculation. Below this speed, correction is disabled. If the correction value of the P control sets the speed below i_wDrvSpdLsp, the second drive will be corrected instead. As the drive is running in Flux Vector Control (FVC) we recommend to set this pin to zero (0) as well as set the drives LSP to zero (0).

Input	Data Type	Description
i_rPidKpSync	REAL	This represents the proportional gain of position controller. Range: 0...3 Default value: 0.3
i_xRst	BOOL	On a rising edge, attempts to clear all alarms. If alarms stay active despite rising edge of this input, the cause of the alarm is still present.

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

When the i_xEn input is set back to TRUE, the function block resumes its normal operation. If synchronization command is present at the time of enabling of the function block and all conditions for synchronization are fulfilled, the axes enter synchronous mode at their current positions.

The fallback states of HoistPositionSync are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_xSync	BOOL	FALSE
q_diEncPosDiff	DINT	0
q_xDrvFwdMstr	BOOL	i_xDrvFwdMstr
q_xDrvRefMstr	BOOL	i_xDrvRevMstr
q_xDrvFwdSlav	BOOL	i_xDrvFwdSlav
q_xDrvRevSlav	BOOL	i_xDrvRevSlav
q_wDrvAccMstr	WORD	i_wDrvAccMstr
q_wDrvDecMstr	WORD	i_wDrvDecMstr
q_wDrvAccSlav	WORD	i_wDrvAccSlav
q_wDrvDecSlav	WORD	i_wDrvDecSlav
q_wDrvSpdTargMstr	WORD	i_wDrvSpdTargMstr
q_wDrvSpdTargSlav	WORD	i_wDrvSpdTargSlav
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

i_xDrvRdyStatMstr, i_xDrvRdyStatSlav

The input needs to be logically connected with the CANopen status of the controlled drive. This could be found in the CANopen status word bit 1 / ready state.

i_xSync

Command to enter or leave the synchronized state. Both drives must be ready and stopped to enter the Synchronous mode and must be stopped to exit it.

In Synchronous mode (*i_xSync* is TRUE), the master and the slave are controlled from the command inputs of the master. For example, *i_xDrvFwdMstr* starts the master and slave drive in forward direction. The input for the slave, for example, *i_xDrvFwdSlav* is ignored. The speed reference for both drives is given at *i_wDrvSpdRefMstr*; both speed reference are recalculated internally by the P control.

The master and slave is controlled individually in Asynchronous mode (*i_xSync* is FALSE). That is, the master is controlled by the master command (*i_xDrvFwdMstr*), the slave is controlled by the slave command (*i_xDrvFwdSlav*). The reference speeds are given by *i_wDrvSpdRefMstr* for the master and *i_wDrvSpdRefSlav* for the slave.

Acceleration and Deceleration Parameter

Description

The following are the acceleration and deceleration parameters of the master and slave drive, when the synchronization is active:

- `i_wDrvAccSlav`
- `i_wDrvDecSlav`
- `i_wDrvAccMstr`
- `i_wDrvDecMstr`

Active synchronization:

When synchronization is active, these values are used to recalculate output acceleration and deceleration ramps to reach required target speed simultaneously. The block calculates internally the required ramp time values.

Inactive synchronization (or the function block disabled):

`q_wDrvAccSlav = i_wDrvAccSlav`

`q_wDrvDecSlav = i_wDrvDecSlav`

`q_wDrvAccMstr = i_wDrvAccMstr`

`q_wDrvDecMstr = i_wDrvDecMstr`

Output Pin Description

Output Pin Description

The following table includes the different outputs of the function block along with description:

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_xSync	BOOL	TRUE: Sync Enabled FALSE: Sync Disabled
q_diEncPosDiff	DINT	Range: -2147483648...2147483647 count This is the actual difference of master and slave encoder position after activating the Synchronous mode. The output is reset to 0 in Asynchronous mode. For Example: $q_diEncPosDiff = \text{Position of master drive} - \text{Position of slave drive}$ 16-bit information about position received from drives through CANopen is recalculated to 32 DINT values. These values are then used by the function block. The flip over of the word input is monitored.
q_xDrvFwdMstr	BOOL	TRUE: Forward FALSE: Not Forward Master drive forward run command.
q_xDrvRevMstr	BOOL	TRUE: Reverse FALSE: Not Reverse Master drive reverse run command.
q_wDrvSpdTargMstr	WORD	Range: 0...6000 RPM Master drive target speed.
q_wDrvAccMstr	WORD	Range: 1...9999 s Scaling/Unit: 0.1 s Master drive acceleration.
q_wDrvDecMstr	WORD	Range: 1...9999 s Scaling/Unit: 0.1 s Master drive deceleration.
q_xDrvFwdSlav	BOOL	TRUE: Forward FALSE: Not Forward Slave drive forward run command.
q_xDrvRevSlav	BOOL	TRUE: Reverse FALSE: Not Reverse Slave drive reverse run command.
q_wDrvSpdTargSlav	WORD	Range: 0...6000 RPM Slave drive target speed.

Output	Data Type	Description
q_wDrvAccSlav	WORD	Range: 1...9999 s Scaling/Unit: 0.1 s Slave drive acceleration.
q_wDrvDecSlav	WORD	Range: 1...9999 s Scaling/Unit: 0.1 s Slave drive deceleration.
q_xAlrm	BOOL	TRUE: Alarm detected FALSE: No alarm detected This output displays the detected alarm status of the function block. TRUE indicates a detected alarm is currently present. FALSE means the function block has no detected alarm.
q_wAlrmId	WORD	This is the detected alarm identification

NOTE: The following commands, target speed and ramp value outputs must be connected to the corresponding ATV function block input pins of the master to allow operation:

- q_xDrvFwdMstr
- q_xDrvRevMstr
- q_wDrvSpdTargMstr
- q_wDrvAccMstr
- q_wDrvDecMstr

The following commands, target speed and ramp value outputs must be connected to the corresponding ATV function block input pins of the master to allow operation:

- q_xDrvFwdSlav
- q_xDrvRevSlav
- q_wDrvSpdTargSlav
- q_wDrvAccSlav
- q_wDrvDecSlav

Notifications

Bit Position	Description Represented by Bit Position
0	The Actual error $q_diEncPosDiff$ is more than the $i_dwSyncMaxThsh$
1	Master or the Slave drive is not ready, $i_xDrvRdyStatMstr$ or $i_xDrvRdyStatSlav$ inputs are false.
2	The $i_wDrvSpdHsp$ value is smaller or equal than $i_wDrvSpdLsp$.

NOTE: If an alarm based on $q_diEncPosDiff > i_dwSynchMax$ arises, the movement will be stopped automatically and you need to turn off the Synchronization function ($i_xSync=FALSE$). Bring 1 of the 2 drives back into the correct position for the synchronization manually. When your load is back in the desired position, restart the synchronization ($i_xSync=TRUE$). If the alarm arises again, verify your configuration for correctly set parameters or your hardware for defective parts.

Section 10.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The bit 0 of <code>q_wAlrmId</code> is TRUE.	Setting of the proportional gain could be incorrect (<code>i_rPidKpSync</code>) or the position difference tolerance threshold could be too low	Check for the proportion gain of the function block and tune this parameter. Check the value of <code>i_dwSyncMaxThsh</code> and increase it if it is too low.
	Resolution of the master and slave encoders are not the same	Use encoders with identical resolution. Check the encoder setting on the drive
	Encoder or motor phases are wired or configured incorrectly	Check wiring and setting of encoders and motors
The bit 1 of <code>q_wAlrmId</code> is TRUE	Either the master or the slave is not in ready state or there is a communication problem.	Check for the CANopen connection, and establish the CANopen communication between controller and drives. Reset the error state detected on both drives
The bit 2 of <code>q_wAlrmId</code> is TRUE.	Drive HSP is less than or equal to LSP	Set HSP higher than LSP
Motors continue to run at low speed even after command has been released.	Value of <code>i_wDrvSpdLsp</code> is lower than LSP parameter set in the drive.	Set LSP in drive configuration to a value equal to setting of the FB.
Position error increases during direction change.	One of the brakes closes when zero speed is reached.	Set brake engage delay to value higher than zero in drive configuration.
One of the drives is in event state or lost connection to the controller and the other one continues to run.	Function block does not have information about event of the drive connected or bus supervision is not configured.	Connect the information about status of the CANopen node to corresponding <code>i_xDrvRdyStat</code> input together with information about ready status of the drive. Input in logical AND with information about ready status of the drive. Make sure that the bus supervision is configured.
The actual motor speed does not correspond to speed reference.	Preset speeds configuration interferes with speed reference from controller.	Disable preset speeds in application functions setting of the drive.

Issue	Cause	Solution
The actual motor speed does not correspond to speed reference.	Limit switch configuration interferes with speed reference from controller.	Disable limit switch function in application functions setting of the drive.
The actual motor speed or read actual speed do not correspond to speed reference.	PDO mapping is inconsistent.	Make sure that both speed reference and actual speed values have the same unit (Hz or RPM).
The brake is excessively stressed when starting hoisting movement in downwards direction.	Brake impulse is not active.	Activate brake impulse in Application functions – Brake logic of ATV71.

Chapter 11

HoistPositionSync_2: Position Synchronization of Two Crane Axes

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
11.1	Functional Overview	360
11.2	Architecture	362
11.3	Function Block Description	363
11.4	Pin Description	364
11.5	Commissioning Guide	375
11.6	Troubleshooting	376

Section 11.1

Functional Overview

Functional Overview

Functional Description

The `HoistPositionSync_2` function block for industrial cranes is designed to synchronize positions of 2 axes for simultaneous handling of large objects. It can synchronize axes with identical or different motors, gears, and encoders. The block retains information about positions of both axes and synchronization status when the crane is switched off. This means that the synchronization can continue after a power cycle of the crane.

Why Use the `HoistPositionSync_2` Function Block?

Synchronized hoists are needed to move long loads. The function block synchronizes 2 axes.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

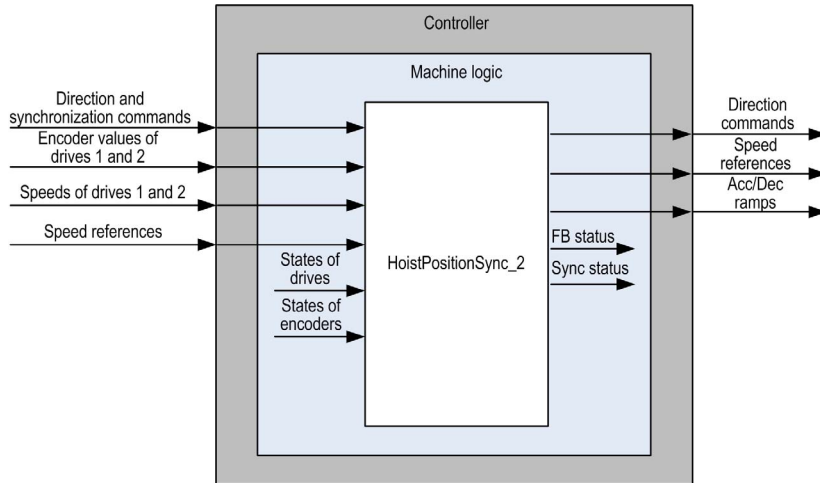
Solution with the `HoistPositionSync_2` Function Block

This function block uses information about position of both axes to synchronize them. The function block supports both incremental and absolute encoders. A difference in position between axes is used as a position deviation input to the proportional controller algorithm within the function block (hereafter referred to as the position controller). The output of the controller is then used to correct speeds of both motors.

Design and Realization Assumptions

It is assumed that when the motor moves in the forward direction, the hoist moves up and when the motor moves in reverse direction, the hoist moves down.

Functional View



Section 11.2

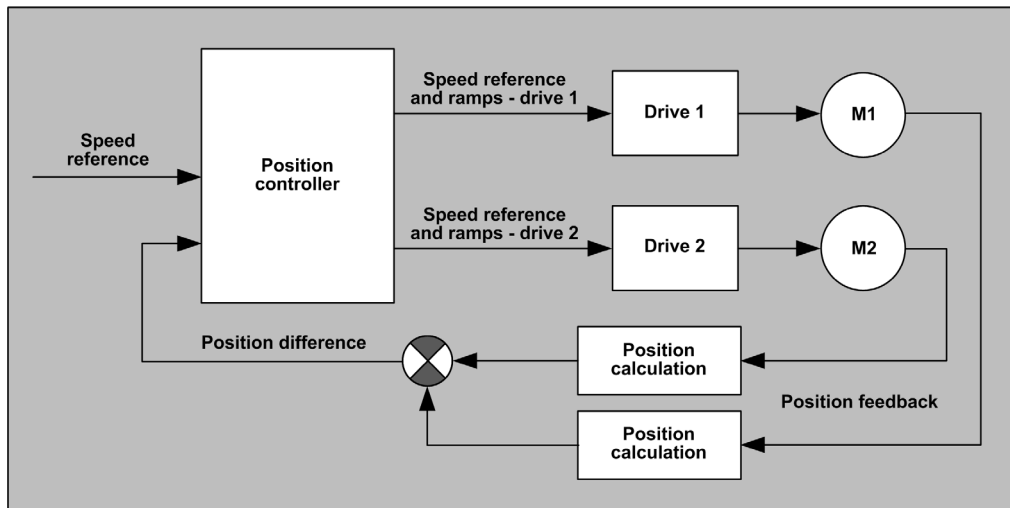
Architecture

Software Architecture

DataFlow Overview

The motors are equipped with encoders for the position feedback. The function block uses these encoder feedbacks in synchronous mode. The encoder position deviation is given to the proportional position controller. The controller corrects speeds, acceleration, and deceleration ramps of both motors to minimize the position deviation.

In non-synchronous mode, the user can operate the drives independently.



Section 11.3

Function Block Description

HoistPositionSync_2 Function Block

Pin Diagram



Function Block Description

The HoistPositionSync_2 function block has the following features:

- Synchronizes motion of 2 axes to keep their relative position.
- Supports synchronization of axes with different motors, gears, and encoders.
- Supports both absolute and incremental encoders.
- Supports permanent synchronization mode (Synchronization continues after power cycle).

Section 11.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	365
Structured Variable Description	370
Output Pin Description	373

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block FALSE: Disables the function block Refer to detailed description (see page 366) of i_xEn.
i_xSync	BOOL	TRUE: Enables the synchronization (Synchronous mode) FALSE: Disables the synchronization (Asynchronous mode) NOTE: The function block can enter Synchronous or Asynchronous mode only when both drives are stopped. Refer to detailed description (see page 367) of i_xSync.
i_xDrv1Fwd	BOOL	This is the input for the forward command for the drive 1. In Synchronous mode, this input controls both drives.
i_xDrv1Rev	BOOL	This is the input for the reverse command for the drive 1. In Synchronous mode, this input controls both drives.
i_xDrv1RdyStat	BOOL	TRUE: Drive ready FALSE: Drive not ready Refer to detailed description (see page 367) of i_xDrv1RdyStat.
i_xEnc1RdyStat	BOOL	TRUE: Position information valid FALSE: Position information invalid Refer to detailed description (see page 368) of i_xEnc1RdyStat.
i_iDrv1SpdAct1	INT	Range: -6000...6000 RPM This is the actual speed of the drive 1
i_wDrv1SpdRef	WORD	Range: 0...6000 RPM This is the target speed input for the drive 1. In Synchronous mode, it is the target speed input of both drives.
i_wDrv1EncPos	WORD	Range: 0...65535 pulses This is the position feedback of the encoder of the axis corresponding to drive 1.
i_xDrv2Fwd	BOOL	This is the input for the forward command for the drive 2. In Synchronous mode, this input does not have any effect.

Input	Data Type	Description
i_xDrv2Rev	BOOL	This is the input for the reverse command for the drive 1. In Synchronous mode, this input does not have any effect.
i_xDrv2RdyStat	BOOL	TRUE: Drive ready FALSE: Drive not ready. Refer to detailed description (see page 367) of i_xDrv2RdyStat.
i_xEnc2RdyStat	BOOL	TRUE: Position information valid FALSE: Position information invalid Refer to detailed description (see page 368) of i_xEnc2RdyStat.
i_iDrv2SpdAct1	INT	Range: -6000...6000 This is the actual speed of the drive 2.
i_wDrv2SpdRef	WORD	Range: 0...6000 RPM This is the target speed input for the drive 2. In Synchronous mode, this input does not have any effect.
i_wDrv2EncPos	WORD	Range: 0...65535 pulses This is the position feedback of the encoder of the axis corresponding to drive 2.
i_stPHPS	PHPS	Structure of parameters.
i_xRst	BOOL	detected alarm reset input.

i_xEn

By setting the i_xEn input to FALSE, the output states will be overwritten by a defined fallback state.

When the i_xEn input is set back to TRUE, the function block resumes its normal operation. If Synchronization command is present at the time of enabling of the function block and all conditions for synchronization are fulfilled, the axes enter synchronous mode at their current positions.

The fallback states of HoistPositionSync_2 are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_xSync	BOOL	FALSE
q_diEncPosDiff	DINT	0
q_xDrv1Fwd	BOOL	i_xDrv1Fwd
q_xDrv1Ref	BOOL	i_xDrv1Ref
q_xDrv2Fwd	BOOL	i_xDrv2Fwd
q_xDrv2Rev	BOOL	i_xDrv2Rev

Output	Data Type	Fallback State
q_wDrv1Acc	WORD	i_wDrv1Acc
q_wDrv1Dec	WORD	i_wDrv1Dec
q_wDrv2Acc	WORD	i_wDrv2Acc
q_wDrv2Dec	WORD	i_wDrv2Dec
q_wDrv1SpdTarg	WORD	i_wDrv1SpdTarg
q_wDrv2SpdTarg	WORD	i_wDrv2SpdTarg
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

i_xSync

This is the command to enter or leave the synchronized state. Both drives must be ready and stopped ($i_iDrv1SpdAct1$ and $i_iDrv1SpdAct1$ equal 0) and both encoder values must be valid to enter the Synchronous mode.

In Synchronous mode (i_xSync is TRUE), both axes are controlled from the command inputs of the drive 1. For example, $i_xDrv1Fwd$ starts both drives in forward direction. The input for the drive 2, for example, $i_xDrv2Fwd$ is ignored. The speed reference for both drives is given at $i_wDrv1SpdRef$; output target speeds are calculated by the FB based on the speed reference and relative position of both axes.

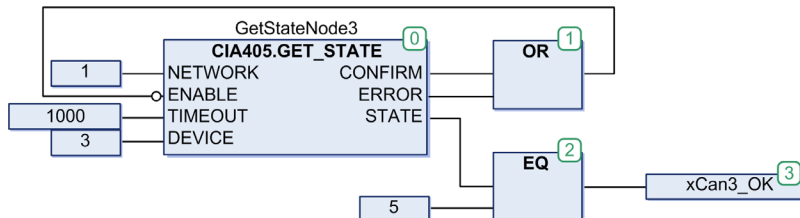
The FB can leave Synchronous mode only when both axes are stopped.

Drives 1 and 2 are controlled individually in Asynchronous mode (q_xSync is FALSE). That is, the drive 1 is controlled by the commands $i_xDrv1Fwd$ and $i_xDrv1Rev$ and the drive 2 is controlled by the commands $i_xDrv2Fwd$ and $i_xDrv2Rev$. The reference speeds are given by $i_wDrv1SpdRef$ for the drive 1 and $i_wDrv2SpdRef$ for the drive 2.

i_xDrv1RdyStat, i_xDrv2RdyStat

These inputs correspond to ready status of both drives. The information can be obtained from bit 1 of status word of Altivar drives. This information has to be interlocked (logical AND) with an information that the drive is communicating on CANopen and is in operational state.

The following figure shows an example of obtaining the CANopen network status of the hoisting drive, assuming that the drive has been assigned a Node ID of 3:



⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that the status of CANopen communication of both drives is interlocked with the ready status of respective drive.

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

If the communication with any of the 2 drives is interrupted, the FB has to get the information in order to stop the movement.

The FB needs to have information about status of both drives. If one of the drives is not ready, the FB does not authorize start of the movement. If one of the ready bits turns FALSE during movement, both drives are stopped.

i_xEnc1RdyStat, i_xEnc2RdyStat

These inputs correspond to ready status of both encoder interfaces. These inputs must be set to TRUE when values connected to inputs `i_wDrv2EncPos` and `i_wDrv2EncPos` are valid. Both of these inputs must be TRUE for the whole duration of movement in Synchronous mode.

When using the internal encoder interface of Altivar 71 and communicating with the drive via CANopen, set the corresponding input to TRUE when the drive is in an operational state (refer to description of `i_xDrv1RdyStat` and `i_xDrv2RdyStat`).

When using an external encoder interface, set the corresponding input to TRUE when this encoder interface delivers valid position value.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that both encoder interfaces deliver valid position data.

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

i_wDrv1EncPos, i_wDrv2EncPos

These inputs correspond to encoder positions of axes 1 and 2. The values may come from various sources such as the internal encoder interface of Altivar 71, absolute encoder connected through CANopen, value of bi-directional counter or others. The resolution of the encoder interface must be at least 16 bits. If the resolution is higher than 16 bits, the FB uses the lowest 16 bits of both encoder interfaces for position synchronization.

Structured Variable Description

i_stPHPS

The following table describes the structure of parameters of data type PHPS:

Structured Parameter	Data Type	Description
wSyncMaxThsh	WORD	Range: 0...65535 Maximum allowed position difference of the axes.
rKpPosSync	REAL	Range: 0.0...100.0 Proportional gain of position controller.
wDrv1SpdNom	WORD	Nominal speed 1 of axis 1 in RPM.
rDrv1SpdLin	REAL	Linear speed of axis 1 in m/s.
wDrv1EncPuls	WORD	Number of encoder pulses per revolution of motor 1.
xDrv1EncType	BOOL	TRUE: When the connected encoder is absolute. FALSE: When it is incremental type.
wDrv1SpdHsp	WORD	Range: 0...6000 RPM This is the maximum speed of the motor 1.
wDrv1SpdLsp	WORD	Range: 0...6000 RPM This is the minimum speed input for the motor 1 .
wDrv1Acc	WORD	Acceleration ramp time for the drive 1 in 0.1 s.
wDrv1Dec	WORD	Deceleration ramp time for the drive 1 in 0.1 s.
wDrv2SpdNom	WORD	Nominal speed1 of axis 2 in RPM.
rDrv2SpdLin	REAL	Linear speed of axis 2 in m/s.
wDrv2EncPuls	WORD	Number of encoder pulses per revolution of motor 2.
xDrv2EncType	BOOL	TRUE: When the connected encoder is absolute. FALSE: When it is incremental type.
wDrv2SpdHsp	WORD	Range: 0...6000 RPM This is the maximum speed of the motor 2.
wDrv2SpdLsp	WORD	Range: 0...6000 RPM This is the minimum speed input for the motor 2.

Structured Parameter	Data Type	Description
wDrv2Acc	WORD	Acceleration ramp time for the drive 2 in 0.1 s.
wDrv2Dec	WORD	Deceleration ramp time for the drive 2 in 0.1 s.

wSyncMaxThsh

If the actual position difference exceeds this value, the FB raises an alarm and stops the movement of both axes.

When 2 axes are not identical, actual position of the second axis is re-calculated to fit the scaling of the first axis.

Example:

If $wDrv1EncPuls = 1024$ and $wDrv2EncPuls = 4096$, and the alarm should be raised when position difference exceeds 2 motor revolutions, set $wSyncMaxThsh$ to 2048.

rKpPosSync

This parameter corresponds to proportional gain of position controller. Start with a low gain (for example, 0.01) and increase it slowly.

When the gain is too low, position controller cannot compensate for position difference. When the gain is too high, the position controller over-compensates and causes oscillations of speeds of both drives and the position difference.

The proportional gain of position controller is internally normalized and is therefore independent of the number of encoder pulses per revolution. It is not necessary to find a new value when a different encoder is used.

wDrv1SpdNom, wDrv2SpdNom

These parameters corresponds to nominal speed of both axes in RPM. Enter the synchronous values. (for example, 1500 for 4-pole motor at 50 Hz and 1800 for 4-pole motor at 60 Hz). These values are used for calculation of gear ratio when the synchronized axes are not identical. When the axes are identical, leave these inputs unconnected.

rDrv1SpdLin, rDrv2SpdLin

These parameters corresponds to linear speed of both axes in m/s. These values are used for calculation of gear ratio when the synchronized axes are not identical. When the axes are identical, leave these inputs unconnected.

wDrv1EncPuls, wDrv2EncPuls

These parameters corresponds to number of encoder pulses per revolution of motors 1 and 2. These values are used for calculation of encoder positions of both axes when the used encoders are not identical. When the axes are identical, leave these inputs unconnected.

xDrv1EncType, xDrv2EncType

These parameters are TRUE when the connected encoder is absolute and FALSE when it is incremental type. When an incremental encoder is used, the FB stores actual position of the axis when the crane is power cycled. When an absolute encoder is used, the actual position is retained by the encoder.

It is necessary to select correct encoder type for both encoders. The FB supports mixed encoder types on axes 1 and 2.

Incremental encoder does not register movement of the axis while the encoder interface is powered Off. If the axis can move while the encoder interface is not powered, use a multi turn absolute encoder.

wDrv1SpdHsp, wDrv2SpdHsp

These parameters define the maximum allowed speed of both motors in RPM. The values are used in both Asynchronous and Synchronous modes. In Synchronous mode, the speed of synchronous movement is limited by the slower axis. Gear ratios are considered when the axes are not identical.

wDrv1SpdLsp, wDrv2SpdLsp

These parameters define the minimum allowed speed of both motors in RPM. The values are used in both Asynchronous and Synchronous modes. In Synchronous mode, the speed of synchronous movement is limited by the faster axis. Gear ratios are considered when the axes are not identical. If possible, keep both of these values at 0. If one or both of the values need to be higher than 0 (for example, because of using other than closed loop motor control mode), set the values as low as possible.

wDrv1Acc, wDrv2Acc

These parameters correspond to acceleration ramp times for both axes. In Synchronous mode, acceleration time value for drive 1 is used to calculate acceleration time values for both axes. The values are calculated in every cycle and must be written to both drives. Correction of acceleration time is important part of the position synchronization process.

wDrv1Dec, wDrv2Dec

These parameters correspond to deceleration ramp times for both axes. In Synchronous mode, deceleration time value for drive 1 is used to calculate deceleration time values for both axes. The values are calculated in every cycle and must be written to both drives. Correction of deceleration time is important part of the position synchronization process.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled FALSE: Function block disabled
q_xSync	BOOL	TRUE: Synchronous mode FALSE: Asynchronous mode
q_diEncPosDiff	DINT	Position difference of axes 1 and 2 Range: -2147483648 ... 2147483647 pulses
q_xDrv1Fwd	BOOL	Forward command for drive 1
q_xDrv1Rev	BOOL	Reverse command for drive 1
q_wDrv1SpdTarg	WORD	Target speed for drive 1 Range: 0...6000 RPM
q_wDrv1Acc	WORD	Drive 1 acceleration time Range: 1...9999 Scaling/Unit: 0.1 s
q_wDrv1Dec	WORD	Drive 1 deceleration time Range: 1...9999 Scaling/Unit: 0.1 s
q_xDrv2Fwd	BOOL	Forward command for drive 2
q_xDrv2Rev	BOOL	Reverse command for drive 2
q_wDrv2SpdTarg	WORD	Target speed for drive 2 Range: 0...6000 RPM
q_wDrv2Acc	WORD	Drive 2 acceleration time Range: 1...9999 Scaling/Unit: 0.1 s
q_wDrv2Dec	WORD	Drive 2 deceleration time Range: 1...9999 Scaling/Unit: 0.1 s
q_xAlrm	BOOL	TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	This is the detected alarm identification.

q_diEncPosDiff

This is the actual difference of encoder positions in the Synchronous mode. The output is reset to 0 in Asynchronous mode.

q_wDrv1Acc, q_wDrv2Acc

These parameters correspond to acceleration ramp times for both drives. In Synchronous mode, these values must be written to acceleration time parameters of both drives in every cycle. Correction of acceleration time is important part of the position synchronization process.

q_wDrv1Dec, q_wDrv2Dec

These parameters correspond to deceleration ramp times for both drives. In Synchronous mode, these values must be written to deceleration time parameters of both drives in every cycle. Correction of deceleration time is important part of the position synchronization process.

q_xAlrm

This output corresponds to the alarm output of the FB. TRUE indicates an detected alarm state. FALSE indicates an no alarm detected state.

q_wAlrmId

Alarm identification output contains information about actually present alarms.

Bit	Alarm Description
0	The actual position difference <code>q_diEncPosDiff</code> is greater than the <code>i_wSyncMaxThsh</code> .
1	At least one of drives is not ready, <code>i_xDrv1RdyStat</code> or <code>i_xDrv2RdyStat</code> input is FALSE.
2	At least one of encoder values is not valid, <code>i_xEnc1RdyStat</code> or <code>i_xEnc2RdyStat</code> input is FALSE.
3	The <code>i_wDrv1SpdHsp</code> value is smaller than or equal to <code>i_wDrv1SpdLsp</code> .
4	The <code>i_wDrv2SpdHsp</code> value is smaller than or equal to <code>i_wDrv2SpdLsp</code> .
5	Combination of high and low speeds of both axes does not allow synchronization with actual speed ratio. Verify nominal and linear speeds of both axes and validate it at high and low speeds.
6	At least one of the linear speeds of both motors is equal or lower than 0. At the same time, at least one of nominal speeds or encoder pulses values is higher than 0. The linear speed values may be equal to 0 only if the nominal speeds and encoder pulses inputs equal to 0 as well.
7	At least one of the nominal speeds of both motors equals 0. At the same time, at least one of linear speeds or encoder pulses values is higher than 0. The nominal speed values may be equal to 0 only if the linear speeds and encoder pulses inputs equal to 0 as well.
8	At least one of the encoder pulse values of both motors equals 0. At the same time, at least one of linear speeds or nominal speeds values is higher than 0. The encoder pulses values may be equal to 0 only if the linear speeds and nominal speeds inputs equal to 0 as well.

The actual position difference detected alarm (bit 0) is reset by re-synchronizing the axes.

Other detected alarms are reset by rising edge on the `i_xRst` input.

Section 11.5

Commissioning Guide

Commissioning Guide

Commissioning of Variable Speed Drives

1. Set both drives to factory settings.
2. Configure correct motor control parameters.
3. Configure brake logic control.
4. Perform motor tuning and encoder test (if motor encoder is present).
5. Set the motor control mode to desired mode (preferably FVC).
6. In drive menu application functions, disable preset speeds. Set command and reference channel to CANopen.
7. Configure CANopen addresses and bus speed, drive power cycle is needed after change of CANopen configuration.

Commissioning of Hoist Position Synchronization 2 Function Block

1. Without moving the motors, check that all commands and target speeds are received correctly from the operator interface.
2. Instantiate the Hoist position synchronization 2 FB in your program and parameterize it with initial parameters.
3. Instantiate it in a program executed in a cyclic task. Preferably in the Master task. The same task should be a parent task for CANopen communication. Master task is the parent task by default.
4. Configure the execution period of the corresponding task.
5. Download the application, put it to run and check status of CANopen communication.
6. Set the proportional gain for position synchronization `i_stPHPS.rKpPosSync`. The default gain is 0.05 and the FB internally compensates for resolution of the encoder. Therefore, it is not necessary to change the value when encoder with different resolution is used.
7. Test the behavior of position synchronization. Move the axes in both directions by giving command to the drive 1. Take a trace of speeds of both motors and the position difference output `q_diEncPosDiff`.

When the position difference is greater than required and if the motor speeds are not oscillating, increase the value of `i_stPHPS.rKpPosSync`. If motor speeds and torques begin to oscillate due to the increase of the parameter, decrease the value of `i_stPHPS.rKpPosSync`.

Section 11.6

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The bit 0 of <code>q_wAlrmId</code> is TRUE.	Setting of the proportional gain could be incorrect (<code>i_stPHPS.rKpPosSync</code>) or the position difference tolerance threshold could be too low.	Verify the proportion gain of the function block and tune this parameter. Verify the value of <code>i_dwSyncMaxThsh</code> and increase it if it is too low.
Detected position error increases during movement.	Encoder or motor phases are wired or configured incorrectly	Verify wiring and setting of encoders and motors.
The bit 1 of <code>q_wAlrmId</code> is TRUE.	At least one of the drives is not ready state or there is a communication interruption.	Verify the CANopen connection, and establish the CANopen communication between controller and drives. Reset detected alarm state of both drives.
Detected position error increases during direction change.	One of the brakes closes when zero speed is reached.	Set brake engage delay to value higher than zero in drive configuration.
One of the drives is in detected alarm state or lost connection to the controller and the other drive continues to run.	Function block does not have information about event of the drive connected or bus supervision is not configured.	Connect the information about status of the CANopen node to corresponding ready status input together with information about ready status of the drive.
The actual motor speed does not correspond to speed reference.	Preset speeds configuration interferes with speed reference from the controller.	Disable preset speeds in application functions setting of the drive.

Part IX

Limit Switch Management

Overview

This part explains the functionality and implementation of `LimitSwitch` function block.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
12	LimitSwitch: Surveillance of the Used Limit Switches of a Crane with Fast/Slow Zone Identification	379
13	LimitSwitch_AR: Monitoring of the Limit Switches of a Crane with Adaptive Speed Reference in Slow Zone	403
14	DoubleLimitSwitch_AR: Administers Two Devices in Synchronous Mode	425
15	DoubleLimitSwitch_AR_2: Administers Two Devices in Synchronous Mode	449

Chapter 12

LimitSwitch: Surveillance of the Used Limit Switches of a Crane with Fast/Slow Zone Identification

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
12.1	Functional and Machine Overview	380
12.2	Architecture	384
12.3	Function Block Description	388
12.4	Pin Description	396
12.5	Troubleshooting	401

Section 12.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	381
Machine Overview	383

Functional Overview

Functional Description

The `LimitSwitch` function block is applicable to the following types of cranes:

- Industrial Cranes (trolley, bridge and hoisting movement)
- Construction Cranes (trolley, slewing and hoisting movement)

The `LimitSwitch` function block reads limit switch inputs from the field. It checks the limit switch status and generates the control outputs which are used to control the movements of the crane.

The function block is designed for use with cross and screw limit switches using Normally Closed (NC) contacts. The contacts indicate the specific status, for example, the stop position.

Why Use the `LimitSwitch` Function Block?

The `LimitSwitch` function block monitors and controls the movement of the trolley/bridge to help prevent it from crashing into the mechanical barrier at each end of the rails.

This block can also be used for hoisting and slewing movement. For example, when the block is used in hoisting movement, it stops the hook from crashing into the trolley or from over-winding the drum.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

Solution with the LimitSwitch Function Block

The function block has various options:

- 2 slow switches on each end
- 2 slow switches and 2 stop switches on each end
- 1 slow switch and 1 stop switch on each end
- Other combinations are also possible,

For example, if only the slow-switch combination is used, then the stop can be executed using the integrated Stop on Distance function which automatically calculates the remaining distance and brings the system to a stop.

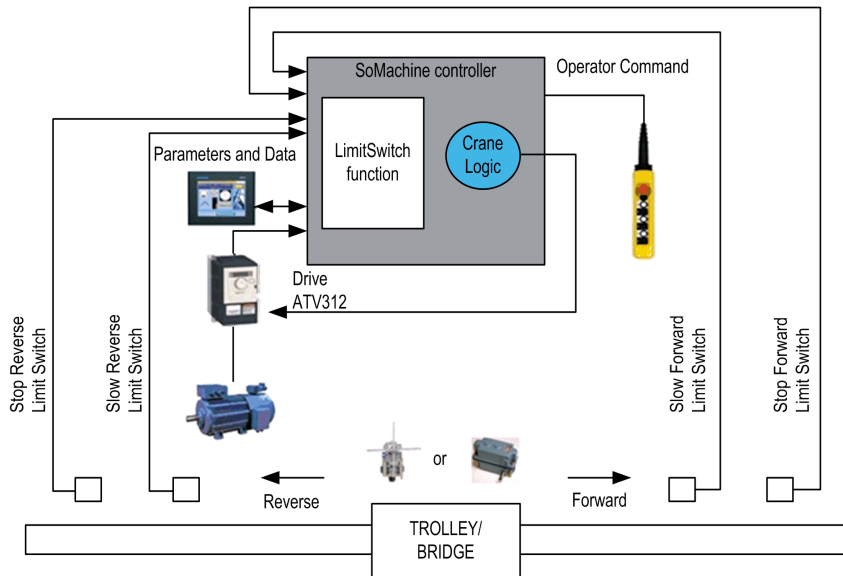
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use impulse or normally open contact switches in association with the function blocks.
- Only use Normally Closed contact switches with the function blocks.

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

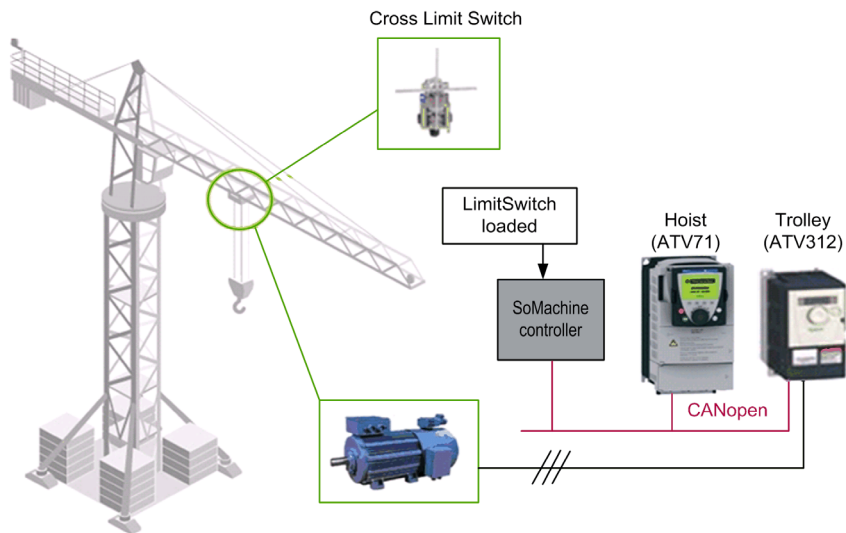
Functional View



Machine Overview

Machine View

The following figure illustrates a machine view of Limit switch management being used in a tower crane.



Section 12.2

Architecture

What Is in This Section?

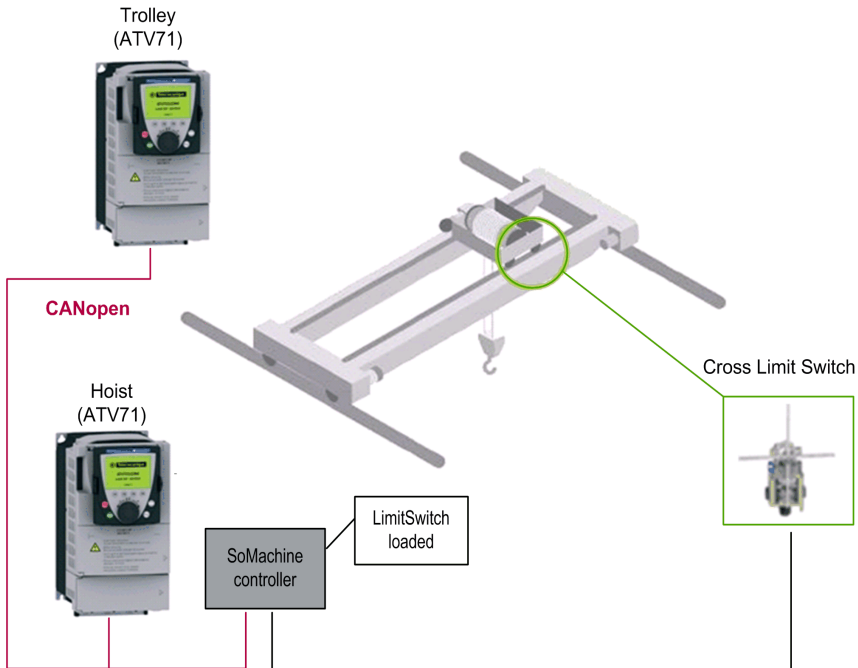
This section contains the following topics:

Topic	Page
Hardware Architecture	385
Software Architecture	387

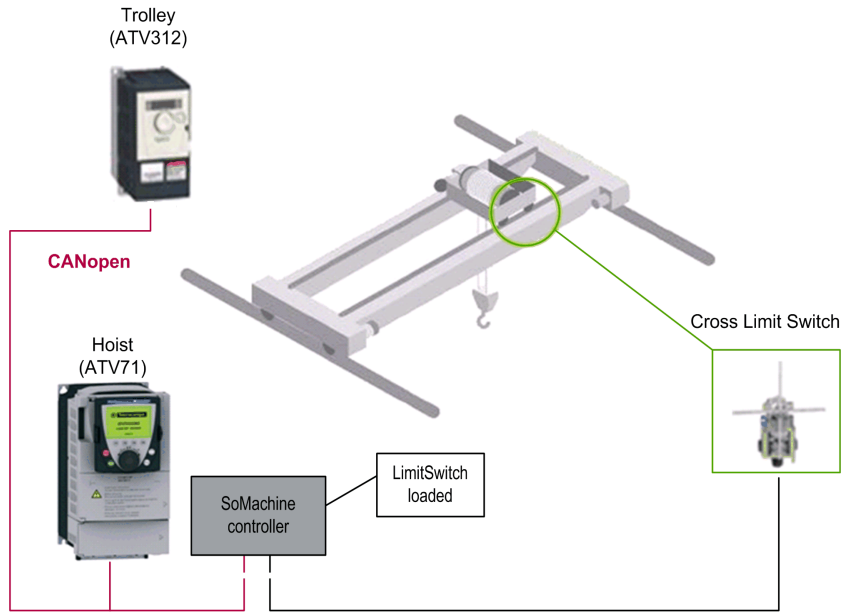
Hardware Architecture

Hardware Architecture Overview

Hardware Architecture with ATV71

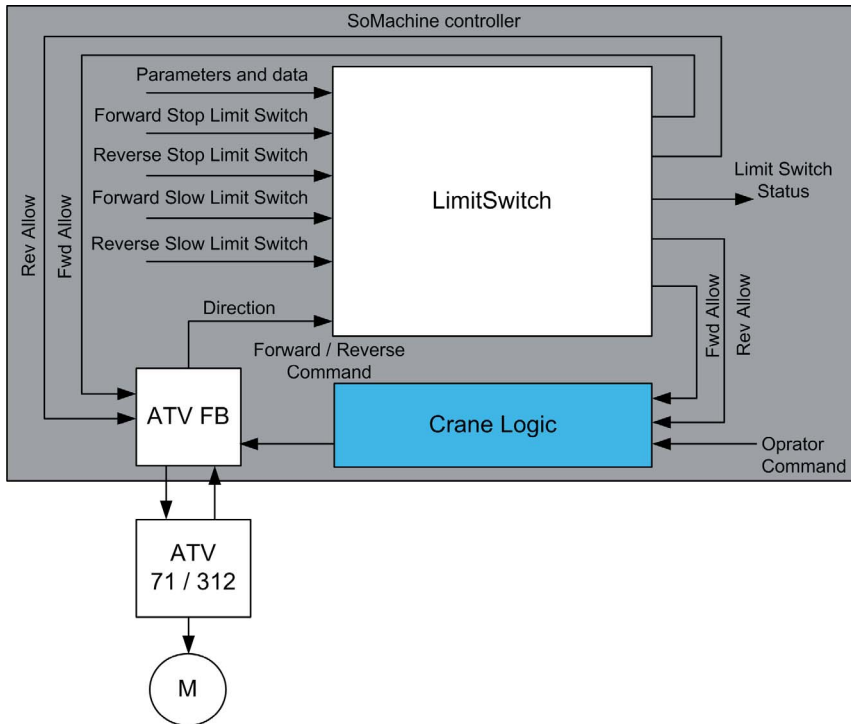


Hardware Architecture with ATV312



Software Architecture

Data Flow Overview



Section 12.3

Function Block Description

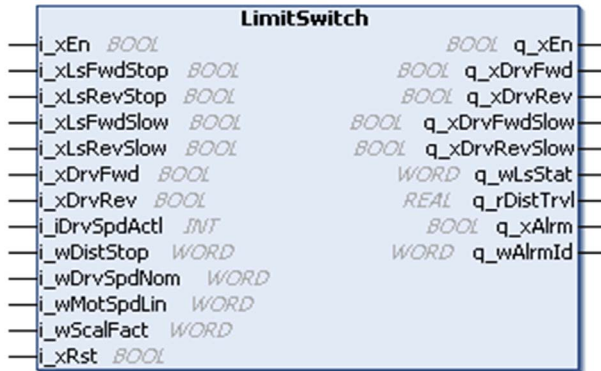
What Is in This Section?

This section contains the following topics:

Topic	Page
LimitSwitch Function Block	389
Configuration of the Movement Positions	390
Timing Chart	394

LimitSwitch Function Block

Pin Diagram

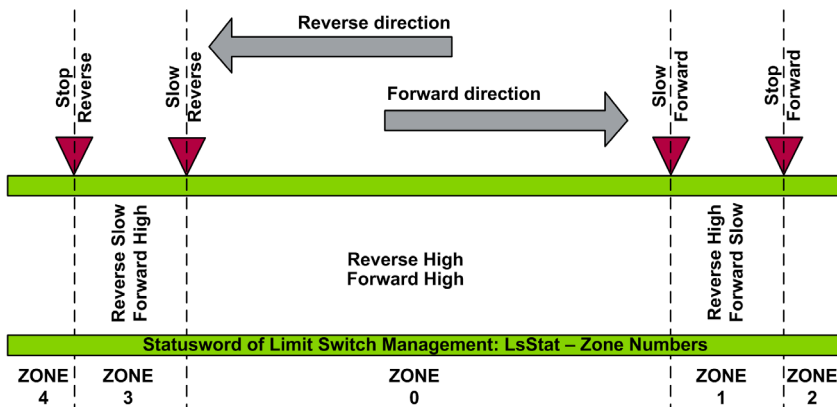


Function Block Description

The `LimitSwitch` function block handles up to 4 movement positions for trolley, bridge, hoisting or slewing. This includes the following:

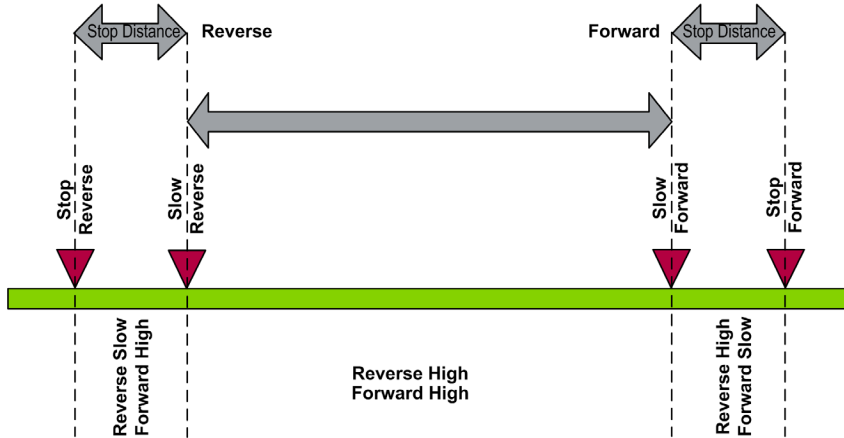
- Forward stop position
- Forward slow position
- Reverse stop position
- Reverse slow position

The following figure shows the zones of the `q_wLsStat` (status word output of the function block).

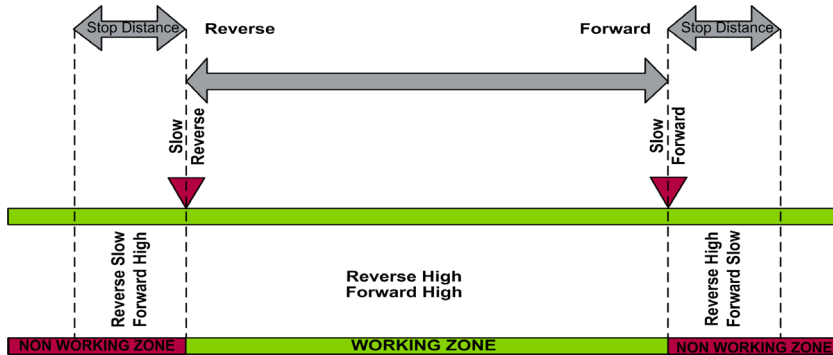


Configuration of the Movement Positions

Fwd/Rev Stop and Fwd/Rev Slow: Configuration of 4 Positions

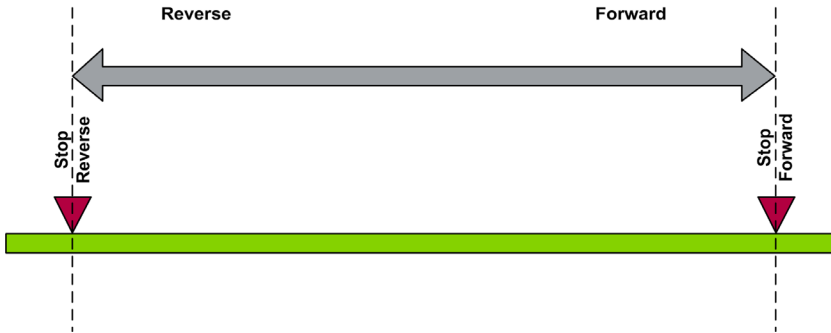


Fwd/Rev Slow: Configuration with Stop on Distance

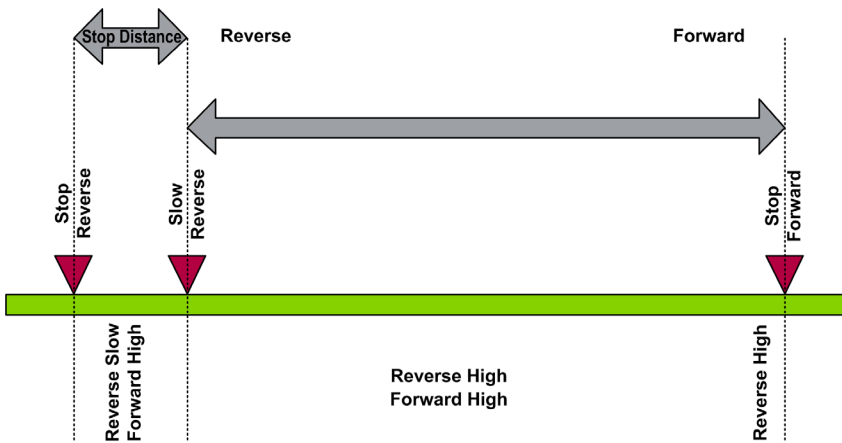


NOTE: If a Stop on Distance is executed while in the non-working zone, movement towards the slow switch is possible, but movement in the opposite direction is blocked until the slow limit switch is passed.

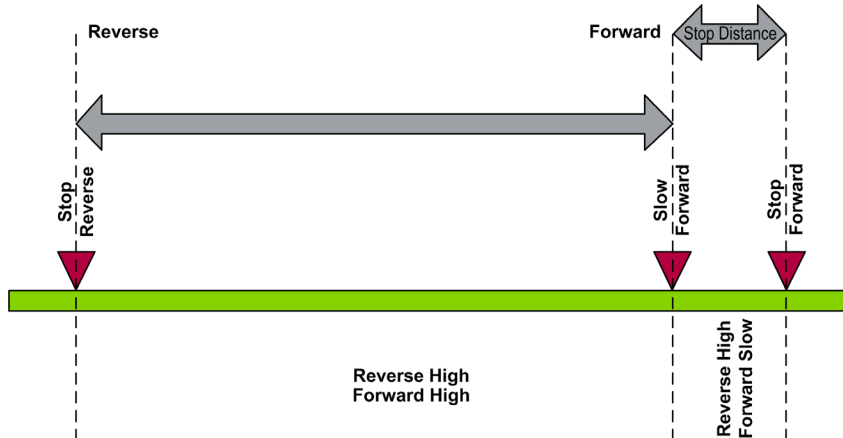
Fwd/Rev Stop: Configuration of 2 Positions



Fwd/Rev Stop and Rev Slow: Configuration of 3 Positions



Fwd/Rev Stop and Fwd Slow: Configuration of 3 Positions



Remark

If any of the limit switch positions are not used, the related input on the function block must be set to TRUE, as the `LimitSwitch` function block is designed for N.C. configuration.

Normal Cycle

When the system is moving in the forward direction, and Forward Slow position (`i_xLsFwdSlow`) is initiated, the function block enables the Forward Slow signal (`q_xDrvFwdSlow`). When the Forward Stop position (`i_xLsFwdStop`) is initiated, the function block turns off the Forward Allow signal (`q_xDrvFwd`).

When the system is moving in the reverse direction and Reverse Slow position (`i_xLsRevSlow`) is initiated, the function block enables the Reverse Slow signal (`q_xDrvRevSlow`). When the Reverse Stop position (`i_xLsRevStop`) is initiated, the function block turns off the Reverse Run signal (`q_xDrvRev`).

Stop on Distance

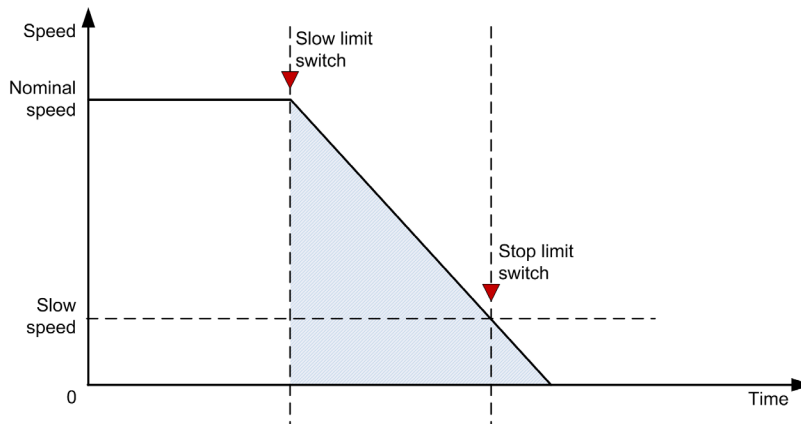
The function block has the functionality to stop the trolley/bridge on distance after passing the Slow Forward or Slow Reverse position. To enable this functionality, enter any value greater than zero at `i_wDistStop`. If the input is equal to zero, the Stop on Distance function is disabled.

The function block converts the actual RPM from the drive to the equivalent actual linear speed in m/s. From this conversion, the function block calculates the traveled distance. When the traveled distance is greater than the configured stop distance, the Forward Run signal (`q_xDrvFwd`) or the Reverse Run signal (`q_xDrvRev`) is turned off (depending on which way the crane is moving).

Example:

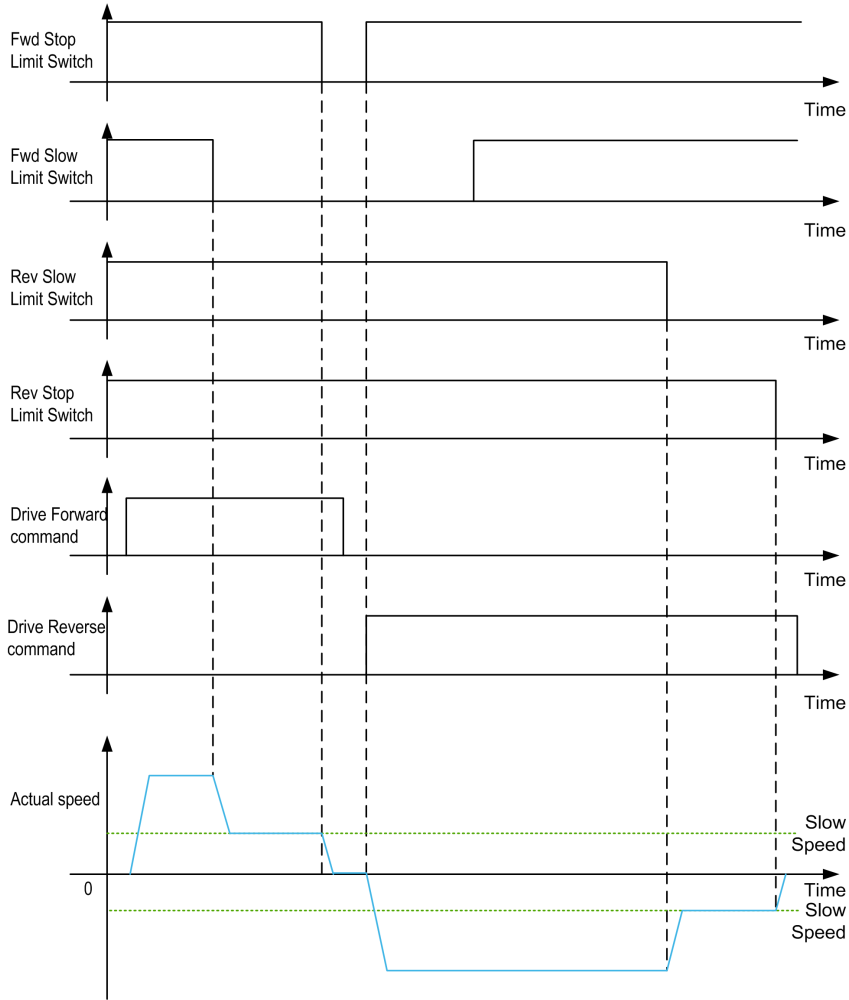
- Stop distance: 3 m
- Nominal speed of the drive: 1500 RPM
- Nominal linear speed: 1 m/s
- If the actual speed of the drive = 600 RPM, the actual linear speed (m/s) = $1 \text{ m/s} * 600 \text{ RPM} / 1500 \text{ RPM} = 0.4 \text{ m/s}$
- Distance traveled in meters during one sample time = $((0.4 \text{ m/s} * \text{Sample Rate in ms})/1000)$
- When the distance traveled is greater than the stop distance, the drive stops and further movement in the same direction is not allowed.

The following figure represents the speed time curve of LimitSwitch function block.

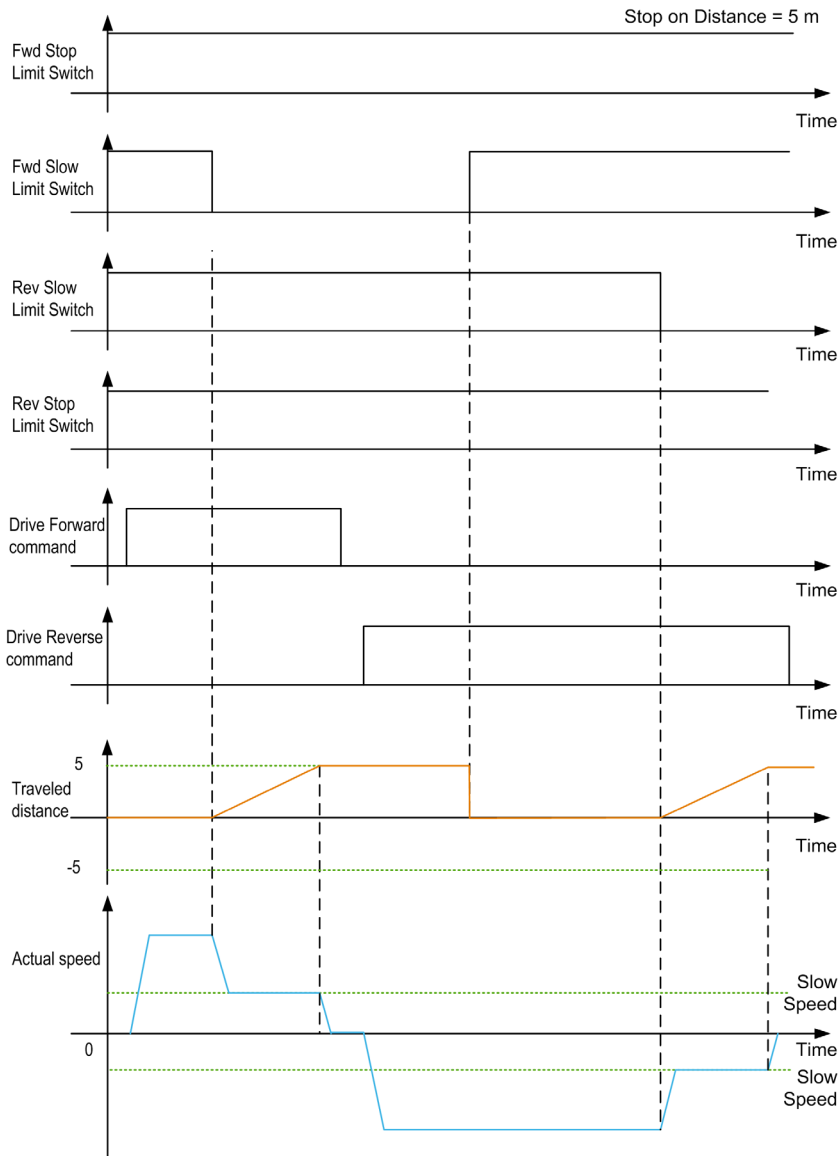


Timing Chart

Timing and Signals Diagram with Limit Switch



Timing and Signals Diagram with Stop on Distance



Section 12.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	397
Output Pin Description	399

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (see page 398) of i_xEn.
i_xLsFwdStop	BOOL	Limit switch Forward Stop position. Used to stop movement of the trolley/bridge in the forward direction. Normally it is closed. TRUE: Set to this condition when not used by a switch.
i_xLsRevStop	BOOL	Limit switch Reverse Stop position. Used to stop movement of the trolley/Bridge in the reverse direction. Normally it is closed. TRUE: Set to this condition when not used by a switch.
i_xLsFwdSlow	BOOL	Limit switch Forward Slow position. Used to move the trolley/bridge at low speed in the forward direction. Normally it is closed. TRUE: Set to this condition when not used by a switch.
i_xLsRevSlow	BOOL	Limit switch Reverse Slow position. Used to move the trolley/bridge at low speed in the reverse direction. Normally it is closed. TRUE: Set to this condition when not used by a switch.
i_xDrvFwd	BOOL	Drive Forward run command. TRUE: Forward FALSE: Not forward
i_xDrvRev	BOOL	Drive Reverse run command. TRUE: Reverse FALSE: Not reverse
i_iDrvSpdAct1	INT	Drive actual motor speed. This value is provided as a feedback signal from the drive. Range: -6000...6000 RPM
i_wDistStop	WORD	Allowed distance for travel after passing the slow position when Stop on Distance functionality is used. Range: 0...65535 Scaling/Unit: 0.1 m If set to zero or left unconnected functionality is disabled.
i_wDrvSpdNom	WORD	Drive nominal motor speed used in distance calculation. Range: 0...6000 RPM Factory setting: 1500 RPM

Input	Data Type	Description
i_wMotSpdLin	WORD	Linear speed of movement at Nominal Motor Speed Range: 0...500 Scaling/Unit: 0.01 m/s
i_wScalFact	WORD	Scale factor for correction of distance calculation Range: 0...200% Factory setting: 100%
i_xRst	BOOL	On a rising edge, attempts to clear all alarms. If alarms stay active despite the rising edge of this input, the cause of the alarm is still present.

The linear speed of movement is entered at this pin in m/s when the motor is at nominal motor speed (i_wDrvSpdNom). This is used in the distance calculation.

Example:

Nominal speed of motor in RPM: 1500

Nominal linear speed in m/s: 1.00

When the motor runs at 1500 RPM, the nominal speed of trolley/bridge is 1.0 m/s.

LxEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of LimitSwitch are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_xDrvFwd	BOOL	FALSE
q_xDrvRef	BOOL	FALSE
q_xDrvFwdSlow	BOOL	FALSE
q_xDrvRevSlow	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_xDrvFwd	BOOL	This is Forward Run command output to the drives TRUE: Forward FALSE: Not forward
q_xDrvRev	BOOL	This is Reverse Run command output to the drives. TRUE: Reverse FALSE: Not reverse
q_xDrvFwdSlow	BOOL	This is Forward Slow run command output to the drives. TRUE: Forward slow FALSE: Not forward slow
q_xDrvRevSlow	BOOL	This is Reverse Slow run command output to the drives. TRUE: Reverse slow FALSE: Not reverse slow
q_wLsStat	WORD	Status of limit switch management Status: 0, 1, 2, 3, 4
q_rDistTrvl	REAL	Output of the Stop on Distance calculation. Whenever this value exceeds the value of <code>i_wDistStop</code> , a stop is executed. NOTE: Calculated traveled distance <code>q_rDistTrvl</code> is retained after warm start. Range: 1.175494351e-38F...3.402823466e+38F m
q_xAlrm	BOOL	Detected alarm bit. NOTE: On detection of Alarm, the outputs <code>q_xDrvFwd</code> , <code>q_xDrvFwdSlow</code> , <code>q_xDrvRev</code> , <code>q_xDrvRevSlow</code> are set to FALSE. TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification

q_wLsStat

Indicates the status of the `LimitSwitch` function block where:

- 0: trolley/bridge in the working area
- 1: trolley/bridge between the Slow Forward and Stop Forward positions
- 2: trolley/bridge beyond the Slow Forward and Stop Forward positions
- 3: trolley/bridge between the Slow Reverse and Stop Reverse positions
- 4: trolley/bridge beyond the Slow Reverse and Stop Reverse positions

Notifications

Bit Number	Description
0	incorrect order of limit switch signals detected
1	<code>i_wDrvSpdNom</code> is zero although Stop on Distance is enabled
2	<code>i_wMotSpdLin</code> is zero although Stop on Distance is enabled
3	<code>i_wScalFact</code> is zero although Stop on Distance is enabled

Section 12.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The bit 0 of <code>q_wAlrmId</code> is TRUE.	Wrong order of the limit switch detected.	Check for the wiring of the limit switches. The function block is designed for the NC type of input signals from the limit switches only.
The bits 1...3 of <code>q_wAlrmId</code> are TRUE.	The parameterization of the function block is incorrect if the stop on distance function is enabled.	Check the parameterization of the function block.
Raised alarm in first cycle for Limit switch error bit 0.	Limit switches are connected on CANopen devices.	Create filter for first cycle until CANopen devices are refreshed and Limit switch information updated.

Chapter 13

LimitSwitch_AR: Monitoring of the Limit Switches of a Crane with Adaptive Speed Reference in Slow Zone

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
13.1	Functional Overview	404
13.2	Architecture	406
13.3	Function Block Description	409
13.4	Pin Description	413
13.5	Quick Reference Guide	420

Section 13.1

Functional Overview

Functional Overview

Functional Description

The `LimitSwitch_AR` function block interprets the information received from limit switch sensors in the system, so that the information can be easily exploited by the user application in a controller.

Why Use the `LimitSwitch_AR` Function Block?

This function block helps to prevent moving parts of a crane from reaching areas out of their operational range. It can be used on the trolley, bridge, hoist or slewing movement of any crane equipped with limit switch sensors.

The adaptive ramp feature allows higher performance of a crane while moving in the slow-down area by calculating the maximum allowed velocity at the actual position rather than simply slowing down to a pre-defined slow speed after a slow-down limit switch is activated.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

Solution With the LimitSwitch_AR Function Block

The function block has 2 inputs for stop limit switches and 2 inputs for slow-down limit switches (forward and reverse).

All signals must come from Normally Closed switches. When an axis enters a slow or stop area, the limit switch must give a constant signal.

NOTE: Impulse limit switches are not supported.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

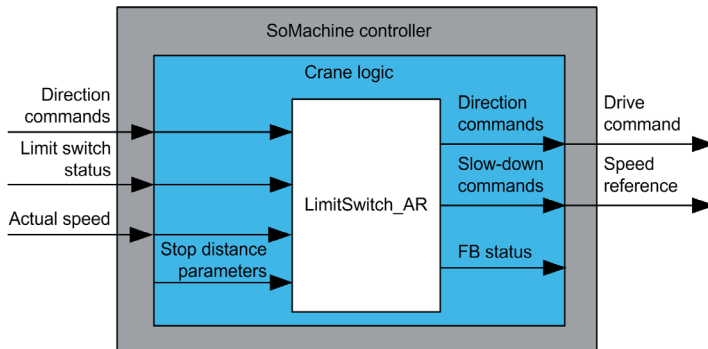
- Do not use impulse or normally open contact switches in association with the function blocks.
- Only use Normally Closed contact switches with the function blocks.

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

An arbitrary configuration of inputs is supported. Inputs that are not used must be set to TRUE.

The adaptive ramp function calculates the maximum speed the axis can move in a slow-down area depending on the calculated position (without using a position sensor). This solves the problem of a standard limit switch which limits the speed to a pre-defined slow speed directly after entering the slow-down area and significantly increases performance of cranes which have to enter the slow-down area regularly.

Functional View



Section 13.2

Architecture

What Is in This Section?


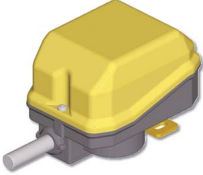
This section contains the following topics:

Topic	Page
Hardware Architecture	407
Software Architecture	408

Hardware Architecture

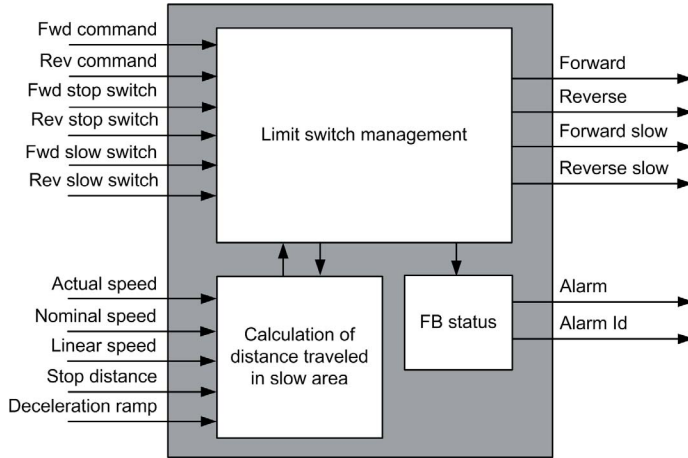
Necessary Equipment to Fulfill Limit Switch Functionality

The following table includes a list of the devices used in the system, including the description.

Symbol	Description	Remarks
	Limit switch, cross type	XCKMR
	Limit switch, screw type	3rd party product

Software Architecture

Data Flow Overview



Section 13.3

Function Block Description

LimitSwitch_AR Function Block

Pin Diagram



Function Block Description

This function block handles up to 4 movement positions for trolley, bridge, hoisting or slewing:

- Forward stop position
- Forward slow position
- Reverse stop position
- Reverse slow position

Remark

If any of the limit switch positions are not used, the related input on the function block must be set to TRUE, as the limit switch management function is designed for NC configuration.

Standard Limit Switch Behavior

The standard limit switch function block (*see page 379*) allows the axis to move at any given speed until either of the slow limit switches is reached. To enable this function, enter a value of zero into the `i_rDistStop` input. Any other value enables the adaptive ramp function.

Example:

When the system moves in the forward direction and the Forward slow position `i_xLsFwdSlow` is reached, the function block enables the Forward slow signal `q_xDrvFwdSlow`. When the Forward stop position `i_xLsFwdStop` is reached, the function block turns OFF the Forward command output signal `q_xDrvFwd`.

When the system moves in the reverse direction and the Reverse slow position `i_xLsRevSlow` is reached, the function block enables the Reverse slow signal `q_xDrvRevSlow`. When the Reverse stop position `i_xLsRevStop` is reached, the function block turns OFF the reverse command output signal `q_xLsRevStop`.

Adaptive Ramp Behavior

The adaptive ramp function allows the axis to move at any given speed as long as it is possible. To enable this function, enter the desired stop distance in meters into the `i_rDistStop` input. If a value of zero is entered, the adaptive ramp function will be disabled.

Example:

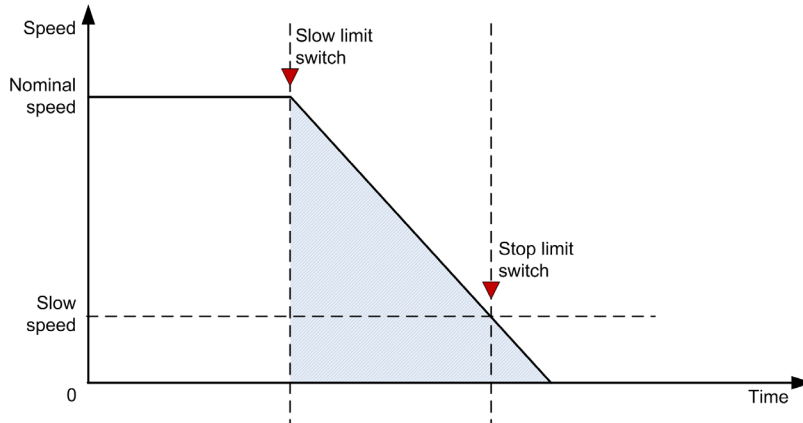
When the system is moving in the forward direction and the Forward slow position `i_xLsFwdSlow` is initiated, the function block enables the internal calculation of the remaining distance according to the actual speed. The function block calculates the traveled distance by integrating actual speed of the drive over time. The adaptive ramp function allows the FB to calculate the highest available speed while the axis is in a slow-down area.

Once the last possible point in time is reached to slow down the movement according to the chosen ramp, the Forward slow signal `q_xDrvFwdSlow` is set TRUE and the movement is stopped. When the Forward stop position `i_xLsFwdStop` is initiated, the function block turns OFF the Forward allow signal `q_xDrvFwd`. The function works the same for the reverse direction.

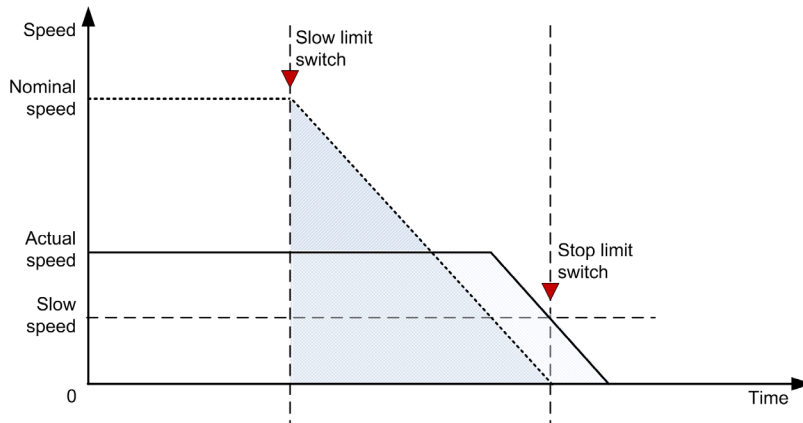
The following figures describe the benefit of using an adaptive ramp function.

NOTE: The following chart shows the actual speed of the drive.

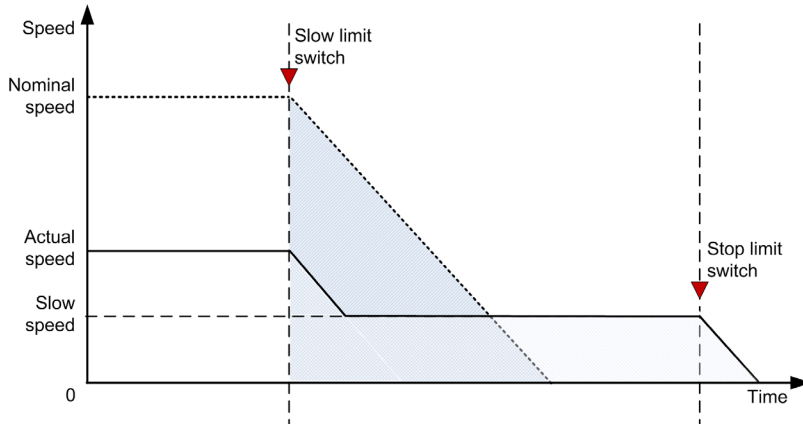
The first chart shows the behavior when entering a properly configured slow-down area at nominal speed:



The second chart describes entering the slow-down area at a speed that is approximately half of the nominal speed with an adaptive ramp function:



The third chart shows the behavior without the adaptive ramp function. The shaded area corresponds to the distance traveled in slow-down area at a given speed:



The function block can work in 2 modes depending on the value of the `i_rDistStop` input:

Mode	Meaning
Adaptive ramp	The FB removes the direction command after the preset stop distance has been traveled.
Slow-down only	After the Slow switch became FALSE, the <code>q_xDrvFwdSlow</code> or <code>q_xDrvRevSlow</code> is set to TRUE, until the Hardware Stop Limit switch is reached.

Section 13.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	414
Output Pin Description	418

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	<p>TRUE: Enables the function block. FALSE: Disables the function block.</p> <p>NOTE: When FB is disabled, q_xEn goes to FALSE, output q_xAlrm, q_xDrvFwd, q_xDrvFwdSlow, q_xDrvRev and q_xDrvRevSlow are FALSE, q_wAlrmId is set to zero.</p>
i_xLsFwdStop	BOOL	<p>Stop forward position. When the Stop forward is initiated the Forward output command (q_xDrvFwd) signal is disabled. This signal is used to stop movement of the trolley/bridge in the forward direction.</p> <p>NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.</p>
i_xLsRevStop	BOOL	<p>Stop reverse position. When the Stop reverse is initiated the Reverse output command (q_xDrvRevMstr) signal is disabled. This signal is used to stop movement of the trolley/bridge in the reverse direction.</p> <p>NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.</p>
i_xLsFwdSlow	BOOL	<p>Slow-down forward position. The Slow forward initiates the Slow forward (q_xDrvFwdSlow) signal. This signal is used to move the trolley/bridge at low speed in the forward direction.</p> <p>NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.</p>
i_xLsRevSlow	BOOL	<p>Slow-down reverse position. The Slow reverse initiates the Slow reverse (q_xDrvRevSlow) signal. This signal is used to move the trolley/bridge at low speed in the reverse direction.</p> <p>NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.</p>
i_xDrvFwd	BOOL	<p>Drive forward run command. TRUE: Forward FALSE: Not forward</p>

Input	Data Type	Description
i_xDrvRev	BOOL	Drive reverse run command. TRUE: Reverse FALSE: Not reverse
i_iDrvSpdAct1	INT	Actual drive speed. These values are provided as a feedback signal from the drives. Range: -6000...+6000 Scaling/Unit: 1 RPM
i_rDistStop *	REAL	Stop Distance. Allowed distance for travel after passing the slow position, when Stop on Distance is used. To disable the Stop on Distance function, these inputs must be set to zero. Range: 1.175494351e-38F ... 3.402823466e+38F Scaling/Unit: 1 m Refer to detailed description below this table.
i_wDrvSpdNom *	WORD	Nominal motor speed used in the distance calculation. Range: 0...6000 Scaling/Unit: 1 RPM
i_rMotSpdLin *	REAL	Linear speed of movement at nominal motor speed used in the distance calculation. Range: 0...5 Unit: m/s
i_wScalFact *	WORD	Scaling factor for correction of distance calculation. Range: 0...200 Scaling/Unit: 1% (default is 100%) Refer to detailed description below this table.
i_wRampDrv	WORD	Deceleration ramp for the drive. Range: 0...9999 Scaling/Unit: 0.1 s Refer to detailed description below this table.
i_xRst	BOOL	On a rising edge, attempts to clear all alarms. If alarms stay active despite the rising edge of this input, the cause of the alarm is still present.
i_xDisLsAlrm	BOOL	Disables plausibility check of limit switch signals combination. Refer to detailed description below this table.

* optional, used for adaptive ramps

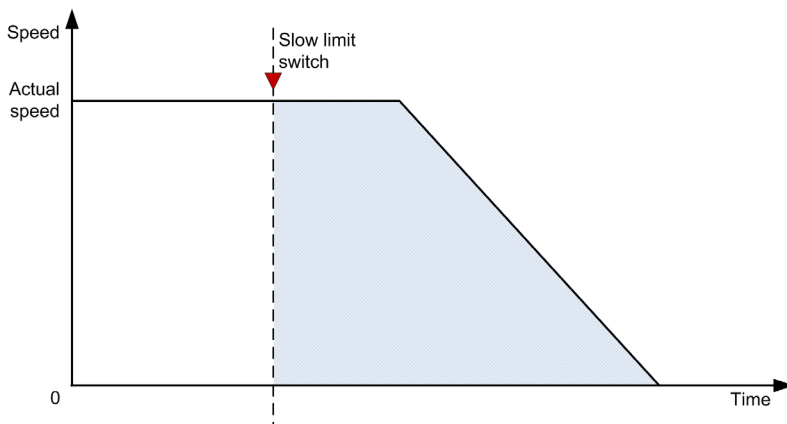
i_rDistStop

Distance between slow-down limit switch and a desired stop position.

i_rDistStop > 0 enables adaptive ramp function

$i_rDistStop = 0$ disables adaptive ramp function

The shaded area in following figure corresponds to the distance traveled in the slow-down area using the adaptive ramp.



$i_xDisLsAlm$

Disables the plausibility check of the sequence of limit switch inputs.

NOTE: Set this parameter to TRUE only if disabling of plausibility check is needed for correct function of the application.

$i_wScalFact$

A scaling factor of 0...200% used to adjust the calculated distance traveled to the real distance.

The overall distance calculation is as follows:

Traveled distance = Scaling Factor * Σ (Actual Speed * Δ Time)

= Scaling Factor * Σ (Linear Speed * Sample Rate * Actual Speed / Nominal Speed)=

$i_wScalFct[\%] / 100 * \Sigma(i_wMotSpdLin[m/s] * i_wSmplRate[ms] / 1000) *$

$i_iDrvSpdAct1[RPM] / i_wDrvSpdNom[RPM]$)

Example:

Nominal speed of motor in RPM	1500 (FB parameter)
Linear speed at nominal RPM in m/s	1 (FB parameter)
Scaling factor in %	100 (FB parameter)
Actual speed in RPM	750 (actual value)
Sample rate in ms	40 (calculated internally)

Distance in current cycle = $100 \% / 100 * (1 \text{ m/s} * 40 \text{ ms} / 1000 * 750 \text{ RPM} / 1500 \text{ RPM}) = 0.02 \text{ m}$

i_wRampDrv

This input corresponds to the deceleration ramp time of the drive.

For example:

Nominal speed of motor in RPM	1500
Deceleration ramp in seconds	5
i_wRampDrv	50

NOTE: The parameter `Ramp increment` assumes a drive default ramp increment configuration of 0.1 seconds. The unit-base used for this parameter must be the same as that configured in the drive. Further, the drive deceleration ramp time must be the same as for this parameter as it is configured in the drive.

This means that the time needed to reach the zero speed from the nominal speed of 1500 RPM is 5 seconds.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled FALSE: Function block disabled
q_xDrvFwd	BOOL	Drive forward run command. TRUE: Forward FALSE: Not forward
q_xDrvRev	BOOL	Drive reverse run command. TRUE: Reverse FALSE: Not reverse
q_xDrvFwdSlow	BOOL	Drive forward slow-down command to a safe speed. TRUE: Forward slow FALSE: Not forward slow
q_xDrvRevSlow	BOOL	Drive reverse slow-down command to a safe speed. TRUE: Reverse slow FALSE: Not reverse slow
q_wLsStat	WORD	Status of limit switch management FB. Range: 0, 1, 2, 3, 4 Refer to detailed description below this table.
q_rDistTrvl	REAL	Actual distance traveled in slow-down area. Range: 1.175494351e-38...3.402823466e+38 Scaling/Unit: 1 m
q_xAlrm	BOOL	Detected alarm bit. TRUE: Alarm detected FALSE: No alarm detected Refer to detailed description below this table.
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (<i>see page 419</i>). Range: 0...15

q_wLsStat

Status Bit	Description
0	axis is in the working area
1	axis is in the slow forward area
2	axis is in the stop forward position
3	axis is in the slow reverse area
4	axis is in the stop reverse position

q_xAlrm

Is TRUE if the FB is in an alarm state. Drive direction commands are set to FALSE if an alarm is present. The alarm must be reset using the `i_xRst` input after the cause of the alarm was removed.

See the alarm description list in Notifications.

Notifications

The alarm ID when `q_xAlrm` is TRUE. The ID is indicated by a bit set in this output.

The bit values have the following meaning:

Alarm Bit <code>q_wAlrmId</code>	Description
0	wrong order of limit switch signals detected
1	<code>i_wDrvSpdNom</code> is zero although <code>i_rDistStop</code> >0
2	<code>i_rMotSpdLin</code> is zero although <code>i_rDistStop</code> >0
3	<code>i_wScalFact</code> is zero although <code>i_rDistStop</code> >0
4	<code>i_rMotSpdLin</code> is negative

Section 13.5

Quick Reference Guide

What Is in This Section?

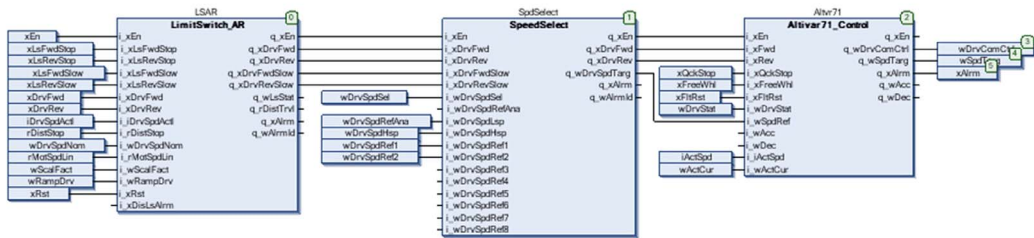
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	421
Commissioning Procedure	422
Troubleshooting	423

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the LimitSwitch_AR function block:



Commissioning Procedure

Commissioning Procedure of the LimitSwitch_AR Function Block

1. Create an instance of the LimitSwitch_AR FB and connect its inputs and outputs.
2. Set the deceleration ramp of the drive to the same as value given to the FB.
3. The distance between slow and stop limit switch should be sufficient for the axis to decelerate from the maximum speed to a defined slow speed.
4. Set the stop behavior using i_xSelStopMode input.
5. Set the i_rDistStop parameter to a value higher than or equal to the distance necessary to decelerate from the maximum speed to a defined slow speed and lower or equal do a distance of slow and stop limit switch if stop limit switch is present.
6. Test the function by moving the axis into the slow-down area at various speeds.

Alarm Fallback State, Disable Fallback State

In the case of an alarm, the FB removes (sets to FALSE) movement command outputs until the cause of the alarm is removed and the alarm is reset.

Troubleshooting

Troubleshooting

Issue	Cause	Solution
Alarm is raised: <code>q_xAlarm = TRUE</code> <code>q_wAlarmId bit 0 = TRUE</code>	An alarm has been detected for an unexpected input signal combination. (<code>q_wLsStat</code>)	<ul style="list-style-type: none"> ● Check the position of movement in regards to the limit switch signals indicated by the <code>LimitSwitch_AR</code> function block. ● Clear the alarm signal or reboot the system (Power OFF and Power ON). ● In the case of incorrect position signals, set the input to TRUE for intermediate operation and verify the wiring or limit switch sensors.
Alarm is raised: <code>q_xAlarm = TRUE</code> <code>q_wAlarmId bit 1, 2, 3, 4 = TRUE</code>	Incorrect parameterization of the FB.	Correct the input parameters.

Chapter 14

DoubleLimitSwitch_AR: Administers Two Devices in Synchronous Mode

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `DoubleLimitSwitch_AR_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
14.1	Functional Overview	426
14.2	Architecture	428
14.3	Function Block Description	431
14.4	Pin Description	436
14.5	Quick Reference Guide	445

Section 14.1

Functional Overview

Functional Overview

Functional Description

The function reads limit switch inputs from the field. It checks the limit switch status on 2 limit switches and generates the control outputs which are used to control the crane's movements. The function block is designed for use with cross and screw limit switches using Normally Closed (NC) contacts. The contacts retain the current status, for example, the stop position.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use impulse or normally open contact switches in association with the function blocks.
- Only use Normally Closed contact switches with the function blocks.

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

Unlike the single limit switch management function block (`LimitSwitch_AR`), the double limit switch management function block administers 2 devices in Synchronous mode which are intended to be used with the `HoistPositionSync` function block.

NOTE: The function block is constructed by combining 2 `LimitSwitch_AR` function blocks into one. The behavior of the FB in the Asynchronous mode is similar to 2 separate `LimitSwitch_AR` function blocks.

Why Use the `DoubleLimitSwitch_AR` Function Block?

The function is required to monitor and control the movement of 2 trolleys/bridges to help prevent it from crashing into the mechanical barrier at either end of the rails. This block can also be used for hoisting and slewing movements. When the block is used in hoisting movement, it stops the hook from crashing into the trolley or from over-winding the drum. Letting the FB handle all the necessary interlocks between synchronous and asynchronous behavior is helpful, especially when 2 axes have to be synchronized.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

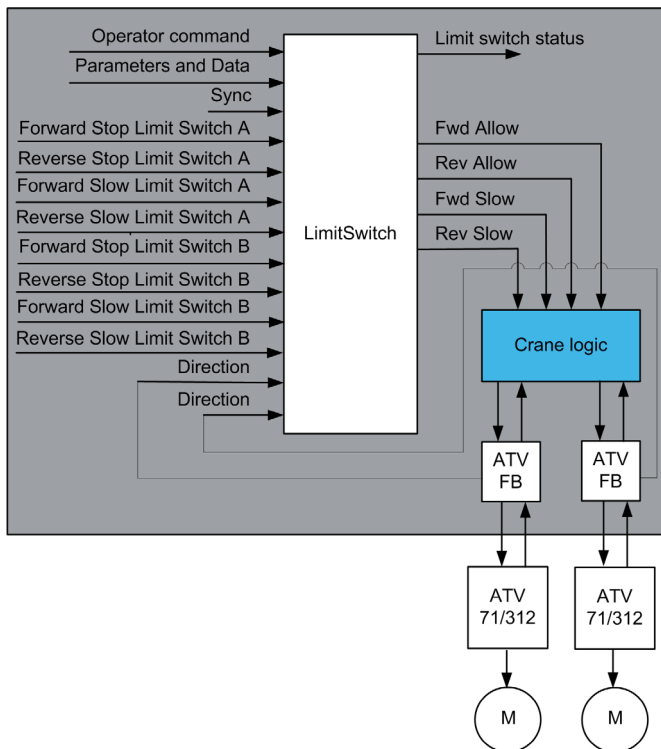
WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

Functional View



Section 14.2

Architecture

What Is in This Section?


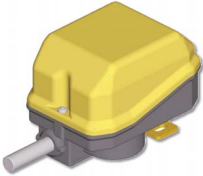
This section contains the following topics:

Topic	Page
Hardware Architecture	429
Software Architecture	430

Hardware Architecture

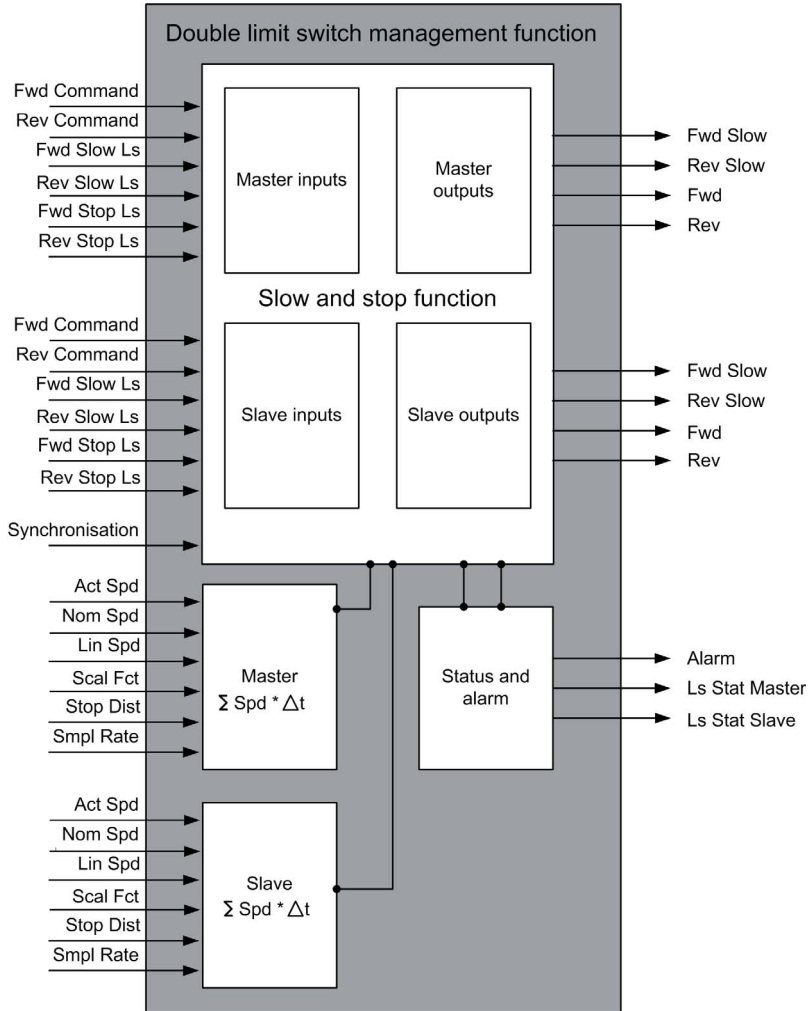
Necessary Equipment to Fulfill Limit Switch Functionality

The following table includes a list of the devices used in the system, including the description, followed by some remarks to clarify the description of the device.

Symbol	Description	Remarks
	Limit switch, cross type	XCKMR
	Limit switch, screw type	3rd party product

Software Architecture

Data Flow Overview

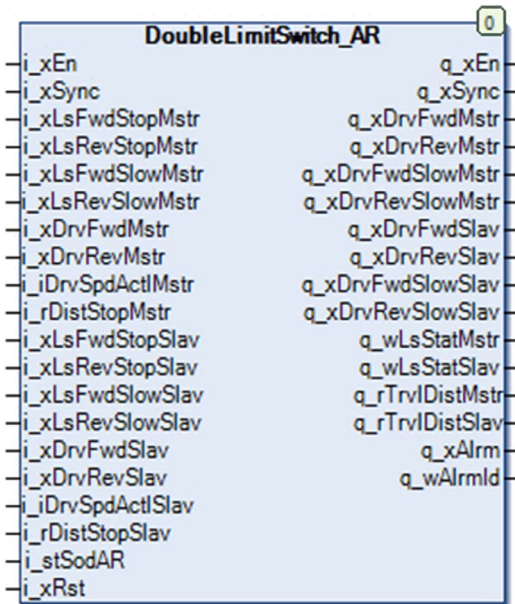


Section 14.3

Function Block Description

DoubleLimitSwitch_AR Function Block

Pin Diagram



Function Block Description

This function block handles up to 4 movement positions for trolley, bridge, hoisting or slewing.

This includes the following:

- Forward stop position
- Forward slow position
- Reverse stop position
- Reverse slow position

Remark

If any of the limit switch stop positions is not used, or the slow position is only used in one direction, the related input on the function block must be set to TRUE, as the limit switch management function is designed for N.C. configuration. If using the adaptive ramp function of the block the corresponding slow limit switch for the desired direction must be present.

Standard Limit Switch Behavior

The standard limit switch function allows the axis to move at any given speed until either of the slow limit switches is reached. To enable this function, `i_rDistStopMstr` and `i_rDistStopSlav` value to be set to zero.

Example:

When the system moves in the forward direction and the Forward slow position `i_xLsFwdSlowMstr` is reached, the function block enables the Forward slow signal `q_xDrvFwdSlowMstr`. When the Forward stop position `i_xLsFwdStopMstr` is reached, the function block turns OFF the Forward command output signal `q_xDrvFwdMstr`.

The function is the same for the slave axis.

When the system moves in the reverse direction and the Reverse slow position `i_xLsRevSlowMstr` is reached, the function block enables the Reverse slow signal `q_xDrvRevSlowMstr`. When the Reverse stop position `i_xLsFwdStopMstr` is reached, the function block turns OFF the Reverse command output signal `i_xLsRevStopMstr`.

The function is the same for the slave axis.

If `i_xSync` is TRUE, the signal information of both connected limit switches is analyzed simultaneously and the result is written to both, the master and the slave outputs. If either one of the Forward slow signals or Reverse slow signals is FALSE, the output is set to slow movement. If either one of the Forward stop signals or Reverse stop signals is FALSE, the output disables the movement in that direction.

Adaptive Ramp Behavior

The adaptive ramp function allows the axis to move at any given speed as long as it is possible. To enable this function, `i_rDistStopMstr` and `i_rDistStopSlav` should be greater than zero.

Example:

When the system is moving in the forward direction and the Forward slow position (`i_xLsFwdSlowMstr` or `i_xLsFwdSlowSlav`) is initiated, the function block enables the internal calculation of the remaining distance according to the actual speed.

The function block calculates the traveled distance by integrating actual speed of the drive over time.

The adaptive ramp function allows the FB to calculate the highest available speed while the axis is in a slow-down area.

When the distance traveled is equal to $i_rDistStop$, the FB turns OFF the Forward command signal ($q_xDrvFwdMstr$ or $q_xDrvFwdSlav$). When the Forward stop position ($i_xLsFwdStopMstr$ or $i_xLsFwdStopSlav$) is initiated, the function block turns OFF the Forward allow signal ($q_xDrvFwdMstr$ or $q_xDrvFwdSlav$). The logic is the same for the reverse direction.

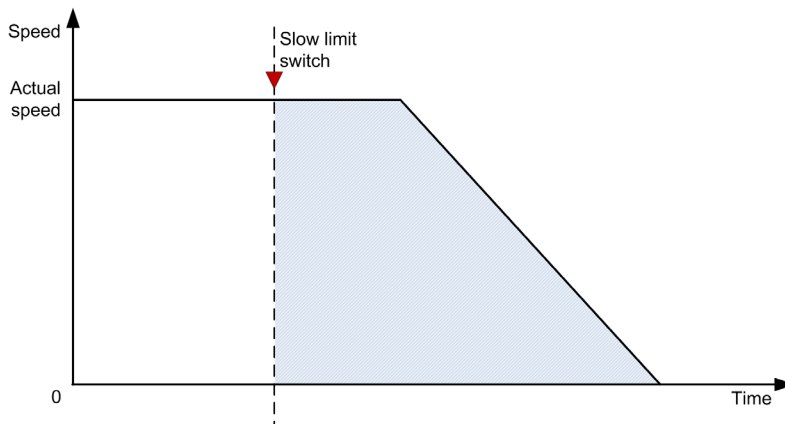
If i_xSync is TRUE, the signal information of both connected limit switches is analyzed simultaneously and the result is written to both, the master and the slave outputs. If either one of the Forward or Reverse slow signals is FALSE, the output is set to slow movement. If either one of the Forward or Reverse stop signals is FALSE, the output disables the movement in that direction.

Example

- Stop distance: 3 m
- Nominal speed of the drive: 1500 RPM
- Nominal linear speed: 1 m/s
- If the actual speed of the drive = 600 RPM, the actual linear speed (m/s) = $1\text{ m/s} * 600\text{ RPM} / 1500\text{ RPM} = 0.4\text{ m/s}$
- Distance traveled in meters = $((0.4\text{ m/s} * \text{internal calculated cycle time}) / 1000)$
- When the distance traveled is greater than the stop distance, the drive stops and further movement in the same direction is not allowed.

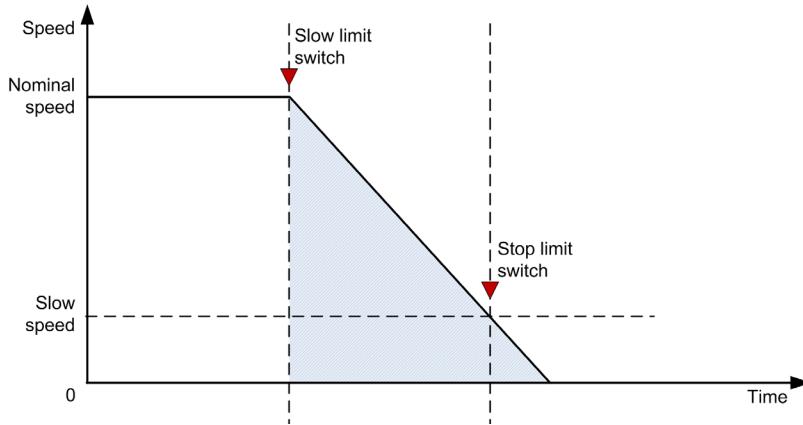
NOTE: The following chart shows the actual speed of the drive.

The following figure represents the speed time curve of `DoubleLimitSwitch_AR` function block:

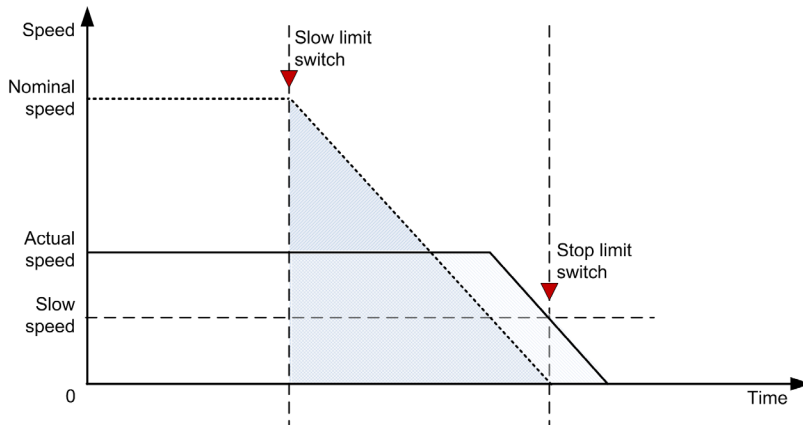


The following figures describe the benefit of using an adaptive ramp function.

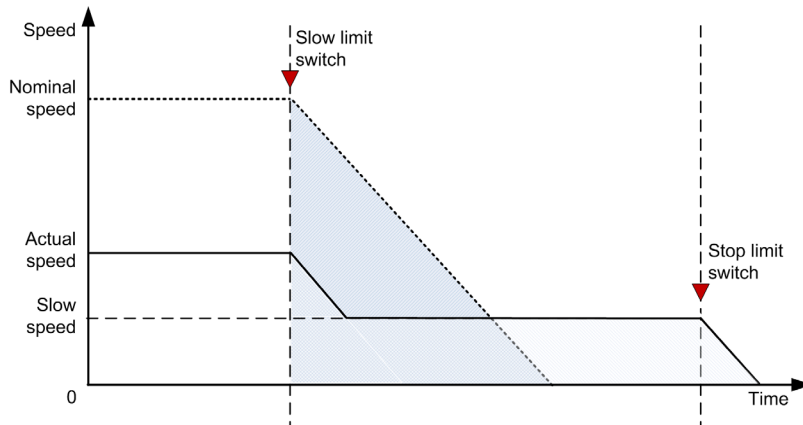
The first chart shows the behavior when entering a property configured slow-down area at nominal speed:



The second chart describes entering a slow-down area at a speed that is approximately half of the nominal speed with an adaptive ramp function:



The third chart shows the behavior without the adaptive ramp function. The shaded area corresponds to the distance traveled in a slow-down area at a given speed.



Section 14.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	437
Structured Variable Description	440
Output Pin Description	442

Input Pin Description

Input Pin Description

The `DoubleLimitSwitch_AR` function block is designed for use exclusively with Normally Closed (NC) sensors. If any of the limit switch inputs are not connected to a Normally Closed (NC) sensors, they have to be set to TRUE.

Input	Data Type	Description
<code>i_xEn</code>	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
<code>i_xSync</code>	BOOL	TRUE: Enables the synchronization (Synchronous mode). FALSE: Disables the synchronization (Asynchronous mode).
<code>i_xLsFwdStopMstr</code>	BOOL	Stop forward position master. When the Stop forward is initiated the Forward output command (<code>q_xDrvFwdMstr</code>) signal is disabled. This signal is used to stop movement of the trolley/bridge in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
<code>i_xLsRevStopMstr</code>	BOOL	Stop reverse position master. When the Stop reverse is initiated the Reverse output command (<code>q_xDrvRevMstr</code>) signal is disabled. This signal is used to stop movement of the trolley/bridge in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
<code>i_xLsFwdSlowMstr</code>	BOOL	Slow-down forward position master. The Slow forward initiates the Slow forward (<code>q_xDrvFwdSlowMstr</code>) signal. This signal is used to move the trolley/bridge at low speed in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
<code>i_xLsRevSlowMstr</code>	BOOL	Slow-down reverse position master. The Slow reverse initiates the Slow reverse (<code>q_xDrvRevSlowMstr</code>) signal. This signal is used to move the trolley/bridge at low speed in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.

Input	Data Type	Description
i_xDrvFwdMstr	BOOL	Input for the forward commands of the master drive. In Synchronous mode, this input will control the slave drive as well. TRUE: Forward FALSE: Not forward
i_xDrvRevMstr	BOOL	Input for the reverse commands of the master drive. In Synchronous mode, this input will control the slave drive as well. TRUE: Reverse FALSE: Not reverse
i_iDrvSpdActlMstr *	INT	Actual drive speed master. These values are provided as a feedback signal from the drives. Range: -6000...+6000 Scaling/Unit: 1 RPM
i_rDistStopMstr *	REAL	Stop Distance. Allowed distance for travel after passing the slow position, when Stop on Distance is used. To disable the Stop on Distance function, these inputs must be set to zero. Range: 1.175494351e-38F ... 3.402823466e+38F Scaling/Unit: 1 m
i_xLsFwdStopSlav	BOOL	Stop forward position slave. When the Stop forward is initiated the Forward output command (q_xDrvFwdSlav) signal is disabled. This signal is used to stop movement of the trolley/bridge in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsRevStopSlav	BOOL	Stop reverse position slave. When the Stop reverse is initiated the Reverse output command (q_xDrvRevSlav) signal is disabled. This signal is used to stop movement of the trolley/bridge in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsFwdSlowSlav	BOOL	Slow-down forward position slave. The Slow forward initiates the Slow forward (q_xDrvFwdSlowSlav) signal. This signal is used to move the trolley/bridge at low speed in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.

Input	Data Type	Description
i_xLsRevSlowSlav	BOOL	Slow-down reverse position slave. The Slow reverse initiates the Slow reverse (<code>q_xDrvRevSlowSlav</code>) signal. This signal is used to move the trolley/bridge at low speed in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xDrvFwdSlav	BOOL	Input for the forward commands of the slave drive. This command is ignored in Synchronous mode. TRUE: Forward FALSE: Not forward
i_xDrvRevSlav	BOOL	Input for the reverse commands of the slave drive. This command is ignored in Synchronous mode. TRUE: Reverse FALSE: Not reverse
i_rDistStopSlav *	REAL	Stop Distance. Allowed distance for travel after passing the slow position, when Stop on Distance is used. To disable the Stop on Distance function, these inputs must be set to zero. Range: 1.175494351e-38F ... 3.402823466e+38F Scaling/Unit: 1 m
i_iDrvSpdAct1Slav *	INT	Actual drive speed slave. These values are provided as a feedback signal from the drives. Range: -6000...+6000 Scaling/Unit: 1 RPM
i_stSodAR *	SODAR	Parameter data structure for motor data of master and slave for Stop on Distance. Range: 0...65535 (each value) Refer to Structured Variable Description (see page 440).
i_xRst	BOOL	On a rising edge, attempts to clear all alarms. If alarms stay active despite the rising edge of this input, the cause of the alarm is still present.

* optional, used for Stop on Distance

Structured Variable Description

SODAR (I_stSodAR)

Structure Parameter	Data Type	Description
wDrvSpdNomMstr *	WORD	Nominal motor speeds of the master motor used in the distance calculation. Range: 0..6000 Scaling/Unit: 1 RPM
rMotSpdLinMstr *	REAL	Linear speed of movement at nominal motor speed (wDrvSpdNomMstr). This is used in the distance calculation. Range: 0..5 Scaling/Unit: 1 m/s
wRampDrvMstr	WORD	Deceleration ramp for the master drive. Range: 0..9999 Scaling/Unit: 0.1 s
wScalFactMstr *	WORD	Scale factor for corrections of distance calculation. Range: 0..200 Scaling/Unit: 1% (default is 100%) Refer to detailed description below this table.
wDrvSpdNomSlav *	WORD	Nominal motor speeds of the slave motor used in the distance calculation. Range: 0..6000 Scaling/Unit: 1 RPM
rMotSpdLinSlav *	REAL	Linear speed of movement at nominal motor speed (wDrvSpdNomSlav). This is used in the distance calculation. Range: 0..5 Unit: m/s
wRampDrvSlav	WORD	Deceleration ramp for the drive. Range: 0..9999 Scaling/Unit: 0.1 s
wScalFactSlav *	WORD	Scale factor for correction of distance calculation. Range: 0..200 Scaling/Unit: 1% (default is 100%) Refer to detailed description below this table.

* optional, used for Stop on Distance

wScalFactMstr, wScalFactSlav

A scaling factor of 0...200% used to adjust the calculated distance traveled to the real distance.

The overall distance calculation is as follows:

$$D = \text{Scaling Factor} * \Sigma (\text{Actual Speed} * \Delta\text{Time})$$

$$= \text{Scaling Factor} * \Sigma (\text{Linear Speed} * \text{Actual Speed} * \text{Sample Rate} / \text{Nominal Speed}) = \\ i_wScalFct[\%] * \Sigma (i_wLinSpd[\text{m/s}] * i_iActSpd[\text{RPM}] * i_wSmpRate[\text{ms}] / \\ i_wNomSpd[\text{RPM}])$$

Example:

Nominal speed of motor in RPM	1500 (from user)
Nominal linear speed in m/s	1.00 (from user)
Scaling factor in %	100 (from user)
Actual speed in RPM	750 (from drive)
Sample rate in ms	40 (from controller)

$$\text{Distance in current cycle} = 100\% * 1 \text{ m/s} * 750 \text{ RPM} * 40 \text{ ms} / 1500 \text{ RPM} = 0.02 \text{ m}$$

wRampDrvMstr, wRampDrvSlav

This input corresponds to the deceleration ramp time of the drive.

Example:

Nominal speed of motor in RPM	1500
Deceleration ramp in seconds	5
i_wRampDrv	50

NOTE: The parameter assumes a drive default ramp increment configuration of 0.1 seconds. The unit-base used for this parameter must be the same as that configured in the drive. Further, the drive deceleration ramp time must be the same as for this parameter as it is configured in the drive.

This means that the time needed to reach the zero speed from the nominal speed of 1500 RPM is 5 seconds.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled FALSE: Function block disabled
q_xSync	BOOL	Synchronization output. TRUE: Function block in Synchronous mode FALSE: Function block in Asynchronous mode
q_xDrvFwdMstr	BOOL	Master drive forward run command. TRUE: Forward FALSE: Not forward
q_xDrvRevMstr	BOOL	Master drive reverse run command. TRUE: Reverse FALSE: Not reverse
q_xDrvFwdSlowMstr	BOOL	Force low speed forward motion only on master or master and slave while synchronized. TRUE: Forward slow FALSE: Not forward slow
q_xDrvRevSlowMstr	BOOL	Force low speed reverse motion only on master or master and slave while synchronized. TRUE: Reverse slow FALSE: Not reverse slow
q_xDrvFwdSlav	BOOL	Slave drive forward run command. TRUE: Forward FALSE: Not forward
q_xDrvRevSlav	BOOL	Slave drive reverse run command. TRUE: Reverse FALSE: Not reverse
q_xDrvFwdSlowSlav	BOOL	Force low speed forward motion only on slave or master and slave while synchronized. TRUE: Forward slow FALSE: Not forward slow
q_xDrvRevSlowSlav	BOOL	Force low speed reverse motion only on slave or master and slave while synchronized. TRUE: Reverse slow FALSE: Not reverse slow

Output	Data Type	Description
q_wLsStatMstr	WORD	Status of limit switch management on master limit switch. Range: 0, 1, 2, 3, 4 Refer to detailed description below this table.
q_wLsStatSlav	WORD	Status of limit switch management on slave limit switch. Range: 0, 1, 2, 3, 4 Refer to detailed description below this table.
q_rDistTrvlMstr	REAL	Output of the Stop on Distance calculation. Whenever this value exceeds the value of i_wStopDistMstr, a stop is executed. Range: 0...65535 Scaling/Unit: 1 m
q_rDistTrvlSlav	REAL	Output of the Stop on Distance calculation. Whenever this value exceeds the value of i_wStopDistSlav, a stop is executed. Range: 0...65535 Scaling/Unit: 1 m
q_xAlrm	BOOL	Detected alarm bit. TRUE: Alarm detected FALSE: No alarm detected. Refer to detailed description below this table.
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (<i>see page 444</i>). Range: 0...63

q_wLsStatMstr, q_sLsStatSlav

Status Bit	Description
0	trolley/bridge/hoist is in the working area.
1	trolley/bridge/hoist is between the Slow forward and Stop forward positions.
2	trolley/bridge/hoist is beyond the Slow forward and Stop forward positions.
3	trolley/bridge/hoist is between the Slow reverse and Stop reverse positions.
4	trolley/bridge/hoist is beyond the Slow reverse and Stop reverse positions.

q_xAlrm

An alarm signal is generated in the cases of incorrect limit switch configuration or improper operation of a limit switch.

The function block detects any incorrect signaling of the position with respect to the movement, i.e. during movement, the signaling of any limit switch position located in the opposite direction of the movement generates an alarm.

Notifications

Alarm Bit q_wAlrmlId	Description
0	wrong order of limit switch signals detected on axis
1	SOD.wDrvSpdNomMstr is zero although Stop on Distance is enabled
2	SOD.wMotSpdLinMstr is zero although Stop on Distance is enabled
3	SOD.wScalFactMstr is zero although Stop on Distance is enabled
4	SOD.rMotSpdLinMstr is negative
5	wrong order of limit switch signals detected on slave axis
6	SOD.wDrvSpdNomSlav is zero although Stop on Distance is enabled
7	SOD.wMotSpdLinSlav is zero although Stop on Distance is enabled
8	SOD.wScalFactSlav is zero although Stop on Distance is enabled
9	SOD.rMotSpdLinSlav is negative

Section 14.5

Quick Reference Guide

What Is in This Section?

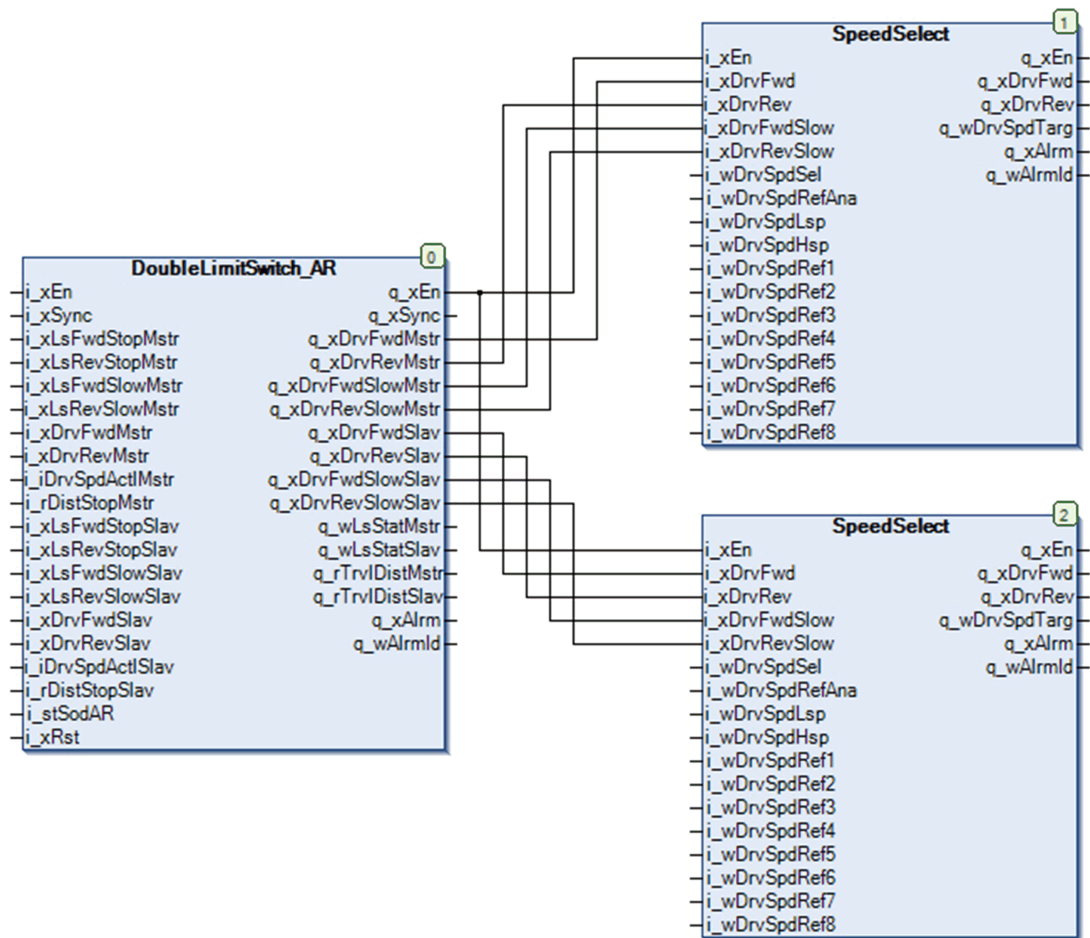
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	446
Commissioning Procedure	447
Troubleshooting	448

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the DoubleLimitSwitch_AR function block:



Commissioning Procedure

Commissioning Procedure of the DoubleLimitSwitch_AR Function Block

The following steps explain the procedure on how to configure the Double limit switch management - adaptive ramp solution:

1. Create an instance of the DoubleLimitSwitch_AR function block and connect its inputs and outputs.
2. Set the deceleration ramp of the drive to the same as value given to the FB.
3. The distance between slow and stop limit switch should be sufficient for the axis to decelerate from the maximum speed to a defined slow speed.
4. If you intent to use the FBs adaptive ramp functionality you need to set all parameters of the SODAR input pin `i_stSODAR` as well as `i_rDistStopMstr` and `i_rDistStopSlav`.
5. To use the non-adaptive ramps operation put zero to the `i_rDistStopMstr` and `i_rDistStopSlav` inputs.
6. Test the function by moving the axis into the slow-down area at various speeds.

Alarm Fallback State, Disable Fallback State

In the case of an alarm, or if the FB is disabled, all output speed authorizations are set to FALSE:

<code>q_xDrvFwdMstr</code>	<code>:=False;</code>
<code>q_xDrvRevMstr</code>	<code>:=False;</code>
<code>q_xDrvFwdSlowMstr</code>	<code>:=False;</code>
<code>q_xDrvRevSlowMstr</code>	<code>:=False;</code>
<code>q_xDrvFwdSlav</code>	<code>:=False;</code>
<code>q_xDrvRevSlav</code>	<code>:=False;</code>
<code>q_xDrvFwdSlowSlav</code>	<code>:=False;</code>
<code>q_xDrvRevSlowSlav</code>	<code>:=False;</code>

Troubleshooting

Troubleshooting

Issue	Cause	Solution
q_xAlrm of the FB is TRUE.	One or more of the input parameters are out of range.	Check the input parameters.

Chapter 15

DoubleLimitSwitch_AR_2: Administers Two Devices in Synchronous Mode

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
15.1	Functional Overview	450
15.2	Architecture	453
15.3	Function Block Description	456
15.4	Pin Description	462
15.5	Quick Reference Guide	471

Section 15.1

Functional Overview

Functional Overview

Functional Description

The function reads limit switch inputs from the field. It checks the status on 2 limit switches and generates the control outputs which are used to control the crane's movements. The function block is designed for use with cross and screw limit switches using Normally Closed (NC) contacts and a photoelectric sensor. The contacts retain the current status, for example, the stop position.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use impulse or normally open contact switches in association with the function blocks.
- Only use Normally Closed contact switches with the function blocks.

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

NOTE: The function block is constructed by combining 2 `LimitSwitch_AR` function blocks into one. The behavior of the FB in the Asynchronous mode is similar to 2 separate `LimitSwitch_AR` function blocks.

Why Use the `DoubleLimitSwitch_AR_2` Function Block?

The function is required to monitor and control the movement of 2 bridges/trolleys to help prevent it from crashing into the mechanical barrier at either end of the rails and to avoid the collision between adjacent bridges/trolleys running on the same axis.

This block can also be used for hoisting and slewing movements. When the block is used in hoisting movement, it stops the hook from crashing into the trolley or from over-winding the drum. The FB handles all the necessary interlocks between synchronous and asynchronous behavior of limit switches.

NOTE: The `DoubleLimitSwitch_AR_2` function block is compatible with the `GrabControl` function block. This means, that the commands for Hold Drive and Close Drive are computed from the `DoubleLimitSwitch_AR_2` while the FB is in synchronized mode.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

 **WARNING****UNINTENDED EQUIPMENT OPERATION**

Validate all function block input values before and while the function block is enabled.

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

Solution with the DoubleLimitSwitch_AR_2 Function Block

The DoubleLimitSwitch_AR_2 manages the motion of 2 hoists of a construction crane or controls bridges/trolleys of an industrial crane along an axis with a screw or cross limit switch sensors which are fixed at the either end of the rail and using 2 points (limit switch and limit stop) of detection on each side.

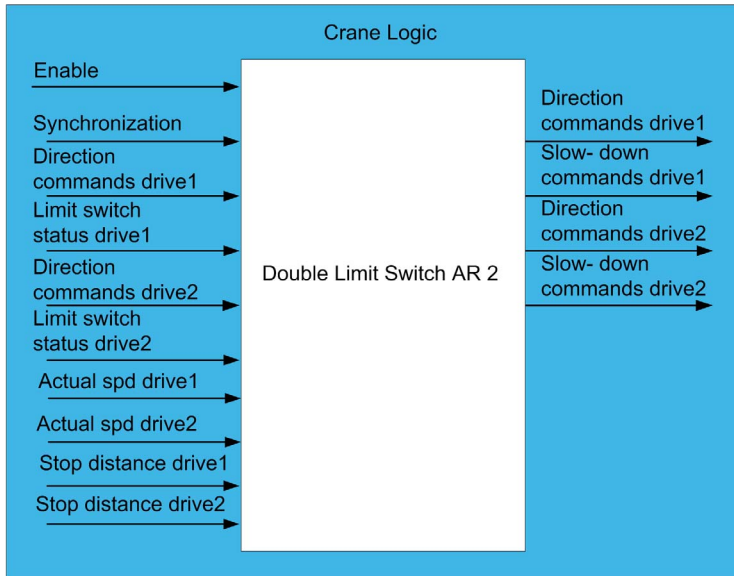
It is recommended to use the photoelectric sensor which is specially used for 2 cranes on the same rail; when 2 cranes approach each other, the sensor head receives signal reflected from the reflector and stop the movement of approaching crane and prevent unintentional collision.

The function block has the following various options:

1. 2 slow switches and 2 stop switches on either end of rail
2. 1 photoelectric sensor mounted at the first bridge/trolley
3. 1 reflector mounted at the second bridge/trolley

If only slow switch combinations are used, the stop can be executed using the integrated Stop on Distance function which automatically calculates the remaining distance and brings the system to stop. In synchronization mode, both axes are stopped if either one of them indicates the end of the working zone or rail has been reached.

Functional View



Section 15.2

Architecture

What Is in This Section?


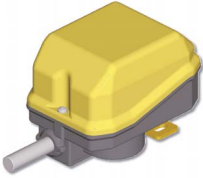

This section contains the following topics:

Topic	Page
Hardware Architecture	454
Software Architecture	455

Hardware Architecture

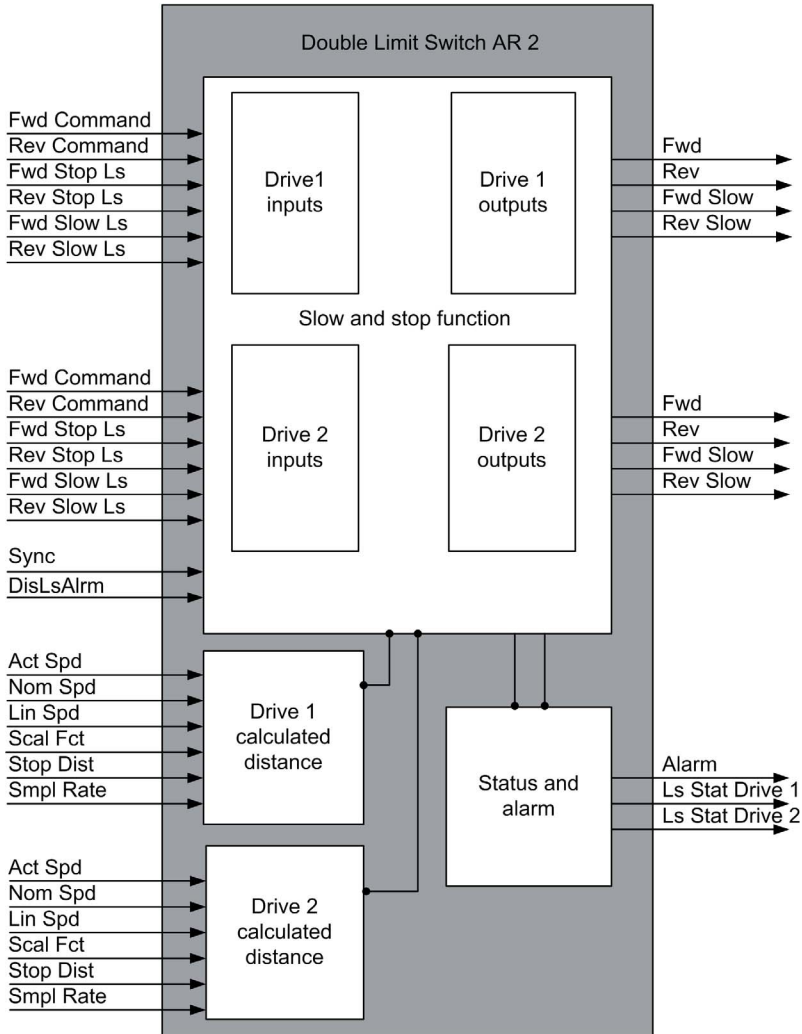
Necessary Equipment to Fulfill Limit Switch Functionality

The following table includes a list of the devices used in the system, including the description, followed by some remarks to clarify the description of the device.

Symbol	Description	Remarks
	Limit switch, cross type	XCKMR
	Limit switch, screw type	3rd party product
	Photoelectric sensor	OsiSense XUE

Software Architecture

Data Flow Overview

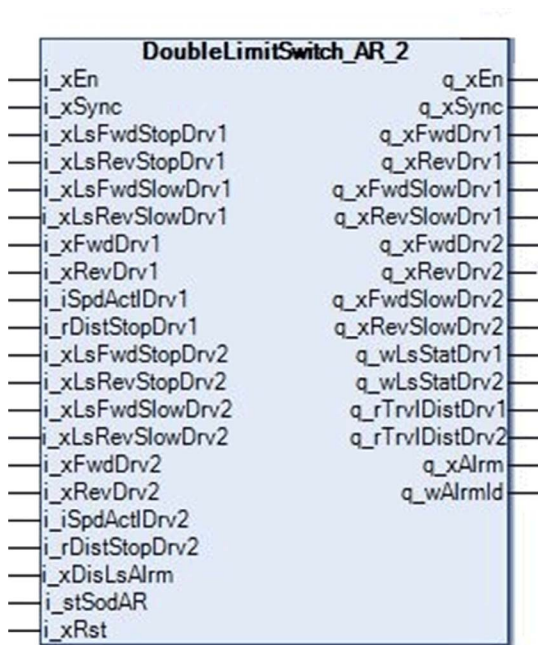


Section 15.3

Function Block Description

DoubleLimitSwitch_AR_2 Function Block

Pin Diagram



Function Block Description

This function block handles up to 4 movement positions for trolley, bridge, hoisting or slewing.

This includes the following:

- Forward stop position
- Forward slow position
- Reverse stop position
- Reverse slow position

Remark

If any of the limit switch stop positions is not used, or the slow position is only used in one direction, set the related input on the function block to TRUE, as the limit switch management function is designed for N.C. configuration. If using the adaptive ramp function of the block, the corresponding slow limit switch for the desired direction must be present.

By using the `DoubleLimitSwitch_AR_2` FB for managing the movement of 2 bridges or 2 trolleys on industrial crane, extra protection is required to be provided when they approach each other.

WARNING

COLLISION OF TWO BRIDGES/TROLLEYS

Use proximity sensors to avoid collisions between bridges/trolleys.

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

Adaptive Ramp Behavior

In the adaptive ramp mode, the FB calculates positions at which the axes have to slow down based on their estimated positions. To enable this function, the parameters `i_rDistStopDrv1` and `i_rDistStopDrv2` must be not equal zero.

NOTE: In synchronization mode to prevent collision between the both axis the value of the parameters `i_rDistStopDrv1` and `i_rDistStopDrv2` is compared and the lowest value is written to the both parameters.

No Synchronization

If...	Then ...
the system moves in the forward direction and the Forward slow position <code>i_xLsFwdSlowDrv1</code> is reached and the parameter <code>i_rDistStopDrv1</code> is equal zero,	the function block enables the Forward slow signal <code>q_FwdSlowDrv1</code> .
the Forward stop position <code>i_xLsFwdStopDrv1</code> is reached,	the function block turns OFF the Forward command output signal <code>q_xFwdDrv1</code> .
the system moves in the reverse direction and the Reverse slow position <code>i_xLsRevSlowDrv1</code> is reached and the parameter <code>i_rDistStopDrv1</code> is equal zero,	the function block enables the Reverse slow signal <code>q_xRevSlowDrv1</code> .
the Reverse stop position <code>i_xLsRevStopDrv1</code> is reached,	the function block turns OFF the Reverse command output signal <code>q_xRevdDrv1</code> .

The function is the same for the axis of drive 2.

Synchronization

If...	Then ...
i_xSync is TRUE,	the signal information of both connected limit switches is analyzed simultaneously and the result is written to the Drive 1 and the Drive 2 outputs.
the system moves in the forward direction and the Forward slow position (i_xLsFwdSlowDrv1 or i_xLsFwdSlowDrv2) is reached and both parameters (i_rDistStopDrv1, i_rDistStopDrv2) are zero,	the function block enables the Forward slow signal of both drives q_FwdSlowDrv1 and q_FwdSlowDrv2.
the Forward stop position (i_xLsFwdStopDrv1 or i_xLsFwdStopDrv2) is reached,	the function block turns OFF the Forward command output signals q_xFwdDrv1 and q_xFwdDrv2.
the system moves in the reverse direction and the reverse slow position (i_xLsRevSlowDrv1 or i_xLsRevSlowDrv2) is reached and the both parameters (i_rDistStopDrv1, i_rDistStopDrv2) are zero,	the function block enables the Reverse slow signal of both drives q_xRevSlowDrv1 and q_xRevSlowDrv2.
the Reverse stop position (i_xLsRevStopDrv1 or i_xLRevdStopDrv2) is reached,	the function block turns OFF the Reverse command output signals q_xRevDrv1 and q_xRevDrv2.
the system is moving in the forward direction and the Forward slow position i_xLsFwdSlowDrv1 or i_xLsFwdSlowDrv2 is initiated while the parameters i_rDistStopDrv1 and i_rDistStopDrv2 are not equal zero,	the function block enables the internal calculation of the remaining distance according to the actual speed.
the calculated distance is greater than lower of the two stop distance values (i_rDistStopDrv1 or i_rDistStopDrv2),	the function block turn OFF the Forward command output signals q_xFwdDrv1 and q_xFwdDrv2.
the system is moving in the reverse direction and the reverse slow position (i_xLsRevSlowDrv1 or i_xLsRevSlowDrv2) is initiated while the parameters i_rDistStopDrv1 and i_rDistStopDrv2 are not equal zero,	the function block enables the internal calculation of the remaining distance according to the actual speed.
the calculated distance is greater than lower of the two stop distance values (i_rDistStopDrv1 or i_rDistStopDrv2),	the function block turn OFF the Forward command output signals q_xFwdDrv1 and q_xFwdDrv2.

The function block calculates the traveled distance by integrating actual speed of the drive over time.

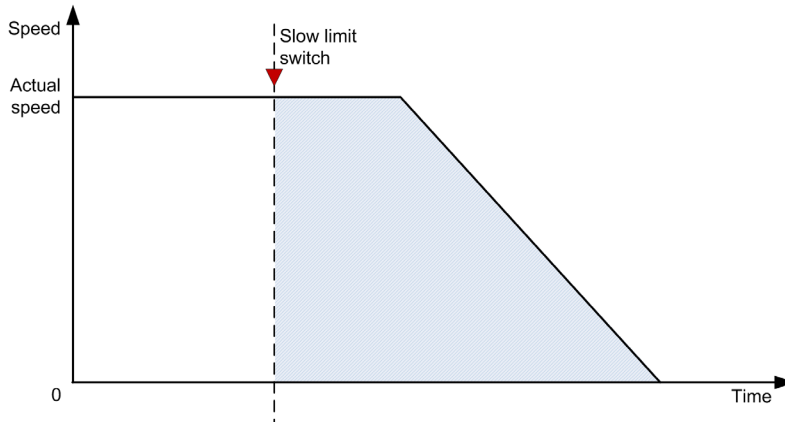
The adaptive ramp function allows the FB to calculate the highest available speed while the axis is in a slow-down area.

Example

- Stop distance: 3 m
- Nominal speed of the drive: 1500 RPM
- Nominal linear speed: 1 m/s
- If the actual speed of the drive = 600 RPM, the actual linear speed (m/s) = $1 \text{ m/s} * 600 \text{ RPM} / 1500 \text{ RPM} = 0.4 \text{ m/s}$
- Distance traveled in meters = $((0.4 \text{ m/s} * \text{internal calculated cycle time}) / 1000)$
- When the distance traveled is greater than the stop distance, the drive stops and further movement in the same direction is not allowed.

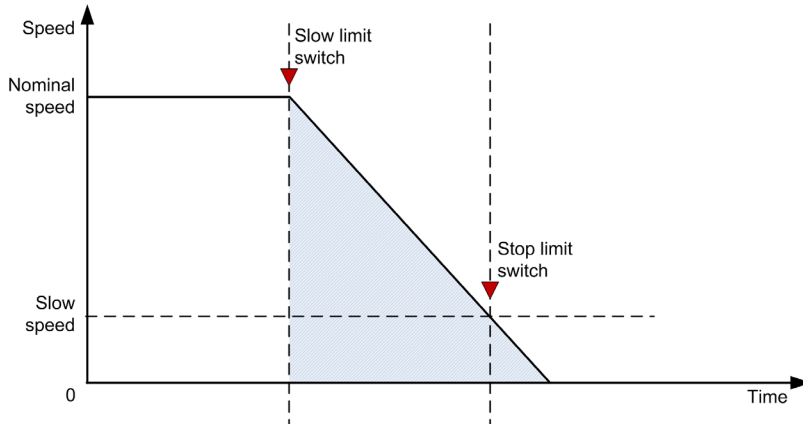
NOTE: The following chart shows the actual speed of the drive.

The following figure represents the speed time curve of DoubleLimitSwitch_AR_2 function block:

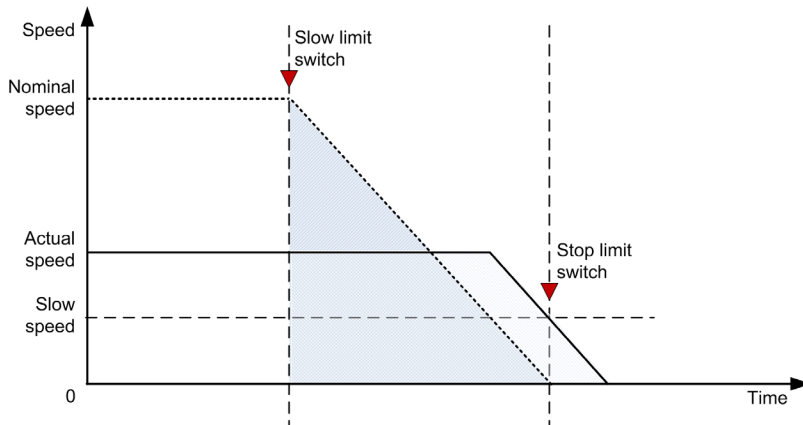


The following figures describe the benefit of using an adaptive ramp function.

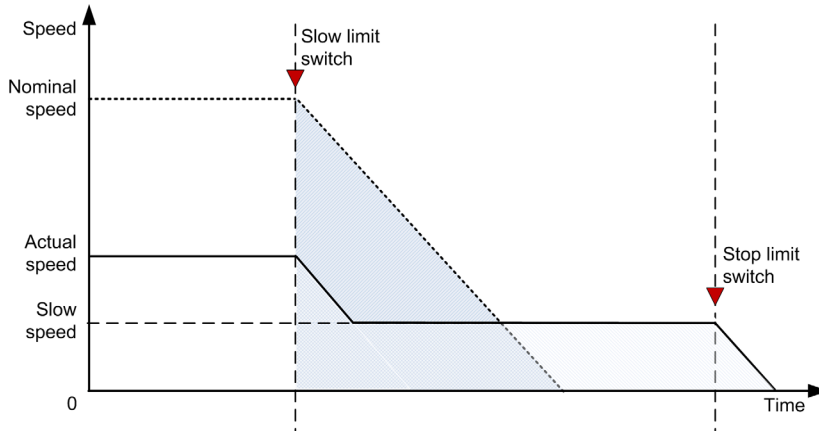
The first chart shows the behavior when entering a property configured slow-down area at nominal speed:



The second chart describes entering a slow-down area at a speed that is approximately half of the nominal speed with an adaptive ramp function:



The third chart shows the behavior without the adaptive ramp function. The shaded area corresponds to the distance traveled in a slow-down area at a given speed.



By using the `DoubleLimitSwitch_AR_2` for 2 cranes (bridges or trolleys) moving on the same rail, the input `i_xDisLsAlrm` has to set to `TRUE` to disable alarms based on combination of limit switch states.

⚠ WARNING

COLLISION OF TWO BRIDGES/TROLLEYS

Use proximity sensors to avoid collisions between bridges/trolleys.

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

When 2 cranes approach each other, the sensor head receives signal a reflected from the reflector. With this, presence of another crane is sensed and movement of approaching crane is stopped in that particular direction. The crane can move in any other direction (away from another crane) any time. Likewise, another unit is fixed on another crane and its reflector is fixed on the first crane. Therefore, each time the cranes approach each other within a specified distance, movement is stopped and collision of the cranes is avoided.

Section 15.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	463
Structured Variable Description	466
Output Pin Description	468

Input Pin Description

Input Pin Description

The DoubleLimitSwitch_AR_2 function block is designed for use exclusively with Normally Closed (NC) sensors. If any of the limit switch inputs are not connected to a Normally Closed (NC) sensors, they have to be set to TRUE.

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_xSync	BOOL	TRUE: Enables the synchronization (Synchronous mode). FALSE: Disables the synchronization (Asynchronous mode).
i_xLsFwdStopDrv1	BOOL	Stop forward position drive 1. When the Stop forward is initiated the Forward output command ($q_xFwdDrv1$) signal is disabled. This signal is used to stop movement of the trolley/bridge/hoist in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsRevStopDrv1	BOOL	Stop reverse position drive 1. When the Stop reverse is initiated the Reverse output command ($q_xRevDrv1$) signal is disabled. This signal is used to stop movement of the trolley/bridge/hoist in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsFwdSlowDrv1	BOOL	Slow-down forward position drive 1. The Slow forward initiates the Slow forward ($q_xFwdSlowDrv1$) signal. This signal is used to move the trolley/bridge/hoist at low speed in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsRevSlowDrv1	BOOL	Slow-down reverse position drive 1. The Slow reverse initiates the Slow reverse ($q_xRevSlowDrv1$) signal. This signal is used to move the trolley/bridge/hoist at low speed in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xFwdDrv1	BOOL	Input for the forward commands of the drive 1. In Synchronous mode, the whole limit switch signals will control both drives. TRUE: Forward FALSE: Not forward

Input	Data Type	Description
i_xRevDrv1	BOOL	Input for the reverse commands of the drive 1. In Synchronous mode, the whole limit switch signals will control both drives. TRUE: Reverse FALSE: Not reverse
i_iSpdAct1Drv1 *	INT	Actual drive speed of drive 1. These values are provided as a feedback signal from the drives. Range: -6000...+6000 Scaling/Unit: 1 RPM
i_rDistStopDrv1 *	REAL	Stop Distance. Allowed distance for travel after passing the slow position, when Stop on Distance is used. To disable the Stop on Distance function, set this input to zero. Range: 1.175494351e-38F ... 3.402823466e+38F Scaling/Unit: 1 m
i_xLsFwdStopDrv2	BOOL	Stop forward position drive 2. When the Stop forward is initiated the Forward output command (q_xFwdDrv2) signal is disabled. This signal is used to stop movement of the trolley/bridge/hoist in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsRevStopDrv2	BOOL	Stop reverse position drive 2. When the Stop reverse is initiated the Reverse output command (q_xRevDrv2) signal is disabled. This signal is used to stop movement of the trolley/bridge/hoist in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsFwdSlowDrv2	BOOL	Slow-down forward position drive 2. The Slow forward initiates the Slow forward (q_xFwdSlowDrv2) signal. This signal is used to move the trolley/bridge/hoist at low speed in the forward direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xLsRevSlowDrv2	BOOL	Slow-down reverse position drive 2. The Slow reverse initiates the Slow reverse (q_xRevSlowDrv2) signal. This signal is used to move the trolley/bridge/hoist at low speed in the reverse direction. NOTE: If a limit switch sensor is not employed in the application, set the input to TRUE.
i_xFwdDrv2	BOOL	Input for the forward commands of the slave drive. The commands for both drives will be computed while the FB is in Synchronous mode. TRUE: Forward FALSE: Not forward

Input	Data Type	Description
i_xRevDrv2	BOOL	Input for the reverse commands of the slave drive. The commands for both drives will be computed while the FB is in Synchronous mode. TRUE: Reverse FALSE: Not reverse
i_rDistStopDrv2 *	REAL	Stop Distance. Allowed distance for travel after passing the slow position, when Stop on Distance is used. To disable the Stop on Distance function, set this input to zero. Range: 1.175494351e-38F ... 3.402823466e+38F Scaling/Unit: 1 m
i_iSpdActlDrv2 *	INT	Actual drive speed of drive 2. These values are provided as a feedback signal from the drives. Range: -6000...+6000 Scaling/Unit: 1 RPM
i_xDisLsAlrm	BOOL	Disables the plausibility check of combination of limit switches for both axes. The FB will not raise any alarms based on combination of limit switch inputs. TRUE: Disable the limit switch alarms. FALSE: Enable the limit switch alarms.
i_stSodAR *	SODAR_2	Parameter data structure for motor data of drive 1 and drive 2 for Stop on Distance. Range: 0...65535 (each value) Refer to Structured Variable Description (see page 466).
i_xRst	BOOL	On a rising edge, attempts to clear all alarms. If alarms stay active despite the rising edge of this input, the cause of the alarm is still present.

* optional, used for Stop on Distance

Structured Variable Description

SODAR_2 (i_stSodAR)

Structure Parameter	Data Type	Description
wSpdNomDrv1 *	WORD	Nominal motor speeds of the drive 1 motor used in the distance calculation. Range: 0..6000 Scaling/Unit: 1 RPM
rMotSpdLinDrv1 *	REAL	Linear speed of movement at nominal motor speed (wSpdNomDrv1). This is used in the distance calculation. Range: 0..5 Scaling/Unit: 1 m/s
wRampDrv1	WORD	Deceleration ramp for the drive 1. Range: 0..9999 Scaling/Unit: 0.1 s
wScalFactDrv1 *	WORD	Scale factor for corrections of distance calculation. Range: 0..200 Scaling/Unit: 1% (default is 100%) Refer to detailed description below.
wSpdNomDrv2 *	WORD	Nominal motor speeds of the drive 2 motor used in the distance calculation. Range: 0..6000 Scaling/Unit: 1 RPM
rMotSpdLinDrv2 *	REAL	Linear speed of movement at nominal motor speed (wSpdNomDrv2). This is used in the distance calculation. Range: 0..5 Unit: m/s
wRampDrv2	WORD	Deceleration ramp for the drive 2. Range: 0..9999 Scaling/Unit: 0.1 s
wScalFactDrv2 *	WORD	Scale factor for correction of distance calculation. Range: 0..200 Scaling/Unit: 1% (default is 100%) Refer to detailed description below.

* optional, used for Stop on Distance

NOTE: It is ill-advised to use the “slow down” limit switch for the calculation of stop distance between two bridges/ trolleys. When using the photoelectric sensor between the both bridges/ trolleys the configured stop distance between slow and stop positions must be reduced relative the speed of each crane, for example, by 50% if both cranes are moving at the same speed. This can be achieved in user application.

wScalFactDrv1, wScalFactDrv2

A scaling factor of 0...200% used to adjust the calculated distance traveled to the real distance.

The overall distance calculation is as follows:

$$D = \text{Scaling Factor} * \Sigma (\text{Actual Speed} * \Delta\text{Time})$$

$$= \text{Scaling Factor} * \Sigma (\text{Linear Speed} * \text{Actual Speed} * \text{Sample Rate} / \text{Nominal Speed}) = \\ i_wScalFct[\%] * \Sigma(i_wLinSpd[\text{m/s}] * i_iActSpd[\text{RPM}] * i_wSmpRate[\text{ms}] / \\ i_wNomSpd[\text{RPM}])$$

Example:

Nominal speed of motor in RPM	1500 (from user)
Nominal linear speed in m/s	1.00 (from user)
Scaling factor in %	100 (from user)
Actual speed in RPM	750 (from drive)
Sample rate in ms	40 (from controller)

$$\text{Distance in current cycle} = 100\% * 1 \text{ m/s} * 750 \text{ RPM} * 40 \text{ ms} / 1500 \text{ RPM} = 0.02 \text{ m}$$

wRampDrv1, wRampDrv2

This input corresponds to the deceleration ramp time of the drive.

Example:

Nominal speed of motor in RPM	1500
Deceleration ramp in seconds	5
i_wRampDrv1	50

NOTE: The parameter assumes a drive default ramp increment configuration of 0.1 seconds. The unit-base used for this parameter must be the same as that configured in the drive. Further, the drive deceleration ramp time must be the same as for this parameter as it is configured in the drive.

This means that the time needed to reach the zero speed from the nominal speed of 1500 RPM is 5 seconds.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled FALSE: Function block disabled
q_xSync	BOOL	Synchronization output. TRUE: Function block in Synchronous mode FALSE: Function block in Asynchronous mode
q_xFwdDrv1	BOOL	Drive 1 forward run command. TRUE: Forward FALSE: Not forward
q_xRevDrv1	BOOL	Drive 1 reverse run command. TRUE: Reverse FALSE: Not reverse
q_xFwdSlowDrv1	BOOL	Force low speed forward motion on drive 1. While synchronization both drives will be forced. TRUE: Forward slow FALSE: Not forward slow
q_xRevSlowDrv1	BOOL	Force low speed reverse motion on drive 1. While synchronization both drives will be forced. TRUE: Reverse slow FALSE: Not reverse slow
q_xFwdDrv2	BOOL	Drive 2 forward run command. TRUE: Forward FALSE: Not forward
q_xRevDrv2	BOOL	Drive 2 reverse run command. TRUE: Reverse FALSE: Not reverse
q_xFwdSlowDrv2	BOOL	Force low speed forward motion on drive 2. While synchronization both drives will be forced. TRUE: Forward slow FALSE: Not forward slow
q_xRevSlowDrv2	BOOL	Force low speed reverse motion on drive 2. While synchronization both drives will be forced. TRUE: Reverse slow FALSE: Not reverse slow
q_wLsStatDrv1	WORD	Status of limit switch management on drive 1 limit switch. Range: 0, 1, 2, 3, 4 Refer to detailed description below.
q_wLsStatDrv2	WORD	Status of limit switch management on drive 2 limit switch. Range: 0, 1, 2, 3, 4 Refer to detailed description below.

Output	Data Type	Description
q_rDistTrvlDrv1	REAL	Output of the Stop on Distance calculation. Whenever this value exceeds the value of i_wStopDistDrv1, a stop is executed. Range: 0...65535 Scaling/Unit: 1 m
q_rDistTrvlDrv2	REAL	Output of the Stop on Distance calculation. Whenever this value exceeds the value of i_wStopDistDrv2, a stop is executed. Range: 0...65535 Scaling/Unit: 1 m
q_xAlrm	BOOL	Detected alarm bit. TRUE: Alarm detected FALSE: No alarm detected. Refer to detailed description below.
q_wAlrmId	WORD	Detected alarm identification. Refer to q_wAlrmId (<i>see page 470</i>). Range: 0...63

q_wLsStatDrv1, q_sLsStatDrv2

Status Bit	Description
0	trolley/bridge/hoist is in the working area.
1	trolley/bridge/hoist is between the Slow forward and Stop forward positions.
2	trolley/bridge/hoist is beyond the Slow forward and Stop forward positions.
3	trolley/bridge/hoist is between the Slow reverse and Stop reverse positions.
4	trolley/bridge/hoist is beyond the Slow reverse and Stop reverse positions.
5	Detection of active area and check of limit switch input combination is disabled.

q_xAlrm

An alarm signal is generated in the cases of incorrect limit switch configuration or improper operation of a limit switch.

The function block detects any incorrect signalling of the position with respect to the movement, i.e. during movement, the signalling of any limit switch position located in the opposite direction of the movement generates an alarm.

When the FB is applied for 2 bridges/trolleys moving on the same rail and if the input i_xDisLsAlrm is set to TRUE the function block disables the plausibility check of the limit switch inputs which detects a wrong order of limit switch signals.

q_wAlrmId

Bit Position	Description Represented by Bit Position
0	wrong order of limit switch signals detected on drive 1 axis
1	DrvSpdNomDrv1 is zero although Stop on Distance is enabled
2	wMotSpdLinDrv1 is zero although Stop on Distance is enabled
3	wScalFactDrv1 is zero although Stop on Distance is enabled
4	rMotSpdLinDrv1 is negative
5	wrong order of limit switch signals detected on drive 2 axis
6	wDrvSpdNomDrv2 is zero although Stop on Distance is enabled
7	wMotSpdLinDrv2 is zero although Stop on Distance is enabled
8	wScalFactDrv2 is zero although Stop on Distance is enabled
9	rMotSpdLinDrv2 is negative

Section 15.5

Quick Reference Guide

What Is in This Section?

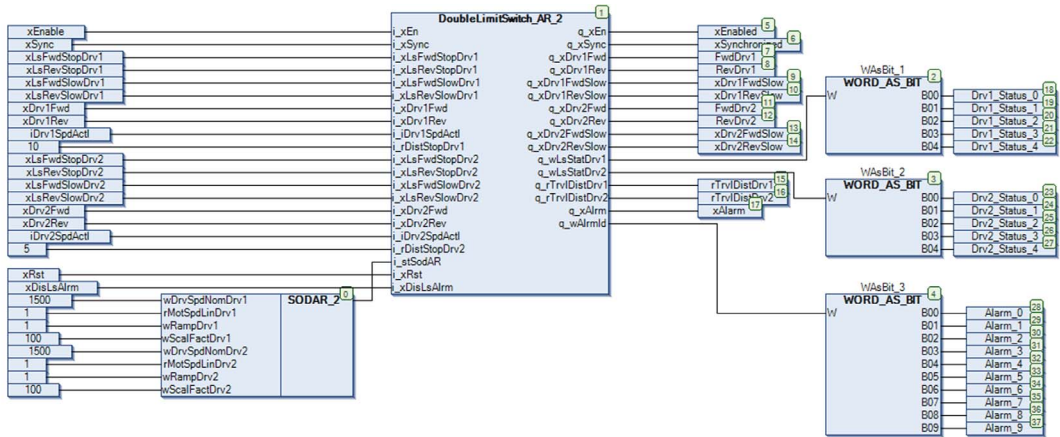
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	472
Commissioning Procedure	473
Troubleshooting	474

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the DoubleLimitSwitch_AR_2 function block:



Commissioning Procedure

Commissioning Procedure of the DoubleLimitSwitch_AR_2 Function Block

The following steps explain the procedure on how to configure the DoubleLimitSwitch_AR_2 function block:

1. Create an instance of the DoubleLimitSwitch_AR_2 function block and connect its inputs and outputs.
2. Set the deceleration ramp of the drive to the same as value given to the FB.
3. The distance between slow and stop limit switch should be sufficient for the axis to decelerate from the maximum speed to a defined slow speed. This one must be the same for both drives.
4. If you intend to use the FBs adaptive ramp functionality, set all parameters of the SODAR_2 input pin `i_stSODAR` as well as `i_rDistStopDrv1` and `i_rDistStopDrv2`.
5. To use the non-adaptive ramps operation, put zero to the `i_rDistStopDrv1` and `i_rDistStopDrv2` inputs.
6. Test the function by moving the axis into the slow-down area at various speeds.

Alarm Fallback State, Disable Fallback State

In the case of an alarm, or if the FB is disabled, all output speed authorizations are set to FALSE:

<code>q_xFwdDrv1</code>	<code>:=False;</code>
<code>q_xRevDrv1</code>	<code>:=False;</code>
<code>q_xFwdSlowDrv1</code>	<code>:=False;</code>
<code>q_xRevSlowDrv1</code>	<code>:=False;</code>
<code>q_xFwdDrv2</code>	<code>:=False;</code>
<code>q_xRevDrv2</code>	<code>:=False;</code>
<code>q_xFwdSlowDrv2</code>	<code>:=False;</code>
<code>q_xRevSlowDrv2</code>	<code>:=False;</code>

Troubleshooting

Troubleshooting

Issue	Cause	Solution
q_xAlrm of the FB is TRUE.	One or more of the input parameters are out of range.	Check the input parameters.

Part X

Load Overspeed Control

Overview

This part explains the functionality and implementation of `LoadOverspeedCtrl` function block in the Hoisting Library.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
16	LoadOverspeedCtrl: Detects Load Overspeed, Brake Wear and Sensor Feedback on Hoist	477
17	LoadOverspeedCtrl_2: Detects Load Overspeed, Brake Wear and Sensor Feedback on Hoist	501

Chapter 16

LoadOverspeedCtrl: Detects Load Overspeed, Brake Wear and Sensor Feedback on Hoist

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `LoadOverspeedCtrl_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
16.1	Functional and Machine Overview	478
16.2	Architecture	482
16.3	Function Block Description	485
16.4	Pin Description	493
16.5	Troubleshooting	500

Section 16.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	479
Machine Overview	481

Functional Overview

Functional Description

The Load Overspeed Control function block (`LoadOverspeedCtrl`) is used to detect overspeed, brake wear, alarm and sensor feedback alarms.

The Load Overspeed Control function (`LoadOverspeedCtrl`) is a function which helps to protect hoisting against load overspeed and detects brake wear. The Load Overspeed Control function detects a load overspeed by monitoring the pulse input of the controller. The brake wear function checks the wear and tear of the hoist brake. This function detects any movement on the load when the drive is not running.

The function is used to detect:

- Load overspeed
- Brake wear
- Sensor feedback

The `LoadOverspeedCtrl` function block is applicable to:

- Construction cranes
- Industrial cranes

The `LoadOverspeedCtrl` function block is designed to be used on the ATV71 and ATV312 drives.

The function block monitors the drive movement and a reference point, which is usually a simple proximity sensor linked to a cog behind the gear box. The sensor must be connected to the high speed input of the controller.

The function compares the drive movement information with the maximum speed of the drive and generates an alarm, if a higher speed than the specified is detected.

For the brake wear detection, the function block generates an alarm, if the sensor signal indicates movement when the drive is not in RUN mode.

A sensor feedback alarm is generated if the speed sensor does not deliver any signal although the drive is in RUN mode.

NOTE: A sensor feedback alarm is detected when the actual speed from the drive is higher than the threshold speed.

Why Use the `LoadOverspeedCtrl` Function Block?

Load overspeed and brake wear have to be taken into account for the controlled working of the hoisting mechanism. Any situation caused by a load moving too fast or that cannot be stopped by the brake could be potentially dangerous for humans and machinery.

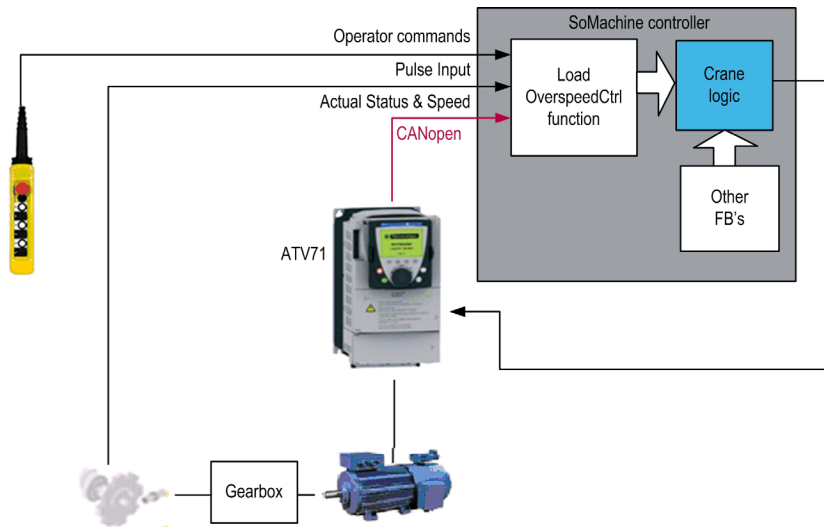
Solution with the LoadOverspeedCtrl Function Block

The LoadOverspeedCtrl function block monitors the speed using toothed wheel at the load and in turn converts function block into frequency. This frequency is compared with the set frequency to generate alarms.

Design & Realization Constraints and Assumptions

- The frequency calculation for load overspeed detection can only be done once the drive is in RUN mode.
- The frequency calculation for brake wear detection is done only when the drive is not in RUN mode.

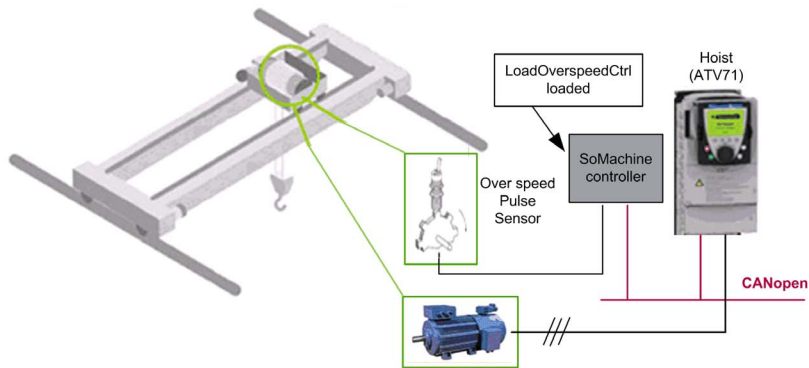
Functional View



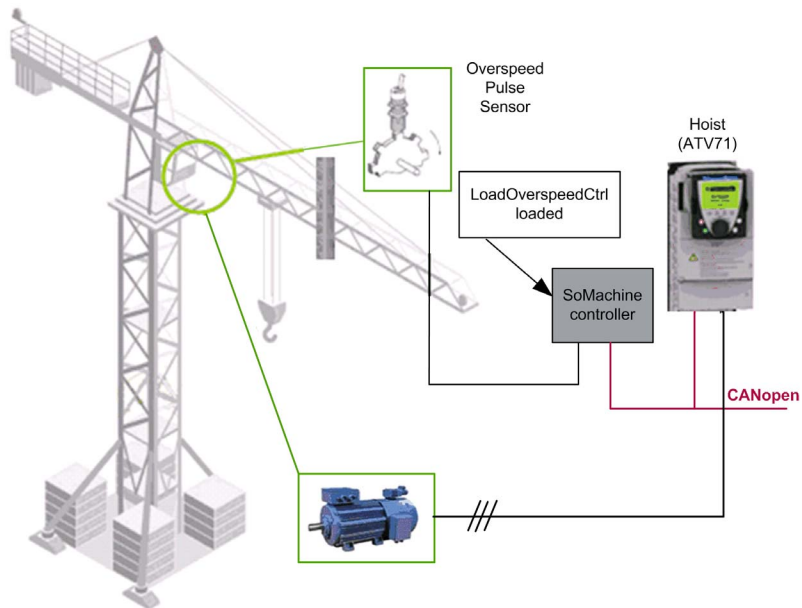
Machine Overview

Machine View

The following figure represents the machine view of the function block with an industrial crane.



The following figure represents the machine view of the function block with a tower crane.



Section 16.2

Architecture

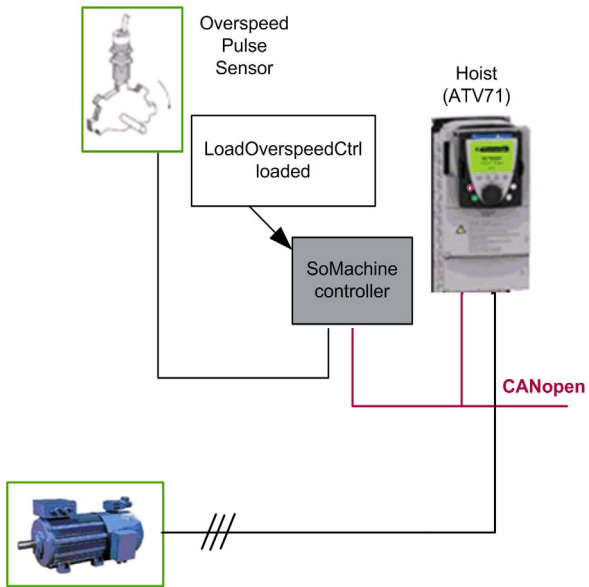
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	483
Software Architecture	484

Hardware Architecture

Hardware Architecture Overview

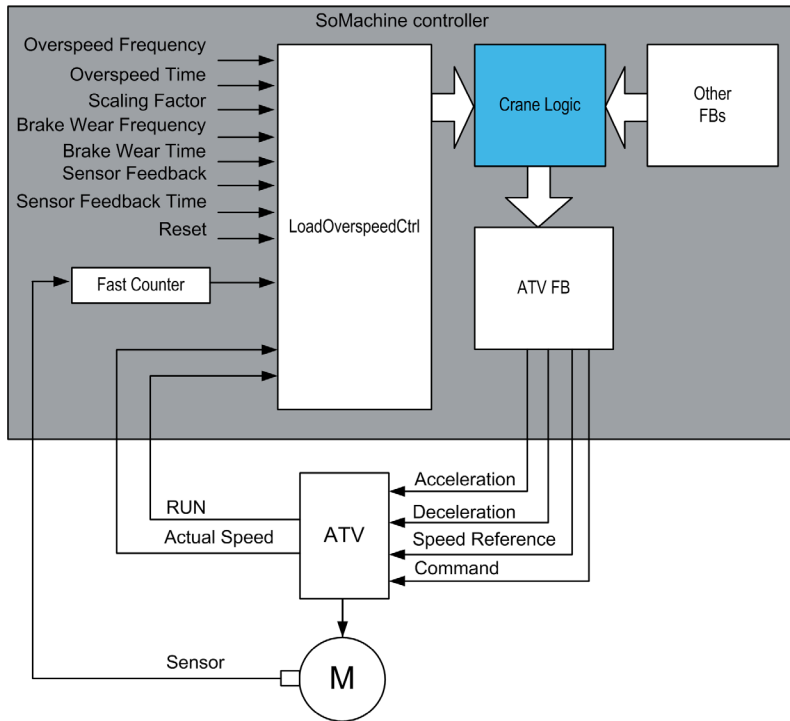


NOTE: The high speed pulses from the proximity sensor must be connected to a high speed input.

Software Architecture

Software Architecture Overview

In the Load overspeed control function, the drive is an ATV71 with an SoMachine controller connected through CANopen. The threshold values and alarm time-out values (parameters) are configured for the function block. The pulses from the proximity sensor are passed to one of the fast counter inputs of the controller. In the controller, specific counter function blocks are used to count the pulses. If a load overspeed or brake wear or sensor feedback incident occurs for longer than the specified time, an alarm is detected.



Section 16.3

Function Block Description

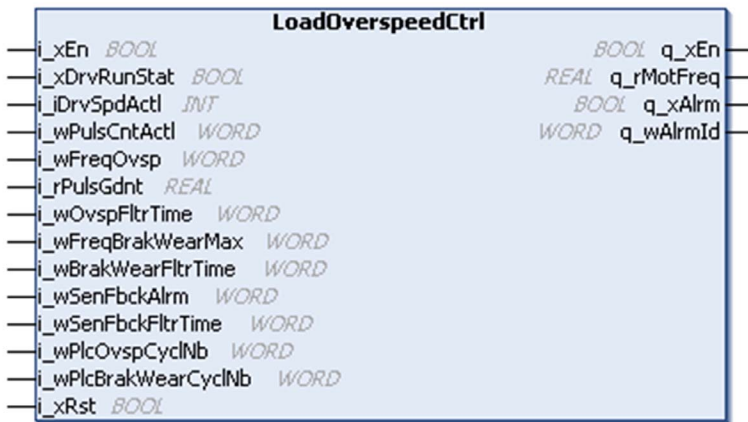
What Is in This Section?

This section contains the following topics:

Topic	Page
LoadOverspeedCtrl Function Block	486
Load Overspeed Detection and Sensor Feedback Alarm Detection	488
Brake Wear Detection	491

LoadOverspeedCtrl Function Block

Pin Diagram

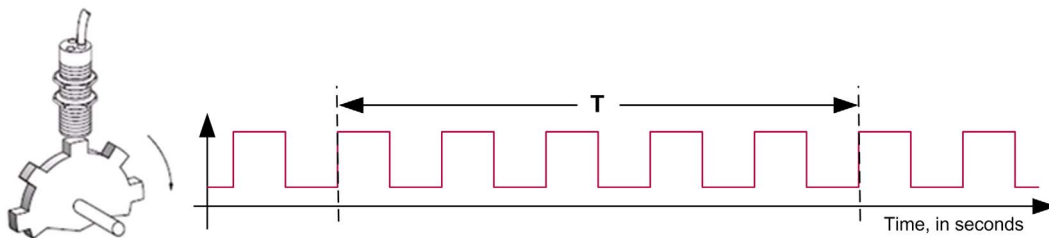


Function Block Description

The Load overspeed control function is used to generate an overspeed alarm and brake wear alarm whenever the calculated frequency exceeds the threshold frequency of overspeed and brake wear respectively. The method of frequency calculation is explained below:

The pulses from the fast counter are passed to the LoadOverspeedCtrl function block through the *i_wPulsCntActl* input.

The following figure represents the frequency calculation by LoadOverspeedCtrl function block.



T Represents the period for which the pulses are measured.

Load Overspeed Detection

This function compares the pulse readings with a user-defined threshold (`i_wFreqOvsp`) when the drive is in the RUN state. If pulse reading is greater than load overspeed frequency, load overspeed alarm is generated.

Brake Wear Detection

This function detects a movement of the load when the drive is not in the **RUN** state. The detected alarm is active as long as the pulse input is higher than the frequency threshold value (`i_wFreqBrakWearMax`).

The following steps explain how the alarm is detected:

- The scaling factor value for frequency, brake wear frequency, and brake wear delay time must be configured.
- The pulse input is converted internally into a frequency, where the number of pulses are monitored over a period of controller scan.
- If the sensor pulse multiplied with a scaling factor is greater than the brake wear frequency, even after the delay time, a brake wear alarm is generated.

Sensor Feedback Alarm Detection

If the actual motor frequency (`i_iDrvSpdAct1`) is higher than the feedback threshold frequency input (`i_wSenFbckAlrm`), and the pulse input is zero, then a sensor feedback alarm is generated.

NOTE: The alarms are reset with a power cycle to the controller or over the reset input `i_xRst`.

Load Overspeed Detection and Sensor Feedback Alarm Detection

Calculation of the Frequency

Calculation of the frequency for the load overspeed detection or sensor feedback alarm is done using the number of pulses over a definite number of controller cycles. The number of controller cycles is defined at `i_wPlcOvspCyclNb` input.

Calculated frequency is the rotation of the shaft of the motor and not the output frequency of the variable speed drive.

Example:

Controller scan time = 100 ms

Pulse Gradient = 1

Number of controller cycles for load overspeed detection = 2

If the number of pulses received over 2 controller cycles is 10, then the frequency is: (Number of pulses received over 2 controller cycles * 1000 * Pulse Gradient) / (controller scan time for 2 cycles)
 Frequency = $(10 * 1000 * 1) / (200) = 50 \text{ Hz}$

NOTE:

- The frequency calculation for load overspeed detection can only be done once the drive is in **RUN** mode.
- You should set both inputs `i_wPlcOvspCyclNb` and `i_wPlcBrakWearCyclNb` according to the cycle time of the controller and the response time of the CANopen bus.
- You need to verify that the actual pulse values reach the function block with sufficient time to calculate a frequency. Check the frequency output of the block and compare it to the target speed of the drive.

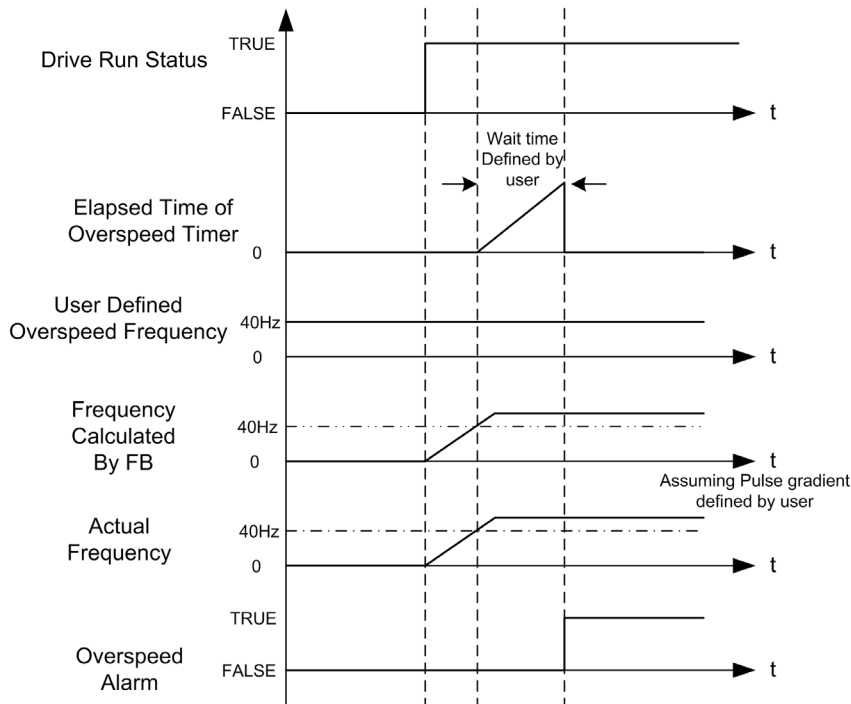
Generation of a Load Overspeed Alarm

The steps for Load overspeed detection are as follows:

Step	Action
1	The frequency threshold value above which a load overspeed alarm generated is defined with the <code>i_wFreqOvsp</code> input of the function block.
2	If the value of the calculated frequency exceeds the frequency defined at <code>i_wFreqOvsp</code> input of the function block, then a timer is started. (The duration of this time period is defined at the <code>i_wOvspFltrTime</code> input of the function block.)
3	The frequency calculation for the load overspeed detection is done by using the number of pulses over a number of controller scans defined at <code>i_wPlcOvspCyclNb</code> .

Step	Action
4	If the calculated frequency continues to exceed the defined frequency at <code>i_wFreqOvsp</code> , even after the load overspeed alarm timer has timed out, then a load overspeed alarm is generated at the output pin (Bit 0 of <code>q_wAlrmId</code>) of the function block.

The following figure represents the timing diagram for Load overspeed alarm.



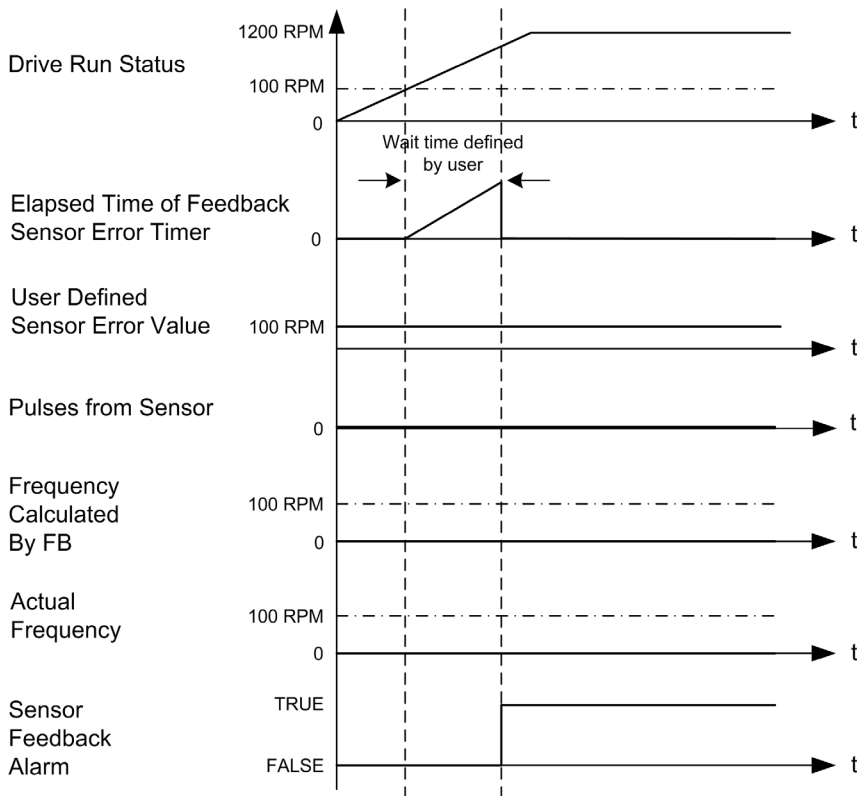
Generation of a Sensor Feedback Alarm

The steps for generating a sensor feedback alarm are as follows:

Step	Action
1	The RPM value above which a Sensor feedback alarm must be generated is defined at the <code>i_wSenFbckAlrm</code> input of the function block.
2	If the actual speed of the drive read at <code>i_iDrvSpdAct1</code> exceeds the value defined at <code>i_wSenFbckAlrm</code> , then a timer is started. (The duration of this time period is defined at the <code>i_wSenFbckFltrTime</code> input of the function block.)

Step	Action
3	If the actual speed of the drive continues to exceed the user defined value at <code>i_wSenFbckAlrm</code> , even after the sensor feedback alarm timer has timed out and no pulses are registered, then a Sensor feedback alarm is generated at the output pin (Bit 2 of <code>q_wAlrmId</code>) of the function block.

The following figure represents the timing diagram for Sensor feedback alarm.



Brake Wear Detection

Calculation of the Frequency

Frequency calculation for brake wear detection is the same as Load overspeed detection. In this case, the number of controller cycles for brake wear detection is at the `i_wPlcBrakWearCyclNb` input.

Calculated frequency is the rotation of the shaft of the motor and not the output frequency of the variable speed drive.

Example:

controller scan time = 100 ms

Pulse Gradient (`i_rPulsGdnt`) = 1

Number of controller cycles for Brake wear detection = 20

If the number of pulses received over 20 controller cycles is = 100,

then the frequency is: (Number of pulses received over 20 controller cycles * 1000 * Pulse Gradient) / (controller scans time for 20 cycles)

Calculated frequency = $(100 * 1000 * 1) / (2000) = 50 \text{ Hz}$

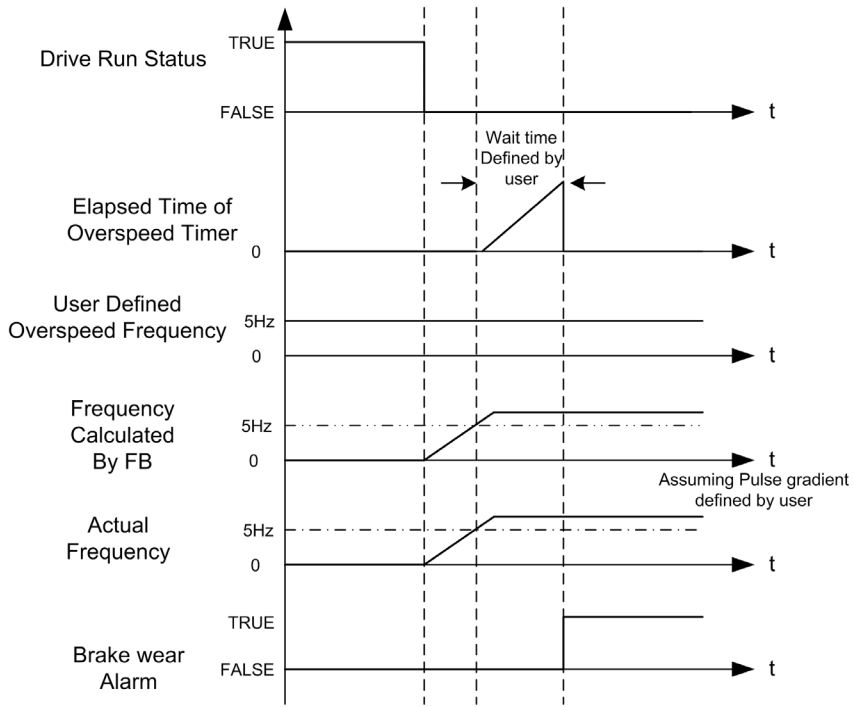
NOTE: The frequency calculation for brake wear detection is done only when the drive is not in RUN mode.

Generation of a Brake Wear Alarm

The steps for generating a brake wear alarm are as follows.

Step	Action
1	The frequency value above which a brake wear alarm is generated is defined at the <code>i_wFreqBrakWearMax</code> input of the function block.
2	If the value of the calculated frequency exceeds the brake wear frequency defined at the <code>i_wFreqBrakWearMax</code> input of the function block, then a timer is started. (The duration of this time period is defined at the <code>i_wBrakWearFltrTime</code> input of the function block.)
3	The frequency calculation for the brake wear detection is done using the number of pulses over a time period and number of controller scans defined at <code>i_wPlcBrakWearCyclNb</code> .
4	If the calculated frequency continues to exceed the user defined frequency at <code>i_wFreqBrakWearMax</code> even after the brake wear alarm timer has timed out, then a brake wear alarm is generated at the output pin (Bit 1 of <code>q_wAlrmId</code>) of the function block.

The following figure represent the timing diagram for Brake wear alarm.



Section 16.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	494
Output Pin Description	498

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (<i>see page 496</i>) of i_xEn.
i_xDrvRunStat	BOOL	Drive run status. During normal usage, you need to extract the RUN bit from the drive status word. See the documentation of the associated drive. TRUE: Drive run. FALSE: Drive not run.
i_iDrvSpdActl	INT	Actual speed of the motor in RPM units as reported by the drive through CANopen. Range: -6000...6000 RPM
i_wPulsCntActl	WORD	Actual value of the pulse counter. Range: 0...65535 count
i_wFreqOvsp	WORD	Frequency above which the overspeed alarm is triggered. This input represents the set point frequency which is compared with the actual frequency from sensor connected at the load end. Range: 0...1200 Scaling/Unit: 0.1 Hz
i_rPulsGdnt	REAL	Ratio between motor and sensor wheel. The number of teeth on the sensor cog and also the gear-box ratio (if any) should be taken into consideration. Refer to detailed description (<i>see page 496</i>) of i_rPulsGdnt. Range: 1.18E-38...100
i_wOvspFltrTime	WORD	User defined time period to wait before registering a load overspeed alarm. Used to avoid the possibility that small jerks in the system are registered as alarms. Range: 0...200 Scaling/Unit: 0.1 s

Input	Data Type	Description
i_wFreqBrakWearMax	WORD	Frequency for monitoring the brake wear alarm. If the drive is not in RUN state but movement is still registered by the sensor wheel (frequency more than i_wFreqBrakWearMax), then it indicates that the mechanical brakes are slipping or inoperative. Range: 0...1200 Scaling/Unit: 0.1 Hz
i_wBrakWearFltrTime	WORD	User defined time period to wait before registering a brake wear alarm. Used to avoid the possibility that small jerks in the system are registered as alarms. Range: 0...200 Scaling/Unit: 0.1 s
i_wSenFbckAlrm	WORD	User defined RPM value below which the sensor feedback alarm should not be triggered. Example: If the actual speed is more than i_wSenFbckAlrm and no pulses are registered, then a sensor feedback alarm occurs. Range: 0...6000 RPM
i_wSenFbckFltrTime	WORD	User defined time period to wait before registering a sensor feedback alarm. Used to avoid the possibility that small jerks in the system are registered as alarms. Range: 0...200 Scaling/Unit: 0.1 s
i_wPlcOvspCyclNb	WORD	Number of controller scans required for the load overspeed detection. The i_wPulsCntActl input should increase by sufficient number of pulses during given number of cycles (executions) to be able to calculate motor frequency with a good resolution. Setting this value too high results in slow response time. Range: 1...65535 count Factory setting = 1

Input	Data Type	Description
i_wPlcBrakWearCyclNb	WORD	Number of controller scans required for the brake wear detection. The i_wPulsCntAct1 input should increase by sufficient number of pulses during given number of cycles (executions) to be able to calculate motor frequency with a good resolution. In case of a brake wear causing movement of the load in stop state, an appropriate value should be set to detect the undesirable movement correctly. Range: 1...65535 count Factory setting = 1
i_xRst	BOOL	Resets detected alarms TRUE: Active. FALSE: Inactive.

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of LoadOverspeedCtrl are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_rMotFreq	REAL	0
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

i_rPulsGdnt

The LoadOverspeedCtrl function block calculates the pulse frequency of incoming pulses from the cog in Hz. This frequency and the pulse gradient are used to calculate the speed of the motor.

The condition for overspeed is as follows:

Pulse Frequency * Pulse Gradient = Motor Frequency.

The pulse gradient needs to take into account the geometry of the gear box (ratio) as well as geometry of the cog wheel (number of teeth on the cog).

Example

Nominal motor speed = 1500 RPM

Nominal speed at the tooth wheel end = 750 RPM

Gearing ratio = $1500 / 750 = 2$

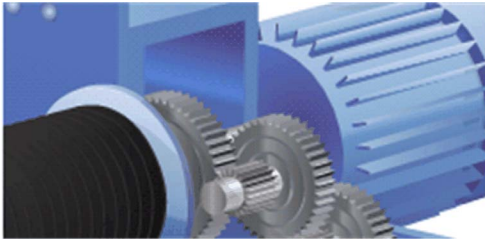
Number of teeth on the tooth wheel = 50

The pulse gradient is equal to:

Pulse gradient = $\text{Gear ratio} / \text{Number of tooth} = 2 / 50 = 0.04$.

So, if pulses arrive with a frequency of 750 Hz, the shaft of the motor turns with $750 \text{ Hz} * 0.04 = 30 \text{ Hz}$. The overspeed frequency can be directly set as a maximum allowed frequency.

The following figure represents the motor with gear box.



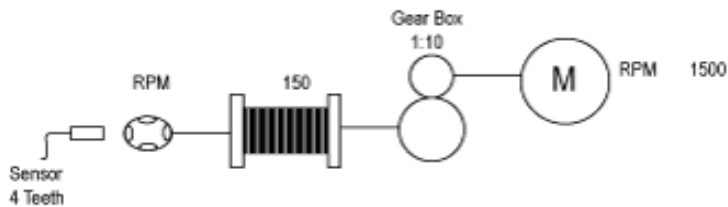
Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_rMotFreq	REAL	Calculated rotations of the shaft of the motor based on the received pulses. Refer to detailed description (see page 498) of q_rMotFreq. Range: 0...1200 Scaling/Unit: 1,0 Hz
q_xAlrm	BOOL	Alarm detection indicator TRUE: Alarm detected. FALSE: No alarm detected.
q_wAlrmId	WORD	Detected Alarm identification. Refer to Notifications (see page 499). Range: 0...7

q_rMotFreq

This output gives information about the (mechanical) motor frequency based on the received pulses from the sensor.



Example:

1. 1 rotation of the drum causes 4 pulses on the sensor.

2. 1 rotation of the drum needs 10 rotations of the motor axis.

==> 4 pulses per second (4 Hz as pulse frequency) in 10 rotations of the motor axis per second is 10 Hz. So the pulse gradient is $10 / 4 = 2.5$

Motor Frequency = Pulse Gradient * Pulse frequency

The function block detects overspeed if the motor frequency is higher than the overspeed frequency.

Brake wear is detected if the motor frequency is higher than the brake wear frequency.

Calculated frequency is the rotation of the shaft of the motor and not the output frequency of the Variable Speed Drive.

Notifications

Bit Number	Description
0	Overspeed
1	Brake wear
2	Sensor feedback

NOTE: The individual bits of the alarm status output need to be extracted using the '.' (DOT) and interlocked suitably with the crane operation program. The function does not have direct control of the drive.

Section 16.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
Parameter <code>q_rMotFreq</code> does not correspond to value in the display of drive	Settings for calculation wrong	Check value calculation for input parameter <code>i_rPulsGdnt</code>
Feedback sensor provides no information or is not connected but does not generate an alarm	Value set too high for filter time	Input parameter <code>i_wSenFbckFltrTime</code> needs to be decreased from current value, factory setting is 0.1 s

Chapter 17

LoadOverspeedCtrl_2: Detects Load Overspeed, Brake Wear and Sensor Feedback on Hoist

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
17.1	Functional and Machine Overview	502
17.2	Architecture	507
17.3	Function Block Description	510
17.4	Pin Description	515
17.5	Quick Reference Guide	521

Section 17.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	503
Machine Overview	505

Functional Overview

Functional Description

The Load Overspeed Control 2 function block (`LoadOverspeedCtrl_2`) is used to detect load overspeed and brake wear. If the hoist moves at a speed faster than configured overspeed threshold the function block raises an overspeed alarm. The function block also detects movement of the hoist when the drive is not in RUN mode. The sensor feedback function tests the connection between the sensor and the controller input and generates an alarm if the sensor does not deliver any signal although the motor is running.

The function is used to detect:

- Load overspeed
- Brake wear
- Sensor feedback loss

The `LoadOverspeedCtrl_2` function block is applicable to:

- Construction cranes
- Industrial cranes

Why Use the `LoadOverspeedCtrl_2` Function Block?

Load overspeed and brake wear must be considered for the correct operation of the hoisting mechanism. Any situation caused by a load moving too fast or that cannot be stopped must be detected.

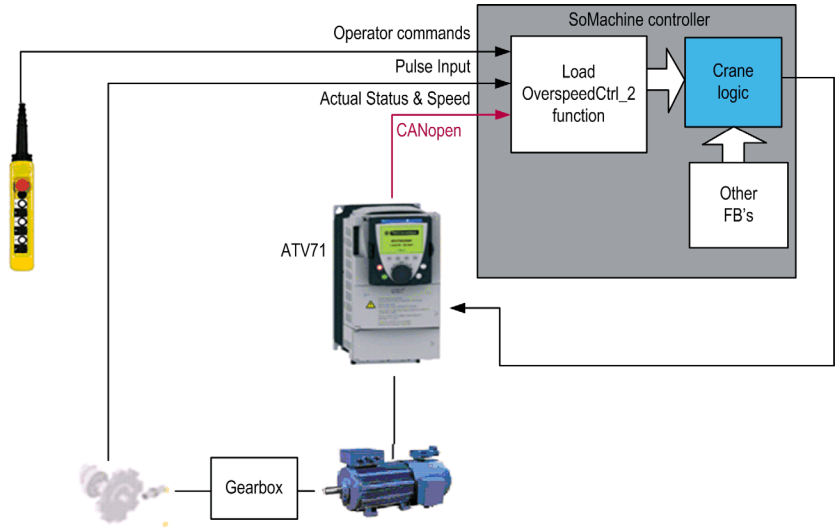
This FB is an evolution of `LoadOverspeedCtrl` FB. It has a simplified interface and higher precision of measurement. There are several modifications to minimize the reaction time after the detecting of overload, brake wear and sensor feedback alarms and to improve the compatibility with other hoisting function blocks.

Solution with the `LoadOverspeedCtrl_2` Function Block

`LoadOverspeedCtrl_2` FB monitors the speed using a proximity sensor. It senses teeth of a cogwheel connected to the drum of the hoisting axis. The FB uses the pulses to calculate speed of rotation of the motor shaft in RPM (revolutions per minute). This calculated speed is compared with the predefined threshold speed and an alarm is raised if the calculated speed exceeds the overspeed threshold.

An encoder can be used instead of a proximity sensor as long as it is connected to the drum of the hoisting axis and not directly to the motor shaft. It is important to measure actual speed of the drum rather than speed of the motor shaft in case the mechanical connection between the motor and the drum was interrupted.

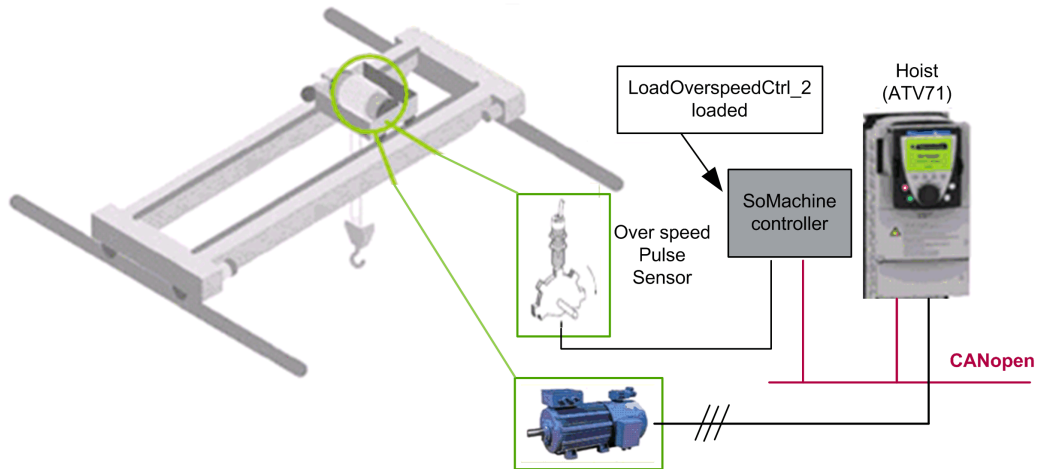
Functional View



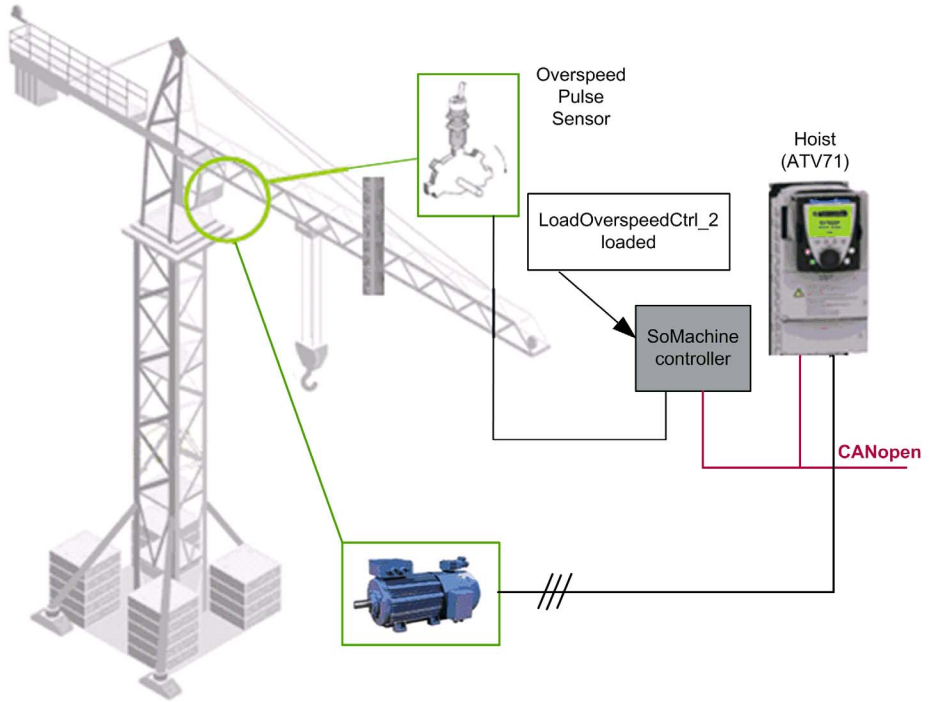
Machine Overview

Machine View

The following figure represents the machine view of the function block with an industrial crane.



The following figure represents the machine view of the function block with a tower crane.



Section 17.2

Architecture

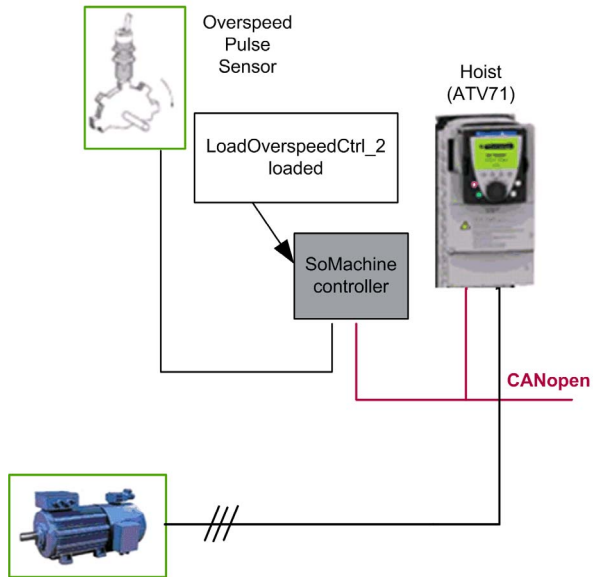
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	508
Software Architecture	509

Hardware Architecture

Hardware Architecture Overview



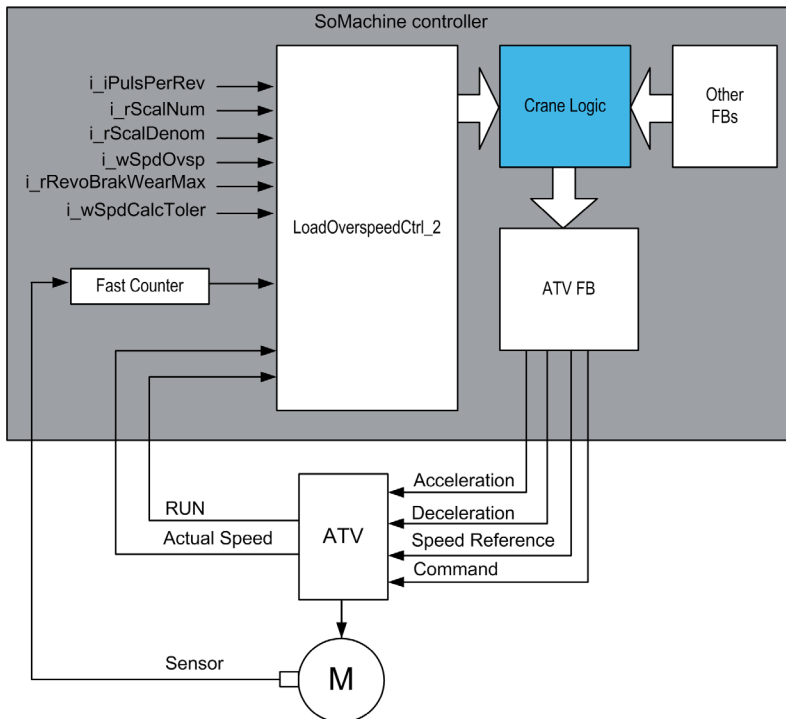
NOTE: The high speed pulses from the proximity sensor must be connected to a high speed input of the controller.

Software Architecture

Software Architecture Overview

The maximum allowed speed for detecting overspeed, the maximum allowed rotation of motor for detecting brake wear, tolerance of motor speed accuracy and gear box ratio factors have to be set during commissioning. The pulses from the proximity sensor are passed to one of the fast counter inputs of the controller. If a load overspeed, brake wear or sensor feedback incident occurs, an alarm is raised.

The function block contains an alarm which is raised if the inputs are not configured correctly.



Section 17.3

Function Block Description

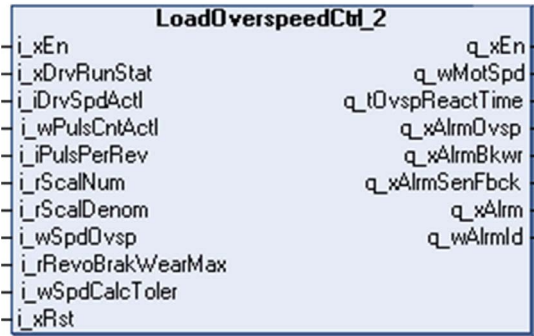
What Is in This Section?

This section contains the following topics:

Topic	Page
LoadOverspeedCtrl_2 Function Block	511
Load Overspeed Detection	512
Brake Wear Detection	514

LoadOverspeedCtrl_2 Function Block

Pin Diagram



Load Overspeed Detection

This function collects pulses from the proximity sensor or encoder and converts them into motor speed in RPM according to the predefined speed tolerance, gear box ratio and cogwheel parameters. The proximity sensor can be mounted directly behind the gear box on the cogwheel or on an additional wheel which is connected to the drum shaft.

If the drive is in RUN state and the calculated speed exceeds the user-defined threshold value `i_wSpdOvsp`, the function block raises a load overspeed alarm.

Brake Wear Detection

This function detects a sinking of the load when the drive is not in RUN state. The detected alarm is raised if the calculated revolutions are higher than the user-defined threshold value `i_rRevoBrakWearMax`.

Sensor Feedback Alarm Detection

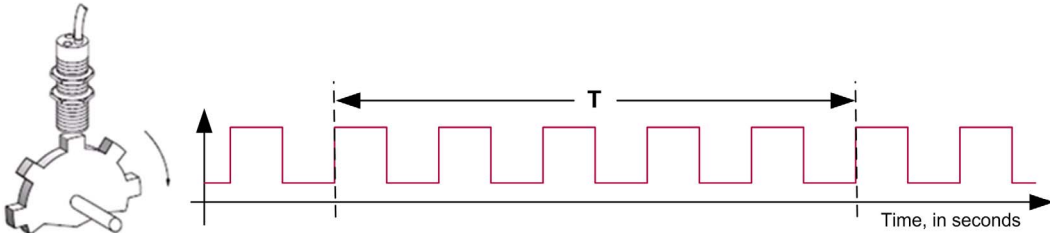
If the actual speed of the motor is higher than tolerance `i_wSpdCalcToler` of the predefined Load overspeed value `i_wSpdOvsp` and the sensor does not detect incoming pulses, a sensor feedback alarm is raised.

NOTE:

- The alarms are reset with a power cycle to the controller or on a rising edge of the reset input `i_xRst`.
- The alarm status outputs have to be used with the crane control program. This function block does not have direct control of the drive.

Load Overspeed Detection

Calculation of the Pulse Frequency

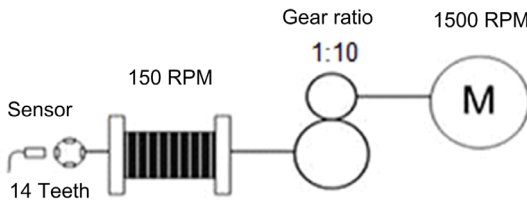


T Represents the period for which the pulses are measured.

The most important feature of an overspeed detection function is a fast reaction in case of an overspeed. The function block adjusts the period of measurement according to requested accuracy of the measurement and number of detected pulses. Measurement period gets shorter as the frequency of pulses increases. This allows the FB to react quickly in case of an overspeed situation. The accuracy and reaction time at low speeds are secondary.

Example:

The following example illustrates the ratio between motor and cogwheel fixed on the drum.



- 1 rotation of the drum causes 14 pulses on the sensor
- 1 rotation of the drum causes 10 rotations of the motor shaft
- number of teeth on the cogwheel is 14
- numerator factor is 10
- denominator factor is 1

The function block calculates the pulse frequency of incoming pulses from the cogwheel. These pulses and the motor gear box ratio (numerator factor / denominator factor) are used to calculate the speed of the motor.

In case that only the gear box ratio is given on the nameplate of the gear box, the numerator factor is set to the gear box ratio and the denominator to 1.

If the sensor is not attached directly on the drum behind the gear box but further on the cogwheel, the ratio between drum and the cogwheel must be taken into account.

Brake Wear Detection

Calculation of the Pulse Frequency

When the drive is not in RUN state and the motor moves, the function block collects the pulses and converts them to the revolutions of the motor shaft. If the calculated rotations are greater than the defined revolution threshold value `i_rRevoBrakWearMax` input, the break wear alarm is raised.

Section 17.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	516
Output Pin Description	519

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (see page 517) of i_xEn.
i_xDrvRunStat	BOOL	Drive run status. During normal usage, you need to extract the RUN bit from the drive status word. See the documentation of the associated drive. TRUE: Drive run. FALSE: Drive not run.
i_iDrvSpdActl	INT	Actual speed of the motor in RPM units as reported by the drive through CANopen. Range: -6000...6000 RPM
i_wPulsCntActl	WORD	Actual value of the pulse counter. Range: 0...65535 pulses
i_iPulsPerRev	INT	Number of pulses per revolution of the cogwheel or encoder. Range: 0...32767 encoder
i_rScalNum	REAL	Numerator of the gear ratio between the motor and the cogwheel or encoder used for speed measurement. Range: 0...3.4E+38
i_rScalDenom	REAL	Denominator of the gear ratio between the motor and the cogwheel or encoder used for speed measurement. Range: 0.1...3.4E+38
i_wSpdOvsp	WORD	Overspeed threshold. Refer to detailed description (see page 517) of i_wSpdOvsp. By setting this input to zero (0), the detection of load overspeed alarm is deactivated. Range: 0...65535 RPM
i_rRevoBrakWearMax	REAL	Brake wear distance threshold. Refer to detailed description (see page 518) of i_rRevoBrakWearMax. By setting this input to zero (0), the detection of brake wear alarm is deactivated. Range: 0...3.4E+38 revolutions

Input	Data Type	Description
i_wSpdCalcToler	WORD	Tolerance of calculated speed. Refer to detailed description (<i>see page 518</i>) of i_wSpdCalcToler. Range: 1.0...10.0 %
i_xRst	BOOL	Resets detected alarms. The reset can also be performed by a warm restart of the controller (power cycle). TRUE: Active. FALSE: Inactive.

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of LoadOverspeedCtrl_2 are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wMotSpd	WORD	0
q_tOvspReactTime	TIME	0 ms
q_xAlrmOvsp	BOOL	FALSE
q_xAlrmBkwr	BOOL	FALSE
q_xAlrmSenFbck	BOOL	FALSE
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

i_wSpdOvsp

The i_wSpdOvsp is the threshold speed above which the overspeed alarm is raised. The TRUE value must be higher than the maximum allowed speed. The difference should be at least a value defined by i_wSpdCalcToler (percent of i_wSpdOvsp).

i_rRevoBrakWearMax

This is the user-defined value of rotations of the motor. The hoist is not allowed to move while the drive is not running before the brake wear alarm is raised. If the drive is not in RUN state, but the movement is still registered by the sensor wheel and the number of rotations of the motor shaft is greater than `i_rRevoBrakWearMax`, the function indicates that the mechanical brakes are slipping or inoperative.

i_wSpdCalcToler

This input defines the requested accuracy of motor speed measurement. The value can be set between 1 and 10, meaning 1...10% of tolerance. Higher tolerance will result in lower accuracy of the measurement, but faster reaction time. Tolerance around 5% is considered as a compromise between accuracy of the measurement and reaction time. The estimated reaction time is available at the output `q_tOvspReactTime`.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: FB enabled. FALSE: FB disabled.
q_wMotSpd	WORD	This output gives information about the motor speed based on the received pulses from the sensor. Range: 0...6000 RPM
q_tOvspReactTime	TIME	Load overspeed reaction time. Refer to detailed description (<i>see page 519</i>) of q_tOvspReactTime. Scaling/Unit: ms
q_xOvrSpdAlrm	BOOL	Overspeed alarm detection indicator TRUE: Alarm detected. FALSE: No alarm detected.
q_xBrakWearAlrm	BOOL	Brake wear alarm detection indicator TRUE: Alarm detected. FALSE: No alarm detected.
q_xSenFbckAlrm	BOOL	Sensor feedback alarm detection indicator TRUE: Alarm detected. FALSE: No alarm detected.
q_xAlrm	BOOL	Alarm detection indicator TRUE: Alarm detected. FALSE: No alarm detected.
q_wAlrmId	WORD	Detected Alarm identification. Refer to Notifications (<i>see page 520</i>). Range: 0...2

q_tOvspReactTime

This output gives an estimated worst case reaction time of overspeed detection.

The value depends on

- combination the overspeed threshold value,
- gear ratio given by the nominator and denominator,
- number of pulses on the cogwheel,
- configured tolerance for speed measurement and
- execution period of the function block.

Notifications

Bit Number	Description
0	Number of pulses per revolution ($i_iPulsPerRevo \leq 0$)
1	Numerator ratio factor ($i_rScalNum \leq 0.0$)
2	Denominator ratio factor ($i_rScalDenom \leq 0.0$)

Section 17.5

Quick Reference Guide

What Is in This Section?

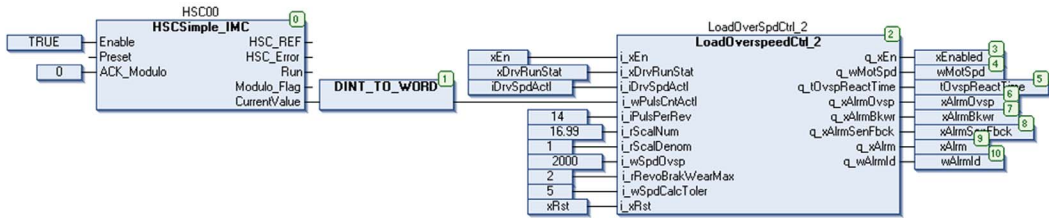
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	522
Troubleshooting	523

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example how to combine the LoadOverspeedCtrl_2 function block with the high speed counter input of the controller.



Troubleshooting

Troubleshooting

Issue	Cause	Solution
Brake wear alarm	Brake wear alarm generated	Verify the correct functioning of the motor brake. If the brake is functioning correctly, verify the parameters of the brake logic control of the drive.
Sensor feedback alarm	Wire breakage in the pulse sensor input.	Verify wire connection. Verify if sensor maintenance is required.
By commissioning of the function block the motor speed is not equal to the actual speed of the drive.	The inputs are not configured correctly.	Verify the value of numerator and denominator factors and encoder increments.

Part XI

Monitoring Data Storage

Chapter 18

Monitoring Data Storage Function: Collect and Monitor the Operational Data of Cranes

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
18.1	Functional and Machine Overview	528
18.2	Architecture	534
18.3	Function Block Description - StatisticDataStorage	535
18.4	Function Block Description - StatisticDataStorage_2	544
18.5	Function Block Description - AlarmDataStorage	549
18.6	Function Block Description - AlarmDataStorage_2	553
18.7	Function Block Description - OvldAlarmDataStorage	557
18.8	Function Block Description - OvtgAlarmDataStorage	561
18.9	Function Block Description - OvspAlarmDataStorage	565
18.10	Function Block Description - LdslAlarmDataStorage	569
18.11	Function Block Description - EncAlarmDataStorage	573
18.12	Function Block Description - MaintenanceDataStorage	577
18.13	Function Block Description - MaintenanceDataStorage_2	587
18.14	Function Block Description - PasswordDataStorage	597
18.15	Quick Reference Guide	601

Section 18.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	529
Machine Overview	533

Functional Overview

Functional Description

All cranes are constructed based on a theoretical lifetime. This lifetime is calculated using the nominal load, nominal use and a theoretical usage concentration. Once this lifetime is up, the crane is recommended to be refurbished and critical components have to be replaced.

With an accurate record of all movements and loads, the function block assists the user in choosing the right time for maintenance. In the case of many movements with a high load, it is possible to plan maintenance work before a component breaks down and eliminate long down times. When loading is occasional it is possible to postpone maintenance, leading to a longer lifecycle and effective maintenance costs.

NOTE: The function block retains the recorded data despite shut down or power cut to the controller. For a proper retention of values, a working battery in the controller is required for those platforms which have battery backed memory.

Monitoring data storage is a function for collecting and monitoring the crane's operational data to improve handling efficiency:

- It allows monitoring of the machine usage under normal and abnormal conditions. All movements of the crane are monitored.
- Usage rates are tallied and an alarm is raised when the system or individual components require maintenance.
- Abnormal conditions such as detected alarms are stored individually.

The function is applicable to the following types of cranes:

- Industrial cranes (trolley, bridge and hoisting movement)
- Construction cranes (trolley, slewing and hoisting movement)

The Monitoring data storage consist of set of a function blocks for SoMachine controller which is dedicated to collect operational and detected alarm data from the crane. Following are the function blocks of Monitoring data storage:

- `StatisticDataStorage`
- `StatisticDataStorage_2`
- `AlarmDataStorage`
- `AlarmDataStorage_2`
- `MaintenanceDataStorage`
- `MaintenanceDataStorage_2`

These function blocks are using an internal function to get the system cycle time for recording the events.

NOTE: The function blocks `StatisticDataStorage`, `AlarmDataStorage` and `MaintenanceDataStorage` have become obsolete, but are retained to maintain compatibility for older applications. We recommend using the following new function blocks in your projects: `StatisticDataStorage_2`, `AlarmDataStorage_2` and `MaintenanceDataStorage_2`. The new function blocks are not pin-compatible with the obsolete function blocks.

StatisticDataStorage Function Block

The `StatisticDataStorage` function block analyzes the crane movements (hoist, slewing, traveling and trolley) and records the necessary information including:

- Operation hours of the hoist, the trolley, slewing and traveling
- Number of the hoist, trolley, slewing and traveling operations
- Number of backtracking operations of the hoist, trolley, slewing and traveling
- Number of pulsating operations of the hoist, trolley slewing and traveling

Backtracking: If different direction movement commands (forward and reverse) are given within a two seconds period, a backtracking event is recorded and a count is incremented by one. If both forward and reverse movement commands are received simultaneously, no backtracking operation is recorded.

Pulsating: If the same direction movement command (forward or reverse) is given within a period of two seconds, a pulsating event is recorded and a count is incremented by one.

StatisticDataStorage_2 Function Block

The `StatisticDataStorage_2` function block is an evolution of the `StatisticDataStorage` function block. For detailed description refer to the function block description ([see page 545](#)).

AlarmDataStorage Function Block

The `AlarmDataStorage` function block (also included in the Library as sub-functions in Overload Data Storage, Overspeed Data Storage, Overtorque Data Storage, Load Slipping Data Storage and Encoder Alarm Data Storage) records the alarms detected in the system. Alarm data storage is involved in the following functions:

- Number of alarm events detected
- Duration of the detected alarm event
- History of the detected alarm event, (time and date of occurrence and duration of the detected alarm)
- The number of hours since the last detected alarm event

AlarmDataStorage_2 Function Block

The `AlarmDataStorage_2` function block is an evolution of the `AlarmDataStorage` function block. For detailed description refer to the function block description ([see page 554](#)).

MaintenanceDataStorage Function Block

The `MaintenanceDataStorage` function block is used to record statistical information about the hoist drive and calculates the remaining hours before maintenance, including:

- Hours spent by the hoist drive working with the actual load
- Remaining hours that the hoist drive can work with the actual load

- Hours 300 class of hoist (amount of hours with more than 300 operations per hour)
- Hours 600 class of hoist (amount of hours with more than 600 operations per hour)

MaintenanceDataStorage_2 Function Block

The MaintenanceDataStorage_2 function block is an evolution of the Maintenance-DataStorage function block. For detailed description refer to the function block description (*see page 588*).

PasswordDataStorage Function Block

The PasswordDataStorage function block validates different passwords and indicates a valid password at the output. The block is usually used in combination with the other blocks and protects the reset input of the other blocks against unauthorized reset.

Why Use the Monitoring Data Storage Function?

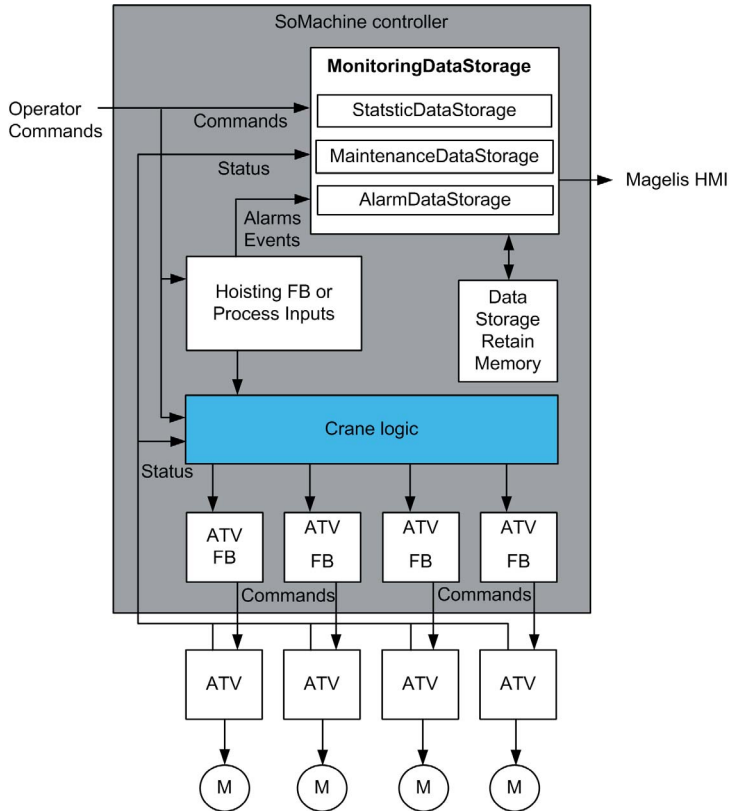
The Monitoring data storage function is essential for the following reasons:

- Collection and monitoring of operation data
- Calculation of effective maintenance intervals
- Inspections and revisions of the cranes are necessary according to applicable safety
- Increased demand for periodic and preventive maintenance

Solution with the Monitoring Data Storage Function

The function uses internal real time clock of the SoMachine controller to time stamp the events such as alerts or detected alarms.

Functional View



Machine Overview

Machine View

The Monitoring data storage function is a software function and although hardware independent, it requires the monitored hardware that provides the information to be logged and collected.

Section 18.2

Architecture

Hardware Architecture

Hardware Architecture Overview

Monitoring data storage is a hardware independent function. Only, the SoMachine controller as the calculation and storage controller is needed. All inputs of the function are event controlled.

Section 18.3

Function Block Description - `StatisticDataStorage`

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `StatisticDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

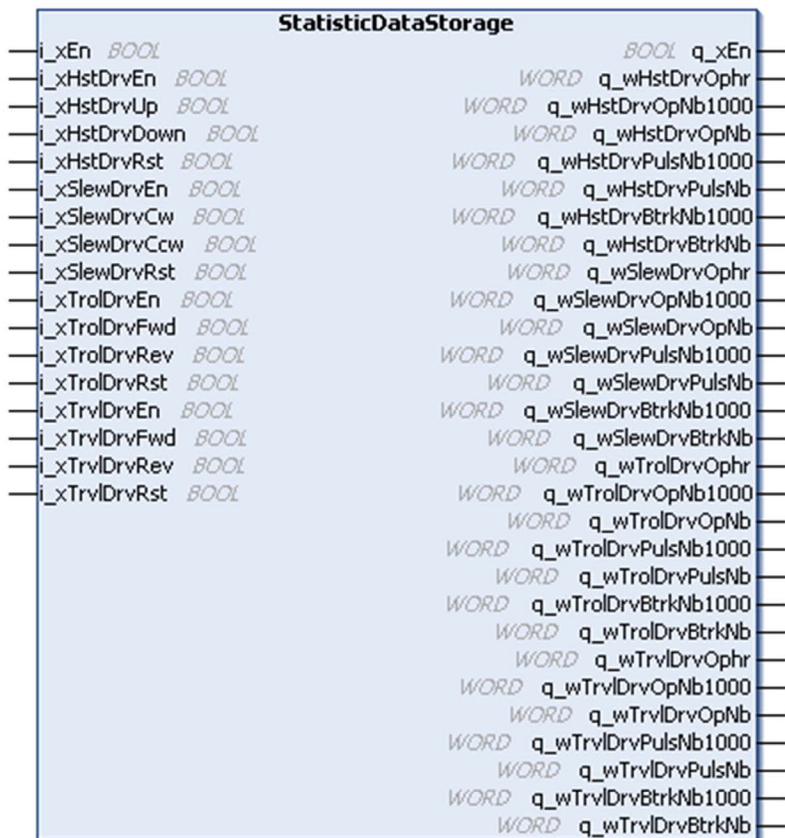
What Is in This Section?

This section contains the following topics:

Topic	Page
<code>StatisticDataStorage</code> Function Block	536
Pin Description - <code>StatisticDataStorage</code>	539

StatisticDataStorage Function Block

Pin Diagram



Function Block Description

The StatisticDataStorage function block is composed of 4 different functions, one for each different movement of the crane. This block keeps a track of all movements as well as all backtracking and pulsating operations for hoist, slew, trolley and traveling movements. The StatisticDataStorage function block records the following movements:

- **Hoist movement**

This block records the number of up or down movements, the number of hoist operational hours in Up or Down movements and the number of backtracking and pulsating operations carried out during the hoisting operations.

NOTE: The reset of the hoisting operations (up, down movements and the number of backtracking and pulsating) can only be carried out when the `i_xHstDrvUp`, `i_xHstDrvDown` is FALSE and the `i_xHstDrvEn` enabled.

- **Slew movement**

This block records the number of Right or Left movements, the number of slew operational hours during Right or Left movements and the number of backtracking and pulsating operations carried out during the slewing operations.

NOTE: The reset of the slewing operations (Right, Left movements and the number of backtracking and pulsating) can only be carried out when the `i_xSlewDrvCw`, `i_xSlewDrvCcw` is FALSE and the `i_xSlewDrvEn` enabled.

- **Trolley movement**

This block records the number of Forward or Reverse movements, the number of trolley operational hours during Forward or Reverse movements and the number of backtracking and pulsating operations carried out during the trolley movement.

NOTE: The reset of the trolley operations (Forward, Reverse movements and the number of backtracking and pulsating) can only be carried out when the `i_xTrolDrvFwd`, `i_xTrolDrvRev` is FALSE and the `i_xTrolDrvEn` enabled.

- **Traveling movement**

This block records the number of Traveling Forward or Traveling Reverse movements, the number of traveling operational hours during Traveling Forward or Traveling Reverse movements, and the number of backtracking and pulsating operations carried out during the traveling movement.

NOTE: All outputs of this block are retained after each expected (switch Off) and unexpected power shutdown as long as there is sufficient battery power available.

NOTE: The reset of the traveling operations (Forward, Reverse movements and the number of backtracking and pulsating) can only be carried out when the `i_xTrvlDrvFwd`, `i_xTrvlDrvRev` is FALSE and the `i_xTrvlDrvEn` enabled.

Controller	Backup Autonomy	Comments
Modicon M238 Logic controller	Internal battery = 3 days Optional battery = 1 year	The charging time of the internal battery is 22 hours for a full charge. The service life of the internal battery is maximum 10 years (derating depending on the operating temperature).

Controller	Backup Autonomy	Comments
Altivar ATV-IMC Drive controller	Product life cycle as the data is saved in the FRAM (Ferroelectric Random Access Memory)	-

Pin Description - StatisticDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer also to detailed description (<i>see page 540</i>) of i_xEn.
i_xHstDrvEn	BOOL	TRUE: Enables for hoist movement recording. FALSE: Disables for hoist movement recording.
i_xHstDrvUp	BOOL	Up hoist movement command. TRUE: Up move FALSE: No action
i_xHstDrvDown	BOOL	Down hoist movement command. TRUE: Down move FALSE: No action
i_xHstDrvRst	BOOL	Reset for hoist movement, resets stored data. TRUE: Reset hoist FALSE: No action
i_xSlewDrvEn	BOOL	TRUE: Enables the slew movement recording. FALSE: Disables the slew movement recording.
i_xSlewDrvCw	BOOL	Clockwise slew movement command. TRUE: Clockwise move FALSE: No action
i_xSlewDrvCcw	BOOL	Counter clockwise slew movement command. TRUE: Counter clockwise move FALSE: No action
i_xSlewDrvRst	BOOL	Reset for slew movement, resets stored data. TRUE: Reset slew FALSE: No action
i_xTrolDrvEn	BOOL	TRUE: Enables for trolley movement recording. FALSE: Disables for trolley movement recording.
i_xTrolDrvFwd	BOOL	Forward trolley movement command. TRUE: Forward move FALSE: No action

Input	Data Type	Description
i_xTrolDrvRev	BOOL	Reverse trolley movement command. TRUE: Reverse move FALSE: No action
i_xTrolDrvRst	BOOL	Reset function for trolley movement, resets stored data. TRUE: Reset trolley FALSE: No action
i_xTrvlDrvEn	BOOL	TRUE: Enables the input for traveling movement recording. FALSE: Disables the input for traveling movement recording.
i_xTrvlDrvFwd	BOOL	Forward traveling movement command. TRUE: Forward move FALSE: No action
i_xTrvlDrvRev	BOOL	Reverse traveling movement command. TRUE: Reverse move FALSE: No action
i_xTrvlDrvRst	BOOL	Reset function for traveling movement, resets stored data. TRUE: Reset traveling movement data FALSE: No action

i_xEn

By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `StatisticDataStorage` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE

Outputs that are not listed will retain their current values.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.

Output	Data Type	Description
q_wHstDrvOphr	WORD	Number of hours with hoist up and down movement activated. Range: 0...65535 h
q_wHstDrvOpNb1000	WORD	Number of hoist operations of up and down movement. Range: 0...65535
q_wHstDrvOpNb	WORD	Number of hoist operations of up and down movement. Range: 0...999
q_wHstDrvPulsNb1000	WORD	Number of pulsating operations associated with hoist up and down movement. Range: 0...65535
q_wHstDrvPulsNb	WORD	Number of pulsating operations associated with hoist up and down movement. Range: 0...999
q_wHstDrvBtrkNb1000	WORD	Number of backtracking operations associated with hoist up and down movement. Range: 0...65535
q_wHstDrvBtrkNb	WORD	Number of backtracking operations associated with hoist up and down movement. Range: 0...999
q_wSlewDrvOphr	WORD	Number of hours with slew (clockwise/counter clockwise) movement activated. Range: 0...65535 h
q_wSlewDrvOpNb1000	WORD	Number of slew operations of (clockwise/counter clockwise) movement. Range: 0...65535
q_wSlewDrvOpNb	WORD	Number of slew operations of (clockwise/counter clockwise) movement. Range: 0...999
q_wSlewDrvPulsNb1000	WORD	Number of pulsating operations associated with slew (clockwise/counter clockwise) movement. Range: 0...65535
q_wSlewDrvPulsNb	WORD	Number of pulsating operations associated with slew (clockwise/counter clockwise) movement. Range: 0...999

Output	Data Type	Description
q_wSlewDrvBtrkNb1000	WORD	Number of backtracking operations associated with slew (clockwise/counter clockwise) movement. Range: 0...65535
q_wSlewDrvBtrkNb	WORD	Number of backtracking operations associated with slew (clockwise/counter clockwise) movement. Range: 0...999
q_wTrolDrvOphr	WORD	Number of hours with trolley (forward/reverse) movement. Range: 0...65535 h
q_wTrolDrvOpNb1000	WORD	Number of trolley operations of (forward/reverse) movement. Range: 0...65535
q_wTrolDrvOpNb	WORD	Number of trolley operations of (forward/reverse) movement. Range: 0...999
q_wTrolDrvPulsNb1000	WORD	Number of pulsating operations associated with trolley (forward/reverse) movement. Range: 0...65535
q_wTrolDrvPulsNb	WORD	Number of pulsating operations associated with trolley (forward/reverse) movement. Range: 0...999
q_wTrolDrvBtrkNb1000	WORD	Number of backtracking operations associated with trolley (forward/reverse) movement. Range: 0...65535
q_wTrolDrvBtrkNb	WORD	Number of backtracking operations associated with trolley (forward/reverse) movement. Range: 0...999
q_wTrvlDrvOphr	WORD	Number of hours with traveling (forward/reverse) movement. Range: 0...65535 h
q_wTrvlDrvOpNb1000	WORD	Number of traveling operations of (forward/reverse) movement. Range: 0...65535
q_wTrvlDrvOpNb	WORD	Number of traveling operations of (forward/reverse) movement. Range: 0...999
q_wTrvlDrvPulsNb1000	WORD	Number of pulsating operations associated with traveling (forward/reverse) movement. Range: 0...65535

Output	Data Type	Description
q_wTrvlDrvPulsNb	WORD	Number of pulsating operations associated with traveling (forward/reverse) movement. Range: 0...999
q_wTrvlDrvBtrkNb1000	WORD	Number of backtracking operations associated with slew (clockwise/counter clockwise) movement. Range: 0...65535
q_wTrvlDrvBtrkNb	WORD	Number of backtracking operations associated with slew (clockwise/counter clockwise) movement. Range: 0...999

Section 18.4

Function Block Description - `StatisticDataStorage_2`

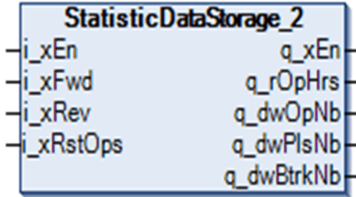
What Is in This Section?

This section contains the following topics:

Topic	Page
<code>StatisticDataStorage_2</code> Function Block	545
Pin Description - <code>StatisticDataStorage_2</code>	546

StatisticDataStorage_2 Function Block

Pin Diagram



Function Block Description

The `StatisticDataStorage_2` function block is an evolution of `StatisticDataStorage` function block. It has a simpler interface. The `StatisticDataStorage` function block contains inputs for 4 pre-defined axes while the `StatisticDataStorage_2` function block provides the same function for one axis and can be instantiated as many times as required. This block keeps a track of all movements as well as all backtracking and pulsating operations for arbitrary axis.

Pin Description - *StatisticDataStorage_2*

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer also to detailed description (<i>see page 546</i>) of i_xEn.
i_xFwd	BOOL	Forward command from the hoist axis. NOTE: It is not possible to run forward (i_xFwd) and reverse (i_xRev) at the same time. The FB accepts the first command in time if both inputs are TRUE. TRUE: Forward FALSE: Not forward
i_xRev	BOOL	Reverse command from the hoist axis. NOTE: It is not possible to run forward (i_xFwd) and reverse (i_xRev) at the same time. The FB accepts the first command in time if both inputs are TRUE. TRUE: Reverse FALSE: Not reverse
i_xRstOps	BOOL	Reset input to clear the following retained values on a rising edge: <ul style="list-style-type: none"> ● q_rOpHrs ● q_dwOpNb ● q_dwPlsNb ● q_dwBtrkNb TRUE: Active FALSE: Inactive

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `StatisticDataStorage_2` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_rOpHrs</code>	REAL	not altered - retained value
<code>q_dwOpNb</code>	DWORD	not altered - retained value
<code>q_dwPlsNb</code>	DWORD	not altered - retained value
<code>q_dwBtrkNb</code>	DWORD	not altered - retained value

Output Pin Description

Output	Data Type	Description
<code>q_xEn</code>	BOOL	Displays the status of the enable parameter input <code>i_xEn</code> . If the output stays FALSE although <code>i_xEn</code> is TRUE, an SoMachine controller platform cannot be detected. TRUE: Function block enabled. FALSE: Function block disabled.
<code>q_rOpHrs</code>	REAL	Displays the actual value of operating hours of the monitored axis. Scaling/Unit: 1 hour
<code>q_dwOpNb</code>	DWORD	Displays the actual number of operations of the monitored axis. Each forward and each reverse command is increasing the number by one.. Scaling/Unit: 1 operation
<code>q_dwPlsNb</code>	DWORD	Displays the actual number of pulsating incidents of the monitored axis. Scaling/Unit: 1 pulse Refer to detailed description below this table.
<code>q_dwBtrkNb</code>	DWORD	Displays the actual number of backtracking incidents of the monitored axis. Scaling/Unit: 1 operation Refer to detailed description below this table.

`q_dwPlsNb`

A pulsating incident is defined as followed:

- A falling edge on a forward command followed by a rising edge of a forward command within 2 seconds.
- A falling edge on a reverse command followed by a rising edge of a reverse command within 2 seconds.

q_dwBtrkNb

A backtracking incident is defined as followed:

- A falling edge on a forward command followed by a rising edge of a reverse command within 2 seconds.
- A falling edge on a reverse command followed by a rising edge of a forward command within 2 seconds.

Section 18.5

Function Block Description - AlarmDataStorage

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
AlarmDataStorage Function Block	550
Pin Description - AlarmDataStorage	551

AlarmDataStorage Function Block

Pin Diagram



Function Block Description

The AlarmDataStorage function block records a specific detected alarm (for example, Overload, Load Overspeed, and so forth). For the specific type of alarm event, the function block records the following data:

- The number of alarm events detected.
- The number of hours since the last detected alarm event occurred.
- Duration of the detected alarm event.
- Maintains a historical record (YYYY MMDD HHMMSS format) of the alarm event detected in the system.

This data is retained after both normal and unexpected power down.

NOTE: The AlarmDataStorage function block is a supporting function block for Overload Alarm Data Storage, Overtorque Alarm Data Storage, Overspeed Alarm Data Storage, Load Slipping Alarm Data Storage, and Encoder Alarm Data Storage blocks.

NOTE: The reset of the alarm history can be only performed in the absence of an alarm (i_xDevAlrm is TRUE).

Pin Description - AlarmDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
i_xDevAlrm	BOOL	Alarm event input. External signal or internal bit from another function block. TRUE: Alarm detected FALSE: No alarm detected
i_wAlrmNb	WORD	Number of historic alarms Range: 0...20
i_xRst	BOOL	Resets the function block outputs. TRUE: Resets function block FALSE: No action

NOTE: When the value entered at `i_wAlrmNb` < 1, the alarm value is 1.

i_xEn

By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `AlarmDataStorage` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE

Outputs that are not listed will retain their current values.

Output Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_wAlrmTohr	WORD	Total hours since the last detected alarm event. Range: 0...65535 h

Output	Data Type	Description
q_dtAlrm	DT	Details of the detected alarm event number i_wAlrmNb. Format: YYYY MMDD HHMMSS
q_wAlrmDur	WORD	Duration of the detected alarm event number i_wAlrmNb. Range: 0...65535 s
q_xAlrm	BOOL	This NC output is an alarm indicating a detected alarm event. TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmNb	WORD	Number of alarm events detected in the system. Range: 0...65535

Section 18.6

Function Block Description - AlarmDataStorage_2

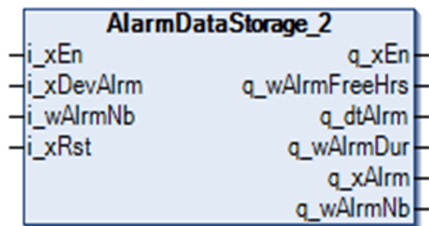
What Is in This Section?

This section contains the following topics:

Topic	Page
AlarmDataStorage_2 Function Block	554
Pin Description - AlarmDataStorage_2	555

AlarmDataStorage_2 Function Block

Pin Diagram



Function Block Description

The AlarmDataStorage_2 function block is an evolution of the AlarmDataStorage function block. It can be used to record an arbitrary alarm.

The function block records the following data:

- The number of alarm events detected.
- The number of hours since the last detected alarm event occurred.
- Duration of the detected alarm event.
- Maintains a historical record (YYYYMMDD HHMMSS format) of the alarm event detected in the system.

This data is retained after both normal and unexpected power down.

Pin Description - AlarmDataStorage_2

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (<i>see page 555</i>) below this table.
i_xDevAlrm	BOOL	Alarm event input monitored by the FB. On a rising edge the monitoring is started until a falling edge is detected. TRUE: Recording active FALSE: Recording inactive
i_wAlrmNb	WORD	Number of incident to be displayed at the output. Range: 0...20 Refer to detailed description (<i>see page 556</i>) below this table.
i_xRst	BOOL	Resets all retained values. A rising edge on this input resets the following retained values: <ul style="list-style-type: none"> • q_wAlrmNb is set to zero • q_dtAlrm [1...20] will be set to 1977-07-07-07:07:07 • q_wAlrmDur [1...20] will be set to zero TRUE: Resets function block FALSE: No action

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of AlarmDataStorage_2 are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wAlrmFreeHrs	WORD	0
q_dtAlrm	DT	1977-07-07-07:07:07
q_wAlrmDur	WORD	0
q_xAlrm	BOOL	FALSE
q_wAlrmNb	WORD	not altered - retained value

i_wAlrmNb

The input is the selector for the alarm event to be displayed at the corresponding outputs `q_dtAlrm` and `q_wAlrmDur`. The FB is able to store 20 alarm incidents with time stamp and duration. The most recent alarm is always stored in position [1] and therefore the oldest recorded event at position [20].

Output Description

Output	Data Type	Description
<code>q_xEn</code>	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
<code>q_wAlrmFreeHrs</code>	WORD	Total hours since the last detected alarm event. Range: 0...65535 Scaling/Unit: 1 hour
<code>q_dtAlrm</code>	DT	Displays the date and start time of the alarm event, which was selected by the number entered at <code>i_wAlrmNb</code> out of register of the last 20 alarms. Range: 1970...21xx Scaling/Unit: YYYY MMDD HHMMSS
<code>q_wAlrmDur</code>	WORD	Displays the duration of the detected alarm event, which was selected by the number entered at <code>i_wAlrmNb</code> out of the register of the last 20 alarms. Range: 0...65535 Scaling/Unit: 1 second
<code>q_xAlrm</code>	BOOL	Displays the status of the alarm input <code>i_xDevAlrm</code> . If <code>i_xDevAlrm</code> is TRUE, <code>q_xAlrm</code> gets TRUE and vice versa. TRUE: Alarm detected FALSE: No alarm detected
<code>q_wAlrmNb</code>	WORD	Displays the total number of alarm events detected since the last reset or initial start. Range: 0...65535 Scaling/Unit: 1 alarm

Section 18.7

Function Block Description - OvldAlarmDataStorage

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
OvldAlarmDataStorage Function Block	558
Pin Description - OvldAlarmDataStorage	559

OvldAlarmDataStorage Function Block

Pin Diagram



Function Block Description

The following blocks are used for specific detected alarms. They are all based on the AlarmDataStorage function block and have the same behavior.

NOTE: For the description of the input pins *i_xEn*, *i_wAlrmNb*, and *i_xRst* of Overload Alarm Data Storage, Overtorque Alarm Data Storage, Overspeed Alarm Data Storage, Load Slipping Alarm Data Storage, and Encoder Alarm Data Storage blocks, refer to the input pin description of the AlarmDataStorage function block.

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Pin Description - OvldAlarmDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
i_xOvld	BOOL	Overload input. External signal or internal bit from crane logic. TRUE: Alarm detected FALSE: No alarm detected
i_wAlrmNb	WORD	Number of historic alarms Range: 1...20
i_xRst	BOOL	Resets the function block outputs. TRUE: Resets function block FALSE: No action

NOTE: When the value entered at `i_wAlrmNb` < 1, the alarm value is 1.

i_xEn

By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `AlarmDataStorage` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE

Outputs that are not listed will retain their current values.

The fallback states of `AlarmDataStorage_2` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wAlrmFreeHrs	WORD	0
q_dtAlrm	DT	1977-07-07-07:07:07
q_wAlrmDur	WORD	0
q_xAlrm	BOOL	FALSE

Output	Data Type	Fallback State
q_wAlrmNb	WORD	not altered - retained value

NOTE: The function block `AlarmDataStorage` has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2` in your project. However, the new function block is not pin-compatible with the obsolete function block.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_wOvldAlrmNb	WORD	Number of overload events detected in the system. Range: 0..65535
q_wOvldAlrmTohr	WORD	Total hours since the last overload event. Range: 0..65535 h
q_dtOvldAlrm	DT	Details of the overload event number <code>i_wAlrmNb</code> Range: 1970-00-00-00:00:00... 2106-02-06-06:28:15 Format: YYYY MMDD HHMMSS
q_wOvldAlrmDur	WORD	Duration of the overload event number <code>i_wAlrmNb</code> Range: 0..65535 s
q_xOvldAlrm	BOOL	TRUE: Alarm detected FALSE: No alarm detected

Section 18.8

Function Block Description - OvtqAlarmDataStorage

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
OvtqAlarmDataStorage Function Block	562
Pin Description - OvtqAlarmDataStorage	563

OvtqAlarmDataStorage Function Block

Pin Diagram



Function Block Description

OvtqAlarmDataStorage function block records a detected number of over torque events. For over torque events, the function block records the following data:

- Number of over torque events.
- Number hours since the last detected over torque events.
- Time duration of over torque events.
- Details of over torque events.

NOTE: The historical records (YYYY MMDD HHMMSS format) of over torque events are maintaining in AlarmDataStorage function block. These data are retained after both normal and unexpected power down.

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Pin Description - `ovtqAlarmDataStorage`

Input Pin Description

Input	Data Type	Description
<code>i_xEn</code>	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
<code>i_xOvtq</code>	BOOL	Over torque input. External signal or internal bit from crane logic. TRUE: Alarm detected FALSE: No alarm detected
<code>i_wAlrmNb</code>	WORD	Number of historic alarms Range: 1...20
<code>i_xRst</code>	BOOL	Resets the function block outputs. TRUE: Resets function block FALSE: No action

NOTE: When the value entered at `i_wAlrmNb` < 1, the alarm value is 1.

`i_xEn`

By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `AlarmDataStorage` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE

Outputs that are not listed will retain their current values.

The fallback states of `AlarmDataStorage_2` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_wAlrmFreeHrs</code>	WORD	0
<code>q_dtAlrm</code>	DT	1977-07-07-07:07:07
<code>q_wAlrmDur</code>	WORD	0
<code>q_xAlrm</code>	BOOL	FALSE

Output	Data Type	Fallback State
q_wAlrmNb	WORD	not altered - retained value

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_wOvtqAlrmNb	WORD	Number of over torque events detected in the system. Range: 0...65535
q_wOvtqAlrmTohr	WORD	Total hours since the last over torque event. Range: 0...65535 h
q_dtOvtqAlrm	DT	Details of the over torque event number i_wAlrmNb. Range: 1970-00-00-00:00:00...2106-02-06-06:28:15 Format: YYYY MMDD HHMMSS
q_wOvtqAlrmDur	WORD	Duration of the over torque event number i_wAlrmNb Range: 0...65535 s
q_xOvtqAlrm	BOOL	TRUE: Alarm detected FALSE: No alarm detected

Section 18.9

Function Block Description - OvspAlarmDataStorage

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
OvspAlarmDataStorage Function Block	566
Pin Description - OvspAlarmDataStorage	567

OvspAlarmDataStorage Function Block

Pin Diagram



Function Block Description

OvspAlarmDataStorage function block records a detected number of over speed events. For over speed events, the function block records the following data:

- Number of over speed events.
- Number hours since the last detected over speed events.
- Time duration of over speed events.
- Details of over speed events.

NOTE: The historical records (YYYY MMDD HHMMSS format) of over speed events are maintaining in AlarmDataStorage function block. These data are retained after both normal and unexpected power down.

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Pin Description - OvspAlarmDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
i_xOvsp	BOOL	Over speed input. External signal or internal bit from crane logic. TRUE: Alarm detected FALSE: No alarm detected
i_wAlrmNb	WORD	Number of historic alarms Range: 1...20
i_xRst	BOOL	Resets the function block outputs. TRUE: Resets function block FALSE: No action

NOTE: When the value entered at i_wAlrmNb < 1, the alarm value is 1.

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of AlarmDataStorage are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE

Outputs that are not listed will retain their current values.

The fallback states of AlarmDataStorage_2 are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wAlrmFreeHrs	WORD	0
q_dtAlrm	DT	1977-07-07-07:07:07
q_wAlrmDur	WORD	0
q_xAlrm	BOOL	FALSE

Output	Data Type	Fallback State
q_wAlrmNb	WORD	not altered - retained value

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_wOvspAlrmNb	WORD	Number of over speed events detected in the system. Range: 0...65535
q_wOvspAlrmTohr	WORD	Total hours since the last over speed event. Range: 0...65535 h
q_dtOvspAlrm	DT	Details of the over speed event number i_wAlrmNb. Format: YYYY MMDD HHMMSS
q_wOvspAlrmDur	WORD	Duration of the detected alarm event number i_wAlrmNb. Range: 0...65535 s
q_xOvspAlrm	BOOL	The NC alarm indicates a detected over speed event. TRUE: Alarm detected FALSE: No alarm detected

Section 18.10

Function Block Description - LdslAlarmDataStorage

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
LdslAlarmDataStorage Function Block	570
Pin Description - LdslAlarmDataStorage	571

LdslAlarmDataStorage Function Block

Pin Diagram



Function Block Description

LdslAlarmDataStorage function block records a detected number of load slipping events. For load slipping events, the function block records the following data:

- Number of load slipping events.
- Number hours since the last detected slipping events.
- Time duration of load slipping events.
- Details of load slipping events.

NOTE: The historical records (YYYY MMDD HHMMSS format) of load slipping events are maintaining in AlarmDataStorage function block. These data are retained after both normal and unexpected power down.

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Pin Description - Lds1AlarmDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
i_xLdsl	BOOL	Load slipping input. External signal or internal bit from crane logic. TRUE: Alarm detected FALSE: No alarm detected
i_wAlrmNb	WORD	Number of historic alarms Range: 1...20
i_xRst	BOOL	Resets the function block outputs. TRUE: Resets function block FALSE: No action

NOTE: When the value entered at `i_wAlrmNb` < 1, the alarm value is 1.

i_xEn

By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `AlarmDataStorage` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE

Outputs that are not listed will retain their current values.

The fallback states of `AlarmDataStorage_2` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wAlrmFreeHrs	WORD	0
q_dtAlrm	DT	1977-07-07-07:07:07
q_wAlrmDur	WORD	0
q_xAlrm	BOOL	FALSE

Output	Data Type	Fallback State
q_wAlrmNb	WORD	not altered - retained value

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_wLdslAlrmNb	WORD	Number of load slipping events detected in the system. Range: 0...65535
q_wLdslAlrmTohr	WORD	Total hours since the last detected load slipping event. Range: 0...65535 h
q_dtLdslAlrm	DT	Details of the detected load slipping event number i_wAlrmNb. Range: 1970-00-00-00:00:00...2106-02-06-06:28:15 Format: YYYY MMDD HHMMSS
q_wLdslAlrmDur	WORD	Duration of the load slipping event number i_wAlrmNb. Range: 0...65535 s
q_xLdslAlrm	BOOL	The NC alarm indicates a detected load slipping event. TRUE: Alarm detected FALSE: No alarm detected

Section 18.11

Function Block Description - EncAlarmDataStorage

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
EncAlarmDataStorage Function Block	574
Pin Description - EncAlarmDataStorage	575

EncAlarmDataStorage Function Block

Pin Diagram



Function Block Description

EncAlarmDataStorage function block records the number of detected encoder alarm events. For encoder alarm events, the function block records the following data:

- Number of encoder alarm events.
- Number hours since the last detected encoder alarm events.
- Time duration of encoder alarm events.
- Details of encoder events.

NOTE: The historical records (YYYY MMDD HHMMSS format) of encoder alarm events are maintaining in AlarmDataStorage function block. These data are retained after both normal and unexpected power down.

NOTE: The function block AlarmDataStorage has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block AlarmDataStorage_2 in your project. However, the new function block is not pin-compatible with the obsolete function block.

Pin Description - EncAlarmDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
i_xEncAlrm	BOOL	Encoder alarm input. External signal or internal bit from crane logic. TRUE: Alarm detected FALSE: No alarm detected
i_wAlrmNb	WORD	Number of historic alarms Range: 1...20
i_xRst	BOOL	Resets the function block outputs. TRUE: Reset function block FALSE: No action

NOTE: When the value entered at `i_wAlrmNb` < 1, the alarm value is 1.

i_xEn

By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `AlarmDataStorage` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE

Outputs that are not listed will retain their current values.

The fallback states of `AlarmDataStorage_2` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wAlrmFreeHrs	WORD	0
q_dtAlrm	DT	1977-07-07-07:07:07
q_wAlrmDur	WORD	0
q_xAlrm	BOOL	FALSE

Output	Data Type	Fallback State
q_wAlrmNb	WORD	not altered - retained value

NOTE: The function block `AlarmDataStorage` has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `AlarmDataStorage_2` in your project. However, the new function block is not pin-compatible with the obsolete function block.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_wEncAlrmNb	WORD	Number of encoder alarm events detected in the system. Range: 0..65535
q_wEncAlrmTohr	WORD	Total hours since the last detected encoder alarm event. Range: 0..65535 h
q_dtEncAlrm	DT	Details of the detected encoder alarm event number <code>i_wAlrmNb</code> . Range: 1970-00-00-00:00:00...2106-02-06-06:28:15 Format: YYYY MMDD HHMMSS
q_wEncAlrmDur	WORD	Duration of the detected encoder alarm event number <code>i_wAlrmNb</code> . Range: 0..65535 s
q_xEncAlrm	BOOL	The NC output alarm indicates a detected encoder alarm event. TRUE: Alarm detected FALSE: No alarm detected

Section 18.12

Function Block Description - MaintenanceDataStorage

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `MaintenanceDataStorage_2`. The new function block is not pin-compatible with the obsolete function block.

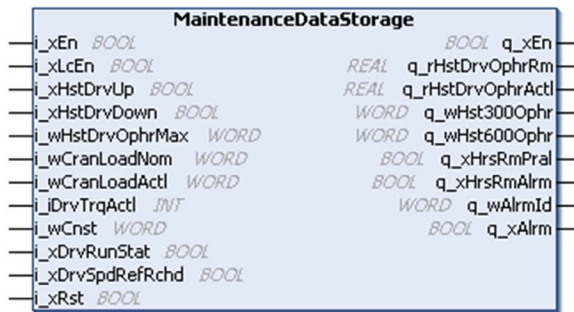
What Is in This Section?

This section contains the following topics:

Topic	Page
MaintenanceDataStorage Function Block	578
Pin Description - MaintenanceDataStorage	582
Commissioning Procedure	585

MaintenanceDataStorage Function Block

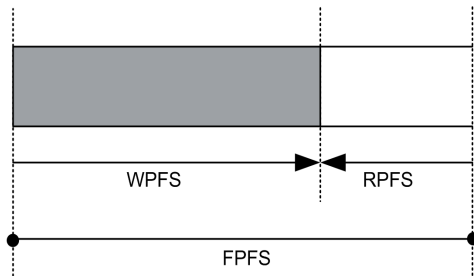
Pin Diagram



Function Block Description

The MaintenanceDataStorage function block is used to supervise all important maintenance functions. This block is used to monitor and provide maintenance information for the Hoist drive only:

PFS	Safety work period hoist main gearbox
FPFS (Factory PFS)	This is the operation time for the hoist main gearbox with nominal load. This information is provided by the manufacturer.
WPFS (Wasted PFS)	This is the proportional number of hours for which the hoist has operated with real load.
RPFS (Remain PFS)	The difference between FPFS and WPFS is the RPFS. RPFS is calculated using the formula: $RPFS = FPFS - WPFS$.



Formula to Calculate WPFS with a Load Cell

In this case, 0...20 mA or 4...20 mA analog output load cell is used to read the value of the real load.

NOTE: The load cell input (0...20 mA / 4...20 mA) is scaled with respect to the nominal input in kg (0 - nominal load) and must be given as an analog input to the SoMachine controller only. WPFS is then calculated using the formula:

$$WPFS = \sum_{i=1}^n T \left(\frac{LACT}{LNOM} \right)^3 \Rightarrow \sum_{i=1}^n T \left(\frac{LdAct}{LdNom} \right)^3$$

Where:

LNOM = Nominal value of the load entered by you.

LACT = Analog output from the load cell (to be scaled using scaling block).

T = Total time since movement started up until movement stops. This time is calculated internally.

LdAct = Actual torque of the drive.

LdNom = Nominal torque of the drive.

Formula to Calculate WPFS without a Load Cell

In this case, the following relationship is used to calculate WPFS:

$$WPFS = \sum_{i=1}^n T (KC)^3 \Rightarrow \sum_{i=1}^n T (Cnst * TrqAct)^3$$

K (i_wCnst) is in % and is the relationship between the nominal crane load and nominal motor torque, usually 100%.

C (i_iDrvTrqAct) is in % and is the nominal torque on the crane for stationary movement (acceleration zero, constant speed). You must configure this input to read the actual torque from the hoist drive.

Example:

$T = 10 \text{ s}, K = 100\%, C = 20\%$

$(KC)^3 = (1 \cdot 0.2)^3 = 0.008$

$T = 10/3600 \text{ h}$

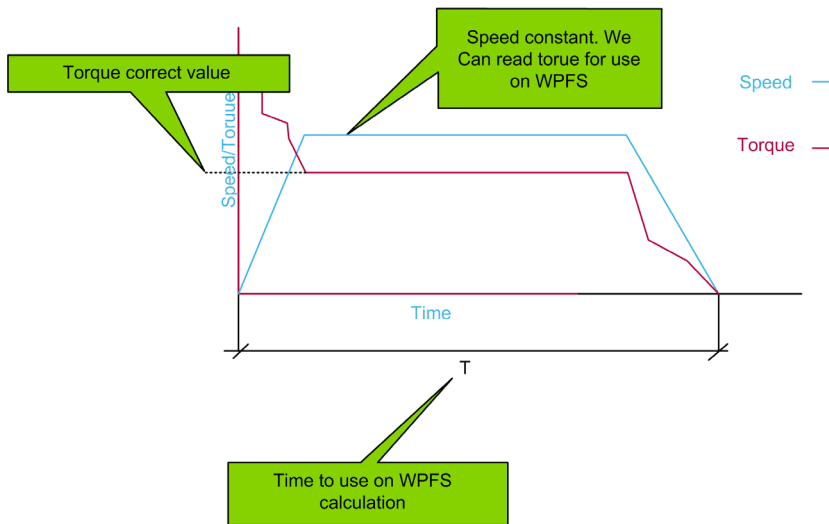
$WPFS = 2.22e-5$

(Since both K and C are in %, while calculating WPFS, both K and C are divided by 100).

T is the total time since movement started up until movement stops. This time is calculated internally.

NOTE: If the actual load is being measured by a load cell, then you must verify that the load cell data acquisition (`i_xLcEn`) input is enabled otherwise this input remains disabled by default.

WPFS Calculation



The figure above explains, whenever a hoist operation (up or down) is carried out, the time T for WPFS calculation is taken into consideration only after a RUN command is given, speed reference is reached and the RUN status becomes FALSE.

NOTE:

WPFS is updated with the following sequence of operations:

1. RUN command is TRUE.
2. Speed reference reached.
3. RUN command is FALSE.

This block provides:

- **WPFS**: The number of hours spent by the hoist drive with the real load.
- **RPFS**: The remaining hours available for the hoist drive to operate the load.
- **H300**: The number of hours of the hoist working with more than 300 hoist movements up or down per hour.
- **H600**: The number of hours of the hoist working with more than 600 hoist movements up or down per hour.

Two alarm outputs:

- Maintenance data pre-alarm is activated, when $RPFS < 0.1(FPFS)$.
- Maintenance data alarm is activated, when $RPFS = 0$ or $FPFS = 0$.
- The above alarms get reset automatically, when their respective conditions are FALSE.

Pin Description - MaintenanceDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (<i>see page 583</i>) of i_xEn.
i_xLcEn	BOOL	Enables use of Load Cell (LC). TRUE: With Load Cell FALSE: Without Load Cell
i_xHstDrvUp	BOOL	Move hoist upwards command. TRUE: Up move FALSE: No action
i_xHstDrvDown	BOOL	Move hoist downwards command. TRUE: Down move FALSE: No action
i_wHstDrvOphrMax	WORD	Factory FPFS hours constant provided by user. Range: 0...65535 h
i_wCranLoadNom	WORD	Nominal crane load. Range: 0...65535 kg
i_wCranLoadAct1	WORD	Scaled value from output of Scaling block. Range: 0...65535 kg
i_iDrvTrqAct1	INT	Actual torque Range: -3000...3000 %
i_wCnst	WORD	Relation between nominal hoist drive torque and nominal load. Range: 0...200 % Factory setting: 100 %
i_xDrvRunStat	BOOL	Status of the drive (drive status word) TRUE: Run FALSE: Stop
i_xDrvSpdRefRchd	BOOL	Input of the drive (drive status word) that the speed reference has been reached. TRUE: Reached FALSE: Stopped NOTE: The exact bit can be found in the <i>Communication Parameters</i> manuals of the respective drives.
i_xRst	BOOL	Resets RPFS, WPFS, q_wHst3000phr and q_wHst6000phr outputs.

i_xEn

By setting the `i_xEn` input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `MaintenanceDataStorage` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_xHrsRmPral</code>	BOOL	FALSE
<code>q_xHrsRmAlrm</code>	BOOL	FALSE
<code>q_xAlrm</code>	BOOL	FALSE
<code>q_wAlrmId</code>	WORD	0

Outputs that are not listed will retain their current values.

Output Pin Description

Output	Data Type	Description
<code>q_xEn</code>	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
<code>q_rHstDrvOphrRm</code>	REAL	Remainder PFS time (RPFS) Range: 0..65535 h
<code>q_rHstDrvOphrAct1</code>	REAL	PFS time proportional hours spent by the hoist main gearbox with real load (WPFS). Range: 0..65535 h
<code>q_wHst300Ophr</code>	WORD	Hours with more than 300 operations/hour. Range: 0..65535 h
<code>q_wHst600Ophr</code>	WORD	Hours with more than 600 operations/hour. Range: 0..65535 h
<code>q_xHrsRmPral</code>	BOOL	Pre-alarm generated when RPFS < 0.1 (FPFS). TRUE: Alarm detected FALSE: No alarm detected
<code>q_xHrsRmAlrm</code>	BOOL	Alarm when RPFS = 0 or FPFS = 0. TRUE: Alarm detected FALSE: No alarm detected

Output	Data Type	Description
q_wAlrmId	WORD	Alarm output to indicate different types of alarm due to improper parameterization of the FB. Range: 0...7 0 No alarm detected. 1 FPFS is 0 2 Load cell calculation enabled (<i>i_xLcEn</i> = 1) but nominal value (<i>i_wCranLoadNom</i> = 0).
q_xAlrm	BOOL	Alarm bit is TRUE when q_wAlrmId is equal to 1 or 2. TRUE: Alarm detected FALSE: No alarm detected

NOTE:

- If over a period of 1 hour, the number of hoist operations carried out exceeds 600, then only the q_wHst600Ophr output will get updated but q_wHst300Ophr will not get updated.
- Similarly, if over a period of 1 hour, the number of hoist is greater than 300 and less than 600, then only q_wHst300Ophr output will get updated but q_wHst600Ophr will not get updated.

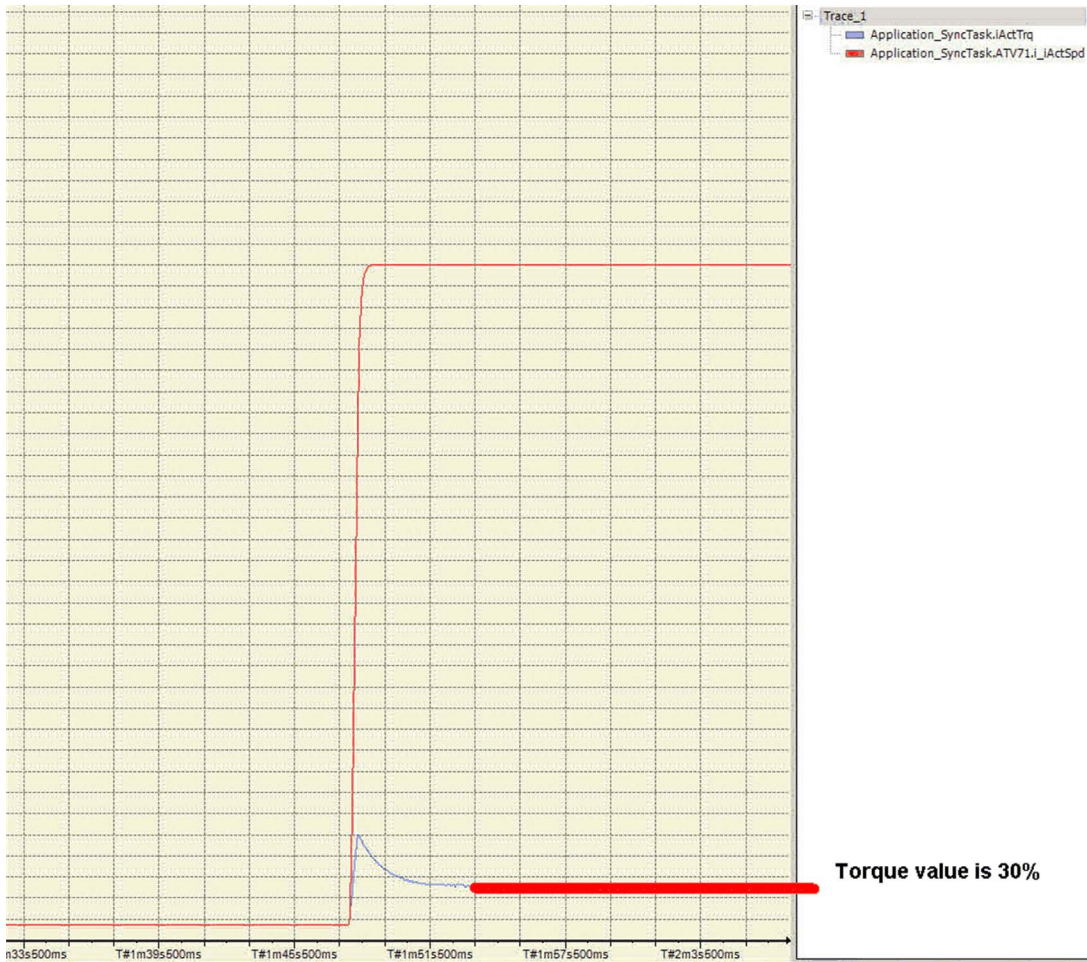
Commissioning Procedure

Commissioning Procedure of the MaintenanceDataStorage Function Block:

The following steps explain the procedure on how to configure the MaintenanceDataStorage function block using the torque of the drive:

1. Insert a trace object into your SoMachine project and map the actual speed and the filtered torque (OTR = actual measured motor torque) to it.
2. Put the nominal load of the crane onto the hock (100% load).
3. Download the trace and start it.
4. Lift the load with nominal speed:
 - 50 Hz outside U.S.
 - 60 Hz inside U.S.
5. Look at the trace and stop it once the torque curve is constant.
6. Determine the average torque during the phase of running at nominal frequency.

Example:



7. Enter the value (here in the example: 30.0%) at the input `i_stMDS.rTrqLoadNom`.
8. Connect all other pins and make sure `i_stMDS.xLcEn` is FALSE.

Section 18.13

Function Block Description - MaintenanceDataStorage_2

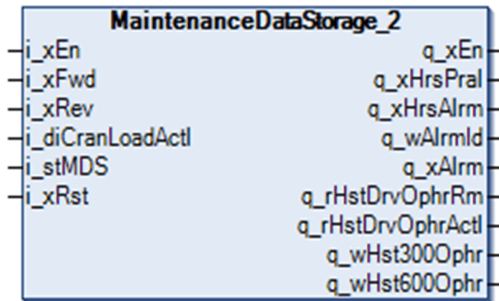
What Is in This Section?

This section contains the following topics:

Topic	Page
MaintenanceDataStorage_2 Function Block	588
Pin Description - MaintenanceDataStorage_2	590
Commissioning Procedure	595

MaintenanceDataStorage_2 Function Block

Pin Diagram

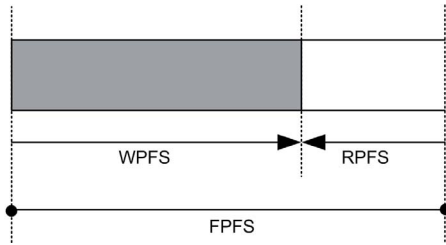


Function Block Description

The MaintenanceDataStorage_2 function block is an evolution of MaintenanceDataStorage function block.

It has a simplified interface and higher precision of measurement. It is used to supervise all important maintenance functions. This block is used to monitor and provide maintenance information for the Hoist axis:

PFS	Safety work period hoist main gearbox
FPFS (Factory PFS)	This is the operation time for the hoist main gearbox with nominal load. This information is provided by the manufacturer.
WPFS (Wasted PFS)	This is the proportional number of hours for which the hoist has operated with real load.
RPFS (Remain PFS)	The difference between FPFS and WPFS is the RPFS. RPFS is calculated using the formula: $RPFS = FPFS - WPFS$.



Formula to Calculate WPFS with a Load Cell

In this case, the following relationship is used to calculate WPFS:

$$WPFS = \sum_1^n T \left(\frac{LdAct}{LdNom} \right)^3$$

Where:

T = Actual cycle time. This time is calculated internally.

LdAct = Actual load of the drive.

LdNom = Nominal load of the drive.

Formula to Calculate WPFS without a Load Cell

In this case, the following relationship is used to calculate WPFS:

$$WPFS = \sum_1^n T \left(\frac{TrqAct}{TrqNom} \right)^3$$

Where:

T = Actual cycle time. This time is calculated internally.

TrqAct = Actual torque of the drive.

TrqNom = Nominal torque of the drive.

Pin Description - MaintenanceDataStorage_2

Overview

You will find the following pin descriptions:

- Input pin description (*see page 590*)
- Sub-structure description (*see page 591*)
- Output pin description (*see page 592*)

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (<i>see page 591</i>) below this table.
i_xFwd	BOOL	Forward command from the hoist axis. NOTE: It is not possible to run forward (i_xFwd) and reverse (i_xRev) at the same time. The FB accepts the first command in time if both inputs are TRUE. TRUE: Forward FALSE: Not forward
i_xRev	BOOL	Reverse command from the hoist axis. NOTE: It is not possible to run forward (i_xFwd) and reverse (i_xRev) at the same time. The FB accepts the first command in time if both inputs are TRUE. TRUE: Reverse FALSE: Not reverse
i_diCranLoadAct1	DINT	Load sensor input or torque input of the drive. Range: -2147483648...+2147483647 Scaling/Unit: 1 kg or 1.0% torque Refer to detailed description (<i>see page 591</i>) below this table.
i_stMDS	MDS	MDS - structure input. Refer to Sub-Structure Description (<i>see page 591</i>).
i_xRst	BOOL	Reset input to clear all alarms. The detected alarms are reset on a rising edge. TRUE: Active FALSE: Inactive

i_xEn

By setting the `i_xEn` input to `FALSE` the output states will be overwritten by a defined fallback state.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `MaintenanceDataStorage_2` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_xHrsPral</code>	BOOL	FALSE
<code>q_xHrsAlrm</code>	BOOL	FALSE
<code>q_wAlrmId</code>	BOOL	0
<code>q_xAlrm</code>	BOOL	FALSE
<code>q_rHstDrvOphrRm</code>	BOOL	not altered - retained value
<code>q_rHstDrvOphrAct1</code>	BOOL	not altered - retained value
<code>q_wHst300Ophr</code>	BOOL	not altered - retained value
<code>q_wHst600Ophr</code>	BOOL	not altered - retained value

i_diCranLoadActI

This is the sensor input of the load cell or the actual torque of the drive.

If the FB is used with a load cell, the `xLcEn` parameter of the MDS structure needs to be set to `TRUE`. Incoming load value must be scaled to 1.0 kilogram.

If the FB is used with the torque value of the drive, the `xLcEn` parameter of the MDS structure needs to be set to `FALSE` (default). Incoming torque values must be scaled to 1.0 percent of torque.

Sub-Structure Description of i_stMDS

A structure of the data type `MDS` containing configuration parameters.

For example, a declaration of structure variable with initial values:

```
VAR
structure_instance: data_type := (element1 := XX, element2 := YY);
END_VAR
```

Structure Parameter	Data Type	Description
xLcEn	BOOL	Selector to enable load cell usage or torque of the drive. If the FB is used with a load cell, this parameter must be set to TRUE. If the FB is used with the torque value of the drive, this parameter must be set to FALSE (default). TRUE: Enables load cell FALSE: Disables load cell
wHstDrvOphrMax	WORD	Time input according to the FEM classification of the crane, normally referring to the time in hours at 100% load (nominal load) after which the crane needs to be serviced. Range: 1...65535 Scaling/Unit: 1 hour
dwCranLoadNom	DWORD	Nominal load (100% load) of the crane. Range: 1...4294967295 Scaling/Unit: 1 kg
rTrqLoadNom	REAL	Torque value of the drive running at nominal speed of the motor with nominal load of the crane. This parameter needs to be measured during a test with the above mentioned prerequisites, i.e. by a trace of SoMachine. Range: 1...300 Scaling/Unit: 1% Example, see below this table.

Example to the torque value (rTrqLoadNom):

On a 100 tons crane we put a load of 100 tons at the hook and run the European motor at its nominal frequency of 50 Hz. During that run at constant speed we trace the incoming value at `i_diCranLoadAct1` by a trace of SoMachine.

The value at constant speed of 50 Hz is in between 81% and 83% torque of the drive. So we take the average value of 82.0 to enter it at this input.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	Displays the status of the enable parameter input <code>i_xEn</code> . If the output stays FALSE although <code>i_xEn</code> is TRUE, an SoMachine controller platform cannot be detected. TRUE: Function block enabled. FALSE: Function block disabled.

Output	Data Type	Description
q_xHrsPral	BOOL	Status of the pre-alarm of the remaining operating hours. If remaining operating hours are equal or smaller than 10% of <code>i_stMDS.wHstDrvOphrMax</code> , the output is set to TRUE. TRUE: Alarm detected FALSE: No alarm detected
q_xHrsAlrm	BOOL	Status of the alarm of the remaining operating hours. If remaining operating hours are equal or smaller than zero the output is set to TRUE. TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification. Range: 0..127 Refer to Notifications (<i>see page 594</i>).
q_xAlrm	BOOL	Detected alarm bit. TRUE: Alarm detected FALSE: No alarm detected
q_rHstDrvOphrRm	WORD	Value of remaining hours until servicing. That means, nominal hours according to FEM class minus actual operating hours based on real load. Range: 0... <code>i_stMDS.wHstDrvOphrMax</code> Scaling/Unit: 1 hour
q_rHstDrvOphrAct1	WORD	Value of actual operating hours since the start of the measurement based on real load. Range: 0... <code>i_stMDS.wHstDrvOphrMax</code> Scaling/Unit: 1 hour
q_wHst300Ophr	WORD	Hours with more than 300 operations/hour. Range: 1..65535 Scaling/Unit: 1 hour
q_wHst600Ophr	WORD	Hours with more than 600 operations/hour. Range: 1..65535 Scaling/Unit: 1 hour

Notifications

Identification of the alarm, if `q_xAlrm` is TRUE.

Alarm Bit <code>q_wAlrmId</code>	Description
0	<code>i_stMDS.wHstDrvOphrMax</code> was left at zero (default) or is not connected.
1	<code>i_stMDS.rTrqLoadNom</code> was left at zero (default) or is not connected (and <code>i_stMDS.xLcEn</code> is FALSE).
2	<code>i_stMDS.dwCranLoadNom</code> was left at zero (default) or is not connected (and <code>i_stMDS.xLcEn</code> is TRUE).

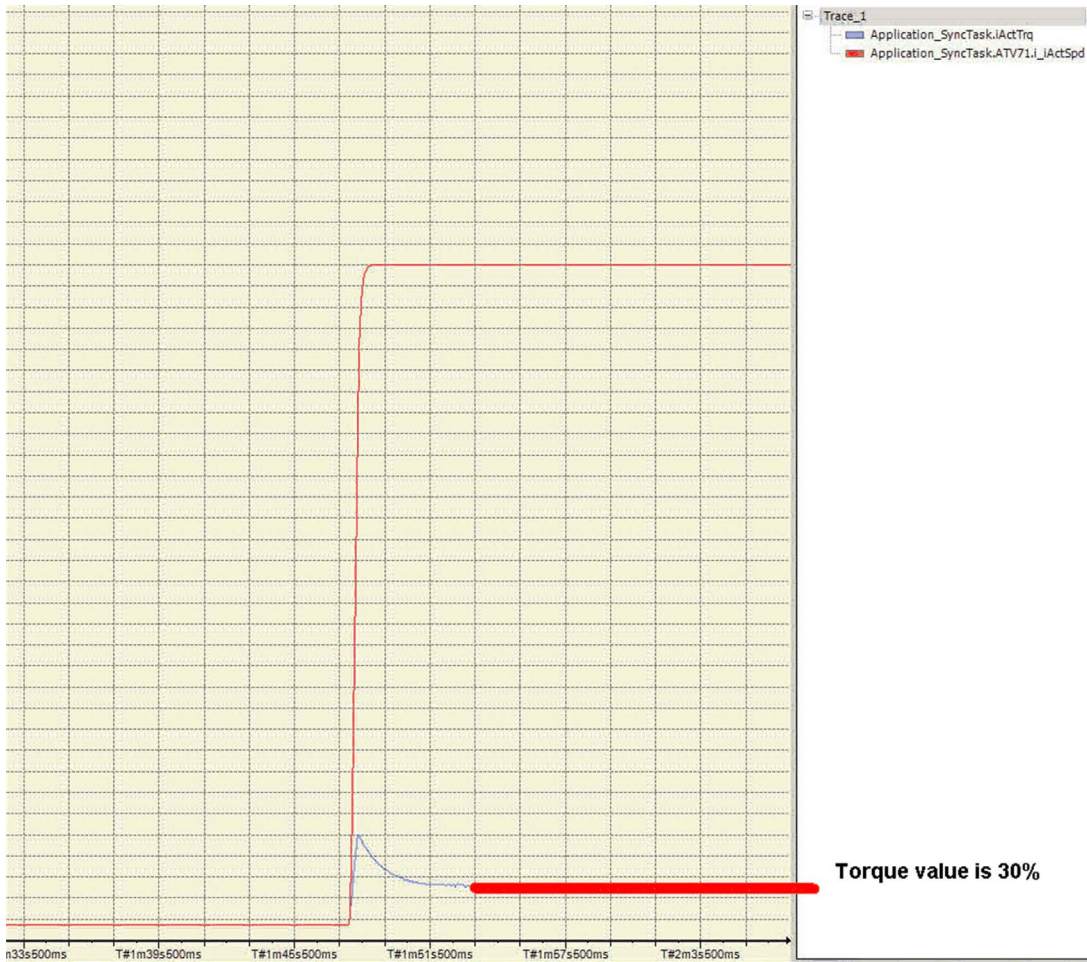
Commissioning Procedure

Commissioning Procedure of the MaintenanceDataStorage_2 Function Block:

The following steps explain the procedure on how to configure the Maintenance-DataStorage_2 function block using the torque of the drive:

1. Insert a trace object into your SoMachine project and map the actual speed and the filtered torque (OTR = actual measured motor torque) to it.
2. Put the nominal load of the crane onto the hock (100% load).
3. Download the trace and start it.
4. Lift the load with nominal speed:
 - 50 Hz outside U.S.
 - 60 Hz inside U.S.
5. Look at the trace and stop it once the torque curve is constant.
6. Determine the average torque during the phase of running at nominal frequency.

Example:



7. Enter the value (here in the example: 30.0%) at the input `i_stMDS.rTrqLoadNom`.
8. Connect all other pins and make sure `i_stMDS.xLcEn` is FALSE.

Section 18.14

Function Block Description - PasswordDataStorage

What Is in This Section?

This section contains the following topics:

Topic	Page
PasswordDataStorage Function Block	598
Pin Description - PasswordDataStorage	600

PasswordDataStorage Function Block

Pin Diagram



Function Block Description

The PasswordDataStorage function block is used to reset the data logged in all Monitoring data storage function blocks (StatisticDataStorage, AlarmDataStorage and MaintenanceDataStorage). An instance of the PasswordDataStorage function block should be connected to the reset pins of all blocks.

For a certain correct password on the password input, the corresponding bit is set to TRUE in the output word.

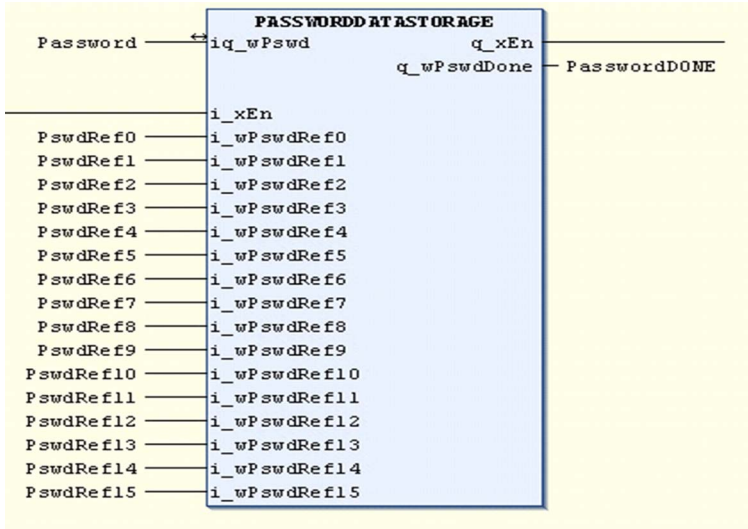
Passwords have to be assigned to reset the StatisticDataStorage (to reset hoist data, slew data, trolley data, and traveling data), the AlarmDataStorage (Overload Alarm Data Storage, Overtorque Alarm Data Storage, Overspeed Alarm Data Storage, Load Slipping Alarm Data Storage, Encoder Alarm Data Storage blocks) and MaintenanceDataStorage function block.

NOTE: The function blocks StatisticDataStorage, AlarmDataStorage and MaintenanceDataStorage have become obsolete, but have been retained to maintain compatibility for older applications. We recommend using the following new function blocks in your projects: StatisticDataStorage_2, AlarmDataStorage_2 and MaintenanceDataStorage_2.

The new function blocks are not pin-compatible with the obsolete function blocks.

Procedure Example

You have to create an instance of the PasswordDataStorage function block as shown in the figure below:



The output variable must be connected bitwise to the reset input (`i_xRst`) of the function blocks that should be reset by the PasswordDataStorage function block.

The connection between a Magelis XBTGT display and controller is supported with the SoMachine protocol. The password coming from the Magelis uses a variable in the Global Variables List (GVL).

After one cycle, the function block sets the incoming password back to zero.

Pin Description - PasswordDataStorage

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description of i_xEn below this table.
i_wPswdRef0...15	WORD	Password for Bit 0...15 Range: 1...65535

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of PasswordDataStorage are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wPswdDone	WORD	0

Outputs that are not listed will retain their current values.

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_wPswdDone	WORD	Used to reset all blocks of Monitoring data storage. Range: 1...65535

Input/Output Pin Description

Input/Output	Data Type	Description
iq_wPswd	WORD	Contains the entered password. Range: 1...65535.

Section 18.15

Quick Reference Guide

What Is in This Section?

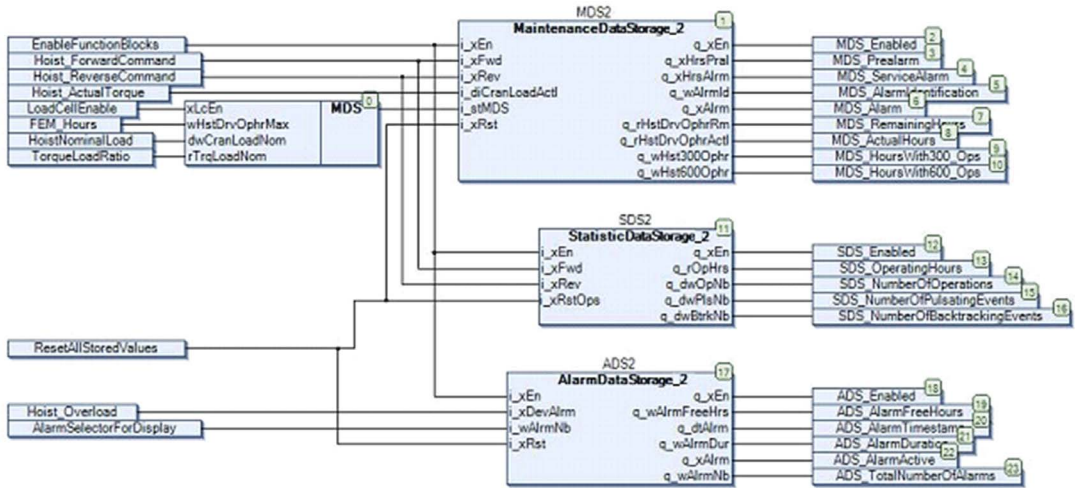
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	602
Troubleshooting	603

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the Monitoring Data Storage function blocks:



Troubleshooting

Troubleshooting

Issue	Cause	Solution
Maintenance block output q_wAlrmId=1	Factory PFS hours input entered by user is 0.	Enter a non-zero value for FPFS input
Maintenance block output q_wAlrmId=2	Load cell acquisition data (i_xLcEn) is enabled, but nominal value of load entered by user (i_wCranLoadNom) is 0.	Enter a non-zero value for LNOM input
Motor torque information read from ATV71 has an incorrect sign	Wrong input for motor torque was used.	Do not use Motor torque or Motor Torque Scope as an input for the actual torque inputs of the drive. Use 4 Quadrant Torque instead. Because 4 Quadrant Torque cannot be read by CANopen directly, use AO1 of the drive to be read and configure 4 Quadrant Torque to the AO1 output. Verify the scaling of AO1, it should be set to 4 to 20 mA. A value of 12000 represents zero torque, 4000 is -300% torque and 20000 is 300% torque.

Part XII

Overload Control

Overview

This part explains the functionality and implementation of the Overload control functions in the Hoisting Library.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
19	Overload Control Function: Detect Mechanical Overload on Hoisting Devices	607
20	Overload_EN15011: Overload Calibration With the Requirements of EN15011	637

Chapter 19

Overload Control Function: Detect Mechanical Overload on Hoisting Devices

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
19.1	Functional and Machine Overview	608
19.2	Architecture	616
19.3	Function Block Description - OverloadCtrlTrq	620
19.4	Function Block Description - OverloadCtrlDist	625
19.5	Function Block Description - OverloadCtrlEnc	630
19.6	Troubleshooting	635

Section 19.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	609
Machine Overview	615

Functional Overview

Functional Description

The Overload control function helps to protect the hoisting equipment from mechanical overload. The function uses the actual torque to detect an overload. In the case of an overload, it stops any ascending motion of the hoist and only allows descent.

The Overload control function is applicable to:

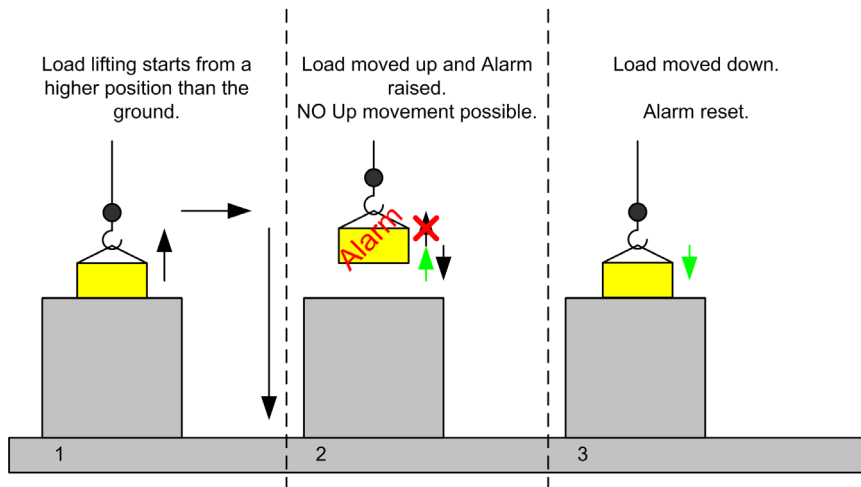
- Industrial cranes (Hoisting movement)
- Construction cranes (Hoisting movement)

The Overload control function is designed to be used on ATV71 and ATV312 drives.

Considerations for the Configuration


A condition may arise where both upward and downward movement of the load may be prevented due to the hook limit switch down stop position during an overload alarm. To better explain this condition, consider the following scenarios.

Scenario A: Movement with reset by lowering the load on the starting position.



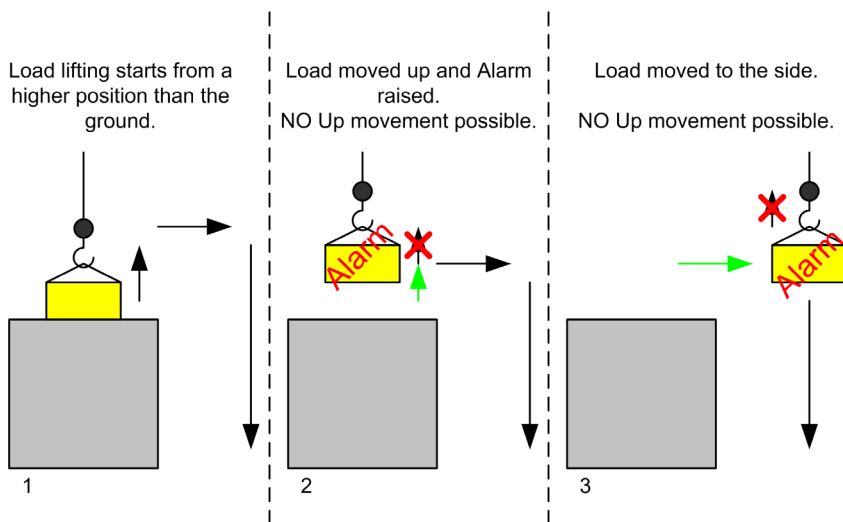
Legend:

Symbol	Meaning
↑	Intended movement
✗	Disallowed movement (restricted by the AFB)




Symbol	Meaning
	Movement done

1. The operator wants to move a load from a higher position to the ground.
2. After lifting the load up, the Overload alarm is raised. A continued up movement is no longer possible.
3. The load is set back on the starting position and the Overload alarm is reset.

Scenario B: Movement with moving the load to the side and attempting to the ground.

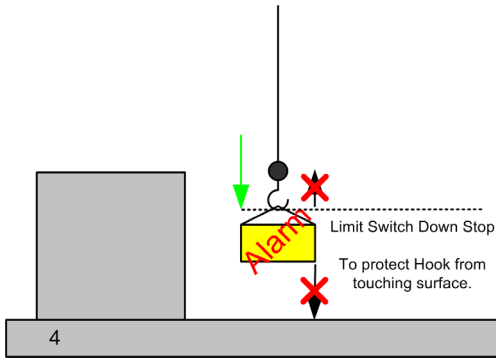


Legend:

Symbol	Meaning
	Intended movement
	Disallowed movement (restricted by the AFB)
	Movement done

1. The operator wants to move a load from a higher position to the ground.
2. After lifting the load up, the Overload alarm is raised. A continued up movement is no longer possible.
3. The load is moved to the side to lower it down to the ground.

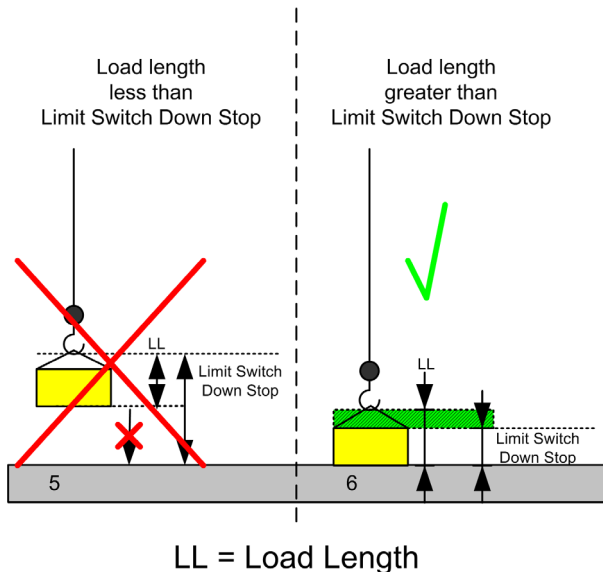
Load lowered beneath start level.
Hook reached the Limit Switch Down Stop.
NO Up OR NO Down movement possible.



4. The load is lowered beneath the starting level. The AFB still maintains the alarm and does not allow an up movement.

If the Hook reaches the Limit switch for the down movement, before lowering the load onto the ground, the operator can not move the load either up or down. The AFB limits upward movement of the load due to the Overload alarm, and the Limit switch does not allow further downward movement. The load, which is overloading the crane, is suspended and the operator can not move it.

To prevent this condition, you must make sure that the load length is greater than the distance imposed by the Limit Switch Down Stop:



5. A high limit switch position for the Hook down stop movement could block either upward or downward movement of the load.
6. To prevent this condition, the load length must be greater than the limit switch for down stop movement level.

⚠ WARNING

PROLONGED SUSPENSION OF AN OVERLOAD

Ensure that the configuration of the mechanical limit switch down stop does not prevent any load from being positioned on the ground.

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

Three Methods to Detect and Reset an Overload

The Overload control solution involves 3 methods that detect overload by torque and reset in 3 different ways:

- Overload reset using the torque method (*see page 618*)
- Overload reset using the distance method (*see page 618*)
- Overload reset using the encoder method (*see page 619*)

Why Use the Overload Control Function

The Overload control helps in protecting hoisting machinery by detecting mechanical overload and stopping the device before any damage can occur. The function block provides monitoring of potential overload conditions. If an overload condition is indicated, upward movement of the hoist must be prevented by the crane control commands and only downward movement should be allowed until the overload condition is corrected.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

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

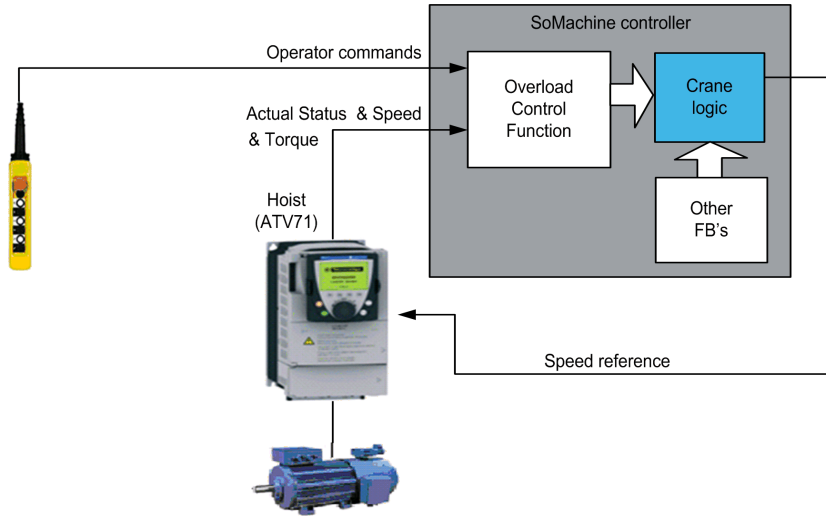
Solution with the Overload Control Function

The actual load at the motor is derived from the torque at the drive. This torque is used for Overload control function block. Detected overload are reset using the torque, encoder feedback, or distance traveled.

Design & Realization Constraints and Assumptions

It is assumed that when the motor moves in forward direction, the hoist moves up and when the motor moves in reverse direction, the hoist moves down. It is necessary that the encoder produces ascending pulses when hoist moving up and descending pulses when the hoist moves down.

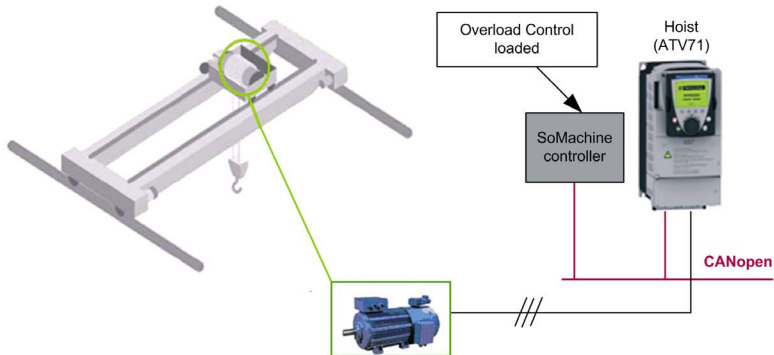
Functional View



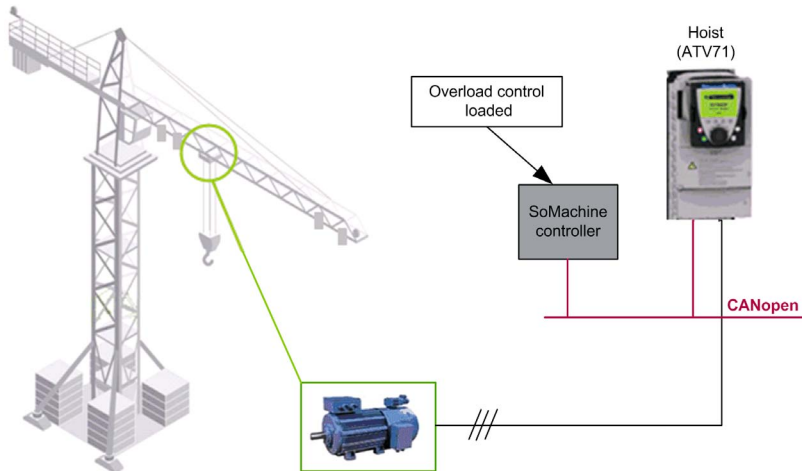
Machine Overview

Machine View

The following figure is the machine overview of the Overload control in the industrial crane.



The following figure is the machine overview of the Overload control in the tower crane.



Section 19.2

Architecture

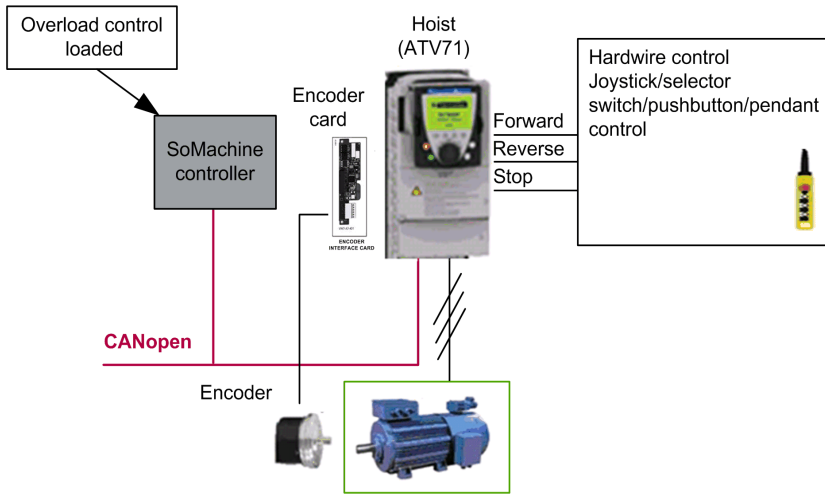
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	617
Software Architecture	618

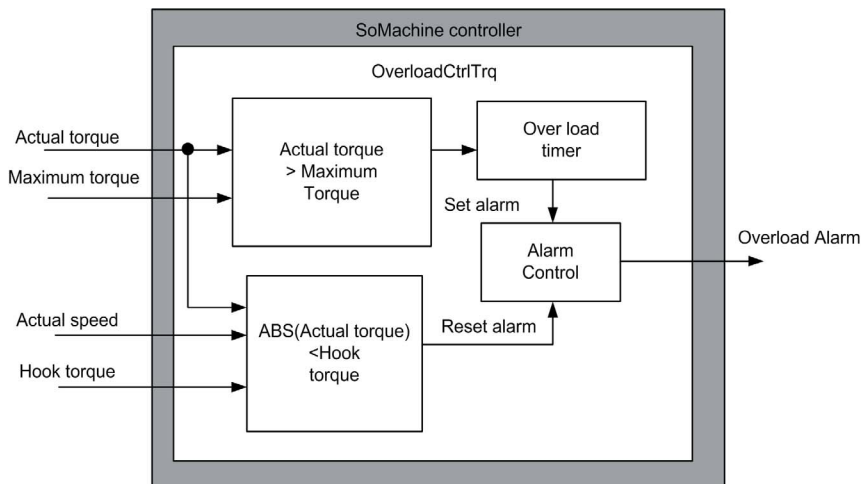
Hardware Architecture

Hardware Architecture Overview



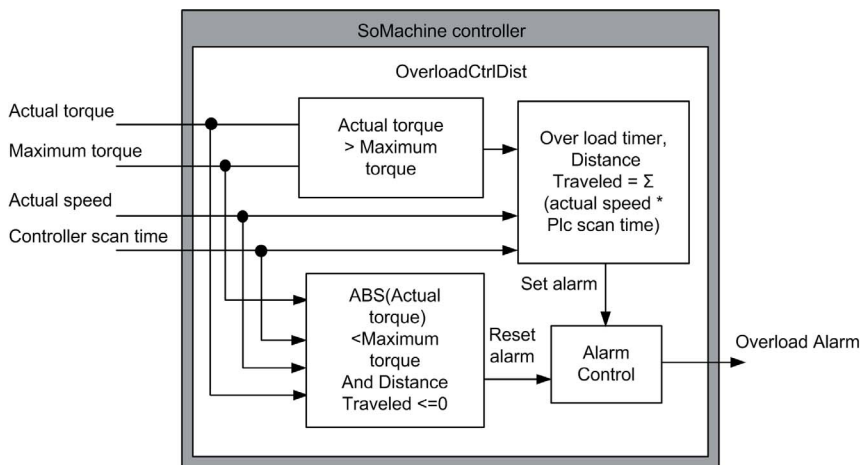
Software Architecture

DataFlow Overview - Overload Control Torque Method



In this overload control torque method, when the actual torque is greater than or equal to the maximum torque for a period of time defined by the user an overload alarm is generated. The overload alarm will only reset when the actual torque is less than the hook torque and the hoist is moving down.

DataFlow Overview - Overload Control Distance Method

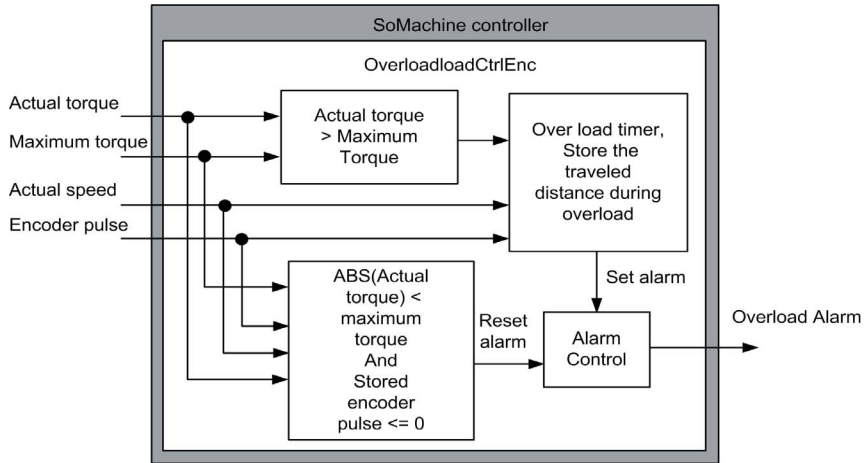


In this overload control distance method, when the actual torque is greater than the maximum torque for a period of time defined, an overload alarm is generated.

$$\text{Distance traveled} = \text{Distance traveled} + \text{Actual speed (RPM)} * (\text{Controller scan time (ms)} / 60000)$$

The overload alarm will only reset when the actual torque is less than the maximum torque and the distance traveled under overload is less than or equal to zero.

DataFlow Overview - Overload Control Encoder Method



In this overload control encoder method, the encoder connected to the hoist motor is used as a feedback to store the distance traveled under overload condition. When the actual torque is greater than the maximum torque for a period of time defined, an overload alarm is generated.

The overload alarm will only reset when the actual torque is less than the maximum torque, and the distance traveled under overload is less than or equal to zero.

Section 19.3

Function Block Description - OverloadCtrlTrq

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `Overload_EN15011`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
OverloadCtrlTrq Function Block	621
Pin Description - OverloadCtrlTrq	623

OverloadCtrlTrq Function Block

Pin Diagram

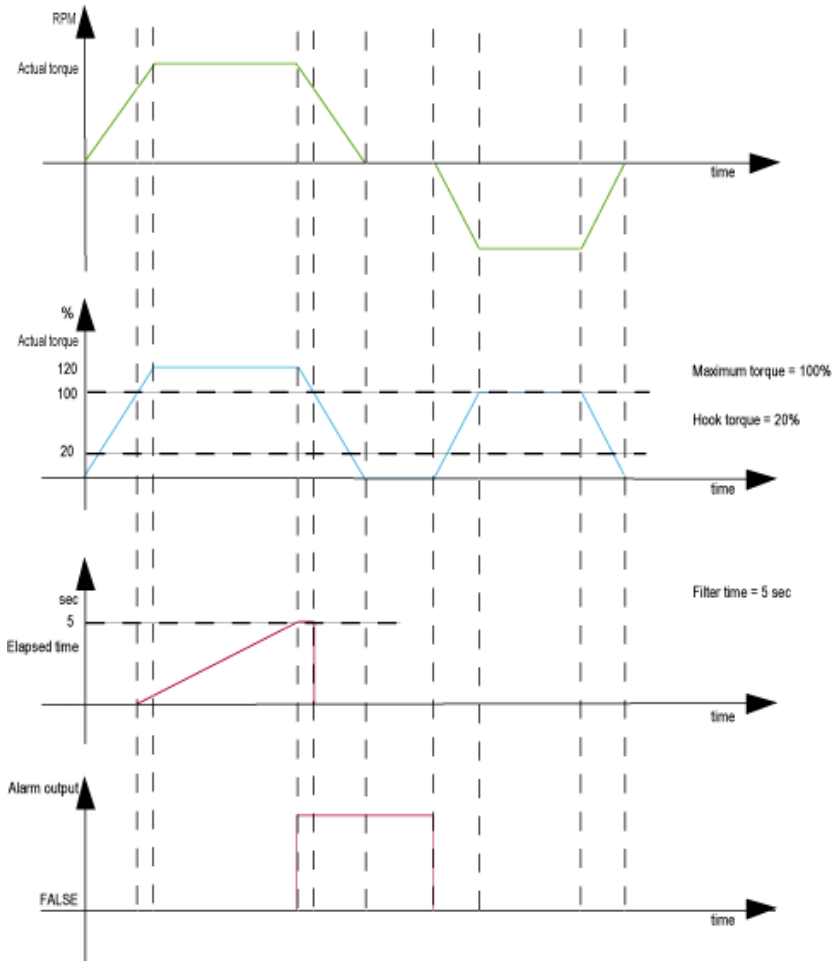


Function Block Description

The actual torque on the motor is read from the drive through CANopen. The motor torque is proportional to the weight of the load. If the actual torque is greater than the preset maximum value *i_wDrvTrqMax*, a timer is started. When the time delay expires, the overload alarm is activated. Clearing of the overload alarm allows upward movement of the load.

The overload alarm is only cleared when the absolute value of motor torque drops below the hook torque *i_wHookTrq* value and the drive is in reverse motion, moving downwards.

Timing Chart



Pin Description - OverloadCtrlTrq

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	The enable function enables or disables the function. TRUE: Enabled FALSE: Disabled Refer to detailed description (see page 623) of i_xEn.
i_iDrvTrqAct1	INT	This is the actual torque on the motor. Range: -3000...3000 Scaling/Unit: 0.1% Refer to detailed description (see page 624) of i_iDrvTrqAct1.
i_wDrvTrqMax	WORD	This is the maximum torque allowed before an overload alarm is generated. Range: 0...3000 Scaling/Unit: 0.1%
i_wFltrTime	WORD	This is the time delay set by the operator for the function block to wait before generating an overload alarm. This is used to help avoid any jerking in the system being registered as an alarm. Range: 0...200 s Scaling/Unit: 0.1 s
i_iDrvSpdAct	INT	This is the actual speed value from the drive input through CANopen. It is just used to know if the hoist is in motion. Range: -6000...6000 RPM
i_wHookTrq	WORD	This is the torque of the drive with gearbox, drums, ropes and a hook. Range: 0...1000 Scaling/Unit: 0.1% Refer to detailed description (see page 624) of i_wHookTrq.

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `OverloadCtrlTrq` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_xOvldAlrm</code>	BOOL	FALSE

Outputs that are not listed will retain their current values.

`i_DrvTrqAct1`

This input `i_iDrvTrqAct1` is read in through the CANopen interface directly from the drive. It is a mandatory input for the function.

The range of the torque is -300% to 300%.

`i_wHookTrq`

You must run the drive without a load on the hook and note the percentage (0.1%/unit) load from the monitoring menu on the drive or by monitoring the corresponding CANopen parameters. This value should then be applied to this parameter. This is mandatory for Overload control torque.

Output Pin Description

Output	Data Type	Description
<code>q_xEn</code>	BOOL	This displays the status of the enable parameter. TRUE: Enabled FALSE: Disabled
<code>q_xOvldAlrm</code>	BOOL	This is the overload alarm output. TRUE: Alarm overload detected FALSE: No alarm detected

`q_xOvldAlrm`

If any drive operation needs to be restricted during an overload condition, the application designer must interlock this particular output to the crane program.

NOTE: The Overload alarm `q_xOvldAlrm` is retained after warm start.

Section 19.4

Function Block Description - OverloadCtrlDist

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `Overload_EN15011`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
OverloadCtrlDist Function Block	626
Pin Description - OverloadCtrlDist	628

OverloadCtrlDist Function Block

Pin Diagram



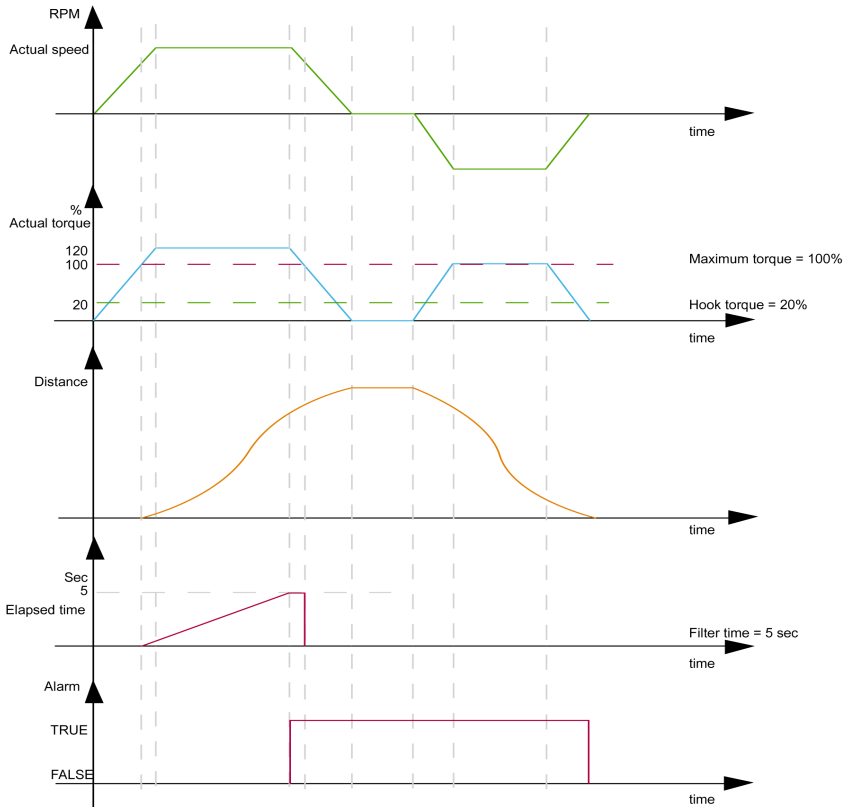
Function Block Description

The distance method is based on the actual position of the hoist. If the actual torque is greater than the preset maximum value `i_wDrvTrqMax`, a timer is started and the distance moved is calculated using the formula:

$$\text{Distance} = (\text{Actual speed of drive} * \text{Cycle Time}) + \text{Distance}$$

When the time delay expires, the overload alarm is activated and upward movement is blocked. The load must now be lowered until the calculated distance is less than or equal to zero and the absolute value of the torque value is less than the maximum allowed value `i_wDrvTrqMax`.

Timing Chart



NOTE: This timing chart does not describe the real behavior of the movement.

Pin Description - OverloadCtrlDist

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	The enable function enables or disables the function. Refer also the i_xEn description below this table. TRUE: Enabled FALSE: Disabled
i_iDrvTrqAct1	INT	This is the actual torque on the motor. Refer also the i_iDrvTrqAct1 description below this table. Range: -3000...3000 Scaling/Unit: 0.1%
i_wDrvTrqMax	WORD	This is the maximum torque allowed before an overload alarm is generated. Range: 0...3000 Scaling/Unit: 0.1%
i_wFltrTime	WORD	This is the time delay set by the operator for the function block to wait before generating an overload alarm. This is used to avoid any jerking in the system that is registered as an alarm. Range: 0...200 s Scaling/Unit: 0.1 s
i_iDrvSpdAct1	INT	This is the actual speed value from the drive input through CANopen. It is used for the calculation of the distance. Range: -6000...6000 RPM

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of OverloadCtrlDist are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_rDistOvld	REAL	0
q_xOvldAlrm	BOOL	FALSE

Outputs that are not listed will retain their current values.

i_iDrvTrqAct1

This input is read in through the CANopen interface directly from the drive. It is a mandatory input for the function. The range of the torque is -300% to 300%.

i_iDrvSpdAct1

The actual speed `i_iDrvSpdAct1` is used to calculate the distance traveled.

Example:

Actual speed = 1500 RPM

Controller scan time = 200 ms (internal measurement)

Then traveled distance by the hoist for this scan (revolutions) = $(1500) * (200 / (1000 * 60))$

Then traveled distance by the hoist for this scan = 5 (revolutions).

The distance traveled by the hoist is calculated by integrating the travel distance in each scan.

Output Pin Description

Output	Data Type	Description
<code>q_xEn</code>	BOOL	This displays the status of the enable parameter. TRUE: Function block enabled FALSE: Function block disabled
<code>q_xOvldAlrm</code>	BOOL	This is the overload alarm output. Refer also the <code>q_xOvldAlrm</code> description below this table. Range: -3000...3000 Scaling/Unit: 0.1%
<code>q_rDistOvld</code>	REAL	This is the traveled distance under overload. Refer also the <code>q_rDistOvld</code> description below this table Range: $1.175494957e-38F \dots 3.402829466e+38F$

q_xOvldAlrm

This is the overload alarm output. If the upward movement of the hoist needs to be restricted during an overload condition, the application designer must interlock this particular output to the crane program.

q_rDistOvld

When an overload condition is detected during an upward movement, the distance traveled until the stop is recorded. The load must be lowered back the position where the overload was first detected to be able to reset the overload alarm.

NOTE: The overload alarm `q_xOvldAlrm` and the distance traveled `q_rDistOvld` under overload are retained after warm start.

Section 19.5

Function Block Description - OverloadCtrlEnc

Notice

This function block has become obsolete, but has been retained to maintain compatibility for older applications. We recommend using the new function block `Overload_EN15011`. The new function block is not pin-compatible with the obsolete function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
OverloadCtrlEnc Function Block	631
Pin Description - OverloadCtrlEnc	633

OverloadCtrlEnc Function Block

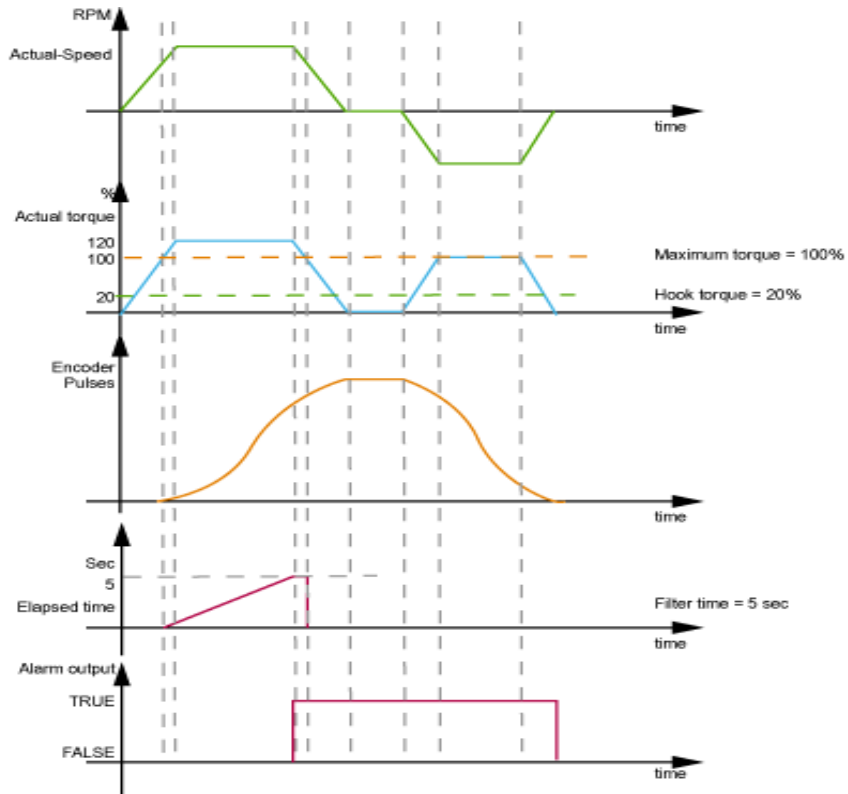
Pin Diagram



Function Block Description

The encoder method utilizes the actual encoder value on the motor. If the actual torque is greater than or equal to the preset maximum value `i_wDrvTrqMax`, a timer is started and the current encoder value is stored. Once the time delay expires the overload alarm is activated and upward movement is blocked. The load must now be lowered until the encoder value drops below the stored value and the absolute value of torque value is lower than the maximum allowed value `i_wDrvTrqMax`.

Timing Chart



NOTE: This timing chart does not describe the real behavior of the movement.

Pin Description - OverloadCtrlEnc

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	The enable function enables or disables the function. Refer also to i_xEn below this table. TRUE: Enabled FALSE: Disabled
i_iDrvTrqAct1	INT	This is the actual torque on the motor. Range: -3000..3000 Scaling/Unit: 0.1%
i_wDrvTrqMax	WORD	This is the maximum torque allowed before an overload alarm is generated. The range is between 0 and 3000 (0.1%). Range: 0..3000 Scaling/Unit: 0.1%
i_wFltrTime	WORD	This is the time delay set by the operator for the function block to wait before generating an overload alarm. This is used to avoid any jerking in the system that is being registered as an alarm. Range: 0..200 s Scaling/Unit: 0.1 s
i_iDrvSpdAct1	INT	This is the actual speed value from the drive input through CANopen. Refer also to i_iDrvSpdAct1 below this table. Range: -6000..6000 RPM
i_wEncPos	WORD	This is the actual value from the encoder through CANopen. Refer also to i_wEncPos below this table. Range: 0..65535 count

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of OverloadCtrlEnc are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_xOvldAlrm	BOOL	FALSE

Outputs that are not listed will retain their current values.

i_iDrvTrqAct1

The input `i_iDrvTrqAct1` is read in through the CANopen interface directly from the drive. It is a mandatory input for the function.

The range of the torque is -300% to 300%.

i_wEncPos

This is used as the feedback to calculate the distance traveled under overload.

NOTE: The 16 bit encoder resolution of `WORD` type is converted to 32 bit resolution `DINT` internally by the function block.

Output Pin Description

Output	Data Type	Description
<code>q_xEn</code>	BOOL	This displays the status of the enable parameter. TRUE: Enabled FALSE: Disabled
<code>q_xOvldAlrm</code>	BOOL	This is the overload alarm output. Refer also <code>q_xOvldAlrm</code> below this table. TRUE: Alarm overload detected. FALSE: No alarm detected.

q_xOvldAlrm

If the upward movement of the hoist needs to be restricted during an overload condition, the application designer must interlock this particular output to the crane program.

NOTE: The overload alarm `q_xOvldAlrm` and the position are retained after warm start.

Section 19.6

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
Overload alarm does not reset in torque method	Hook torque is not assigned	Make sure hook torque is non zero
Motor torque information read from ATV71 has an incorrect sign	Wrong input for motor torque was used.	Do not use Motor torque or Motor Torque Scope as an input for the actual torque inputs of the drive. Use 4 Quadrant Torque instead. Because 4 Quadrant Torque cannot be read by CANopen directly, use AO1 of the drive to be read and configure 4 Quadrant Torque to the AO1 output. Verify the scaling of AO1, it should be set to 4 to 20 mA. A value of 12000 represents zero torque, 4000 is -300% torque and 20000 is 300% torque.

Chapter 20

Overload_EN15011: Overload Calibration With the Requirements of EN15011

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
20.1	Functional Overview	638
20.2	Architecture	640
20.3	Function Block Description	643
20.4	Pin Description	644
20.5	Quick Reference Guide	654

Section 20.1

Functional Overview

Functional Overview

Functional Description

The `Overload_EN15011` function block is applicable to the following types of cranes:

- Industry Cranes (hoisting movement)
- Construction Cranes

The function reads the torque inputs from the hoist drive. It detects an overload situation according to the calibrated torque threshold and gives an alarm indication by lights and horn as well as stopping the upwards movement of the hoist. Additionally an alert indication by lights and horn at 90% of the nominal calibrated load is generated.

The function also shows the actual mass of the load with an accuracy of +/- 5%, depending on the setting of the speed loop of the drive.

The FB is conforms with the requirements of EN15011. Furthermore, it is possible to connect a load cell instead of the drive torque value.

An internal calibration procedure (*see page 657*) allows easy commissioning of the thresholds and the weight output.

Why Use the `Overload_EN15011` Function Block?

The intention of the function is to prevent running the hoist outside of its load capabilities by locking the upwards movement and protecting the mechanical structure of the crane, prolonging its life and reducing maintenance. The actual weight output of the function can be used to calculate lifetime applied load and to adjust the maintenance cycles according to the real use of the crane. The alert output at 90% of the nominal load can be used to operate the hoist at lower speeds to reduce stress to mechanical components, such as the gearbox. The function includes an algorithm to detect a fast and steep rise in torque of the drive to handle sudden changes of the load, i.e. when the lifting accessories, like chains or ropes attached to the hook, are tensioning.

The function includes an algorithm to detect a fast and steep rise in torque of the drive to handle sudden changes of the load, for example, when taking up the slack of the lifting accessories such as chains and ropes before reaching the tension of the bearing of the load on those accessories. This algorithm therefore would make a second measurement to replace the non-load values that were first measured.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

 **WARNING****UNINTENDED EQUIPMENT OPERATION**

Validate all function block input values before and while the function block is enabled.

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

Section 20.2

Architecture

What Is in This Section?



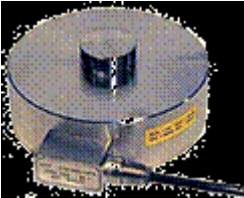
This section contains the following topics:

Topic	Page
Hardware Architecture	641
Software Architecture	642

Hardware Architecture

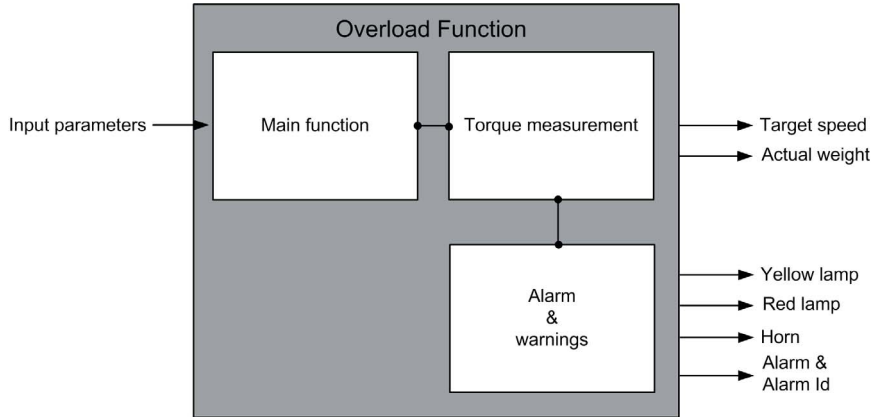
Necessary Equipment to Fulfill Limit Switch Functionality

The following table includes a list of the devices used in the system, including the description, followed by remarks to clarify the description of the device.

Symbol	Description	Remarks
	Incremental encoder	XCC incremental encoder radial solid shaft
	Absolute encoder	Osicoder absolute multiturn encoder CANopen through shaft
	Load cell	Third party item

Software Architecture

Data Flow Overview

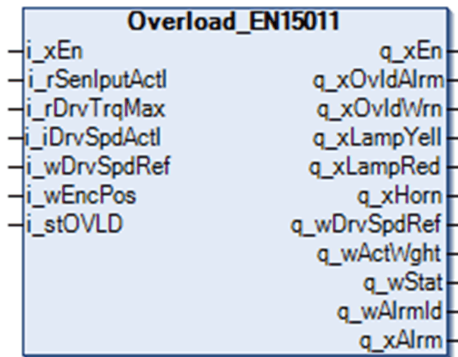


Section 20.3

Function Block Description

Overload_EN15011 Function Block

Pin Diagram



Function Block Description

If a calibration has not been done and the maximum torque override input `i_rDrvTrqMax` is zero, the maximum torque is set to 30% by default.

Section 20.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	645
Structured Variable Description	646
Output Pin Description	651

Input Pin Description

Input Pin Description

This function block requires metric input units.

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
i_rSenInputAct1	REAL	Actual torque input from the drive or load cell input. Range: -30000...+100000 Scaling/Unit: 0.01% for torque, 1 kg for load cell Refer to detailed description below this table.
i_rDrvTrqMax	REAL	Drive torque at 100% nominal load of the crane. Range: 0...30000 Scaling/Unit: 0.01% Refer to detailed description below this table.
i_iDrvSpdAct1	INT	Actual drive speed. Range: -6000...+6000 Scaling/Unit: 1 RPM
i_wDrvSpdRef	WORD	Target speed to the drive. Range: 0...6000 Scaling/Unit: 1 RPM
i_wEncPos	WORD	Encoder input from the drive. Range: 0...65535
i_stOVLd	OVLd	Refer to the Structured Variable Description (see page 646).

i_rSenInputAct1

This is either the actual torque input from the drive, or the input from the load cell sensor scaled to 1 kg.

We recommend using the 4Q-Torque output of the drive scaled to 0.01% accuracy.

i_rDrvTrqMax

This is the torque threshold input that will trigger an overload alarm. It represents the torque of the drive at 100% of the nominal load of the crane. If the measurement state of the Overload_EN15011 FB calculates an actual torque above this value the upwards movement is stopped and the overload alarm indicators `q_xLampRed`, `q_xHorn` are enabled.

If using a load cell this input is not used.

Structured Variable Description

OVLD (l_stOVLD)

A structure of the data type OVLD containing configuration parameters.

Structure Parameter	Data Type	Description
wSpdBnd	WORD	Deadband is the range of speed that defines the attainment of target speed. Range: 2...200 (default is 5) Scaling/Unit: 1 RPM Refer to detailed description below this table.
wTrqBndCnt	WORD	Number of cycles the torque must be inside the deadband. Range: 1...100 (default is 5) Scaling/Unit: 1 cycle Refer to detailed description below this table.
wTrqBndWdth	WORD	Area around the last torque value, deadband. Range: 1...300 (default is 5) Scaling/Unit: 1% Refer to detailed description below this table.
wSmple	WORD	Number of samples to collect for torque measurement. Range: 1...50 (default is 5) Scaling/Unit: 1 cycle Refer to detailed description below this table.
dwLoadNom	DWORD	Nominal load of the crane, 100% load. Range: 1...100000 (default is 88888) Scaling/Unit: 1 kg Refer to detailed description below this table.
wMeasFreq	WORD	Torque measurement frequency. Range: 1...6000 (default is 100) Scaling/Unit: 1 RPM Refer to detailed description below this table.
wTrqThres	WORD	Torque increase threshold to detect change of load. Range: 100...30000 (default is 1000) Scaling/Unit: 0.01% Refer to detailed description below this table.
wAntiTip	WORD	Maximum number of turns of the motor shaft without torque measurement. Range: 1...10000 (default is 50) Scaling/Unit: 1 turn Refer to detailed description below this table.
wEncPuls	WORD	Number of pulses per revolution of the encoder. Range: 1...65535 (default is 1024) Scaling/Unit: 1 pulse

Structure Parameter	Data Type	Description
wNoTrq	WORD	No load torque of the motor, without hook. Range: 1...10000 (default is 500) Scaling/Unit: 0.01% Refer to detailed description below this table.
rOvldThres	REAL	Tolerance of load to engage overload state. Range: 1...1.2 (default is 1.05) Scaling/Unit: 100% Refer to detailed description below this table.
xEnLdCell	BOOL	For the usage of a load cell instead of the torque of the drive. TRUE: Enables the load cell FALSE: Disables the load cell Refer to detailed description below this table.
rAcuWght	REAL	Accuracy of the actual weight output. Range: 1...1000 (default is 1) Refer to detailed description below this table.
wActWghtThres	WORD	Threshold to enable actual load output pin. Range: 1...65535 (default is 1000) Scaling/Unit: 1 kg Refer to detailed description below this table.
dwCalbWght	DWORD	Actual calibration weight. Range: 1...100000 (default is 1) Scaling/Unit: 1 kg Refer to detailed description below this table.
xCalb	BOOL	Enable calibration. TRUE: Calibration enabled FALSE: Calibration disabled Refer to detailed description below this table.
xOvld	BOOL	Enable overload test. TRUE: Overload enabled FALSE: Overload disabled Refer to detailed description below this table.

wSpdDbnd

To be able to measure the actual torque correctly, constant speed is needed. Therefore the FB is only starting to engage into the torque sampling, once the measurement frequency is reached. Depending on the settings of the speed loop of the drive the actual speed is oscillating around the target speed. To compensate this irregularity a dead band around the target speed is needed.

The FB defines target speed attainment as:

actual speed > (target speed – wSpdDbnd)

and

actual speed < (target speed + wSpdDbnd)

wTrqBndCnt

During starting and acceleration the torque of the drive is oscillating. To measure the actual torque correctly, the FB is waiting until the torque has settled and becomes relatively constant. This parameter is the number of cycles the torque values from the drive need to be inside the torque dead band defined by wTrqBndWdth to start the measurement.

wTrqBndWdth

Depending on the settings of the speed loop of the drive, the actual speed is oscillating around the target speed. To compensate this irregularity a dead band for the torque is needed. The FB starts counting values to fulfill wTrqBndCnt if the actual torque values are:

actual torque > (actual torque of the last cycle * (1.0 - wTrqBndWdth))

and

actual torque < (actual torque of the last cycle * (1.0 + wTrqBndWdth))

wSmple

This is the number of samples (equaling controller cycles) that are taken during the torque measurement. The resulting value is the average of all samples.

dwLoadNom

This is the nominal load of the hoist. If the nominal load is variable (e.g. changing based on the position of the trolley on a jib of a tower crane) this value must be updated continuously to reflect the actual nominal load.

wMeasFreq

This is the frequency in RPM at which the torque measurement is conducted. It must be bigger or equal to the LSP setting of the drive. The smaller the value chosen, the earlier the measurement will result in overload detection. We recommend setting this frequency higher than the slip of the motor, even when running in closed loop.

wTrqThres

This is a percentage of torque to detect a change in load after the torque measurement. If the actual value of the torque rises beyond this value, the drive is decelerated to measurement frequency again and the torque is measured one more time. Make sure to set the value above the normal rise of the torque during acceleration with a constant 100% load.

wAntiTip

From the start of the upwards movement of the hoist the distance is monitored. After a successful measurement of the torque the monitoring is disabled. The FB will compare the actual distance with the desired number of turns entered into this parameter. This is necessary to help prevent the user from inching the load upwards without performing a load measurement, and thereby allowing an excessive load to be lifted without triggering an alarm. If `wAntiTip` is engaged, it will stop the upwards movement by setting the overload alarm output to TRUE.

If `i_stOVLd.wAntiTip` is at 55 turns and `i_stOVLd.wEncPuls` is at 1024, the maximum distance available to measure the load is $55 \times 1024 = 56320$.

It is helpful to look at the `q_wStat` output of the FB during commissioning.

wNoTrq

This is a threshold of torque in percent.

Below this threshold value the measurement procedure is disabled.

If the motor is running with very little load, the actual torque value of the drive has high oscillations. Therefore any torque measurement in this range would be too inconsistent to use in application.

rOvldThres

This is the tolerance adjustment of the overload detection. It can be set between 0% to 20% tolerance.

If the maximum torque at 100% load is set to 10000, and `rOvldThres` is set to 1.1, a measured torque of 11000 must be exceeded to detect overload.

xEnLdCell

To enable the usage of a load cell instead of using the torque of the drive, this bit must be set TRUE. The signal of the sensor must be scaled to 1 kg. The measurement run at lifting up will be disabled. All signaling, the overload alarm, the 90% alert, the lamps and the horn remain active.

The input pin `i_rDrvTrqMax` does not need to be connected.

rAcuWght

This is an optional dividing factor to the `q_wActWght` output. By default it is set to 1.0 which puts the weight output to a scaling of 1 kg.

<code>rAcuWght</code>	<code>q_wActWght</code>
1.0	1 equals 1 kg
10.0	1 equals 10 kg
100.0	1 equals 100 kg
1000.0	1 equals 1 metric ton

This value is helpful for a display. A value of 123 with a scaling of 10 can be used to show the actual weight in tons with 2 decimal places, i.e. on an ATV71 display.

wActWghtThres

If the hoist is used with no load or little load, the torque value of the drive is not always reproducible and steady. Therefore it is possible to disable the actual weight output below a certain load, it will then always be zero. If you put a value of 1000.0 to this input, all loads below 1000 kg are represented as 0 kg. Depending on the mechanical circumstances of the crane the threshold for reproducible and steady results is at around 30% of the available torque of the drive.

dwCalbWght

This is the weight value in kilogram used during the calibration, that means, the calibration bit `i_stOVLD.xCalb` is TRUE. All internal calculations of the 100% load threshold are based on this value, therefore it is very important to measure the weight used precisely. If calibration is inactive this input does not have any effect.

xCalb

This is the switch to enable calibration. If the input `i_rDrvTrqMax` is left at zero the result of the calibration is used to analyze the load. If any value unequal to zero is entered at the `i_rDrvTrqMax` input, it will override the calibrated result.

Refer to the calibration procedure description (*see page 657*).

xOvld

This is the switch to enable the overload test during the acceptance test for a new crane.

If set to TRUE, it enables to lift loads with up to 300% torque. It can only be active for a maximum of 5 hours of movement. If left active for longer than 5 hours an overload alarm will be triggered preventing the upwards movement of the hoisting axis.

DANGER

CRANE OVERLOAD

You must provide the necessary security in the application of the `xOvld` input and restrict its use to commission and test.

Failure to follow these instructions will result in death or serious injury.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Function block enabled. FALSE: Function block disabled.
q_xOvldAlrm	BOOL	Overload detected, actual torque is above 100% load. TRUE: Alarm detected FALSE: No alarm detected Refer to detailed description below this table.
q_xOvldWrn	BOOL	90% load detected, actual torque is above 90% load. TRUE: Advisory FALSE: No advisory. Refer to detailed description below this table.
q_xLampYell	BOOL	Yellow lamp connection. TRUE: Lamp on FALSE: Lamp off
q_xLampRed	BOOL	Red lamp connection. TRUE: Lamp on FALSE: Lamp off
q_xHorn	BOOL	Horn connection. TRUE: Horn on FALSE: Horn off
q_wDrvSpdRef	WORD	Target speed to the drive. Range: 0...6000 Scaling/Unit: 1 RPM Refer to detailed description below this table.
q_wActWght	WORD	Actual weight after torque measurement. Range: 0...65535 Scaling/Unit: depends on chosen scaling at <code>i_stOVLd.rAcuWght</code>
q_wStat	WORD	Actual status of the FB. Range: 0...255 Refer to detailed description below this table.
q_xAlrm	BOOL	Detected alarm bit. TRUE: Alarm detected FALSE: No alarm detected Refer to detailed description below this table.
q_wAlrmId	WORD	Detected alarm identification. Refer to <code>q_wAlrmId</code> (<i>see page 653</i>). Range: 0...255

q_xOvldAlrm

Indicates an overload incidence if TRUE. That means, during the measurement process stage of the FB a torque value was calculated higher than `i_rDrvTrqMax`. The distance run since the start of the movement was memorized and the load has to be brought down to the same position where it was moved from to reset the alarm. The alarm is indicated by a constant horn and a flashing red light 1 s on and off.

If using a load cell the input value of `i_rSenInputAct1` is compared to `i_stOVLD.dwLoadNom`.

q_xOvldWrm

The calculated torque value of the measurement process stage of the FB is higher than 90% of `i_rDrvTrqMax`. A advisory is given by the yellow light and the horn by being activated for 1 s on and off. The advisory is reset by a new movement/measurement resulting in a lower torque value than 90% of `i_rDrvTrqMax`. If using a load cell the input value of `i_rSenInputAct1` is compared to 90% of `i_stOVLD.dwLoadNom`.

q_wDrvSpdRef

This is the speed reference to the hoist drive. It is mandatory to transfer the target speed by this block to get proper functionality of the torque measurement. See the instantiation example (*see page 655*).

q_wStat

Status Bit	Description
0	torque measurement in progress
1	weight detection
2	change of load detected, <code>i_stOVLD.wTrqThres</code>
3	Anti-Tip active, <code>i_stOVLD.wAntiTip</code>
4	overload test time greater than 5 hours
5	calibration done
6...12	not used
13	90% load alert detected
14	overload alarm detected
15	load cell enabled

q_xAlrm

An alarm signal is generated in cases of incorrect parameterization of input values. Refer to the `q_wAlrmId` table below. In case of an alarm the yellow light and the horn are flashing 1 s on and off.

q_wAlrmId

Identification of the alarm, if `q_xAlrm` is TRUE.

Bit Position	Description Represented by Bit Position
0	<code>i_stOVLD.wSpdDbnd > i_stOVLD.wMeasFreq</code>
1	<code>i_stOVLD.wAntiTip < 2</code>
2	overload timer greater than 5 hours

Section 20.5

Quick Reference Guide

What Is in This Section?

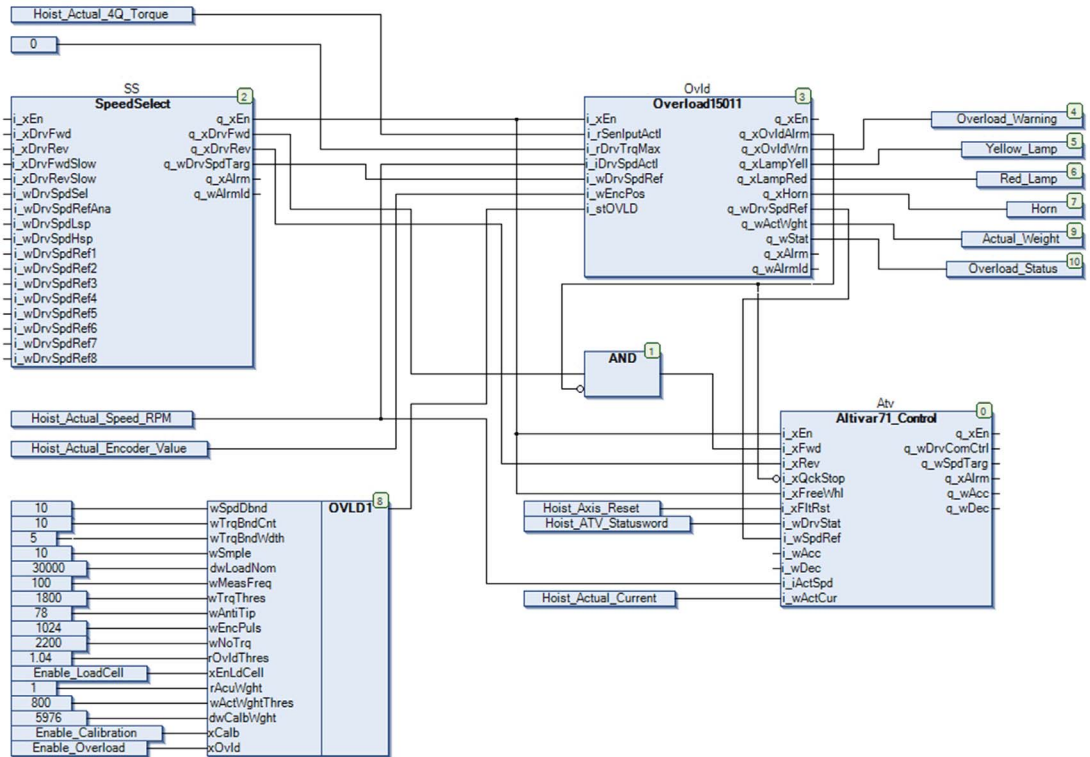
This section contains the following topics:

Topic	Page
Instantiation and Usage Example	655
Commissioning Procedure	656
Calibration Procedure	657
Troubleshooting	658

Instantiation and Usage Example

Instantiation and Usage Example

This figure shows an instantiation example of the Overload_EN15011 function block:



Commissioning Procedure

Commissioning Procedure of the Overload_EN15011 Function Block

1. Start SoMachine and open the relevant project file (*.prj).
2. Add the Hoisting Library to the Library Manager after opening the application file.
3. Instantiate the required function blocks and configure as per the requirements and within the limits specified in this document.
4. Verify the wiring and the application node number for CANopen.
5. Verify the wiring connections for the drive, motor and controller.
6. Use the programming cable, connect the PC to the controller and download the program.
7. For the first start-up of machine and function blocks it is advised to use reduced values before going to the projected settings (for example: full speed, full load, etc.).

Alarm Fallback State, Disable Fallback State

In the case of an alarm the lamp outputs `q_xLampYell` and `q_xLampRed` are flashing and the hoist movement is disabled by setting the output speed to zero.

If the FB is disabled the outputs `q_xOvldAlrm`, `q_xOvldWrn`, `q_xLampYell`, `q_xLampRed`, `q_xHorn` and `q_xAlrm` are set to FALSE.

The input target speed `i_wDrvSpdRef` is transferred to the output `q_wDrvSpdRef` to allow operation and the `q_wAlrmId`, `q_wStat` as well as `q_wActWght` is zero.

Calibration Procedure

Calibration Procedure of the Overload_EN15011 Function Block

1. Find a defined load to put onto the hook with a precisely measured weight in kilogram.
2. Enter the weight at `i_stOVLd.dwCalbWght`.
3. Enter the nominal load of the crane at `i_stOVLd.dwLoadNom`.
4. Put `i_stOVLd.xCalb` to TRUE.
5. Start the movement upwards/forward until the hoist accelerates beyond the desired measurement frequency `i_stOVLd.wMeasFreq`.
6. `q_wStat.5` will be TRUE once calibration is done.
7. Stop the movement and put `i_stOVLd.xCalb` to FALSE.
8. Compare the output value at `q_wActWght` with the actual weight of the load.
9. If the calculated weight differs from the real weight change the parameter setting, i.e. `wSpdBnd`, `wTrqBndCnt`, `wTrqBndWdth`, `wSmplE` and repeat from step 4 to 6.

Troubleshooting

Troubleshooting

Issue	Cause	Solution
q_xAlrm of the FB is TRUE.	One or more of the input parameters are out of range.	Check the input parameters.

Part XIII

Scale Input

Chapter 21

ScaleInput: Change Incoming Input to a Scaled Value

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
21.1	Functional Overview	662
21.2	Architecture	663
21.3	Function Block Description	665
21.4	Pin Description	666
21.5	Troubleshooting	670

Section 21.1

Functional Overview

Functional Overview

Functional Description

The Scale input function scales an analog or counter input from a device to a desired output range.

Why Use the ScaleInput Function Block?

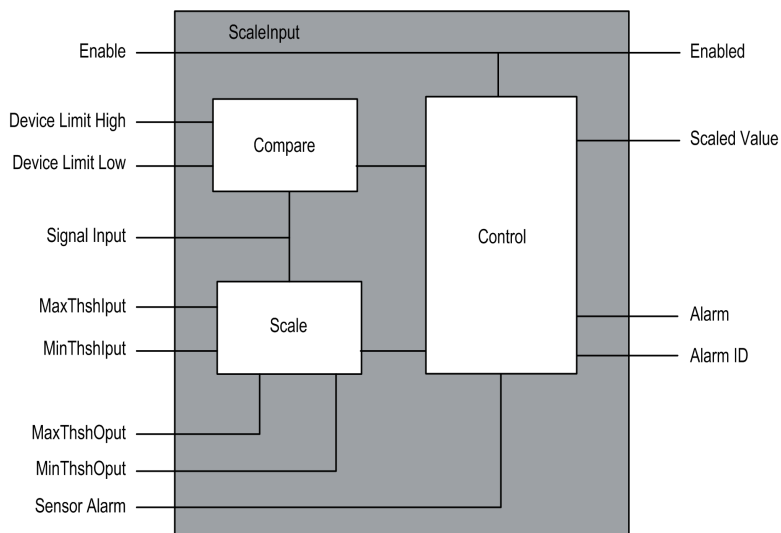
The ScaleInput function block allows to change incoming sensor or encoder input to a scaled value as may be required by an application.

Section 21.2

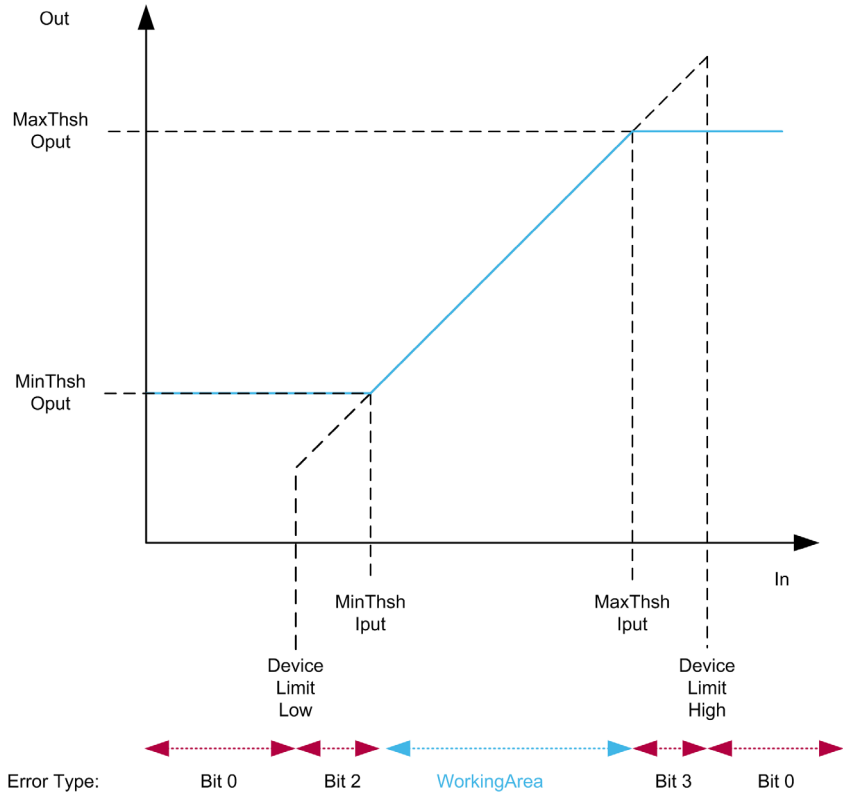
Architecture

Software Architecture

DataFlow Overview



Range Relations and Alarm Bits

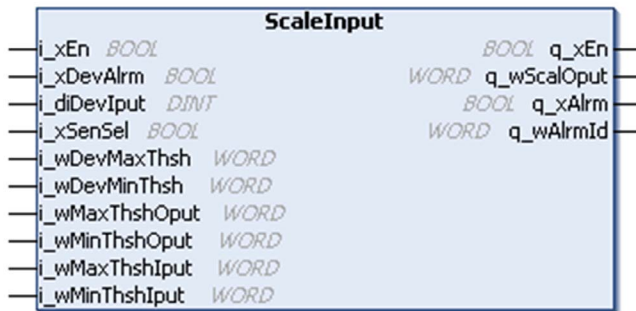


Section 21.3

Function Block Description

ScaleInput Function Block

Pin Diagram



Function Block Description

This function scales an analog or counter input from a device to a desired output range.

The `ScaleInput` function block allows to change incoming sensor or encoder `WORD` input to a scaled `WORD` value for application that needs it.

The input value is monitored to check that it is within the scaled parameters; otherwise an alarm signal is generated. An alarm signal can also be generated if the function block receives an indication that the device generating the analog input signal is not functioning correctly. Alarm bit 2 indicates that the input signal is below the low input limit and alarm bit 3 indicates the input signal is above high input limit.

Section 21.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	667
Output Pin Description	669

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (see page 667) of i_xEn.
i_xDevAlrm	BOOL	Device signal disable alarm. TRUE: Device alarm FALSE: No device alarm
i_diDevIput	DINT	Sensor input signal Range: -2147483648...2147483647 count Factory setting: 0
i_xSenSel	BOOL	Selector input to switch between pulse or analog sensor signal TRUE: Pulse input FALSE: Analog input
i_wDevMaxThsh	WORD	Maximum threshold of sensor output signal Range: 0...65535 Factory setting: 0
i_wDevMinThsh	WORD	Minimum threshold of sensor output signal Range: 0...65535 Factory setting: 0
i_wMaxThshOput	WORD	Maximum threshold of scaled output signal Range: 0...65535 Factory setting: 0
i_wMinThshOput	WORD	Minimum threshold of scaled output signal Range: 0...65535 Factory setting: 0
i_wMaxThshIput	WORD	Maximum threshold of desired scaling range of input signal Range: 0...65535 Factory setting: 0
i_wMinThshIput	WORD	Minimum threshold of desired scaling range of input signal Range: 0...65535 Factory setting: 0

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `ScaleInput` are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wScalOput	WORD	i_diDevInput
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

Output Pin Description

Output Pin Description

Output	Data Type	Data Type
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_wScalOput	WORD	Scaled output of the given input Range: 0...65535 Factory setting: 0
q_xAlrm	BOOL	Alarm detection Indicator TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (see page 669). Range: 0...15

Notifications

Bit Number	Description
0	The sensor output is outside the hardware range.
1	Device hardware alarm detected. The alarm bit from the sensor is high.
2	Device input is lower than the threshold of input-range-min.
3	Device input is higher than the threshold of input-range-max.

NOTE: An alarm does not stop the scaling. The input or output values are restricted to the given limits.

Section 21.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
ScaleInput: Alarm bit 0	The sensor output is outside the hardware range.	Check the sensor and replace if it is damaged. OR Change the range of device-limit-high and device-limit-low.
ScaleInput: Alarm bit 1	Device hardware alarm detected. The alarm bit from the sensor is high.	Check the sensor and replace if it is damaged.
ScaleInput: Alarm bit 2	Device input is lower than the threshold of input-range-low.	This indicates that you should lower the value of low input range if possible (<code>i_wDevMinThsh</code>).
ScaleInput: Alarm bit 3	Device input is higher than the threshold of input-range-high.	This indicates that you should increase the value of high input range if possible (<code>i_wDevMaxThsh</code>).

Part XIV

Smooth Slewing

Chapter 22

Smooth Slewing Function: Stepwise Smooth Crane Movement Based on Speed or Torque Value of the Slewing Drive

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
22.1	Functional and Machine Overview	674
22.2	Architecture	678
22.3	Function Block Description - SmoothSlewingSpd	682
22.4	Function Block Description - SmoothSlewingTrq	688
22.5	Troubleshooting	695

Section 22.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	675
Machine Overview	677

Functional Overview

Functional Description

This function allows to turn construction cranes with different acceleration and deceleration levels. As a result, it helps ensure a soft movement without creating a high force against the construction of the crane.

When starting a slewing (rotation) movement of the jib on a tower or self-erecting crane, it is necessary to overcome the inertia of the jib. There is always a chance of an over-torque (during start) or over-speed (during stop) condition. To help prevent this over-torque or over-speed condition, it is necessary to control the acceleration and deceleration ramps.

This specification explains two different solutions for smooth slewing:

- Type 1 : Speed based
- Type 2 : Torque based

This function influences acceleration and deceleration of the slewing drive:

- In Type 1: Speed based
The acceleration and deceleration time is adjusted based on the varying speed level during the acceleration and deceleration.
- In Type 2: Torque based
The acceleration and deceleration time is adjusted based on the varying torque level during the acceleration and deceleration.

The smooth slewing solution is applicable to:

- Tower cranes
- Self erecting cranes

Why Use the SmoothSlewingTrq or SmoothSlewingSpd Function Block?

The function blocks allow to create a soft turn of a tower or self-erecting crane based on the different levels of torque or speed movement reached. This helps protect the mechanism of the crane from high forces and allows a better stable movement of the load.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

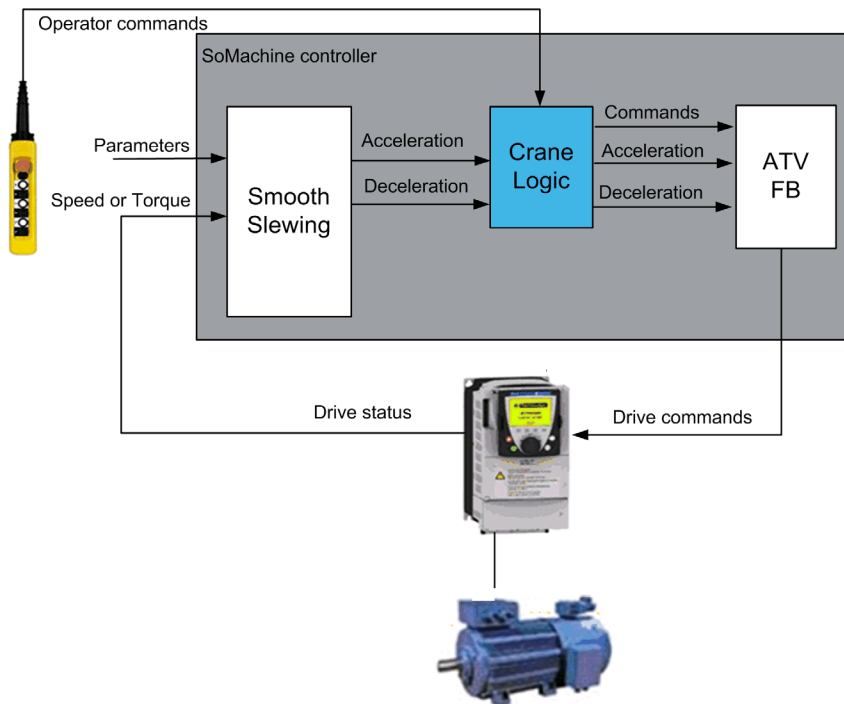
Validate all function block input values before and while the function block is enabled.

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

Solution with the SmoothSlewingTrq or SmoothSlewingSpd Function Block

The function blocks use information about torque or speed out of the drive. The function block is changing depending on the level of torque/speed and the acceleration or deceleration rate of the drive. This creates a flexible ramp which allows the drive to lead the crane structure into a gentle rotation (Smooth slewing).

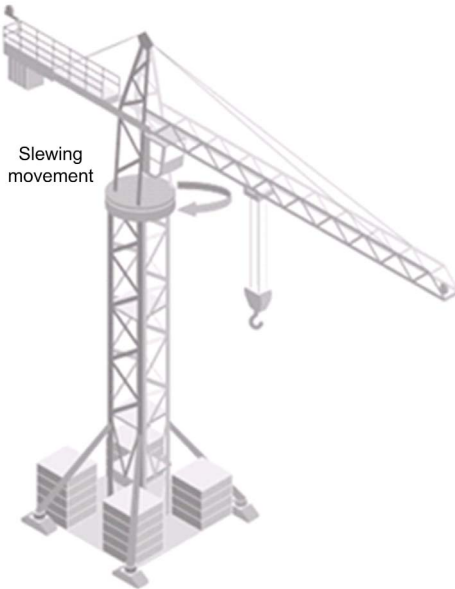
Functional View



Machine Overview

Machine View

The following figure shows a tower crane with jib where the function blocks can be used to achieve smooth slewing movement.



Section 22.2 Architecture

What Is in This Section?

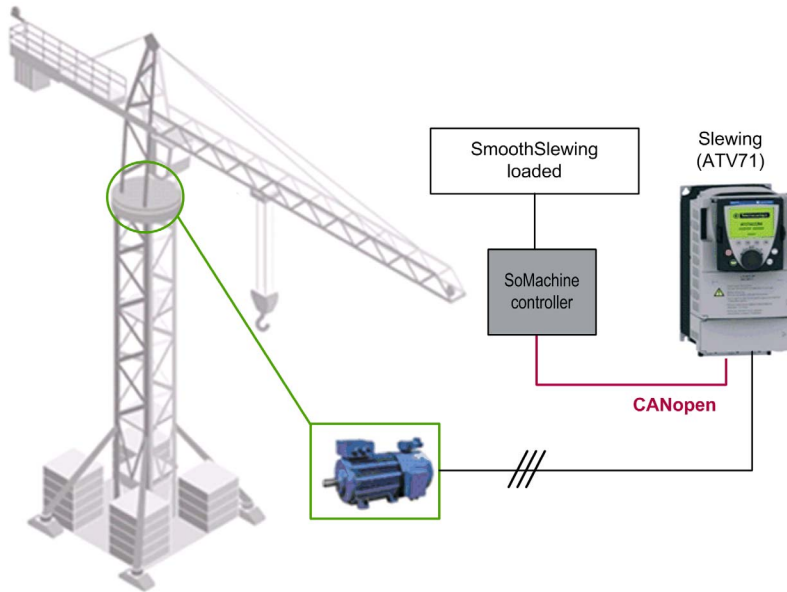
This section contains the following topics:

Topic	Page
Hardware Architecture	679
Software Architecture	680

Hardware Architecture

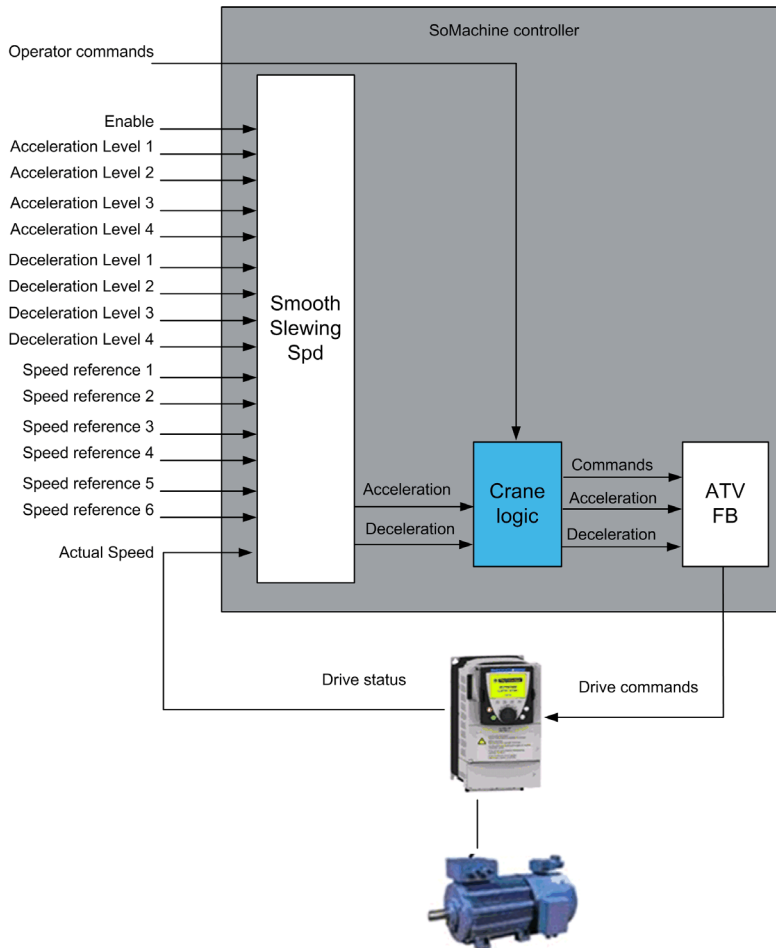
Hardware Architecture Overview

The Smooth slewing function blocks can be used with an ATV71 or ATV312 constant torque drive, for the slewing movement.

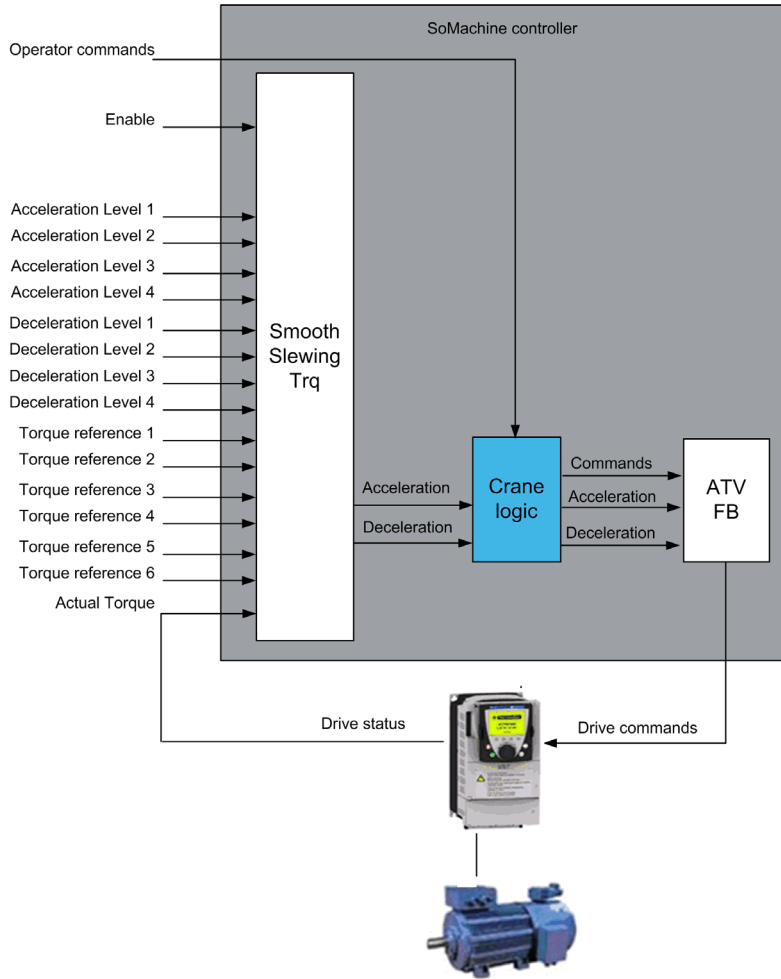


Software Architecture

DataFlow Overview - Speed Based



DataFlow Overview - Torque Based



Section 22.3

Function Block Description - SmoothSlewingSpd

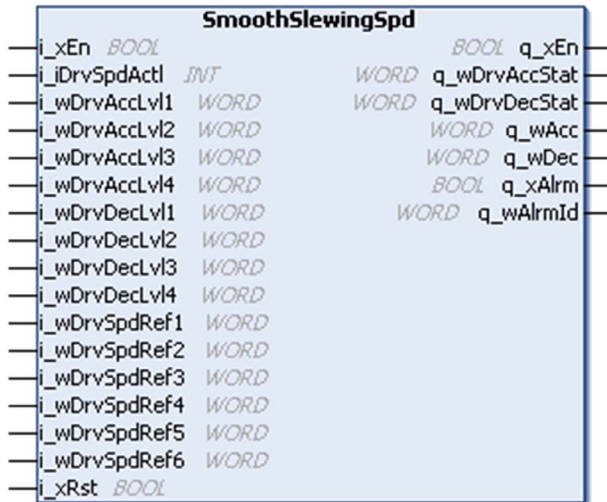
What Is in This Section?

This section contains the following topics:

Topic	Page
SmoothSlewingSpd Function Block	683
Input Pin Description	686
Output Pin Description	687

SmoothSlewingSpd Function Block

Pin Diagram

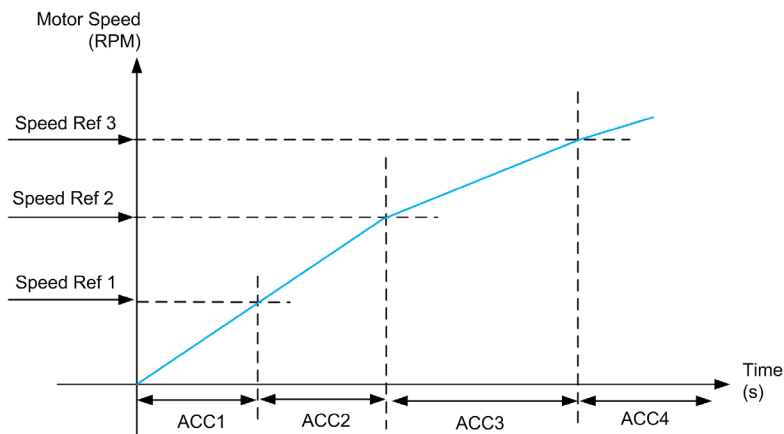


Function Block Description

This function block changes the acceleration and deceleration time parameters based on the actual speed of the motor.

Acceleration Parameter

During acceleration, the acceleration parameter is selected from 4 pre-defined acceleration values (ACC1, ACC2, ACC3 and ACC4) as the actual speed reaches the threshold (speed1, speed2 and speed3 respectively).



Comparing Actual Speed with Defined Speed	Acceleration Value
$i_iDrvSpdAct1 \geq i_wDrvSpdRef3$	$i_wDrvAccLvl4$
$i_iDrvSpdAct1 \geq i_wDrvSpdRef2$	$i_wDrvAccLvl3$
$i_iDrvSpdAct1 \geq i_wDrvSpdRef1$	$i_wDrvAccLvl2$
None of the above condition holds good then	$i_wDrvAccLvl1$

Example:

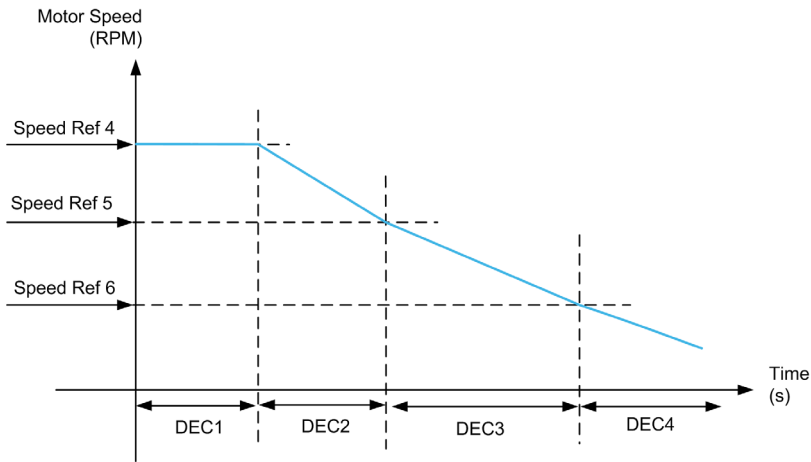
If Spd1 = 500 RPM, Spd2 = 1000 RPM, Spd3 = 1500 RPM (that is, Spd1 < Spd2 < Spd3) then:

If actual speed is between...	Then acceleration is...
0 and 500 RPM,	ACC1
500 and 1000 RPM,	ACC2
1000 and 1500 RPM,	ACC3
1500 and HSP RPM,	ACC4

NOTE: You must set speed levels such that **Spd1 < Spd2 < Spd3** to give a four slope acceleration curve. If less than four levels of acceleration are required, the values for ACC2, ACC3 and ACC4 should be set to the same value.

Deceleration Parameter

During deceleration, the deceleration parameter is selected from four pre-defined deceleration values (DEC1, DEC2, DEC3 and DEC4), as the actual speed reaches threshold (speed4, speed5 and speed6 respectively).



Comparing Actual Speed with Defined Speed	Deceleration Value
$i_iDrvSpdAct1 \geq i_wDrvSpdRef4$	$i_wDrvDecLv11$
$i_iDrvSpdAct1 \geq i_wDrvSpdRef5$	$i_wDrvDecLv12$
$i_iDrvSpdAct1 \geq i_wDrvSpdRef6$	$i_wDrvDecLv13$
None of the above condition holds good then	$i_wDrvDecLv14$

Example:

If Spd4 = 1500 RPM, Spd5 = 1000 RPM, and Spd6 = 500 RPM (that is, Spd4 > Spd5 > Spd6) then:

If actual speed is between...	Then deceleration is...
1500 and HSP RPM,	DEC1
1000 and 1500 RPM,	DEC2
500 and 1000 RPM,	DEC3
0 and 500 RPM,	DEC4

NOTE: You must set Speed levels such that $Spd4 > Spd5 > Spd6$ to get a three slope deceleration curve. If less than four levels of deceleration are required, the values for DEC2, DEC3 and DEC4 should be set the same.

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below this table.
i_iDrvSpdAct1	INT	Actual speed input of drive Range: -32768...32767 RPM
i_wDrvAccLvl1...4	WORD	Drive acceleration for ramp Level 1...4 Range: 1...9999 Scaling/Unit: 0.1 s
i_wDrvDecLvl1...4	WORD	Drive deceleration for ramp Level 1...4 Range: 1...9999 Scaling/Unit: 0.1 s
i_wDrvSpdRef1...6	WORD	Drive pre-select speed 1...6 Range: 0...6000 RPM
i_xRst	BOOL	Resets detected alarms TRUE: Active FALSE: Inactive

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of SmoothSlewingSpd are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wAcc	WORD	i_wdrvAccLvl4
q_wDrvAccStat	WORD	4
q_wDec	WORD	i_wdrvDecLvl4
q_wDrvDecStat	WORD	4
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_wDrvAccStat	WORD	Status of selected acceleration ramp level Range: 1...4
q_wDrvDecStat	WORD	Status of selected deceleration ramp level Range: 1...4
q_wAcc	WORD	Ramp time from 0 to nominal speed Target drive must be set to 0.1 accuracy Range 1...9999 Scaling/Unit: 0.1 s
q_wDec	WORD	Ramp time from nominal speed to 0 Target drive must be set to 0.1 accuracy Range 1...9999 Scaling/Unit: 0.1 s
q_xAlrm	BOOL	TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (see page 687). Range: 0...3

Notifications

Alarm Bit q_wAlrmId	Description
0	Acceleration speed levels are not set in ascending order (i_wDrvSpdRef1, 2, 3)
1	Deceleration speed levels are not set in descending order (i_wDrvSpdRef4, 5, 6)

NOTE:

On detection of alarm:

- q_wDrvAcc is set to the i_wDrvAccLvl4.
- q_wDrvDec is set to the i_wDrvDecLvl4.

Section 22.4

Function Block Description - SmoothSlewingTrq

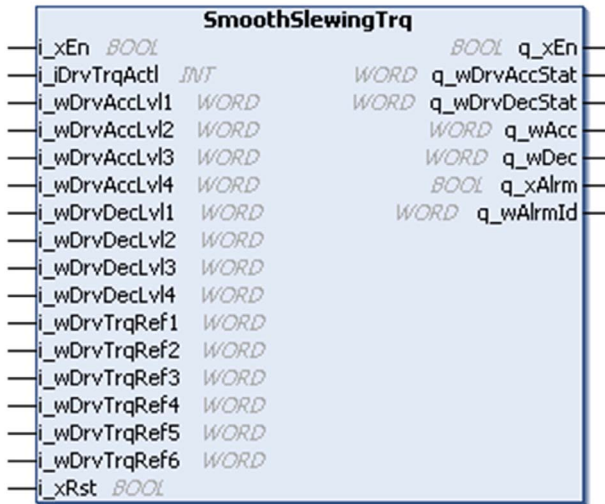
What Is in This Section?

This section contains the following topics:

Topic	Page
SmoothSlewingTrq Function Block	689
Input Pin Description	692
Output Pin Description	694

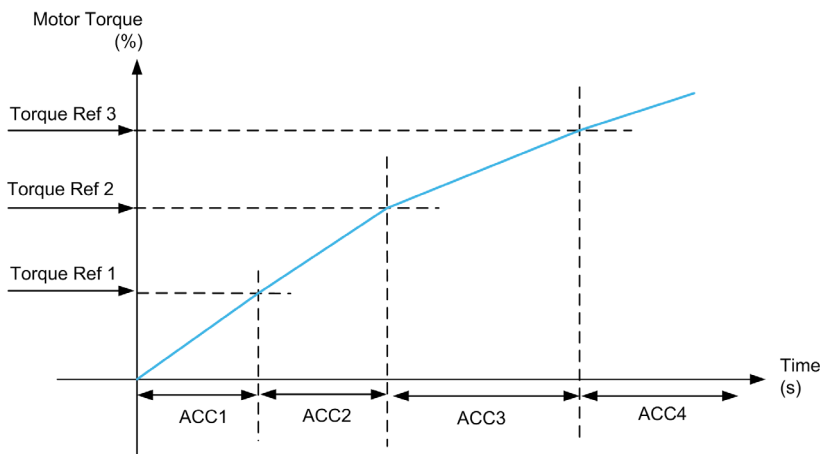
SmoothSlewingTrq Function Block

Pin Diagram



Acceleration Parameter

During acceleration, the acceleration parameter is selected from 4 pre-defined acceleration values (ACC1, ACC2, ACC3 and ACC4) as the actual torque reaches the threshold (Trq1, Trq2 and Trq3 respectively).



Comparing Actual Torque with Defined Torque	Acceleration Value
$i_iDrvTrqAct1 \geq i_wDrvTrqRef3$	$i_wDrvAccLv14$
$i_iDrvTrqAct1 \geq i_wDrvTrqRef2$	$i_wDrvAccLv13$
$i_iDrvTrqAct1 \geq i_wDrvTrqRef1$	$i_wDrvAccLv12$
None of the above condition holds good then	$i_wDrvAccLv11$

Example:

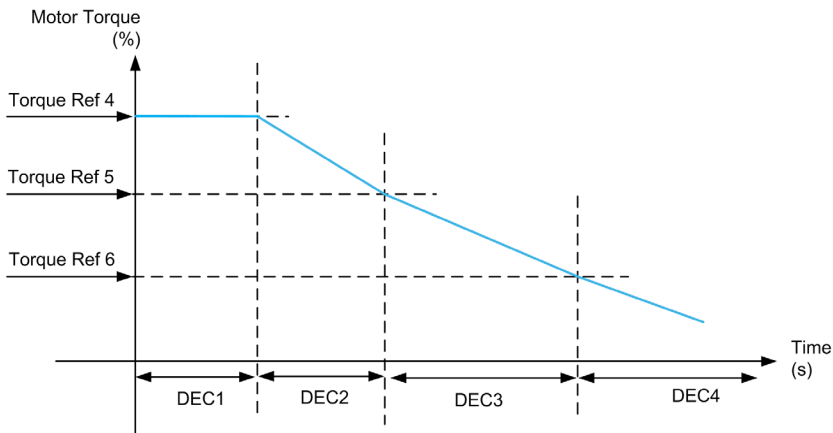
If $Trq1 = 10\%$, $Trq2 = 20\%$, $Trq3 = 30\%$ (that is, $Trq1 < Trq2 < Trq3$) then,

If actual torque is between...	Then acceleration is...
0 and 10%,	ACC1
10 and 20%,	ACC2
20 and 30%,	ACC3
30 and MAX (300)%,	ACC4

NOTE: You must set torque levels such that $Trq1 < Trq2 < Trq3$ to give a four slope acceleration curve. If less than 4 levels of acceleration are required, the values for ACC2, ACC3 and ACC4 should be set to the same value.

Deceleration Parameter

During deceleration, the deceleration parameter is selected from 4 pre-defined deceleration values (DEC1, DEC2, DEC3 and DEC4), as the actual torque reaches threshold $Trq4$, $Trq5$ and $Trq6$ respectively.



Comparing Actual Torque with Defined Torque	Deceleration Value
$i_iDrvTrqAct1 \geq i_wDrvTrqRef4$	$i_wDrvDecLv11$
$i_iDrvTrqAct1 \geq i_wDrvTrqRef5$	$i_wDrvDecLv12$
$i_iDrvTrqAct1 \geq i_wDrvTrqRef6$	$i_wDrvDecLv13$
None of the above condition holds good then	$i_wDrvDecLv14$

Example:

If $Trq4 = 30\%$, $Trq5 = 20\%$, and $Trq6 = 10\%$ (that is, $Trq4 > Trq5 > Trq6$) then,

If actual speed is between...	Then deceleration is...
30 and Max (300)%,	DEC1
20 and 30%,	DEC2
10 and 20%,	DEC3
0 and 10%,	DEC4

NOTE: You must to set torque levels such that $Trq4 > Trq5 > Trq6$ to get a three slope deceleration curve. If less than four levels of deceleration are required, the values for DEC2, DEC3 and DEC4 should be set the same.

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description below the table.
i_iDrvTrqAct1	INT	Actual torque on the motor. Range: -32768...32767 Scaling/Unit: 0.1 %
i_wDrvAccLvl1...4	WORD	Drive acceleration for ramp Level 1...4 Range: 1...9999 Scaling/Unit: 0.1 s
i_wDrvDecLvl1...4	WORD	Drive deceleration for ramp Level 1...4 Range: 1...9999 Scaling/Unit: 0.1 s
i_wDrvTrqRef1...6	WORD	Drive pre-select torque1...6 Range: 0...3000 Scaling/Unit: 0.1 %
i_xRst	BOOL	Resets detected alarms TRUE: Active FALSE: Inactive

i_xEn

By setting the `i_xEn` input to `FALSE` the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of `SmoothSlewingTrq` are given in the table below.

Output	Data Type	Fallback State
<code>q_xEn</code>	BOOL	FALSE
<code>q_wAcc</code>	WORD	<code>i_wdrvAccLv14</code>
<code>q_wDrvAccStat</code>	WORD	4
<code>q_wDec</code>	WORD	<code>i_wdrvDecLv14</code>
<code>q_wDrvDecStat</code>	WORD	4
<code>q_xAlrm</code>	BOOL	FALSE
<code>q_wAlrmId</code>	WORD	0

Outputs that are not listed will retain their current values.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_wDrvAccStat	WORD	Status of selected acceleration ramp level Range: 1...4
q_wDrvDecStat	WORD	Status of selected deceleration ramp level Range: 1...4
q_wAcc	WORD	Ramp time from 0 to nominal speed Target drive must be set to 0.1 accuracy Range 1...9999 Scaling/Unit: 0.1 s
q_wDec	WORD	Ramp time from nominal speed to 0 Target drive must be set to 0.1 accuracy Range 1...9999 Scaling/Unit: 0.1 s
q_xAlrm	BOOL	TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (see page 694). Range: 0...3

Notifications

Alarm Bit q_wAlrmId	Description
0	Acceleration torque levels are not set in ascending order (i_wDrvTrqRef1, 2, 3)
1	Deceleration torque levels are not set in descending order (i_wDrvTrqRef4, 5, 6)

NOTE:

On detection of alarm:

- q_wDrvAcc is set to the i_wDrvAccLvl4.
- q_wDrvDec is set to the i_wDrvDecLvl4.
- It is mandatory to have acceleration parameter i_wDrvAccLvl4 and deceleration parameter i_wDrvDecLvl4

Section 22.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
Motor torque information read from ATV71 has an incorrect sign	Wrong input for motor torque was used.	Do not use Motor torque or Motor Torque Scope as an input for the actual torque inputs of the drive. Use 4 Quadrant Torque instead. Because 4 Quadrant Torque cannot be read by CANopen directly, use AO1 of the drive to be read and configure 4 Quadrant Torque to the AO1 output. Verify the scaling of AO1, it should be set to 4 to 20 mA. A value of 12000 represents zero torque, 4000 is -300% torque and 20000 is 300% torque.

Part XV

Speed Optimization and Rope Slack

Chapter 23

SpeedOptRopeSlack: Reduces Work Cycle Time and Avoid Start Up with Slack on the Rope

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
23.1	Functional and Machine Overview	700
23.2	Architecture	704
23.3	Function Block Description	708
23.4	Pin Description	716
23.5	Troubleshooting	723

Section 23.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	701
Machine Overview	703

Functional Overview

Functional Description

The Speed optimization and rope slack function is designed to:

- Optimize the utilization of power based on the actual torque of the drive.
- Correct any rope-slack commonly found at the start or end of any hoisting operation.

The function is designed to be used on ATV71 drives and is applicable to:

- Tower cranes
- Industrial cranes

Why Use the SpeedOptRopeSlack Function Block?

Speed optimization:

During the movement of a hoist, the load can range between the empty hook to maximum load, but the same nominal speed is always used to drive the crane. The Speed optimization function helps you to maintain an optimum working time and increased productivity.

Rope slack:

When the load or hook is on the ground, there could be slack at the rope. At this moment, a high speed movement could lead to a whiplash motion causing damage to the rope or the rope could jump out of the rope guides. The Rope slack function is used to avoid this.

This function block is intended to have significant influence on the physical movement of the crane and its load. The application of this function block requires accurate and correct input parameters in order to make its movement calculations valid and to avoid hazardous situations. If invalid or otherwise incorrect input information is provided by the application, the results may be undesirable.

WARNING

UNINTENDED EQUIPMENT OPERATION

Validate all function block input values before and while the function block is enabled.

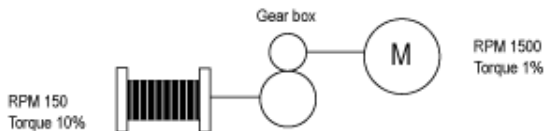
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Solution with the SpeedOptRopeSlack Function Block

The Speed optimization solution maintains operation at constant power consumption in order to reach a speed greater than the rated speed without exceeding the rated motor current. Constant power operation increases the system efficiency and helps to protect the motor by keeping the over-speed value in check based on the actual torque measured.

The Rope slack function is used to avoid extra rope being let out once the hook has touched the floor.

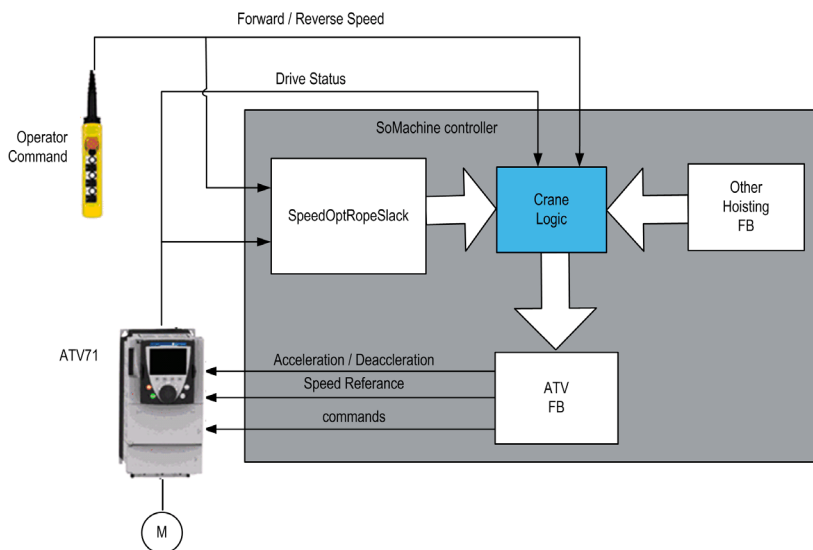
NOTE: The load on the motor at hook level must be at minimum 10% of the nominal load. Otherwise, the function can not work correctly due, at least in part, to the inefficiency of the gear box. If the load torque is below the minimum 10%, the further reduction by the gear box creates a scenario in which the ATV71 or ATV312 is not able to determine between generator and motor mode. The result is a non optimized movement with limited speed.



Design & Realization Constraints and Assumptions

When the motor moves in forward direction, the hoist moves up and when the motor moves in reverse direction, the hoist moves down.

Functional View

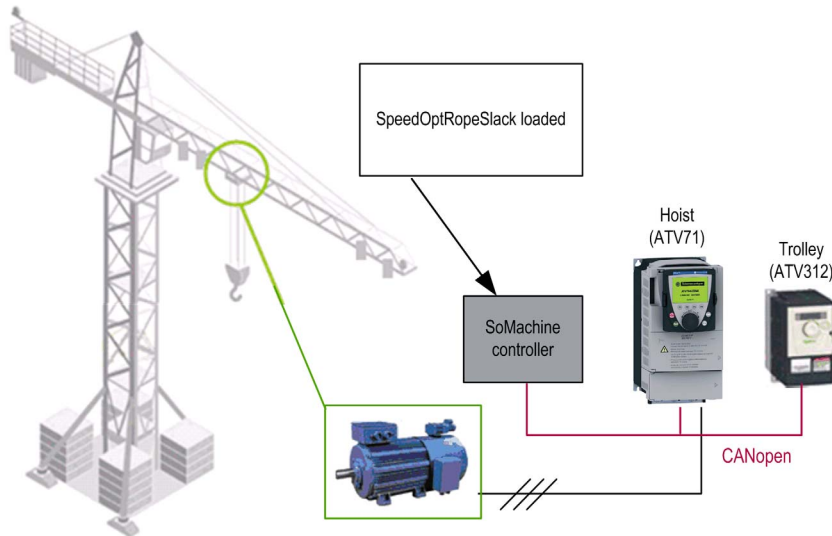


NOTE: The function Speed optimization and rope slack works with ATV312 or ATV71. In the graphic, only ATV71 is shown, but ATV312 could also be used.

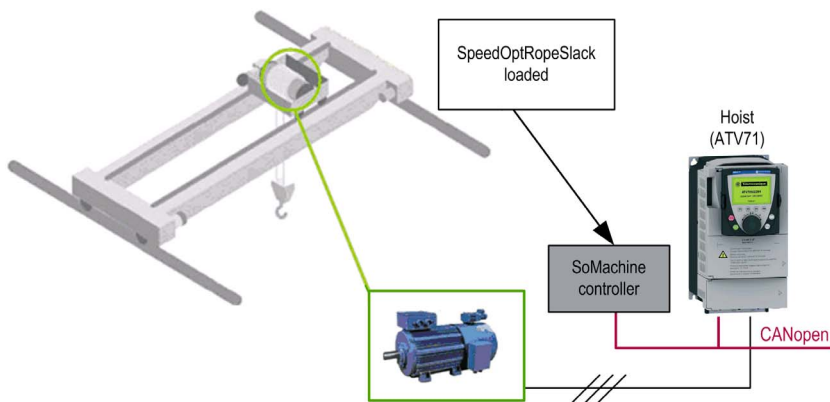
Machine Overview

Machine View

The following figure represents the machine view of tower crane using the Speed optimization and rope slack function.



The following figure represents the machine view of an industrial crane with Speed optimization and rope slack function.



NOTE: The function Speed optimization and rope slack works with ATV312 or ATV71. In the graphic, only ATV71 is shown, but ATV312 could also be used.

Section 23.2

Architecture

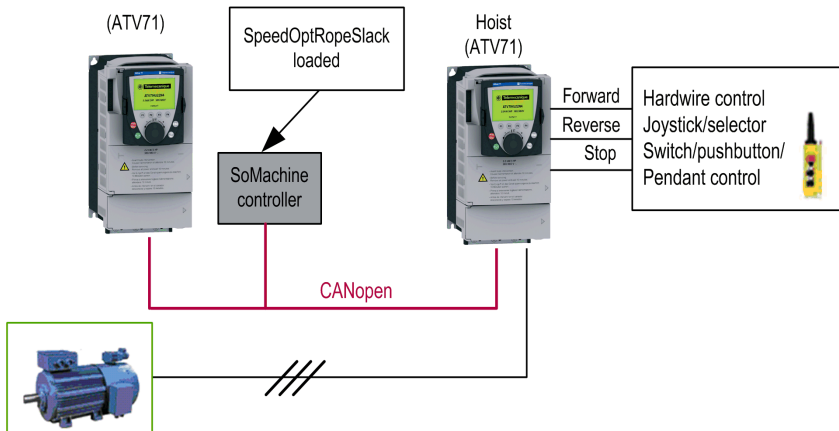
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	705
Software Architecture	706

Hardware Architecture

Hardware Architecture Overview



NOTE: The function Speed optimization and rope slack works with ATV312 or ATV71. In the graphic, only ATV71 is shown, but ATV312 could also be used.

Software Architecture

Software Architecture Overview

The Speed optimization function monitors the actual torque of the motor via drive. When the actual torque goes above the 70% of hook torque, the function block compares the speed reference with the nominal speed. When the speed reference is greater than or equal to the nominal speed of the drive, then the function block calculates the optimized speed.

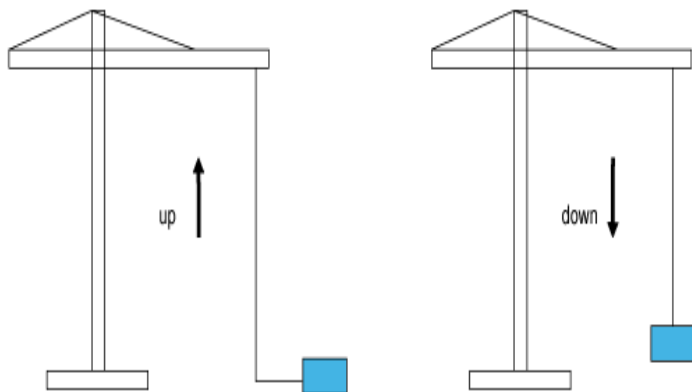
The optimized speed is calculated using dynamic torque when the hoist is moving up.

When the hoist is moving down, you have the following facility:

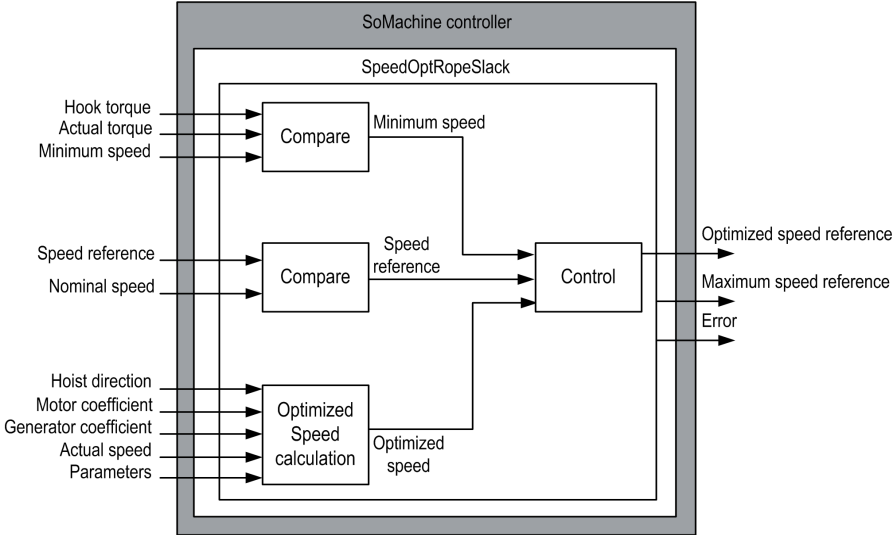
- Calculate the optimized speed only once while coming down.
- Calculate the optimized speed continuously.

The Rope slack function is used to avoid slack in the rope when the hoist is moved in a downward direction. If the actual torque drops below the user-defined hook-torque (for down movement) during any downward movement, any further movement is restricted to the minimum speed until the actual torque reaches the hook-torque. The speed reference is sent to the ATV drive which is connected to the motor and to the mechanical assembly.

The following figure represents the Rope slack function for up and down movement.



DataFlow Overview



Section 23.3

Function Block Description

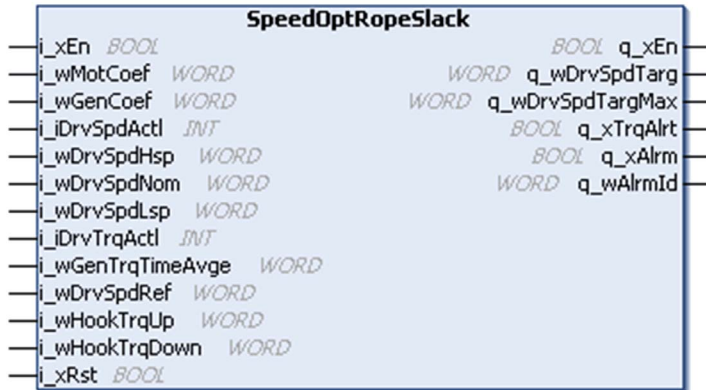
What Is in This Section?

This section contains the following topics:

Topic	Page
SpeedOptRopeSlack Function Block	709
Timing Chart	712

SpeedOptRopeSlack Function Block

Pin Diagram



Function Block Description - Speed Optimization

When the speed reference is greater than or equal to the nominal speed and the actual torque is above 70% of the hook torque, the function block calculates the optimized speed.

NOTE: Nominal torque is equal to 100% considered inside the function block.

Calculation of Optimized Speed During Motor Mode

Formula for calculating optimized speed during motor mode:

Optimized Speed = (Nominal speed in RPM) * (Nominal torque in percentage / (Measured torque in percentage) * (Motor coefficient in percentage)

Example:

During Motor operation

Nominal speed = 1500 RPM

Actual torque = 50%

Motor coefficient = 40%

Optimized speed (RPM) = 1500 * (100 / 50) * (40 / 100)

Optimized speed = 1200 RPM

Calculation of Optimized Speed During Generator Mode

Formula for calculating optimized speed during generator mode:

You have the facility to use the following two modes:

- **Dynamic torque calculation**

In this mode, the Optimized speed is calculated using the actual torque continuously.

Optimized Speed = (Nominal speed in RPM) * (Nominal torque in percentage/(Measured torque in percentage)) * (Generator coefficient in percentage)

Example:

During Generator operation

Nominal speed = 1500 RPM

Actual torque = 25%

Generator coefficient = 50%

Optimized speed (RPM) = $1500 * (100 / 25) * (50 / 100)$

Optimized = 3000 RPM

- **Averaged torque calculation**

In this mode, the actual torque is averaged over a user defined period of time. When the actual speed is greater than or equal to 90% of nominal speed, the optimized speed is calculated using this averaged torque.

This torque is recalculated when the drive is stopped or there is a change in direction of the hoist.

Example:

During Generator operation

Nominal speed = 1500 RPM

Measured averaged torque = 20%

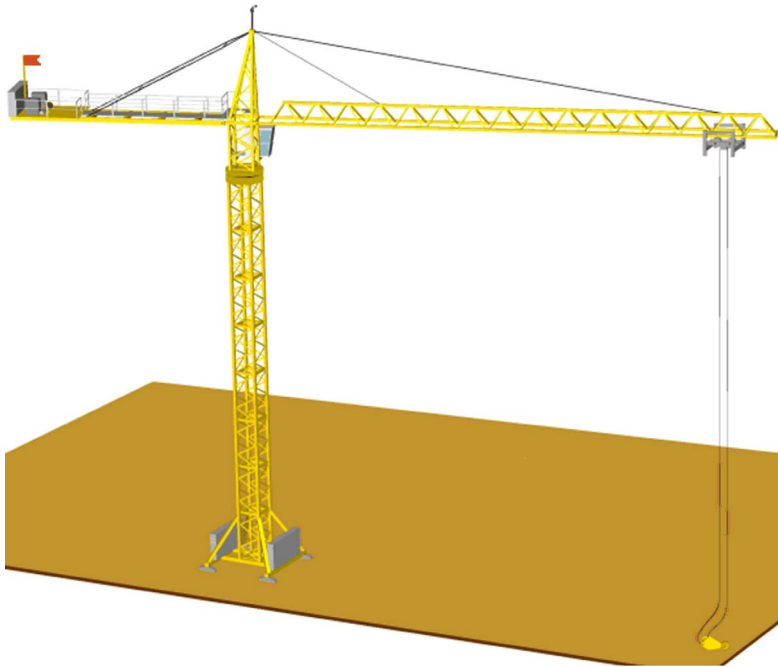
Generator coefficient = 50%

Optimized speed (RPM) = $1500 * (100/20) * (50/100)$

Optimized = 3750 RPM

Function Block Description - Rope Slack

The Rope slack function is used to keep track of the actual load on the drive or motor and helps to avoid any slack that may occur in the hoisting rope when the load is set on the ground.

**NOTE:**

The function block is defined with two distinct inputs for hook torque:

- Hook torque up
- Hook torque down

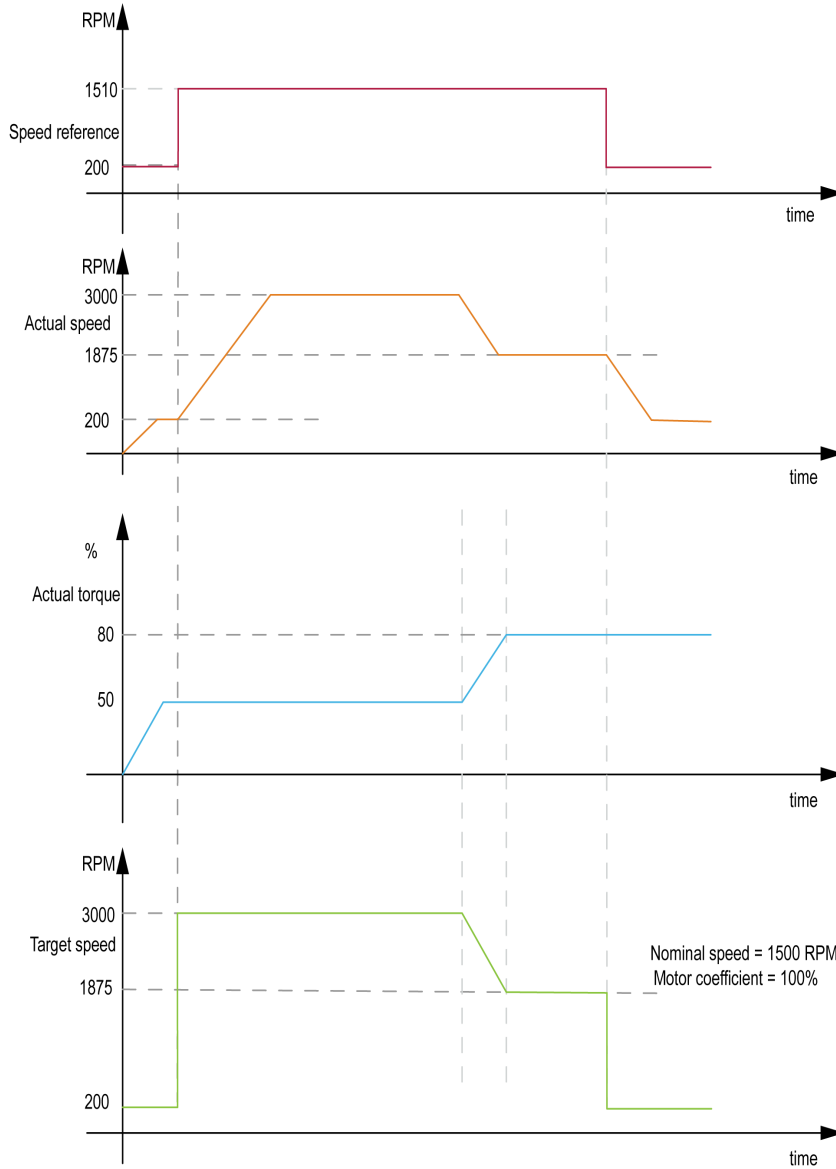
The hook torque up is considered when the hoist is moving up and hook torque down is considered when the hoist is moving down.

The Rope slack function is activated when the actual torque goes below or equal to 70% of the hook torque up value when the hoist is moving up.

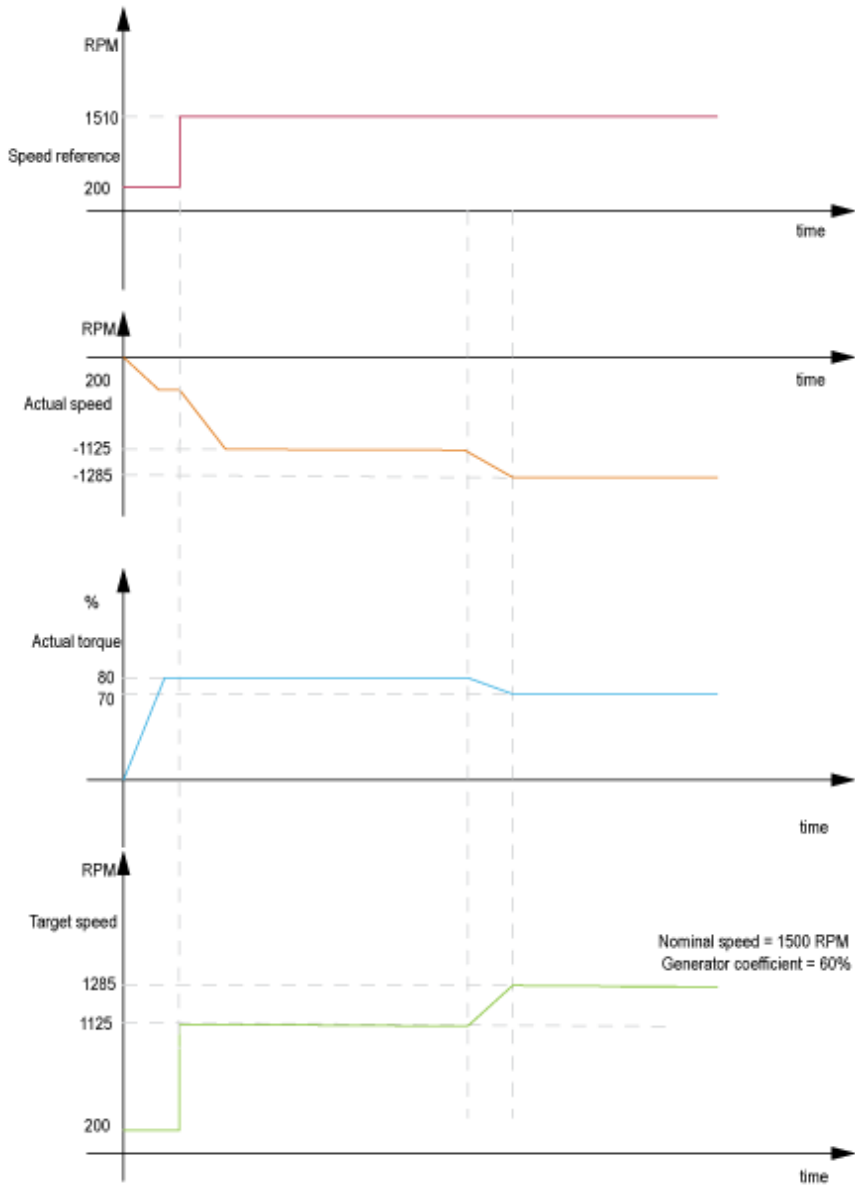
The Rope slack function is activated when the actual torque goes below or equal to 70% of the hook torque down value when the hoist is moving down.

Timing Chart

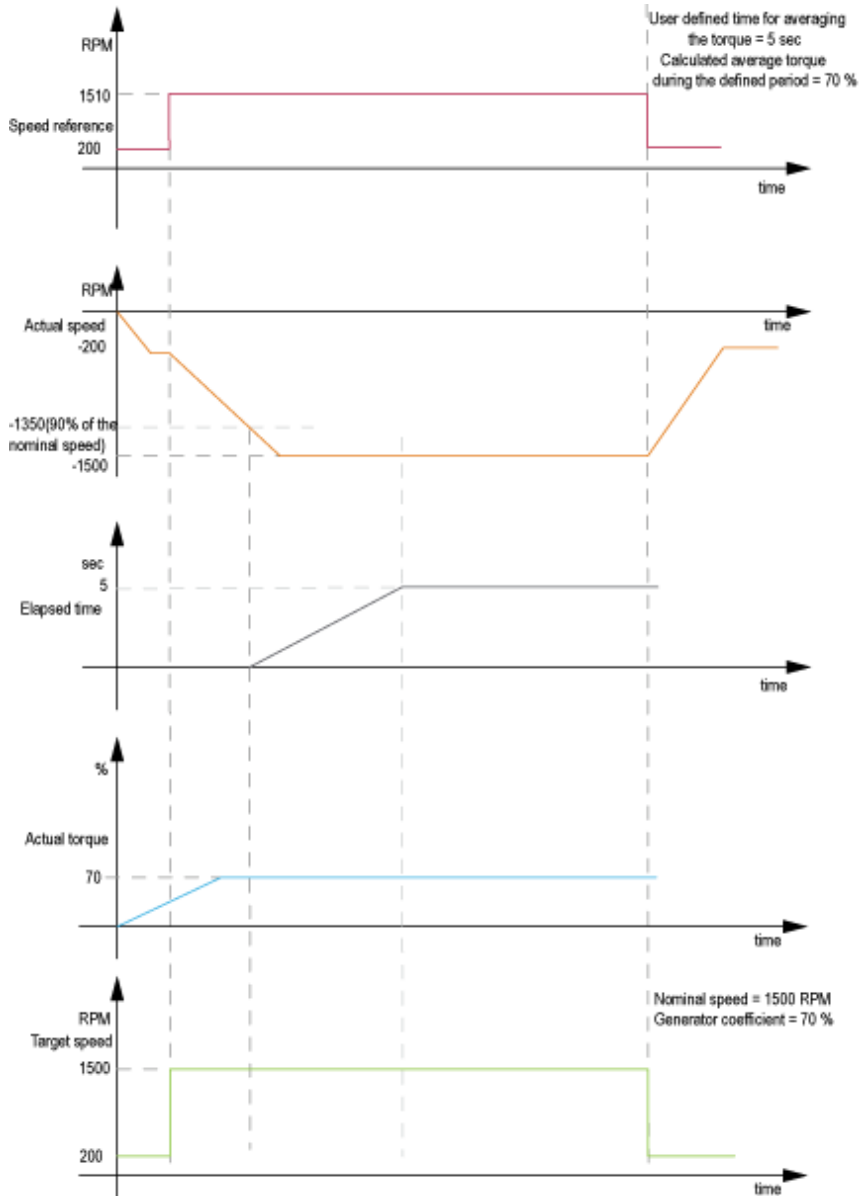
Hoist Moving Up



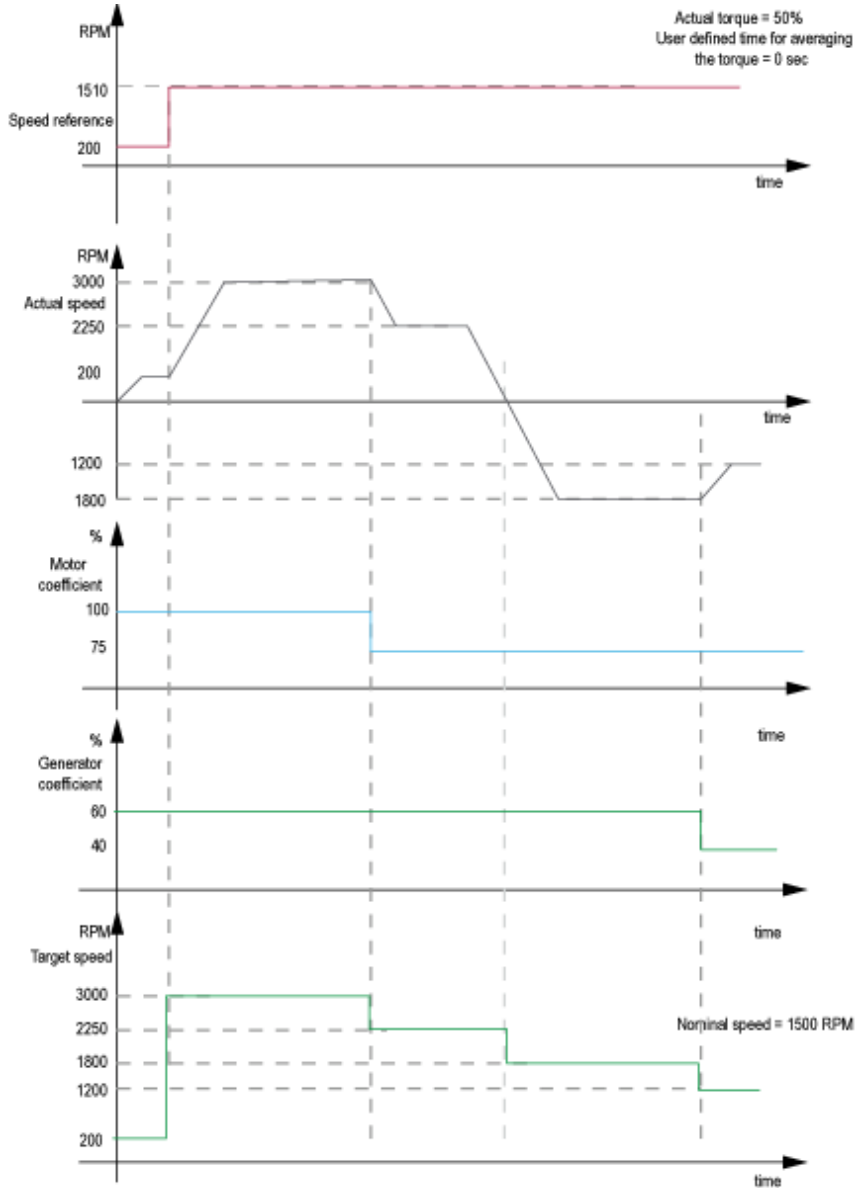
Hoist Moving Down and Dynamic Measurement Activated



Hoist Moving Down and Average Torque Measurement Activated



Hoist Moving Down-Response of Speed on the Change of Motor and Generator Coefficient



NOTE: The timing diagram does not depict the real performance of the hoist.

Section 23.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	717
Output Pin Description	721

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (see page 718) of i_xEn.
i_wMotCoef	WORD	Scaling factor for the limitation in motor mode and is defined by user/programmer. Used to fine tune the function based on the loading characteristics of the hoist. Range: 0...150% Factory setting: 100%
i_wGenCoef	WORD	Scaling factor for the limitation in the generator mode and is defined by you. Used to fine tune the function based on the loading characteristics of the hoist. Range: 0...150% Factory setting: 100%
i_iDrvSpdAct1	INT	Actual speed of the motor received through CANopen. Gives information about whether the hoist is moving up or down or stopped. Also, used as the feedback to check whether the motor has reached 90% of nominal speed to average the torque. Range: -6000...6000 RPM
i_wDrvSpdHsp	WORD	Maximum allowed speed for the drive (HSP). Range: 0...6000 RPM Factory setting: 0 RPM
i_wDrvSpdNom	WORD	Nominal speed of the drive. Range: 0...6000 RPM Factory setting: 1500 RPM
i_wDrvSpdLsp	WORD	Minimum allowed speed for the drive (LSP). Range: 0...6000 RPM Factory setting: 0 RPM
i_iDrvTrqAct1	INT	Actual torque of the motor received from the drive through CANopen. Refer to detailed description (see page 719) of i_iDrvTrqAct1. Range: -3000...3000 Scaling/Unit: 0.1 %

Input	Data Type	Description
i_wGenTrqTimeAvge	WORD	Time for averaging the torque during the hoist down movement. Dynamic calculation of optimized speed enabled. Refer to detailed description (see page 719) of i_wGenTrqTimeAvge. Range: 0...100 Scaling/Unit: 0.1 s
i_wDrvSpdRef	WORD	Speed reference provided by you. Refer to detailed description (see page 719) of i_wDrvSpdRef. Range: 0...6000 RPM
i_wHookTrqUp	WORD	No-load torque when the hoist is moving up. Refer to detailed description (see page 720) of i_wHookTrqUp. Range: 0...1000 Scaling/Unit: 0.1 %
i_wHookTrqDown	WORD	No-load torque when the hoist is moving down. Refer to detailed description (see page 720) of i_wHookTrqDown. Range: 0...1000 Scaling/Unit: 0.1 %
i_xRst	BOOL	On a rising edge, attempts to clear all alarms. If alarms stay active despite rising edge of this input, the cause of the alarm is still present.

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of SpeedOptRopeSlack are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_wDrvSpdTarg	WORD	i_wDrvSpdRef
q_wDrvSpdTargMax	WORD	i_wDrvSpdNom
q_xTrqAlrt	BOOL	FALSE
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

iDrvTrqActl

Do not use Motor Torque or Motor Torque Scope as an input for the actual torque inputs of the drive.

Use 4 Quadrant Torque instead. Because 4 Quadrant Torque cannot be read by CANopen directly, use AO1 of the drive to be read and configure 4 Quadrant Torque to the AO1 output.

Verify the scaling of AO1, it should be set to 4...20 mA. A value of 12 000 represents zero (0) torque, 4 000 is -300% torque and 20 000 is 300% torque.

i_wGenTrqTimeAvge

Average torque calculation is disabled when the value at the pin `i_wGenTrqTimeAvge` is set to zero. In this case the optimized speed for downwards movement is calculated continuously. The average torque is recalculated when the drive is stopped and started again or a change in the direction is detected. Actual torque is averaged only when the actual speed is equal to or greater than the 90% of the nominal speed.

i_wDrvSpdRef

When the speed reference (`i_wDrvSpdRef`) is less than nominal speed (`i_wDrvSpdNom`) of the drive, then

$$q_wDrvSpdTarg = i_wDrvSpdRef$$
$$q_wSpdTargMax = i_wDrvSpdNom$$

When the speed reference (`i_wDrvSpdRef`) is greater than or equal to Nominal speed `i_wDrvSpdNom`, optimized speed is calculated.

i_wHookTrqUp

You are advised to run the drive without a load on the hook and record the percentage (%) load from the Monitoring Menu on the drive or by monitoring the corresponding CANopen parameters.

When the Hoist is moving up and if the actual torque (*i_iDrvTrqAct1*) is less than or equal to 70% of the hook torque (*i_wHookTrqUp*), then the target speed (*q_wDrvSpdTarg*) is set to minimum speed reference (*i_wDrvSpdLsp*) and (*q_wSpdTargMax*) is set to (*i_wDrvSpdNom*).

NOTE:

Following are the points to be respected while using this input:

- The load on the motor at hook level must be at minimum 10% of the nominal load. Otherwise, the function cannot work correctly due to the efficiency of the gear box. If the load torque is below the minimum 10%, the further reduction by the gear box creates the scenario in which the ATV71 is not able to determine between generator and motor mode. The result is a non-optimized movement with limited speed.
- When the *i_wHookTrqUp* input is set to zero, Rope slack detection and correction while moving up is disabled.
- When the *i_wHookTrqDown* input is set to zero, Rope slack detection and correction while moving down is disabled.

i_wHookTrqDown

You are advised to run the drive without a load on the hook and record the percentage (%) load from the Monitoring menu on the drive or by monitoring the corresponding CANopen parameters.

When the Hoist is moving down and if the actual torque (*i_iDrvTrqAct1*) is less than or equal to 70% of the hook torque (*i_wHookTrqDown*), then the target speed (*q_wDrvSpdTarg*) is set to minimum speed reference (*i_wDrvSpdLsp*) and (*q_wSpdTargMax*) is set to (*i_wDrvSpdNom*).

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_wDrvSpdTarg	WORD	Calculated reference for the drive. Normally transferred to the drive through CANopen. Range: 0...6000 RPM
q_wDrvSpdTargMax	WORD	Maximum speed for the Hoisting Position Synchronization function block so that the speed calculation of the function block does not run out of the maximum speed reference limit. Refer to detailed description (see page 721) of q_wDrvSpdTargMax. Range: 0...6000 RPM
q_xTrqAlrt	BOOL	Alert when the measured torque is less than hook torque and that speed is minimized. TRUE: Alert FALSE: No alert
q_xAlrm	BOOL	Alarm detection indicator TRUE: Alarm detected FALSE: No alarm detected.
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (see page 722) Range: 0...3

q_wDrvSpdTargMax

The values of this input are calculated as follows:

$q_wDrvSpdTarg = i_wDrvSpdNom$ if $i_wDrvSpdRef < i_wDrvSpdRef < i_wDrvSpdNom$ (normal mode) or

When $i_iTrqact \leq 70\%$ of $i_wHookTrqUp$ or $i_wHookTrqUp$, $q_wDrvSpdTargMax =$ optimized speed if $i_wDrvSpdRef \geq i_wDrvSpdNom$ (speed optimized mode)

Notifications

Bit Number	Description
0	i_wDrvSpdLsp is greater than or equal to i_wDrvSpdHsp
1	i_wDrvSpdNom is greater than i_wDrvSpdHsp

NOTE: On detection of alarm, the target speed ($q_wDrvSpdTarg$) and maximum speed reference ($q_wMaxSpdTarg$) are set to zero.

Section 23.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
Movement is limited in terms of speed, but not optimized	Motor load on hook level is less than 10% of nominal load	Details for (<i>see page 720</i>) <code>i_wHookTrqUp</code>
Rope slack detection while moving up is not working	Parameter <code>i_wHookTrqUp</code> is set to zero	Check parameter setting for <code>i_wHookTrqUp</code> according to the explanations (<i>see page 720</i>) given for <code>i_wHookTrqUp</code> .
Rope slack detection while moving down is not working	Parameter <code>i_wHookTrqDown</code> is set to zero	Check parameter setting for <code>i_wHookTrqDown</code> according to the explanations (<i>see page 720</i>) given for <code>i_wHookTrqDown</code> .
Motor torque information read from ATV71 has an incorrect sign	Wrong input for motor torque was used.	Do not use Motor torque or Motor Torque Scope as an input for the actual torque inputs of the drive. Use 4 Quadrant Torque instead. Because 4 Quadrant Torque cannot be read by CANopen directly, use AO1 of the drive to be read and configure 4 Quadrant Torque to the AO1 output. Verify the scaling of AO1, it should be set to 4 to 20 mA. A value of 12000 represents zero torque, 4000 is -300% torque and 20000 is 300% torque.
The actual motor speed does not correspond to speed reference.	Preset speeds configuration interferes with speed reference from controller.	Disable preset speeds in application functions setting of the drive.
The actual motor speed does not correspond to speed reference.	Limit switch configuration interferes with speed reference from controller.	Disable limit switch function in application functions setting of the drive.

Issue	Cause	Solution
The actual motor speed or read actual speed do not correspond to speed reference.	PDO mapping is inconsistent.	Make sure that both speed reference and actual speed values have the same unit (Hz or RPM).
The brake is excessively stressed when starting hoisting movement in downwards direction.	Brake impulse is not active.	Activate brake impulse in Application functions – Brake logic of ATV71.

Part XVI

Speed Select

Chapter 24

SpeedSelect: Provides the Option to Choose between 8 Predefined and one Analog Speed

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
24.1	Functional Overview	728
24.2	Architecture	729
24.3	Function Block Description	730
24.4	Pin Description	731
24.5	Troubleshooting	735

Section 24.1

Functional Overview

Functional Overview

Functional Description

The `SpeedSelect` function block allows you to control the speed inside the application that is communicated to the ATV71 or ATV312 drives using an analog input channel or selector switch. You have the flexibility to select 8 pre-selectable speeds using the selector switch.

Why Use the `SpeedSelect` Function Block?

The `SpeedSelect` function block is an adapted multiplexing algorithm compiled into a single function block. It was designed to facilitate user-logic development associated with the pre-selectable speed functions found on Altivar variable speed drives. Implementing the `SpeedSelect` function block may also ease comprehension of the user-logic while monitoring and troubleshooting the application.

Design & Realization Constraints and Assumptions

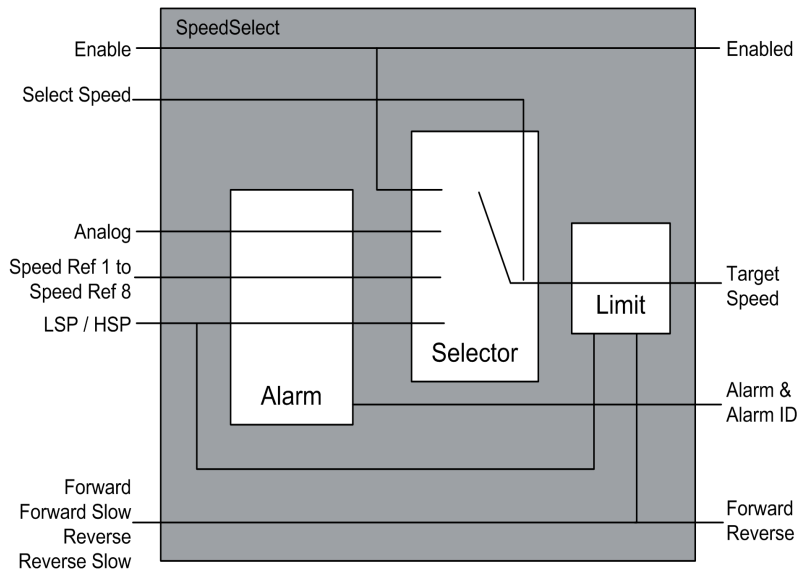
When the motor moves in forward direction, the hoist moves up and when the motor moves in reverse direction, the hoist moves down.

Section 24.2

Architecture

Software Architecture

DataFlow Overview

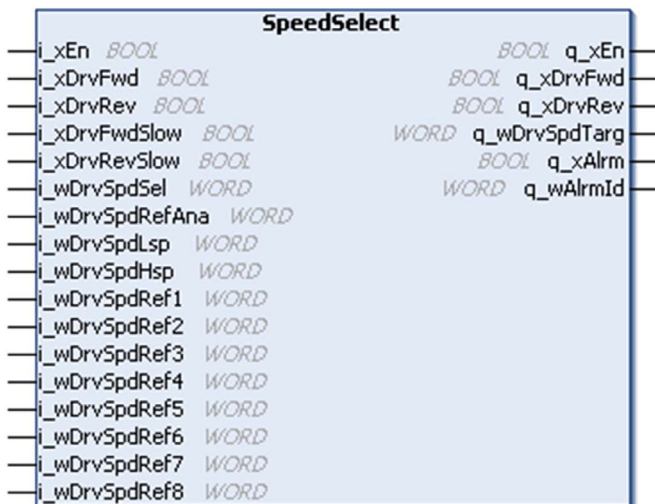


Section 24.3

Function Block Description

SpeedSelect Function Block

Pin Diagram



Function Block Description

The `SpeedSelect` function block allows you to control the speed inside the application that will be communicated to the ATV71 or ATV312 drives using an analog input channel or selector switch. You have the flexibility to select 8 pre-selectable speeds using the selector switch.

Section 24.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	732
Output Pin Description	734

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (<i>see page 733</i>) of i_xEn.
i_xDrvFwd	BOOL	Drive forward run command. TRUE: Forward FALSE: Not forward
i_xDrvRev	BOOL	Drive reverse run command. TRUE: Reverse FALSE: Not reverse
i_xDrvFwdSlow	BOOL	Drive forward slow run command. The activation of this input causes the output speed reference (q_wDrvSpdTarg) set to low speed (LSP). This usually comes from the corresponding Limit Switch Management function block outputs if present. TRUE: Forward slow FALSE: Not forward slow
i_xDrvRevSlow	BOOL	Drive reverse slow run command. The activation of this input causes the output speed reference (q_wDrvSpdTarg) set to low speed (LSP). This usually comes from the corresponding Limit Switch Management function block outputs if present. TRUE: Reverse slow FALSE: Not reverse slow
i_wDrvSpdSel	WORD	Used to set the output reference speed (q_wDrvSpdTarg) to one of the eight pre-selected speed references or analog speed reference Range: 0...8 Default: 1
i_wDrvSpdRefAna	WORD	Analog speed reference. Refer to detailed description (<i>see page 733</i>) of i_wDrvSpdRefAna. Range: LSP...HSP RPM
i_wDrvSpdLsp	WORD	Drive low speed (LSP). This LSP input represents the target speed output of the function block, if working in the slow zones. That means i_xDrvFwdSlow or i_xDrvRevSlow is TRUE. This parameter needs to correspond with the LSP setting of the connected drive. Range: 0...6000 RPM

Input	Data Type	Description
i_wDrvSpdHsp	WORD	Drive high speed (HSP) Range: 0...6000 RPM
i_wDrvSpdRef1...8	WORD	Drive pre-select speed 1...8 Range: LSP...HSP RPM

NOTE:

- The Forward and Reverse command are allowed to pass through, even in the presence of a detected alarm.
- The input Forward Slow (i_xDrvFwdSlow) and Reverse Slow (i_xDrvRevSlow) has the highest priority over the Speed Select input (i_xDrvSpdSel).
- When the Forward Slow or Reverse Slow input is activated, the output Speed Reference is set to i_wDrvSpdLsp.

i_xEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of SpeedSelect are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_xDrvFwd	BOOL	FALSE
q_xDrvRev	BOOL	FALSE
q_xDrvSpdTarg	WORD	0
q_xAlrm	BOOL	FALSE
q_xAlrmId	WORD	0

Outputs that are not listed will retain their current values.

i_wDrvSpdRefAna

This input pin allows you to control the speed of the drive using an analog input channel. It is the speed reference from analog input device like a joystick. i_wDrvSpdSel input must be set to zero to select analog input as reference.

NOTE: You should scale the analog reference input to RPM units. The ScaleInput function block can be used for this purpose.

Output Pin Description

Output Pin Description

Output	Data Type	Data Type
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_xDrvFwd	BOOL	Drive forward run command. TRUE: Forward FALSE: Not forward
q_xDrvRev	BOOL	Drive reverse run command. TRUE: Reverse FALSE: Not reverse
q_wDrvSpdTarg	WORD	Drive target speed Range: 0...6000 RPM
q_xAlrm	BOOL	Alarm detection Indicator TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected alarm identification. Refer to Notifications (see page 734). Range: 0...15

Notifications

Bit Number	Description
0	i_DrvSpdHsp is less than or equal to i_wDrvSpdLsp.
1	One of the pre-selectable speeds (1-8) is lower than i_wDrvSpdLsp.
2	One of the pre-selectable speeds (1-8) is higher than i_wDrvSpdHsp.

Section 24.5

Troubleshooting

Troubleshooting

Troubleshooting

Issue	Cause	Solution
The actual motor speed does not correspond to speed reference.	Preset speeds configuration interferes with speed reference from controller.	Disable preset speeds in application functions setting of the drive.
The actual motor speed does not correspond to speed reference.	Limit switch configuration interferes with speed reference from controller.	Disable limit switch function in application functions setting of the drive.
The actual motor speed or read actual speed do not correspond to speed reference.	PDO mapping is inconsistent.	Make sure that both speed reference and actual speed values have the same unit (Hz or RPM).
The brake is excessively stressed when starting hoisting movement in downwards direction.	Brake impulse is not active.	Activate brake impulse in Application functions – Brake logic of ATV71.

Part XVII

Wind Speed Control

Chapter 25

WindSpeedCtrl: Monitors and Provides Alarms for High Wind Conditions

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
25.1	Functional and Machine Overview	740
25.2	Architecture	743
25.3	Function Block Description	746
25.4	Pin Description	748

Section 25.1

Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Topic	Page
Functional Overview	741
Machine Overview	742

Functional Overview

Functional Description

The Wind speed control function monitors the wind speed and generates an alert and/or alarm when it exceeds a specified limit. It attracts the attention of the operator with yellow and red indicator lamps and a horn.

This function is used to generate an alert when the wind speed is above the set alert threshold.

The WindSpeedCtrl function block is applicable to the following types of cranes:

- Industrial Cranes (outdoor use)
- Construction Cranes

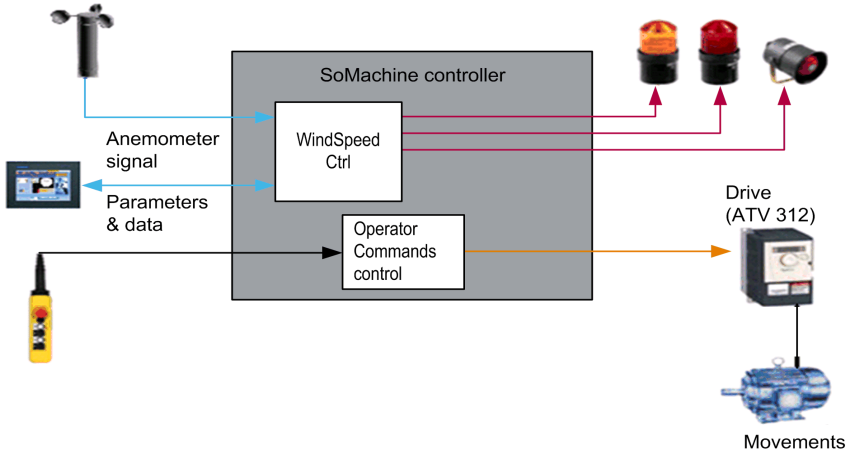
Why Use the WindSpeedCtrl Function Block?

The WindSpeedCtrl function block is required to advise or alarm the operator that he is using the hoist during high speed winds.

Solution with the WindSpeedCtrl Function Block

The WindSpeedCtrl function block monitors the wind speed and compares its speed with set values. If the wind speed exceeds the set value, an advisory or alarm is generated.

Functional View

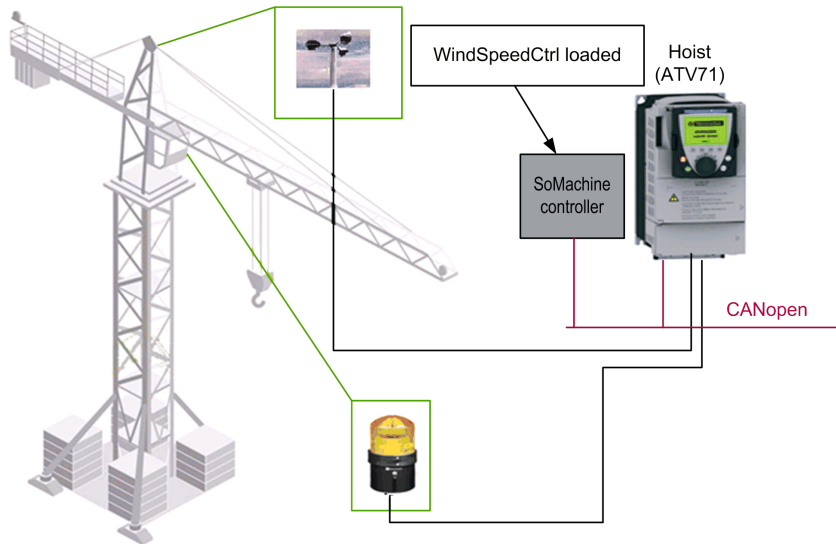


NOTE: The function works with ATV312 or ATV71. In the graphic, only ATV312 is shown, but ATV71 could also be used.

Machine Overview

Machine View

The following illustrates wind speed control being used on a tower crane.



NOTE: The function works with ATV312 or ATV71. In the graphic, only ATV71 is shown, but ATV312 could also be used.

Section 25.2

Architecture

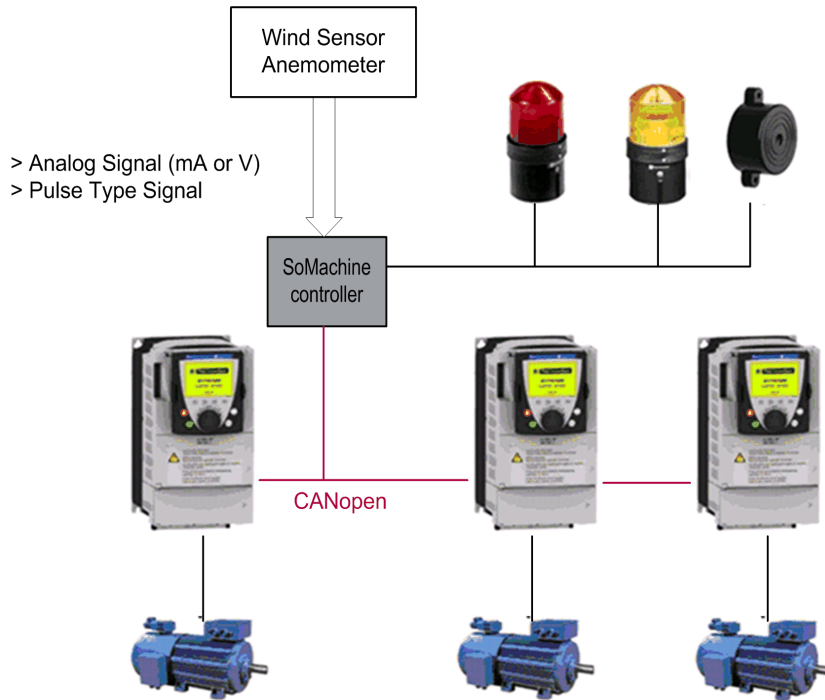
What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Architecture	744
Software Architecture	745

Hardware Architecture

Hardware Architecture Overview



NOTE: Maximum frequency of pulse signal depends on the used controller.

Software Architecture

Software Architecture Overview

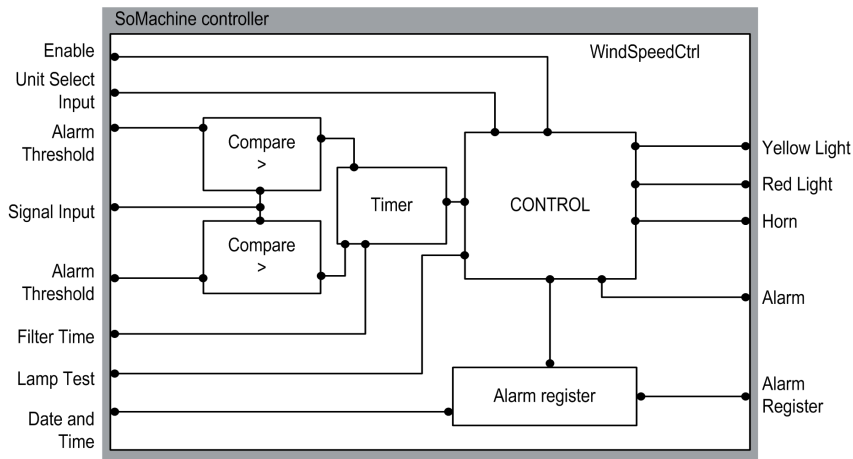
The `WindSpeedCtrl` function block monitors the output signal of an anemometer.

If the actual wind speed value is greater than the set parameter, an alert or alarm is generated (yellow light and/or red light and horn).

If the function is set to imperial units (`i_xUnitSel` input is TRUE), the maximum alert value is 31MPH and the maximum alarm value is 45 MPH. If the function is set to metric units (`i_xUnitSel` input is FALSE), the maximum alert value is 50 km/h and the maximum alarm value is 72 km/h. These limits cannot be exceeded; if greater values are entered, the values are not accepted, and the limits are set to the maximum alert values.

The maximum value of the delay time (`i_wFltrTime`) is limited to 20 sec. This is the time over which a certain wind speed must be monitored before an alert and/or alarm is given, thus filtering out short gusts of wind.

Data Flow Overview



Section 25.3

Function Block Description

WindSpeedCtrl Function Block

Pin Diagram



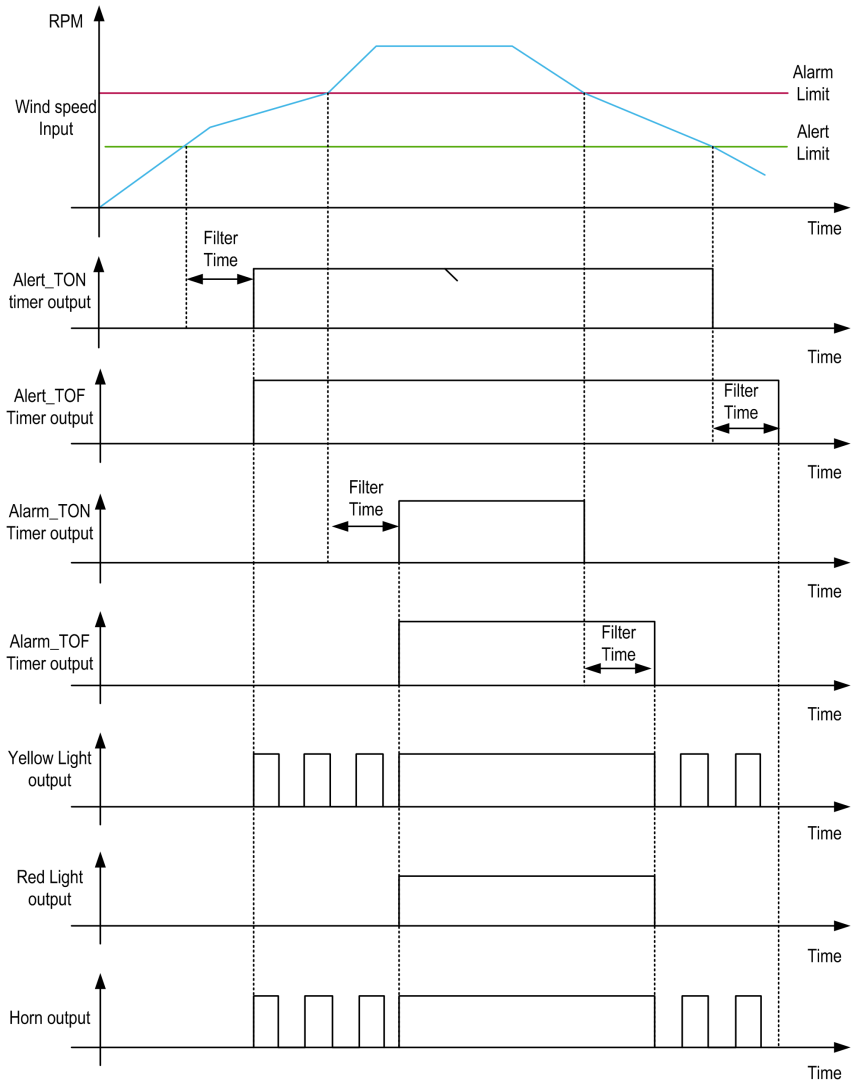
Function Block Description

The Wind speed control function is designed to:

- Provide an alert indication (after filter time) when the wind speed is greater than the set alert threshold. In this case, a blinking yellow light and a pulsed horn are generated.
- Provide an alarm indication (after filter time) when the wind speed is greater than the set alarm threshold. In this case, a continuous red and yellow light and a constant horn signal are generated.
- Provide a test to check the indicator lights.
- Provide an alarm if no signal from the anemometer is detected within a period of 8 hours.
- If an alarm is indicated and the wind speed drops below the alarm limit, or an alert is indicated and the wind speed drops below the alert limit, apply the filter time delay before the alarm/alert is withdrawn.
- Store the last 11 alarm events in a FIFO data array (0 to 10), which can be accessed through the `q_adtDttmAlrm` output pin, where position 0 represents the most recent alarm event.

Timing Chart

The following figure represents the timing and signals diagram for WindSpeedCtrl function block.



Section 25.4

Pin Description

What Is in This Section?

This section contains the following topics:

Topic	Page
Input Pin Description	749
Output Pin Description	751

Input Pin Description

Input Pin Description

Input	Data Type	Description
i_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block. Refer to detailed description (<i>see page 750</i>) of i_xEn.
i_xUnitSel	BOOL	TRUE: Selector input switches to MPH. FALSE: Selector input switches to km/h.
i_wAlrmThsh	WORD	Threshold raises the alarm state. All inputs above 72 (with i_xUnitSel set to FALSE) are set to 72. All inputs above 45 (with i_xUnitSel set to TRUE) are set to 45. Range: 0 to 72 km/h or 0 to 45 MPH depends on i_xUnitSel. Factory setting: 0
i_wAlrtThsh	WORD	Threshold raises the alert state. All inputs above 50 (with i_xUnitSel set to FALSE) are set to 50. All inputs above 31 (with i_xUnitSel set to TRUE) are set to 31. Range: 0 to 50 km/h or 0 to 31 MPH depends on i_xUnitSel. Factory setting: 0
i_wFltrTime	WORD	Timer to delay the alert and alarm level condition. Delay time used to avoid generating false alarms or alerts when very short gusts occur. Whenever the wind speed is greater than the alert or alarm level, the speed wind filter timer is enabled. Only after this delay period, if the wind speed continues to be greater than the alert or alarm level, alert or an alarm is generated. Range: 0...200 [0.1s] Factory setting: 50
i_wWindSpdIput	WORD	Anemometer input from ScaleInput function block. Range: 0 to 65535 Scaling/Unit: km/h or MPH Factory setting: 0

Input	Data Type	Description
i_xLampTest	BOOL	TRUE: Input detects a rising edge. It turns on the following lamps for 6 s: <ul style="list-style-type: none"> ● Yellow lamp (q_xLampYell) ● Red lamp (q_xLampRed) ● Horn (q_xHorn)

LxEn

By setting the i_xEn input to FALSE the output states will be overwritten by a defined fallback state which is specific for each function block.

As soon as the function block gets enabled again the output states will be updated corresponding to the function block behavior as described above. There is no additional action required from you to re-establish the enabled state.

The fallback states of WindSpeedCtrl are given in the table below.

Output	Data Type	Fallback State
q_xEn	BOOL	FALSE
q_xLampYell	BOOL	FALSE
q_xLampRed	BOOL	FALSE
q_xHorn	BOOL	FALSE
q_xAlrm	BOOL	FALSE
q_wAlrmId	WORD	0

Outputs that are not listed will retain their current values.

Output Pin Description

Output Pin Description

Output	Data Type	Description
q_xEn	BOOL	TRUE: Enables the function block. FALSE: Disables the function block.
q_xLampYell	BOOL	Output for the alert light. TRUE: Alert indicator active and inactive for 6 s continuously. FALSE: No alert
q_xLampRed	BOOL	Output for the alarm light. TRUE: Alarm indicator is permanently ON. FALSE: No alarm
q_xHorn	BOOL	Output for the horn during an alarm or alert. TRUE: Alarm PULSE: Alert FALSE: OK
q_xAlrm	BOOL	Alarm detection indicator. TRUE: Alarm detected FALSE: No alarm detected
q_wAlrmId	WORD	Detected Alarm identification. Range: 0...7 For more information refer to Notifications <i>(see page 751)</i> .
q_adtDttmAlrm	Array of DT	Alarm event register output to access from the HMI. It stores the last 11 alarm events which are time stamped (0...10). The 0 position is the most recent event. Range: 0...10

Notifications

Bit Number	Description
0	No signal has been received from anemometer for more than 8 hours.
1	i_wAlrtThsh was too high and has been set to the maximum allowed value.
2	i_wAlrmThsh was too high and has been set to the maximum allowed value.



!

%

According to the IEC standard, % is a prefix that identifies internal memory addresses in the logic controller to store the value of program variables, constants, I/O, and so on.

A

AFB

(application function block)

analog input

Converts received voltage or current levels into numerical values. You can store and process these values within the logic controller.

analog output

Converts numerical values within the logic controller and sends out proportional voltage or current levels.

application

A program including configuration data, symbols, and documentation.

ARRAY

The systematic arrangement of data objects of a single type in the form of a table defined in logic controller memory. The syntax is as follows: `ARRAY [<dimension>] OF <Type>`

Example 1: `ARRAY [1..2] OF BOOL` is a 1-dimensional table with 2 elements of type `BOOL`.

Example 2: `ARRAY [1..10, 1..20] OF INT` is a 2-dimensional table with 10 x 20 elements of type `INT`.

ATV

The model prefix for Altivar drives (for example, ATV312 refers to the Altivar 312 variable speed drive).

C

calibration

The process of setting or maintaining the accuracy of a measuring device by comparing its value to a known and correct standard.

CANmotion

A CANopen-based motion bus with an additional mechanism that provides synchronization between the motion controller and the drives.

CANopen

An open industry-standard communication protocol and device profile specification (EN 50325-4).

closed loop

A closed loop control is a motion control system that used both positional feedback and velocity feedback to generate a correction signal. It does this by comparing its position and velocity to the values of specified parameters. The devices providing the feedback are typically encoders, resolvers, LVTDs, and tachometers.

See also: *open loop*

closing on stack

process of filling the grab with bulk cargo

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

D

derating

A reduction in an operating specification. For devices in general, it is usually a specified reduction in nominal power to facilitate operation at increased ambient conditions like higher temperatures or higher altitudes.

E

element

The short name of the ARRAY element.

encoder

A device for length or angular measurement (linear or rotary encoders).

equipment

A part of a machine including sub-assemblies such as conveyors, turntables, and so on.

expansion bus

An electronic communication bus between expansion I/O modules and a controller.

F

FB

(function block) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

function

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

Examples: boolean (AND) operators, calculations, conversions (BYTE_TO_INT)

G

GVL

(global variable list) Manages global variables that can be passed between controllers on an Ethernet TCP/IP Modbus network.

H

HSP

(high speed) The motor frequency at maximum reference of a drive.

I

I/O

(input/output)

ID

(identifier/identification)

L

LMC

(Lexium motion controller)

LMC058

Lexium motion controller A Modicon logic controller

LSP

The motor frequency at minimum reference of a drive.

M

M258

Modicon M258 logic controller

machine

Consists of several *functions* and/or *equipment*.

ms

(*millisecond*)

N

N/C

(*normally closed*) A contact pair that closes when the actuator is de-energized (no power is applied) and opens when the actuator is energized (power is applied).

N/O

(*normally open*) A contact pair that opens when the actuator is de-energized (no power is applied) and closes when the actuator is energized (power is applied).

node

An addressable device on a communication network.

O

open loop

Open loop control refers to a motion control system with no external sensors to provide position or velocity correction signals.

See also: *closed loop*.

OTB

(*optimized terminal block*) Used in the context of STB I/O distributed modules.

P

PDO

(*process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PID

(*proportional, integral, derivative*) A generic control loop feedback mechanism (controller) widely used in industrial control systems.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

project file

A project file contains information about the developer and purpose of a project, the configuration of the targeted logic controller and associated expansion modules, the source code of a program, symbols, comments, and all other related information.

protocol

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

R

REAL

A data type that is defined as a floating-point number encoded in a 32-bit format.

RPDO

(receive process data object) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

RPM

(revolutions per minute)

run

A command that causes the controller to scan the application program, read the physical inputs, and write to the physical outputs according to solution of the logic of the program.

S

scan

A function that includes:

- reading inputs and placing the values in memory
- executing the application program 1 instruction at a time and storing the results in memory
- using the results to update outputs

setpoint

In a PID controller, the target value set by the user. The main objective of the PID controller is to ensure that the process value reaches the setpoint.

See also *process value*.

STOP

A command that causes the controller to stop running an application program.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

V

variable

A memory unit that is addressed and modified by a program.

W

WORD

A type encoded in a 16-bit format.



A

AdvancedPositioning, *49*
AlarmDataStorage, *549*
AlarmDataStorage_2, *553*
anti sway function, *159*
AntiCrab, *103*
AntiCrab_2, *125*
AntiSwayOpenLoop, *214*
AntiSwayOpenLoop_2, *231*
architecture
 hardware architecture, *45*

C

CableLength_2Pos, *186*
CableLength_3Pos, *186*
CableLength_Enc, *186*
CableLength_Enc_2, *186*

D

DiagnosticCoverage, *257*
DoubleLimitSwitch_AR, *425*
DoubleLimitSwitch_AR_2, *449*

E

ElectronicPotentiometer, *277*
EncAlarmDataStorage, *573*

G

GrabControl, *299*

H

Hoisting
 AdvancedPositioning, *49*
 AlarmDataStorage, *549*
 AlarmDataStorage_2, *553*
 anti sway function, *159*
 AntiCrab, *103*
 AntiCrab_2, *125*
 AntiSwayOpenLoop, *214*
 AntiSwayOpenLoop_2, *231*
 CableLength_2Pos, *186*
 CableLength_3Pos, *186*
 CableLength_Enc, *186*
 CableLength_Enc_2, *186*
 cranes, *31*
 DiagnosticCoverage, *257*
 DoubleLimitSwitch_AR, *425*
 DoubleLimitSwitch_AR_2, *449*
 ElectronicPotentiometer, *277*
 EncAlarmDataStorage, *573*

function block location in the Hoisting Library, *29*
GrabControl, *299*
hardware architecture, *45*
HoistPositionSync, *337*
HoistPositionSync_2, *359*
LdslAlarmDataStorage, *569*
LimitSwitch, *379*
LimitSwitch_AR, *403*
LoadOverspeedCtrl, *477*
LoadOverspeedCtrl_2, *501*
MaintenanceDataStorage, *577*
MaintenanceDataStorage_2, *587*
monitoring data storage function, *527*
overload control function, *607*
Overload_EN15011, *637*
OverloadCtrlDist, *625*
OverloadCtrlEnc, *630*
OverloadCtrlTrq, *620*
OvldAlarmDataStorage, *557*
OvspAlarmDataStorage, *565*
OvtqAlarmDataStorage, *561*
PasswordDataStorage, *597*
ScaleInput, *661*
smooth slewing function, *673*
SmoothSlewingSpd, *682*
SmoothSlewingTrq, *688*
SpeedOptRopeSlack, *699*
SpeedRef_2, *169*
SpeedRef_4, *169*
SpeedRef_AI, *169*
SpeedSelect, *727*
StatisticDataStorage, *535*
StatisticDataStorage_2, *544*
system requirements, *26*
WindSpeedCtrl, *739*
HoistPositionSync, *337*
HoistPositionSync_2, *359*

L

LdslAlarmDataStorage, *569*
LimitSwitch, *379*
LimitSwitch_AR, *403*
LoadOverspeedCtrl, *477*

LoadOverspeedCtrl_2, *501*

M

machine
hardware architecture, *45*
MaintenanceDataStorage, *577*
MaintenanceDataStorage_2, *587*
monitoring data storage function, *527*

O

overload control function, *607*
Overload_EN15011, *637*
OverloadCtrlDist, *625*
OverloadCtrlEnc, *630*
OverloadCtrlTrq, *620*
OvldAlarmDataStorage, *557*
OvspAlarmDataStorage, *565*
OvtqAlarmDataStorage, *561*

P

PasswordDataStorage, *597*

S

ScaleInput, *661*
smooth slewing function, *673*
SmoothSlewingSpd, *682*
SmoothSlewingTrq, *688*
SpeedOptRopeSlack, *699*
SpeedRef_2, *169*
SpeedRef_4, *169*
SpeedRef_AI, *169*
SpeedSelect, *727*
StatisticDataStorage, *535*
StatisticDataStorage_2, *544*
system requirements, *26*

W

WindSpeedCtrl, *739*