

ConfigurationDesk

I/O Function Implementation Guide

For ConfigurationDesk 6.7

Release 2021-A – May 2021

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: <http://www.dspace.com/go/locations>
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: <http://www.dspace.com/go/supportrequest>. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit <http://www.dspace.com/go/patches> for software updates and patches.

Important Notice

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2008 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

About This Document	39
Introduction	41
Overview on the Available Function Block Types.....	41
Allocating SCALEXIO Channel Types to Function Block Types.....	51
Allocating MicroAutoBox III Channel Types to Function Block Types.....	56
Implementing I/O Functionality	61
Basics on Implementing I/O Functionality.....	62
Basics on Instantiating Function Blocks.....	62
Basics on Function Blocks.....	67
Characteristics of Signal Ports.....	71
Characteristics of Function Ports.....	73
Characteristics of Event Ports.....	75
Configuring Function Blocks.....	77
Basics on Function Block Configuration.....	77
Categories of Configurable Parameters.....	79
Basics on Hardware Resources and Channel Types.....	80
How to Rename Function Blocks.....	83
How to View Circuit Diagrams of Hardware Resources.....	84
Configuring Standard Features	87
Configuring Standard Features of Function Ports.....	88
Specifying Initialization and Stop Behavior.....	88
Specifying User Saturation.....	90
Configuring Test Automation Support.....	92
Configuring Standard Features of the Electrical Interface.....	95
Specifying Current and Voltage Values for Channel Multiplication.....	95
Specifying Digital Output Signals (Flexible Out 1).....	98
Specifying Digital Output Signals (Digital Out 1).....	102
Specifying Digital Output Signals (Digital Out 2, Digital In/Out 1).....	106
Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3).....	109
Specifying Digital Output Signals (Digital In/Out 5).....	112
Specifying the Circuit Type for Voltage Input Signals.....	116

Specifying the Circuit Type for Analog Output Signals.....	118
Specifying the Measurement Point for Input Signals.....	119
Configuring Standard Features of Signal Ports.....	121
Specifying Load Settings.....	121
Specifying Fuse Settings.....	122
Specifying Settings for Failure Simulation.....	123
Configuring Standard Functionalities of Function Blocks.....	126
Using Angular Processing Units (APUs).....	126
Current In	131
Introduction (Current In).....	132
Introduction to the Function Block (Current In).....	132
Overviews (Current In).....	133
Overview of Ports and Basic Properties (Current In).....	133
Overview of Tunable Properties (Current In).....	136
Configuring the Function Block (Current In).....	137
Configuring the Basic Functionality (Current In).....	137
Configuring Standard Features (Current In).....	139
Hardware Dependencies (Current In).....	140
SCALEXIO Hardware Dependencies (Current In).....	140
Triggered Current In	143
Introduction (Triggered Current In).....	144
Introduction to the Function Block (Triggered Current In).....	144
Overviews (Triggered Current In).....	145
Overview of Ports and Basic Properties (Triggered Current In).....	145
Overview of Tunable Properties (Triggered Current In).....	148
Configuring the Function Block (Triggered Current In).....	149
Configuring the Basic Functionality (Triggered Current In).....	149
Configuring Standard Features (Triggered Current In).....	153
Hardware Dependencies (Triggered Current In).....	155
SCALEXIO Hardware Dependencies (Triggered Current In).....	155
Voltage In	157
Introduction (Voltage In).....	158
Introduction to the Function Block (Voltage In).....	158

Overviews (Voltage In).....	159
Overview of Ports and Basic Properties (Voltage In).....	159
Overview of Tunable Properties (Voltage In).....	163
Configuring the Function Block (Voltage In).....	164
Configuring the Basic Functionality (Voltage In).....	164
Configuring Standard Features (Voltage In).....	167
Hardware Dependencies (Voltage In)	170
SCALEXIO Hardware Dependencies (Voltage In).....	170
MicroAutoBox III Hardware Dependencies (Voltage In).....	173
Current Sink	175
Introduction (Current Sink).....	176
Introduction to the Function Block (Current Sink).....	176
Overviews (Current Sink).....	177
Overview of Ports and Properties (Current Sink).....	177
Overview of Tunable Properties (Current Sink).....	179
Configuring the Function Block (Current Sink).....	180
Configuring the Basic Functionality (Current Sink).....	180
Configuring Standard Features (Current Sink).....	182
Hardware Dependencies (Current Sink).....	184
SCALEXIO Hardware Dependencies (Current Sink).....	184
Voltage Out	187
Introduction (Voltage Out).....	188
Introduction to the Function Block (Voltage Out).....	188
Overviews (Voltage Out).....	189
Overview of Ports and Basic Properties (Voltage Out).....	189
Overview of Tunable Properties (Voltage Out).....	191
Configuring the Function Block (Voltage Out).....	192
Configuring the Basic Functionality (Voltage Out).....	192
Configuring Standard Features (Voltage Out).....	192
Hardware Dependencies (Voltage Out).....	195
SCALEXIO Hardware Dependencies (Voltage Out).....	195
MicroAutoBox III Hardware Dependencies (Voltage Out).....	197

Multi Bit In	199
Introduction (Multi Bit In).....	200
Introduction to the Function Block (Multi Bit In).....	200
Overviews (Multi Bit In).....	201
Overview of Ports and Basic Properties (Multi Bit In).....	201
Overview of Tunable Properties (Multi Bit In).....	204
Configuring the Function Block (Multi Bit In).....	206
Configuring the Basic Functionality (Multi Bit In).....	206
Configuring Standard Features (Multi Bit In).....	210
Hardware Dependencies (Multi Bit In).....	213
SCALEXIO Hardware Dependencies (Multi Bit In).....	213
MicroAutoBox III Hardware Dependencies (Multi Bit In).....	216
Trigger In	219
Introduction (Trigger In).....	220
Introduction to the Function Block (Trigger In).....	220
Overviews (Trigger In).....	221
Overview of Ports and Basic Properties (Trigger In).....	221
Overview of Tunable Properties (Trigger In).....	223
Configuring the Function Block (Trigger In).....	224
Configuring the Basic Functionality (Trigger In).....	224
Configuring Standard Features (Trigger In).....	226
Hardware Dependencies (Trigger In).....	228
SCALEXIO Hardware Dependencies (Trigger In).....	228
MicroAutoBox III Hardware Dependencies (Trigger In).....	230
Multi Bit Out	231
Introduction (Multi Bit Out).....	232
Introduction to the Function Block (Multi Bit Out).....	232
Overviews (Multi Bit Out).....	233
Overview of Ports and Basic Properties (Multi Bit Out).....	233
Overview of Tunable Properties (Multi Bit Out).....	236
Configuring the Function Block (Multi Bit Out).....	237
Configuring the Basic Functionality (Multi Bit Out).....	237
Configuring Standard Features (Multi Bit Out).....	238

Hardware Dependencies (Multi Bit Out).....	242
SCALEXIO Hardware Dependencies (Multi Bit Out).....	242
MicroAutoBox III Hardware Dependencies (Multi Bit Out).....	245

Digital Pulse Out 247

Introduction (Digital Pulse Out).....	248
Introduction to the Function Block (Digital Pulse Out).....	248
Overviews (Digital Pulse Out).....	249
Overview of Ports and Basic Properties (Digital Pulse Out).....	249
Overview of Tunable Properties (Digital Pulse Out).....	251
Configuring the Function Block (Digital Pulse Out).....	252
Configuring the Basic Functionality (Digital Pulse Out).....	252
Configuring Standard Features (Digital Pulse Out).....	253
Hardware Dependencies (Digital Pulse Out).....	256
SCALEXIO Hardware Dependencies (Digital Pulse Out).....	256
MicroAutoBox III Hardware Dependencies (Digital Pulse Out).....	257

Potentiometer Out 259

Introduction (Potentiometer Out).....	260
Introduction to the Function Block (Potentiometer Out).....	260
Overviews (Potentiometer Out).....	261
Overview of Ports and Basic Properties (Potentiometer Out).....	261
Overview of Tunable Properties (Potentiometer Out).....	262
Configuring the Function Block (Potentiometer Out).....	263
Configuring the Basic Functionality (Potentiometer Out).....	263
Configuring Standard Features (Potentiometer Out).....	264
Hardware Dependencies (Potentiometer Out).....	266
SCALEXIO Hardware Dependencies (Potentiometer Out).....	266

Resistance Out 267

Introduction (Resistance Out).....	268
Introduction to the Function Block (Resistance Out).....	268
Overviews (Resistance Out).....	269
Overview of Ports and Basic Properties (Resistance Out).....	269
Overview of Tunable Properties (Resistance Out).....	271

Configuring the Function Block (Resistance Out).....	272
Configuring Standard Features (Resistance Out).....	272
Hardware Dependencies (Resistance Out).....	274
SCALEXIO Hardware Dependencies (Resistance Out).....	274
PWM/PFM Out	275
Introduction (PWM/PFM Out).....	276
Introduction to the Function Block (PWM/PFM Out).....	276
Basics on PWM/PFM Signals.....	277
Overviews (PWM/PFM Out).....	279
Overview of Ports and Basic Properties (PWM/PFM Out).....	279
Tunable Properties (PWM/PFM Out).....	282
Configuring the Function Block (PWM/PFM Out).....	283
Configuring the Basic Functionality (PWM/PFM Out).....	283
Configuring Standard Features (PWM/PFM Out).....	284
Hardware Dependencies (PWM/PFM Out).....	288
SCALEXIO Hardware Dependencies (PWM/PFM Out).....	288
MicroAutoBox III Hardware Dependencies (PWM/PFM Out).....	291
Multi-Channel PWM Out	293
Introduction to the Functionality (Multi-Channel PWM Out).....	294
Introduction to the Function Block (Multi-Channel PWM Out).....	294
Basics on Multi-Channel PWM Signals.....	295
Overviews (Multi-Channel PWM Out).....	298
Overview of Ports and Basic Properties (Multi-Channel PWM Out).....	298
Overview of Tunable Properties (Multi-Channel PWM Out).....	303
Configuring the Function Block (Multi-Channel PWM Out).....	304
Configuring the Basic Functionality (Multi-Channel PWM Out).....	304
Configuring Standard Features (Multi-Channel PWM Out).....	308
Hardware Dependencies (Multi-Channel PWM Out).....	311
SCALEXIO Hardware Dependencies (Multi-Channel PWM Out).....	311
MicroAutoBox III Hardware Dependencies (Multi-Channel PWM Out).....	312
PWM/PFM In	315
Introduction (PWM/PFM In).....	316
Introduction to the Function Block (PWM/PFM In).....	316
Basics on Signal Pattern Measurement.....	317

Overviews (PWM/PFM In).....	320
Overview of Ports and Basic Properties (PWM/PFM In).....	320
Overview of Tunable Properties (PWM/PFM In).....	323
Configuring the Function Block (PWM/PFM In).....	324
Configuring the Basic Functionality (PWM/PFM In).....	324
Configuring Standard Features (PWM/PFM In).....	328
Hardware Dependencies (PWM/PFM In).....	331
SCALEXIO Hardware Dependencies (PWM/PFM In).....	331
MicroAutoBox III Hardware Dependencies (PWM/PFM In).....	334
Digital Pulse In	337
Introduction (Digital Pulse In).....	338
Introduction to the Function Block (Digital Pulse In).....	338
Overviews (Digital Pulse In).....	339
Overview of Ports and Basic Properties (Digital Pulse In).....	339
Overview of Tunable Properties (Digital Pulse In).....	340
Configuring the Function Block (Digital Pulse In).....	341
Configuring the Basic Functionality (Digital Pulse In).....	341
Configuring Standard Features (Digital Pulse In).....	343
Hardware Dependencies (Digital Pulse In)	344
MicroAutoBox III Hardware Dependencies (Digital Pulse In).....	344
Digital Pulse Capture	345
Introduction (Digital Pulse Capture).....	346
Introduction to the Function Block (Digital Pulse Capture).....	346
Overviews (Digital Pulse Capture).....	347
Overview of Ports and Basic Properties (Digital Pulse Capture).....	347
Overview of Tunable Properties (Digital Pulse Capture).....	351
Configuring the Function Block (Digital Pulse Capture).....	352
Configuring the Basic Functionality (Digital Pulse Capture).....	352
Configuring Standard Features (Digital Pulse Capture).....	355
Hardware Dependencies (Digital Pulse Capture).....	358
SCALEXIO Hardware Dependencies (Digital Pulse Capture).....	358
MicroAutoBox III Hardware Dependencies (Digital Pulse Capture).....	360

Engine Simulation 363

Introduction to Engine Simulation.....	364
Basics on Engine Simulation with SCALEXIO.....	364
Operation Principle of Four-Stroke Piston Engines.....	365
Angular Clock Setup.....	369
Introduction (Angular Clock Setup).....	369
Introduction to the Function Block (Angular Clock Setup).....	369
Overviews (Angular Clock Setup).....	370
Overview of Ports and Basic Properties (Angular Clock Setup).....	370
Configuring the Function Block (Angular Clock Setup).....	371
Configuring the Basic Functionality (Angular Clock Setup).....	371
Configuring Standard Features (Angular Clock Setup).....	372
Engine Simulation Setup.....	373
Introduction (Engine Simulation Setup).....	373
Introduction to the Function Block (Engine Simulation Setup).....	373
Overviews (Engine Simulation Setup).....	374
Overview of Ports and Basic Properties (Engine Simulation Setup).....	374
Overview of Tunable Properties (Engine Simulation Setup).....	375
Configuring the Function Block (Engine Simulation Setup).....	375
Configuring the Basic Functionality (Engine Simulation Setup).....	375
Configuring Standard Features (Engine Simulation Setup).....	376
Injection/Ignition (Voltage In, Current In).....	377
Introduction to Injection and Ignition Pulse Measurement.....	377
Basics on Measurement and Analysis of Injection and Ignition Pulses.....	377
Details on Window-Based Pulse Measurement Results.....	384
Details on Continuous Pulse Measurement Results.....	390
SCALEXIO Versus DS2211 Injection/Ignition Signal Evaluation.....	392
Injection/Ignition Voltage In.....	395
Introduction (Injection/Ignition Voltage In).....	395
Introduction to the Function Block (Injection/Ignition Voltage In).....	395
Overviews (Injection/Ignition Voltage In).....	397
Overview of Ports and Basic Properties (Injection/Ignition Voltage In).....	397
Overview of Tunable Properties (Injection/Ignition Voltage In).....	405
Configuring the Function Block (Injection/Ignition Voltage In).....	406
Configuring the Basic Functionality (Injection/Ignition Voltage In).....	406

Configuring Standard Features (Injection/Ignition Voltage In).....	409
Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Voltage In).....	412
Hardware Dependencies (Injection/Ignition Voltage In).....	423
SCALEXIO Hardware Dependencies (Injection/Ignition Voltage In).....	423
Injection/Ignition Current In.....	425
Introduction (Injection/Ignition Current In).....	426
Introduction to the Function Block (Injection/Ignition Current In).....	426
Overviews (Injection/Ignition Current In).....	427
Overview of Ports and Basic Properties (Injection/Ignition Current In).....	427
Overview of Tunable Properties (Injection/Ignition Current In).....	435
Configuring the Function Block (Injection/Ignition Current In).....	436
Configuring the Basic Functionality (Injection/Ignition Current In).....	436
Configuring Standard Features (Injection/Ignition Current In).....	439
Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Current In).....	441
Hardware Dependencies (Injection/Ignition Current In).....	449
SCALEXIO Hardware Dependencies (Injection/Ignition Current In).....	449
Crank/Cam (Voltage Out, Current Sink, Digital Out).....	451
Basics on Crankshaft and Camshaft Signals and Sensors.....	451
Operation Principle of Crankshaft and Camshaft Sensors.....	451
Basics on Crankshaft and Camshaft Signal Generation.....	454
Basics on Reverse Crankshaft Signal Generation.....	456
Creating Crankshaft/Camshaft Wavetables.....	458
Basics on Crankshaft/Camshaft Wavetable Files.....	458
How to Create Analog Crankshaft Wavetables.....	459
How to Create Digital Crankshaft Wavetables.....	459
How to Create Reverse Crankshaft Wavetables.....	461
How to Create Camshaft Wavetables.....	462
Assigning Wavetables to the Real-Time Application.....	463
How to Import Wavetables to a Crank/Cam Function Block.....	463
Switching the Active Wavetable at Run Time.....	465
Replacing Wavetable Files at Run Time.....	466
Upgrading DS2211 Wavetables.....	467
How to Convert a MAT Wavetable to CSV	467

Crank/Cam Voltage Out.....	468
Introduction (Crank/Cam Voltage Out).....	469
Introduction to the Function Block (Crank/Cam Voltage Out).....	469
Overviews (Crank/Cam Voltage Out).....	470
Overview of Ports and Basic Properties (Crank/Cam Voltage Out).....	470
Overview of Tunable Properties (Crank/Cam Voltage Out).....	472
Configuring the Function Block (Crank/Cam Voltage Out).....	473
Configuring the Basic Functionality (Crank/Cam Voltage Out).....	473
Configuring Standard Features (Crank/Cam Voltage Out).....	474
Hardware Dependencies (Crank/Cam Voltage Out).....	476
SCALEXIO Hardware Dependencies (Crank/Cam Voltage Out).....	476
Crank/Cam Current Sink.....	478
Introduction (Crank/Cam Current Sink).....	478
Introduction to the Function Block (Crank/Cam Current Sink).....	478
Overviews (Crank/Cam Current Sink).....	479
Overview of Ports and Basic Properties (Crank/Cam Current Sink).....	479
Overview of Tunable Properties (Crank/Cam Current Sink).....	482
Configuring the Function Block (Crank/Cam Current Sink).....	482
Configuring the Basic Functionality (Crank/Cam Current Sink).....	483
Configuring Standard Features (Crank/Cam Current Sink).....	486
Hardware Dependencies (Crank/Cam Current Sink).....	487
SCALEXIO Hardware Dependencies (Crank/Cam Current Sink).....	487
Crank/Cam Digital Out.....	488
Introduction (Crank/Cam Digital Out).....	488
Introduction to the Function Block (Crank/Cam Digital Out).....	488
Overviews (Crank/Cam Digital Out).....	489
Overview of Ports and Basic Properties (Crank/Cam Digital Out).....	490
Overview of Tunable Properties (Crank/Cam Digital Out).....	492
Configuring the Function Block (Crank/Cam Digital Out).....	493
Configuring the Basic Functionality (Crank/Cam Digital Out).....	493
Configuring Standard Features (Crank/Cam Digital Out).....	494
Hardware Dependencies (Crank/Cam Digital Out).....	496
SCALEXIO Hardware Dependencies (Crank/Cam Digital Out).....	496

Knock Signal Out.....	498
Introduction (Knock Signal Out).....	498
Introduction to the Function Block (Knock Signal Out).....	498
Basics on Knock Signal Generation.....	499
Overviews (Knock Signal Out).....	502
Overview of Ports and Basic Properties (Knock Signal Out).....	502
Overview of Tunable Properties (Knock Signal Out).....	505
Configuring the Function Block (Knock Signal Out).....	505
Configuring the Basic Functionality (Knock Signal Out).....	506
Configuring Standard Features (Knock Signal Out).....	506
Hardware Dependencies (Knock Signal Out).....	507
SCALEXIO Hardware Dependencies (Knock Signal Out).....	507
Lambda Probe Simulation.....	509
Introduction to Lambda Probe Simulation	509
Definition of Lambda.....	509
SCALEXIO-Compatible Lambda Probe Types.....	510
Basics on the Simulation of Two-Step Lambda Probes.....	511
Basics on Simulation of Wide-Band Lambda Probes (DCR).....	515
Basics on Simulation of Wide-Band Lambda Probes (NCCR).....	518
Lambda DCR.....	523
Introduction (Lambda DCR).....	524
Introduction to the Function Block (Lambda DCR).....	524
Overviews (Lambda DCR).....	525
Overview of Ports and Basic Properties (Lambda DCR).....	525
Overview of Tunable Properties (Lambda DCR).....	526
Configuring the Function Block (Lambda DCR).....	527
Configuring the Basic Functionality (Lambda DCR).....	527
Configuring Standard Features (Lambda DCR).....	529
Mapping Signal Ports (Lambda DCR).....	530
Hardware Dependencies (Lambda DCR).....	533
SCALEXIO Hardware Dependencies (Lambda DCR).....	533
Lambda NCCR.....	534
Introduction (Lambda NCCR).....	535
Introduction to the Function Block (Lambda NCCR).....	535

Overviews (Lambda NCCR).....	536
Overview of Ports and Basic Properties (Lambda NCCR).....	536
Overview of Tunable Properties (Lambda NCCR).....	538
Configuring the Function Block (Lambda NCCR).....	539
Configuring the Basic Functionality (Lambda NCCR).....	539
Configuring Standard Features (Lambda NCCR).....	543
Mapping Signal Ports (Lambda NCCR).....	544
Hardware Dependencies (Lambda NCCR).....	547
SCALEXIO Hardware Dependencies (Lambda NCCR).....	547
Engine Control	549
Introduction to Engine Control.....	550
Basics on Engine Control with MicroAutoBox III.....	550
Operation Principle of Four-Stroke Piston Engines.....	552
Engine Control Setup.....	556
Introduction (Engine Control Setup).....	556
Introduction to the Function Block (Engine Control Setup).....	556
Overviews (Engine Control Setup).....	557
Overview of Ports and Basic Properties (Engine Control Setup).....	557
Overview of Tunable Properties (Engine Control Setup).....	559
Configuring the Function Block (Engine Control Setup).....	559
Configuring the Basic Functionality (Engine Control Setup).....	560
Configuring Standard Features (Engine Control Setup).....	561
Crank In.....	563
Introduction (Crank In).....	563
Introduction to the Function Block (Crank In).....	563
Basics on Crankshaft Sensor Signals.....	564
Supported Crankshaft Wheels.....	565
Basics on Synchronizing the Crankshaft.....	566
Basics on Speed Measurement.....	571
Overviews (Crank In).....	572
Overview of Ports and Basic Properties (Crank In).....	573
Overview of Tunable Properties (Crank In).....	578
Configuring the Function Block (Crank In).....	578
Configuring the Basic Functionality (Crank In).....	579
Configuring Standard Features (Crank In).....	585

Hardware Dependencies (Crank In)	586
MicroAutoBox III Hardware Dependencies (Crank In).....	586
Cam In.....	587
Introduction (Cam In).....	587
Introduction to the Function Block (Cam In).....	587
Basics on Camshaft Sensor Signals.....	588
Basics on Camshaft Phase Shift Measurement.....	589
Overviews (Cam In).....	590
Overview of Ports and Basic Properties (Cam In).....	590
Overview of Tunable Properties (Cam In).....	596
Configuring the Function Block (Cam In).....	596
Configuring the Basic Functionality (Cam In).....	597
Configuring Standard Features (Cam In).....	602
Hardware Dependencies (Cam In)	603
MicroAutoBox III Hardware Dependencies (Cam In).....	603
Injection Out.....	604
Introduction (Injection Out).....	604
Introduction to the Function Block (Injection Out).....	604
Overviews (Injection Out).....	605
Overview of Ports and Basic Properties (Injection Out).....	605
Overview of Tunable Properties (Injection Out).....	609
Configuring the Function Block (Injection Out).....	610
Configuring the Basic Functionality (Injection Out).....	610
Configuring Standard Features (Injection Out).....	614
Hardware Dependencies (Injection Out)	615
MicroAutoBox III Hardware Dependencies (Injection Out).....	615
Ignition Out.....	617
Introduction (Ignition Out).....	617
Introduction to the Function Block (Ignition Out).....	617
Overviews (Ignition Out).....	618
Overview of Ports and Basic Properties (Ignition Out).....	618
Overview of Tunable Properties (Ignition Out).....	622
Configuring the Function Block (Ignition Out).....	623
Configuring the Basic Functionality (Ignition Out).....	623
Configuring Standard Features (Ignition Out).....	626

Hardware Dependencies (Ignition Out)	627
MicroAutoBox III Hardware Dependencies (Ignition Out).....	627
Engine Angular Pulse Out.....	629
Introduction (Engine Angular Pulse Out).....	629
Introduction to the Function Block (Engine Angular Pulse Out).....	629
Overviews (Engine Angular Pulse Out).....	630
Overview of Ports and Basic Properties (Engine Angular Pulse Out).....	630
Overview of Tunable Properties (Engine Angular Pulse Out).....	632
Configuring the Function Block (Engine Angular Pulse Out).....	632
Configuring the Basic Functionality (Engine Angular Pulse Out).....	633
Configuring Standard Features (Engine Angular Pulse Out).....	635
Hardware Dependencies (Engine Angular Pulse Out)	636
MicroAutoBox III Hardware Dependencies (Engine Angular Pulse Out).....	636
Knock In.....	637
Introduction (Knock In).....	637
Introduction to the Function Block (Knock In).....	637
Basics on Knock Signal Measurement.....	638
Overviews (Knock In).....	640
Overview of Ports and Basic Properties (Knock In).....	640
Overview of Tunable Properties (Knock In).....	643
Configuring the Function Block (Knock In).....	644
Configuring the Basic Functionality (Knock In).....	644
Configuring Standard Features (Knock In).....	645
Hardware Dependencies (Knock In)	646
MicroAutoBox III Hardware Dependencies (Knock In).....	646
Lambda Probe In.....	647
Introduction (Lambda Probe In).....	647
Introduction to the Function Block (Lambda Probe In).....	647
Introduction to Wideband Lambda Probe Operation (Lambda Probe In).....	648
Overviews (Lambda Probe In).....	652
Overview of Ports and Basic Properties (Lambda Probe In).....	652
Overview of Tunable Properties (Lambda Probe In).....	657
Configuration of the Function Block (Lambda Probe In).....	657
Configuring the Basic Functionality (Lambda Probe In).....	657

Configuring Standard Features (Lambda Probe In).....	658
Mapping Signal Ports (Lambda Probe In).....	659
Hardware Dependencies (Lambda Probe In).....	662
MicroAutoBox III Hardware Dependencies (Lambda Probe In).....	662
Wavetable (Voltage Out, Current Sink, Digital Out)	663
Introduction to Signal Generation Using Time-Coded Wavetables.....	664
Basics on Signal Generation Using Time-Coded Wavetables.....	664
Specifying Time-Coded Wavetable Files.....	666
Wavetable Voltage Out.....	668
Introduction (Wavetable Voltage Out).....	668
Introduction to the Function Block (Wavetable Voltage Out).....	668
Overviews (Wavetable Voltage Out).....	669
Overview of Ports and Basic Properties (Wavetable Voltage Out).....	669
Overview of Tunable Properties (Wavetable Voltage Out).....	674
Configuring the Function Block (Wavetable Voltage Out).....	674
Configuring the Basic Functionality (Wavetable Voltage Out).....	675
Configuring Standard Features (Wavetable Voltage Out).....	677
Hardware Dependencies (Wavetable Voltage Out).....	679
SCALEXIO Hardware Dependencies (Wavetable Voltage Out).....	679
Wavetable Current Sink.....	682
Introduction (Wavetable Current Sink).....	682
Introduction to the Function Block (Wavetable Current Sink).....	682
Overviews (Wavetable Current Sink).....	683
Overview of Ports and Basic Properties (Wavetable Current Sink).....	683
Overview of Tunable Properties (Wavetable Current Sink).....	688
Configuring the Function Block (Wavetable Current Sink).....	688
Configuring the Basic Functionality (Wavetable Current Sink).....	689
Configuring Standard Features (Wavetable Current Sink).....	693
Hardware Dependencies (Wavetable Current Sink).....	694
SCALEXIO Hardware Dependencies (Wavetable Current Sink).....	694
Wavetable Digital Out.....	696
Introduction (Wavetable Digital Out).....	696
Introduction to the Function Block (Wavetable Digital Out).....	696

Overviews (Wavetable Digital Out).....	697
Overview of Ports and Basic Properties (Wavetable Digital Out).....	697
Overview of Tunable Properties (Wavetable Digital Out).....	702
Configuring the Function Block (Wavetable Digital Out).....	702
Configuring the Basic Functionality (Wavetable Digital Out).....	703
Configuring Standard Features (Wavetable Digital Out).....	705
Hardware Dependencies (Wavetable Digital Out).....	707
SCALEXIO Hardware Dependencies (Wavetable Digital Out).....	707
Angular Wavetable (Voltage Out, Digital Out)	711
Introduction to Signal Generation Using Angular-Coded Wavetables.....	712
Basics on Signal Generation Using Angular-Coded Wavetables.....	712
Specifying Angular-Coded Wavetable Files.....	715
Replacing Angular-Coded Wavetable Files at Run Time.....	716
Angular Wavetable Voltage Out.....	719
Introduction (Angular Wavetable Voltage Out).....	719
Introduction to the Function Block (Angular Wavetable Voltage Out).....	719
Overviews (Angular Wavetable Voltage Out).....	720
Overview of Ports and Basic Properties (Angular Wavetable Voltage Out).....	720
Overview of Tunable Properties (Angular Wavetable Voltage Out).....	723
Configuring the Function Block (Angular Wavetable Voltage Out).....	723
Configuring the Basic Functionality (Angular Wavetable Voltage Out).....	724
Configuring Standard Features (Angular Wavetable Voltage Out).....	725
Hardware Dependencies (Angular Wavetable Voltage Out).....	727
SCALEXIO Hardware Dependencies (Angular Wavetable Voltage Out).....	727
Angular Wavetable Digital Out.....	730
Introduction (Angular Wavetable Digital Out).....	730
Introduction to the Function Block (Angular Wavetable Digital Out).....	730
Overviews (Angular Wavetable Digital Out).....	731
Overview of Ports and Basic Properties (Angular Wavetable Digital Out).....	731
Overview of Tunable Properties (Angular Wavetable Digital Out).....	734
Configuring the Function Block (Angular Wavetable Digital Out).....	735
Configuring the Basic Functionality (Angular Wavetable Digital Out).....	735
Configuring Standard Features (Angular Wavetable Digital Out).....	737

Hardware Dependencies (Angular Wavetable Digital Out).....	738
SCALEXIO Hardware Dependencies (Angular Wavetable Digital Out).....	738
Waveform (Voltage Out, Current Sink, Digital Out)	741
Introduction to Signal Generation Using Waveforms	742
Basics on Signal Generation Using Waveforms.....	742
Waveform Voltage Out.....	745
Introduction (Waveform Voltage Out).....	745
Introduction to the Function Block (Waveform Voltage Out).....	745
Overviews (Waveform Voltage Out).....	746
Overview of Ports and Basic Properties (Waveform Voltage Out).....	746
Overview of Tunable Properties (Waveform Voltage Out).....	751
Configuring the Function Block (Waveform Voltage Out).....	751
Configuring the Basic Functionality (Waveform Voltage Out).....	752
Configuring Standard Features (Waveform Voltage Out).....	753
Hardware Dependencies (Waveform Voltage Out).....	755
SCALEXIO Hardware Dependencies (Waveform Voltage Out).....	755
Waveform Current Sink.....	759
Introduction (Waveform Current Sink).....	759
Introduction to the Function Block (Waveform Current Sink).....	759
Overviews (Waveform Current Sink).....	760
Overview of Ports and Basic Properties (Waveform Current Sink).....	760
Overview of Tunable Properties (Waveform Current Sink).....	765
Configuring the Function Block (Waveform Current Sink).....	765
Configuring the Basic Functionality (Waveform Current Sink).....	766
Configuring Standard Features (Waveform Current Sink).....	768
Hardware Dependencies (Waveform Current Sink).....	769
SCALEXIO Hardware Dependencies (Waveform Current Sink).....	769
Waveform Digital Out.....	772
Introduction (Waveform Digital Out).....	772
Introduction to the Function Block (Waveform Digital Out).....	772
Overviews (Waveform Digital Out).....	773
Overview of Ports and Basic Properties (Waveform Digital Out).....	773
Overview of Tunable Properties (Waveform Digital Out).....	778

Configuring the Function Block (Waveform Digital Out).....	779
Configuring the Basic Functionality (Waveform Digital Out).....	779
Configuring Standard Features (Waveform Digital Out).....	781
Hardware Dependencies (Waveform Digital Out).....	783
SCALEXIO Hardware Dependencies (Waveform Digital Out).....	783
Current Signal Capture	787
Introduction (Current Signal Capture).....	788
Introduction to the Function Block (Current Signal Capture).....	788
Overviews (Current Signal Capture).....	789
Overview of Ports and Basic Properties (Current Signal Capture).....	789
Overview of Tunable Properties (Current Signal Capture).....	797
Configuring the Function Block (Current Signal Capture).....	799
Configuring the Basic Functionality (Current Signal Capture).....	799
Configuring Trigger Functionality (Current Signal Capture).....	802
Configuring Standard Features (Current Signal Capture).....	807
Hardware Dependencies (Current Signal Capture).....	808
SCALEXIO Hardware Dependencies (Current Signal Capture).....	808
Voltage Signal Capture	811
Introduction (Voltage Signal Capture).....	812
Introduction to the Function Block (Voltage Signal Capture).....	812
Overviews (Voltage Signal Capture).....	813
Overview of Ports and Basic Properties (Voltage Signal Capture).....	813
Overview of Tunable Properties (Voltage Signal Capture).....	821
Configuring the Function Block (Voltage Signal Capture).....	823
Configuring the Basic Functionality (Voltage Signal Capture).....	823
Configuring Trigger Functionality (Voltage Signal Capture).....	827
Configuring Standard Features (Voltage Signal Capture).....	832
Hardware Dependencies (Voltage Signal Capture).....	834
SCALEXIO Hardware Dependencies (Voltage Signal Capture).....	834
Voltage Signal Capture (ADC Type 4)	837
Introduction (Voltage Signal Capture - ADC Type 4).....	838
Introduction to the Function Block (Voltage Signal Capture - ADC Type 4).....	838

Overviews (Voltage Signal Capture - ADC Type 4).....	839
Overview of Ports and Basic Properties (Voltage Signal Capture - ADC Type 4).....	839
Overview of Tunable Properties (Voltage Signal Capture - ADC Type 4).....	842
Configuring the Function Block (Voltage Signal Capture - ADC Type 4).....	843
Configuring the Basic Functionality (Voltage Signal Capture - ADC Type 4).....	843
Specifying Trigger Sources (Voltage Signal Capture - ADC Type 4).....	844
Configuring the Standard Features (Voltage Signal Capture - ADC Type 4).....	848
Hardware Dependencies (Voltage Signal Capture - ADC Type 4).....	849
MicroAutoBox III Hardware Dependencies (Voltage Signal Capture - ADC Type 4).....	849
Digital Incremental Encoder In	851
Introduction (Digital Incremental Encoder In).....	852
Introduction to the Function Block (Digital Incremental Encoder In).....	852
Basics on Digital Incremental Encoders.....	853
Basics on Measuring the Position and Speed (Digital Incremental Encoder In).....	855
Overviews (Digital Incremental Encoder In).....	858
Overview of Ports and Basic Properties (Digital Incremental Encoder In).....	858
Overview of Tunable Properties (Digital Incremental Encoder In).....	862
Configuring the Function Block (Digital Incremental Encoder In).....	864
Configuring the Basic Functionality (Digital Incremental Encoder In).....	864
Using Calibration for Position Measurement (Digital Incremental Encoder In).....	869
Configuring Standard Features (Digital Incremental Encoder In).....	873
Hardware Dependencies (Digital Incremental Encoder In).....	874
SCALEXIO Hardware Dependencies (Digital Incremental Encoder In).....	874
MicroAutoBox III Hardware Dependencies (Digital Incremental Encoder In).....	875
Sine Encoder In	877
Introduction (Sine Encoder In).....	878
Introduction to the Function Block (Sine Encoder In).....	878

Basics on Analog Incremental Encoders (Sine Encoder In).....	879
Basics on Measuring the Position and Speed (Sine Encoder In).....	880
Overviews (Sine Encoder In).....	884
Overview of Ports and Basic Properties (Sine Encoder In).....	884
Overview of Tunable Properties (Sine Encoder In).....	889
Configuring the Function Block (Sine Encoder In).....	890
Configuring the Basic Functionality (Sine Encoder In).....	890
Using Calibration for Position Measurement (Sine Encoder In).....	894
Configuring Standard Features (Sine Encoder In).....	899
Hardware Dependencies (Sine Encoder In).....	900
SCALEXIO Hardware Dependencies (Sine Encoder In).....	900
Hall Encoder In	901
Introduction (Hall Encoder In).....	902
Introduction to the Function Block (Hall Encoder In).....	902
Introduction to Hall Encoders.....	903
Overviews (Hall Encoder In).....	906
Overview of Ports and Basic Properties (Hall Encoder In).....	906
Overview of Tunable Properties (Hall Encoder In).....	909
Configuring the Function Block (Hall Encoder In).....	910
Configuring the Basic Functionality (Hall Encoder In).....	910
Configuring the Standard Features (Hall Encoder In).....	916
Hardware Dependencies (Hall Encoder In).....	918
SCALEXIO Hardware Dependencies (Hall Encoder In).....	918
Resolver In	919
Introduction (Resolver In).....	920
Introduction to the Function Block (Resolver In).....	920
Introduction to Resolvers.....	921
Overviews (Resolver In).....	924
Overview of Ports and Basic Properties (Resolver In).....	924
Overview of Tunable Properties (Resolver In).....	927
Configuring the Function Block (Resolver In).....	928
Configuring the Basic Functionality (Resolver In).....	928
Configuring the Standard Features (Resolver In).....	933
Hardware Dependencies (Resolver In).....	934
SCALEXIO Hardware Dependencies (Resolver In).....	934

EnDat Master	935
Introduction (EnDat Master).....	936
Introduction to the Function Block (EnDat Master).....	936
Introduction to the EnDat Interface.....	937
Overviews (EnDat Master).....	939
Overview of Ports and Basic Properties (EnDat Master).....	939
Overview of Tunable Properties (EnDat Master).....	943
Configuring the Function Block (EnDat Master).....	944
Configuring the Basic Functionality (EnDat Master).....	944
Configuring the Standard Features (EnDat Master).....	949
Hardware Dependencies (EnDat Master).....	950
SCALEXIO Hardware Dependencies (EnDat Master).....	950
 SSI Master	951
Introduction (SSI Master).....	952
Introduction to the Function Block (SSI Master).....	952
Introduction to the SSI/BiSS-C Interfaces (SSI Master).....	953
Overviews (SSI Master).....	956
Overview of Ports and Basic Properties (SSI Master).....	956
Overview of Tunable Properties (SSI Master).....	961
Configuring the Function Block (SSI Master).....	963
Configuring the Basic Functionality (SSI Master).....	963
Configuring the Detection and Handling of Transmission Errors (SSI Master).....	969
Configuring Signal Conditioning for Data and Clock Signals (SSI Master).....	972
Predefined Settings for BiSS-C Variant Protocols (SSI Master).....	974
Configuring the Standard Features (SSI Master).....	976
Hardware Dependencies (SSI Master).....	977
SCALEXIO Hardware Dependencies (SSI Master).....	977
 Digital Incremental Encoder Out	979
Introduction (Digital Incremental Encoder Out).....	980
Introduction to the Function Block (Digital Incremental Encoder Out).....	980
Basics on the Generated Encoder Signals.....	981

Overviews (Digital Incremental Encoder Out).....	984
Overview of Ports and Basic Properties (Digital Incremental Encoder Out).....	984
Overview of Tunable Properties (Digital Incremental Encoder Out).....	987
Configuring the Function Block (Digital Incremental Encoder Out).....	988
Configuring the Basic Functionality (Digital Incremental Encoder Out).....	988
Configuring Standard Features (Digital Incremental Encoder Out).....	989
Hardware Dependencies (Digital Incremental Encoder Out).....	992
SCALEXIO Hardware Dependencies (Digital Incremental Encoder Out).....	992
Wheelspeed Out	995
Introduction (Wheelspeed Out).....	996
Introduction to the Function Block (Wheelspeed Out).....	996
Basics on Active Wheel Speed Sensors.....	997
Encoding the Data Protocol with the Wheelspeed Out Function Block.....	1000
Notes on the Data Protocol for Fast Rotating Wheels.....	1002
Notes on Simulating Non-Rotating Wheels.....	1003
Overviews (Wheelspeed Out).....	1005
Overview of Ports and Basic Properties (Wheelspeed Out).....	1005
Overview of Tunable Properties (Wheelspeed Out).....	1010
Configuring the Function Block (Wheelspeed Out).....	1012
Configuring the Basic Functionality (Wheelspeed Out).....	1012
Configuring Standard Features (Wheelspeed Out).....	1016
Hardware Dependencies (Wheelspeed Out).....	1018
SCALEXIO Hardware Dependencies (Wheelspeed Out).....	1018
Block-Commutated PWM Out	1021
Introduction (Block-Commutated PWM Out).....	1022
Introduction to the Function Block (Block-Commutated PWM Out).....	1022
Introduction to Block-Commutated PWM Signals.....	1023
Overviews (Block-Commutated PWM Out).....	1026
Overview of Ports and Basic Properties (Block-Commutated PWM Out).....	1026
Overview of Tunable Properties (Block-Commutated PWM Out).....	1032
Configuring the Function Block (Block-Commutated PWM Out).....	1033
Configuring the Basic Functionality (Block-Commutated PWM Out).....	1033

Configuring Sector and Stationary Signals (Block-Commuted PWM Out).....	1039
Configuring Standard Features (Block-Commuted PWM Out).....	1048
Hardware Dependencies (Block-Commuted PWM Out).....	1050
SCALEXIO Hardware Dependencies (Block-Commuted PWM Out).....	1050
Field-Oriented Control In/Out	1053
Introduction (Field-Oriented Control In/Out).....	1054
Introduction to the Function Block (Field-Oriented Control In/Out).....	1054
Basics on Field-Oriented Control.....	1056
Basics on Using the Function Block with Field-Oriented Control.....	1058
Basics on PWM Signal Generation.....	1059
Overviews (Field-Oriented Control In/Out).....	1061
Overview of Ports and Basic Properties (Field-Oriented Control In/Out).....	1061
Overview of Tunable Properties (Field-Oriented Control In/Out).....	1068
Configuring the Function Block (Field-Oriented Control In/Out).....	1070
Adjusting the Function Block to the Connected Motor (Field-Oriented Control In/Out).....	1070
Configuring the Generation of PWM Signals (Field-Oriented Control In/Out).....	1074
Configuring the Trigger and I/O Event Generation Functionality (Field-Oriented Control In/Out).....	1081
Configuring Standard Features (Field-Oriented Control In/Out).....	1084
Hardware Dependencies (Field-Oriented Control In/Out).....	1085
SCALEXIO Hardware Dependencies (Field-Oriented Control In/Out).....	1085
Bus Configuration	1087
Introduction (Bus Configuration).....	1088
Introduction to the Function Block (Bus Configuration).....	1088
Basics on Bus Configurations.....	1089
CAN	1091
Introduction (CAN).....	1092
Introduction to the Function Block (CAN).....	1092
Overviews (CAN).....	1094
Overview of Ports and Basic Properties (CAN).....	1094
Overview of Tunable Properties (CAN).....	1104

Configuring the Function Block (CAN).....	1106
Configuring the Basic Functionality (CAN).....	1106
Configuring Standard Features (CAN).....	1113
Hardware Dependencies (CAN).....	1115
SCALEXIO Hardware Dependencies (CAN).....	1115
MicroAutoBox III Hardware Dependencies (CAN).....	1117
FlexRay	1119
Introduction (FlexRay).....	1120
Introduction to the Function Block (FlexRay).....	1120
Overviews (FlexRay).....	1121
Overview of Ports and Basic Properties (FlexRay).....	1121
Ports and Properties of the Common Functions Function Group (FlexRay).....	1126
Ports and Properties of the Com Cyclic Control Chx Functions (FlexRay).....	1130
Ports and Properties of the Com Event Control Chx Port Functions (FlexRay).....	1132
Ports and Properties of the CTR<n> Function Groups (FlexRay).....	1135
Overview of Tunable Properties (FlexRay).....	1141
Configuring the Function Block (FlexRay).....	1142
Configuring the Basic Functionality (FlexRay).....	1142
Configuring Standard Features (FlexRay).....	1147
Hardware Dependencies (FlexRay).....	1150
SCALEXIO Hardware Dependencies (FlexRay).....	1150
MicroAutoBox III Hardware Dependencies (FlexRay).....	1151
LIN	1153
Introduction (LIN).....	1154
Introduction to the Function Block (LIN).....	1154
Overviews (LIN).....	1156
Overview of Ports and Basic Properties (LIN).....	1156
Configuring the Function Block (LIN).....	1158
Configuring the Basic Functionality (LIN).....	1158
Configuring Standard Features (LIN).....	1160
Hardware Dependencies (LIN).....	1162
SCALEXIO Hardware Dependencies (LIN).....	1162
MicroAutoBox III Hardware Dependencies (LIN).....	1163

Ethernet	1165
Introduction to Implement Ethernet Communication.....	1166
Basics on Implementing Ethernet Communication in ConfigurationDesk.....	1166
Ethernet Setup.....	1168
Introduction (Ethernet Setup).....	1168
Introduction to the Function Block (Ethernet Setup).....	1168
Overviews (Ethernet Setup).....	1169
Overview of Ports and Properties (Ethernet Setup).....	1169
Configuring the Function Block (Ethernet Setup).....	1170
Configuring the Basic Functionality (Ethernet Setup).....	1170
Assigning an Ethernet Controller (Ethernet Setup).....	1174
Configuring Standard Features (Ethernet Setup).....	1178
Virtual Ethernet Setup.....	1179
Introduction to the Functionality (Virtual Ethernet Setup).....	1179
Introduction to the Function Block.....	1179
Basics on Virtual Ethernet.....	1180
Basics on the VLAN Tag.....	1182
Overviews (Virtual Ethernet Setup).....	1183
Overview of Ports and Properties (Virtual Ethernet Setup).....	1183
Configuring the Function Block (Virtual Ethernet Setup).....	1184
Configuring the Basic Functionality (Virtual Ethernet Setup).....	1184
UDP Receive.....	1188
Introduction (UDP Receive).....	1188
Introduction to the Function Block (UDP Receive).....	1188
Basics on UDP/IP in the Signal Chain and Receiving Data Bytes.....	1189
Overviews (UDP Receive).....	1190
Overview of Ports and Properties (UDP Receive).....	1190
Configuring the Function Block (UDP Receive).....	1193
Configuring the Basic Functionality (UDP Receive).....	1193
Configuring Standard Features (UDP Receive).....	1197

UDP Transmit.....	1198
Introduction (UDP Transmit).....	1198
Introduction to the Function Block (UDP Transmit).....	1198
Basics on UDP/IP in the Signal Chain and the Transmitting of Data Bytes.....	1199
Overviews (UDP Transmit).....	1200
Overview of Ports and Properties (UDP Transmit).....	1200
Configuring the Function Block (UDP Transmit).....	1203
Configuring the Basic Functionality (UDP Transmit).....	1203
Configuring Standard Features (UDP Transmit).....	1206
TCP.....	1207
Introduction to the Functionality (TCP).....	1207
Introduction to the Function Block.....	1207
Basics on TCP/IP in the Signal Chain.....	1208
Controlling the TCP/IP Connection.....	1209
Transmitting and Receiving Data Bytes.....	1211
Overviews (TCP).....	1213
Overview of Ports and Properties (TCP).....	1213
Configuring the Function Block (TCP).....	1219
Configuring the Basic Functionality (TCP).....	1220
Configuring Standard Features (TCP).....	1224
PTP Master.....	1226
Introduction (PTP Master).....	1226
Introduction to the Function Block (PTP Master).....	1226
Basics on Using the PTP Master Function Block.....	1227
Overviews (PTP Master).....	1229
Overview of Ports and Basic Properties (PTP Master).....	1229
Configuration (PTP Master).....	1230
Configuring the Basic Functionality (PTP Master).....	1231
Configuring Standard Features (PTP Master).....	1232
PTP Slave.....	1234
Introduction (PTP Slave).....	1234
Introduction to the Function Block (PTP Slave).....	1234
Basics on Time Values.....	1235
Overviews (PTP Slave).....	1236
Overview of Ports and Basic Properties (PTP Slave).....	1236

Configuration (PTP Slave).....	1238
Configuring the Basic Functionality (PTP Slave).....	1238
Configuring Standard Features (PTP Slave).....	1240
Ethernet Switch.....	1241
Introduction (Ethernet Switch).....	1241
Introduction to the Function Block (Ethernet Switch).....	1241
Basics on Ethernet Switches of the dSPACE Hardware.....	1242
Overviews (Ethernet Switch).....	1245
Overview of Ports and Basic Properties (Ethernet Switch).....	1245
Configuring the Function Block (Ethernet Switch).....	1247
Configuring the Basic Functionality (Ethernet Switch).....	1248
Configuring VLANs Based on the Protocol (Ethernet Switch).....	1251
SENT Communication (SENT In, SENT Out)	1255
Introduction to SENT Communication.....	1256
Basics on the SENT Protocol.....	1256
Using Application-Specific Protocols.....	1260
Basics on User-Defined Protocol Files.....	1262
Basics on Multiplexing SENT Messages.....	1263
Creating User-Defined Protocol Files.....	1266
SENT In.....	1276
Introduction (SENT In).....	1276
Introduction to the Function Block (SENT In).....	1276
Overviews (SENT In).....	1277
Overview of Ports and Basic Properties (SENT In).....	1277
Overview of Tunable Properties (SENT In).....	1289
Configuring the Function Block (SENT In).....	1290
Configuring the Basic Functionality (SENT In).....	1290
Using Application-Specific Protocols to Decode SENT Messages.....	1293
Reading Nibble Values to Decode SENT Messages.....	1295
Configuring Standard Features (SENT In).....	1296
Hardware Dependencies (SENT In).....	1299
SCALEXIO Hardware Dependencies (SENT In).....	1300
MicroAutoBox III Hardware Dependencies (SENT In).....	1301
SENT Out.....	1303
Introduction (SENT Out).....	1303
Introduction to the Function Block (SENT Out).....	1303

Basics on the Message Buffer.....	1304
Basics on Simulating Faulty SENT Messages.....	1306
Overviews (SENT Out).....	1308
Overview of Ports and Basic Properties (SENT Out).....	1308
Overview of Tunable Properties (SENT Out).....	1320
Configuring the Function Block (SENT Out).....	1321
Configuring the Basic Functionality (SENT Out).....	1321
Using Application-Specific Protocols to Encode SENT Messages.....	1323
Encoding SENT Messages in the Behavior Model.....	1326
Configuring Standard Features (SENT Out).....	1327
Hardware Dependencies (SENT Out).....	1329
SCALEXIO Hardware Dependencies (SENT Out).....	1329
SPI Master	1331
Introduction (SPI Master).....	1332
Introduction to the Function Block (SPI Master).....	1332
Basics on Serial Peripheral Interfaces (SPI Master).....	1333
Overviews (SPI Master).....	1336
Overview of Ports and Basic Properties (SPI Master).....	1336
Overview of Tunable Properties (SPI Master)	1341
Configuring the Function Block (SPI Master).....	1342
Configuring the Basic Functionality (SPI Master).....	1342
Configuring SPI Cycles (SPI Master).....	1343
Configuring Standard Features (SPI Master).....	1348
Hardware Dependencies (SPI Master).....	1350
MicroAutoBox III Hardware Dependencies (SPI Master).....	1350
Gigalink	1353
Introduction (Gigalink).....	1354
Introduction to the Function Block (Gigalink).....	1354
Overviews (Gigalink).....	1355
Overview of Ports and Basic Properties (Gigalink).....	1355
Overview of Tunable Properties (Gigalink).....	1356
Configuring the Function Block (Gigalink).....	1357
Configuring the Basic Functionality (Gigalink).....	1357
Configuring Standard Features (Gigalink).....	1359

ECU Interface Configuration 1361

Introduction (ECU Interface Configuration).....	1362
Introduction to the Function Block (ECU Interface Configuration).....	1362
Overviews (ECU Interface Configuration).....	1364
Overview of Ports and Basic Properties (ECU Interface Configuration).....	1364
Configuring the Function Block (ECU Interface Configuration).....	1377
Configuring the Basic Functionality (ECU Interface Configuration).....	1377
Configuring Standard Features (ECU Interface Configuration).....	1379

Power Supply Control 1381

Introduction (Power Supply Control).....	1382
Introduction to the Function Block (Power Supply Control).....	1382
Overviews (Power Supply Control).....	1383
Overview of Ports and Basic Properties (Power Supply Control).....	1383
Overview of Tunable Properties (Power Supply Control).....	1386
Configuring the Function Block (Power Supply Control).....	1388
Configuring the Basic Functionality (Power Supply Control).....	1388
Configuring Standard Features (Power Supply Control).....	1389

Power Switch 1391

Introduction (Power Switch).....	1392
Introduction to the Function Block (Power Switch).....	1392
Overviews (Power Switch).....	1393
Overview of Ports and Basic Properties (Power Switch).....	1393
Overview of Tunable Properties (Power Switch).....	1394
Configuring the Function Block (Power Switch).....	1396
Configuring the Basic Functionality (Power Switch).....	1396
Configuring Standard Features (Power Switch).....	1397
Hardware Dependencies (Power Switch).....	1398
SCALEXIO Hardware Dependencies (Power Switch).....	1398

Power On Signal In 1399

Introduction (Power On Signal In).....	1400
Introduction to the Function Block (Power On Signal In).....	1400

Overviews (Power On Signal In).....	1401
Overview of Ports and Basic Properties (Power On Signal In).....	1401
Overview of Tunable Properties (Power On Signal In)	1402
Configuring the Function Block (Power On Signal In).....	1404
Configuring Standard Features (Power On Signal In).....	1404
System Shutdown	1405
Introduction (System Shutdown).....	1406
Introduction to the Function Block (System Shutdown).....	1406
Basics on Using the System Shutdown Functionality.....	1407
Overviews (System Shutdown).....	1411
Overview of Ports and Basic Properties (System Shutdown).....	1411
Tunable Properties (System Shutdown)	1412
Configuring the Function Block (System Shutdown).....	1413
Configuring Standard Features (System Shutdown).....	1413
System Temperature Monitoring	1415
Introduction (System Temperature Monitoring).....	1416
Introduction to the Function Block (System Temperature Monitoring)	1416
Overviews (System Temperature Monitoring).....	1417
Overview of Ports and Basic Properties (System Temperature Monitoring).....	1417
Overview of Tuneable Properties (System Temperature Monitoring).....	1418
Configuration (System Temperature Monitoring).....	1419
Configuring the Standard Features (System Temperature Monitoring).....	1419
USB Eject	1421
Introduction.....	1422
Introduction to the Function Block (USB Eject).....	1422
Overviews.....	1423
Overview of Ports and Basic Properties (USB Eject).....	1423
Overview of Tunable Properties (USB Eject).....	1424
Configuration.....	1425
Configuring the Standard Features (USB Eject).....	1425

FuSa Setup	1427
Introduction (FuSa Setup).....	1428
Introduction to the Function Block (FuSa Setup).....	1428
Basics on Using FuSa with the MicroAutoBox III.....	1429
Overviews (FuSa Setup).....	1432
Overview of Ports and Basic Properties (FuSa Setup).....	1432
Overview of Tunable Properties (FuSa Setup).....	1433
Configuring the Function Block (FuSa Setup).....	1434
Configuring the Basic Functionality (FuSa Setup).....	1434
Configuring Standard Features (FuSa Setup).....	1435
FuSa Response Trigger	1437
Introduction (FuSa Response Trigger).....	1438
Introduction to the Function Block (FuSa Response Trigger).....	1438
Overviews (FuSa Response Trigger).....	1439
Overview of Ports and Basic Properties (FuSa Response Trigger).....	1439
Overview of Tunable Properties (FuSa Response Trigger).....	1439
Configuring the Function Block (FuSa Response Trigger).....	1441
Configuring the Basic Functionality (FuSa Response Trigger).....	1441
Configuring Standard Features (FuSa Response Trigger).....	1442
FuSa Challenge-Response Monitoring	1443
Introduction (FuSa Challenge-Response Monitoring).....	1444
Introduction to the Function Block (FuSa Challenge-Response Monitoring).....	1444
Overviews (FuSa Challenge-Response Monitoring).....	1446
Overview of Ports and Basic Properties (FuSa Challenge-Response Monitoring).....	1446
Overview of Tunable Properties (FuSa Challenge-Response Monitoring).....	1449
Configuring the Function Block (FuSa Challenge-Response Monitoring).....	1450
Configuring the Basic Functionality (FuSa Challenge-Response Monitoring).....	1450
Configuring Standard Features (FuSa Challenge-Response Monitoring).....	1455

FuSa System Monitoring	1457
Introduction (FuSa System Monitoring).....	1458
Introduction to the Function Block (FuSa System Monitoring).....	1458
Overviews (FuSa System Monitoring).....	1459
Overview of Ports and Basic Properties (FuSa System Monitoring).....	1459
Configuring the Function Block (FuSa System Monitoring).....	1460
Configuring the Basic Functionality (FuSa System Monitoring).....	1460
Domain Clock	1463
Introduction (Domain Clock).....	1464
Introduction to the Function Block (Domain Clock).....	1464
Basics on Using the Domain Clock Function Block.....	1465
Overviews (Domain Clock).....	1467
Overview of Ports and Basic Properties (Domain Clock).....	1467
Configuration (Domain Clock).....	1469
Configuring Standard Features (Domain Clock).....	1469
Non-Volatile Memory Access	1471
Introduction (Non-Volatile Memory Access).....	1472
Introduction to the Function Block (Non-Volatile Memory Access).....	1472
Basics on Using the Non-Volatile Memory of dSPACE Processing Hardware.....	1473
Overviews (Non-Volatile Memory Access).....	1475
Overview of Ports and Basic Properties (Non-Volatile Memory Access).....	1475
Overview of Tunable Properties (Non-Volatile Memory Access)	1476
Configuring the Function Block (Non-Volatile Memory Access).....	1478
Configuring the Basic Functionality (Non-Volatile Memory Access).....	1478
Configuring the Standard Features (Non-Volatile Memory Access).....	1478
Acceleration In	1481
Introduction (Acceleration In).....	1482
Introduction to the Function Block (Acceleration In).....	1482
Overviews (Acceleration In).....	1483
Overview of Ports and Basic Properties (Acceleration In).....	1483
Overview of Tunable Properties (Acceleration In).....	1484

Configuring the Function Block (Acceleration In).....	1485
Configuring the Basic Functionality (Acceleration In).....	1485
Configuring the Standard Features (Acceleration In).....	1487
Atmospheric Pressure In	1489
Introduction.....	1490
Introduction to the Function Block (Atmospheric Pressure In).....	1490
Overviews.....	1491
Overview of Ports and Basic Properties (Atmospheric Pressure In).....	1491
Overview of Tunable Properties (Atmospheric Pressure In).....	1491
Configuration.....	1493
Configuring the Standard Features (Atmospheric Pressure In).....	1493
LED Out	1495
Introduction (LED Out).....	1496
Introduction to the Function Block (LED Out).....	1496
Overviews (LED Out).....	1497
Overview of Ports and Basic Properties (LED Out).....	1497
Overview of Tunable Properties (LED Out).....	1497
Configuring the Function Block (LED Out).....	1499
Configuring the Basic Functionality (LED Out).....	1499
Configuring the Standard Features (LED Out).....	1499
Implementing FPGA Custom Function Blocks	1501
Introduction (FPGA).....	1502
Introduction to the Function Block (FPGA).....	1502
Steps to Implement an FPGA Application.....	1502
Types of FPGA Applications.....	1505
Identifying Elements of the RTI FPGA Programming Blockset.....	1509
Handling FPGA Custom Function Blocks in ConfigurationDesk.....	1511
Hardware Dependencies of FPGA Custom Function Blocks.....	1511
How to Add FPGA Applications to ConfigurationDesk.....	1513
How to Check Hardware Resources Required for FPGA Custom Function Blocks (SCALEXIO).....	1515
How to Update FPGA Custom Function Blocks.....	1518
How to Open the FPGA Model in MATLAB/Simulink.....	1519

Overviews (FPGA).....	1520
Overview of Ports and Properties (FPGA).....	1520
Overview of Tunable Properties (FPGA)	1524
Configuring the Function Block (FPGA).....	1525
Configuring the Basic Functionality (FPGA).....	1525
Configuring Standard Features (FPGA).....	1531
Accessing FPGA Applications At Run Time.....	1532
Accessing the FPGA Application with the Experiment Software.....	1532
Initializing of the RAM Used by the FPGA Application.....	1536
How to Initialize the DDR4 RAM of the DS6602.....	1536
General Hardware Dependencies	1539
Optimal Selection of Analog Output Channels on DS2680 and DS6101.....	1539
Circuit Diagrams of Channel Types	1541
Analog In 1.....	1544
Analog In 2.....	1546
Analog In 4.....	1546
Analog In 5.....	1547
Analog In 6.....	1547
Analog In 7.....	1548
Analog In 8.....	1548
Analog In 9.....	1549
Analog In 10.....	1549
Analog In 11.....	1550
Analog In 12.....	1550
Analog In 14.....	1551
Analog In 15	1551
Analog In 16.....	1552
Analog In 17.....	1552
Analog Out 1.....	1553
Analog Out 2.....	1554
Analog Out 3.....	1554
Analog Out 4.....	1555
Analog Out 6.....	1556
Analog Out 7.....	1556
Analog Out 8.....	1557
Analog Out 9.....	1557

Analog Out 10.....	1558
Analog Out 11.....	1559
Analog Out 12.....	1559
Analog Out 13.....	1560
Bus 1	1560
CAN 1.....	1563
CAN 2.....	1564
CAN 3.....	1566
CAN 4.....	1566
CAN 5.....	1567
CAN 6.....	1568
Digital In 1.....	1569
Digital In 2.....	1571
Digital In 3.....	1572
Digital In 4.....	1572
Digital In 5.....	1573
Digital In 9.....	1573
Digital In 10.....	1574
Digital Out 1.....	1575
Digital Out 2.....	1576
Digital Out 3.....	1577
Digital Out 4.....	1578
Digital Out 5.....	1579
Digital Out 7.....	1580
Digital Out 8.....	1581
Digital In/Out 1.....	1582
Digital In/Out 3.....	1584
Digital In/Out 5.....	1586
Digital In/Out 6.....	1588
Digital In/Out 8.....	1589
Digital In/Out 9.....	1590
Digital In/Out 10.....	1591
Flexible In 1.....	1593
Flexible In 2.....	1595
Flexible In 3.....	1603
Flexible Out 1.....	1604
Flexible In/Out 1.....	1608
FlexRay 1.....	1609
FlexRay 2.....	1610
FlexRay 3.....	1611
FlexRay 4.....	1613
LIN 1.....	1615

LIN 2.....	1615
LIN 3.....	1616
LIN 4.....	1616
Power Switch 1.....	1617
Power Switch 2.....	1618
Resistance Out 1.....	1619
Resistance Out 2.....	1620
Resolver In 2.....	1621
Trigger In 1.....	1621
Trigger In 2.....	1622
Trigger In 3.....	1622
UART 1.....	1623
UART 2.....	1624
UART 3.....	1625
UART 4.....	1626

Index	1627
--------------	-------------

About This Document

Contents

This document provides feature-oriented access to the information you need to work with ConfigurationDesk's function blocks.

Required knowledge

Knowledge in handling the host PC and the Microsoft Windows operating system is presupposed.

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
 DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
 WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
 CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
 NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
 Note	Indicates important information that you should take into account to avoid malfunctions.
 Tip	Indicates tips that can make your work easier.
 ?	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
 	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Special folders

Some software products use the following special folders:

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

`%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>`

or

`%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>`

Documents folder A standard folder for user-specific documents.

`%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>`

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

`%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>`

Accessing dSPACE Help and PDF Files

After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com/go/help.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the  icon in dSPACE Help. The PDF opens on the first page.

Introduction

Motivation

Implementing and configuring I/O functionality is one of the main work steps in implementing a real-time application. In ConfigurationDesk, I/O functionality is added to the signal chain via function blocks which are instantiated from the Function Library.

Where to go from here**Information in this section**

Overview on the Available Function Block Types.....	41
Allocating SCALEXIO Channel Types to Function Block Types.....	51
Allocating MicroAutoBox III Channel Types to Function Block Types.....	56

Overview on the Available Function Block Types

Introduction

The following overview will give you a quick introduction to the function block types. It also guides you to the corresponding chapters in this documentation for further information.

Basic I/O

Function Block Type	Purpose
Current In	The Current In function block type digitizes analog current signals coming from an external device.

Function Block Type	Purpose
Triggered Current In	The Triggered Current In function block type features the digitization of PWM-synchronous current signals coming from an external device, for example, to measure average currents of PWM driven valves and motors.
Voltage In	The Voltage In function block type digitizes analog voltage signals coming from an external device.
Current Sink	The Current Sink function block type lets you simulate sensors with a current interface like most Hall sensors.
Voltage Out	The Voltage Out function block provides the possibility to output analog voltages.
Multi Bit In	The Multi Bit In function block type lets you measure digital signals coming from an external device.
Trigger In	The Trigger In function block type generates a trigger signal each time the external input signal matches the defined triggering conditions. The function block works as a provider: Other function blocks can use the generated trigger signal as trigger source.
Multi Bit Out	The Multi Bit Out function block type lets you stimulate digital inputs of an external device.
Digital Pulse Out	The Digital Pulse Out function block generates a digital pulse with each model step of the behavior model or with each trigger event of another function block.
Potentiometer Out	The Potentiometer Out function block type allows adjustable three-terminal resistances (potentiometers) to be simulated.
Resistance Out	The Resistance Out function block type lets you simulate resistances.
PWM/PFM Out	The PWM/PFM Out function block type can be used to generate one-phase pulse-width modulated signals and frequency output signals.
Multi-Channel PWM Out	The Multi-Channel PWM Out function block type synchronously generates multiple PWM signals with a common frequency. The function block can work as a provider: Other function blocks can use the generated trigger signal as a trigger source.
PWM/PFM In	With the PWM/PFM In function block, you can measure one-phase pulse-width-modulated signal patterns.
Digital Pulse In	The Digital Pulse In function block type lets you measure the pulse duration of digital voltage signals coming from an external device.
Digital Pulse Capture	The Digital Pulse Capture function block type converts signals coming from an external device (e.g., ECU) to digital pulses.

Advanced I/O

Function Block Type	Purpose
Wavetable Current Sink	The Wavetable Current Sink function block type outputs a current sink signal sequence by looking up a table with time-coded values.
	The Wavetable Digital Out function block type outputs a digital signal sequence by looking up a table with time-coded values.
	The Wavetable Voltage Out function block type outputs a voltage signal sequence by looking up a table with time-coded values.
Angular Wavetable Voltage Out	The Angular Wavetable Voltage Out function block outputs a voltage signal sequence by looking up a table with angular-coded values.
	The Angular Wavetable Digital Out function block type outputs a digital signal sequence by looking up a table with angular-coded values.
Waveform Voltage Out	The Waveform Current Sink function block type lets you simulate sensors with a current interface, like Hall sensors. The load can be changed at run time without the concurrence of the behavior model, for example, to produce cyclic current profiles.
	The Waveform Voltage Out function block type can be used to generate periodic voltage output signals, for example, to simulate signals generated by combined oil temperature and pressure sensors in combustion engines.
	The Waveform Digital Out function block type can be used to generate periodic pulses, for example, to simulate low-speed data packages (protocols).
Current Signal Capture	With the Current Signal Capture function block type, you can measure analog current signals (coming from an external device, e.g., ECU) by capturing signal sequences at configurable sample rates.
Voltage Signal Capture	The Voltage Signal Capture function block type measures analog voltage signals from an external device by capturing signal sequences at configurable sample rates.
Voltage Signal Capture (ADC Type 4)	With the Voltage Signal Capture (ADC Type 4) function block, you can measure analog voltage signals (coming from an external device) by capturing signal sequences, for example, at configurable sample rates. The function block type is exclusively designed to be used for the ADC Type 4 module of the DS1511, DS1511B1, and DS1513 Multi-I/O Boards.
Digital Incremental Encoder In	The Digital Incremental Encoder In function block provides access to rotary or linear digital incremental encoders. The function block can be used, for example, to measure the angular position and the speed of an electric motor.
Digital Incremental Encoder Out	The Digital Incremental Encoder Out function block simulates the signals provided by a rotary or linear incremental encoder. The

Function Block Type	Purpose
	function block supports the generation of three digital output signals (Phi0, Phi90 and Index).
Wheelspeed Out	The Wheelspeed Out function block type simulates the signals provided by an active wheel speed sensor.
ECU Interface Configuration	The ECU Interface Configuration function block configures the data exchange between a real-time application (executed on SCALEXIO or MicroAutoBox III) and an ECU application (executed on an ECU) for ECU interfacing.
Sine Encoder In	The Sine Encoder In function block provides access to rotary or linear analog incremental encoders that provide sinusoidal output signals. The function block can be used, for example, to measure the angular position and the speed of an electric motor.
Hall Encoder In	The Hall Encoder In function block provides access to Hall encoders with differential and single-ended signals. The function block can be used to determine the angular position and the speed of an electric motor.
Resolver In	The Resolver In function block type provides access to resolvers to determine the angular position and the speed, for example, of an electric motor.
EnDat Master	The EnDat Master function block provides access to absolute encoders (single-turn and multi-turn encoders) via the EnDat 2.1 or EnDat 2.2 protocol. The function block can be used to determine the angular position, the revolution count, and the speed, for example, of an electric motor.
SSI Master	The SSI Master function block provides access to absolute encoders (single-turn and multiturn encoders) that support the SSI or the BiSS-C interface. The function block can be used to determine the angular position, the revolution count, and the speed of an electric motor, for example.

Engine I/O

Function Block Type	Purpose
Engine Control Setup	The Engine Control Setup function block specifies basic characteristics of an existing (real) piston engine that you want to control. The function block works as a provider: Other function blocks can use it to obtain information on the characteristics of the specified piston engine.

Function Block Type	Purpose
Crank In	The Crank In function block type lets you measure the crankshaft of a piston engine and calculates the current position, speed and rotational direction of the engine. For performing these measurements, the Crank In function block must be used in combination with at least one Cam In function block.
Cam In	The Cam In function block type lets you measure the phase shift angle between a camshaft and the coupled crankshaft. Cam In must be used with the Crank In function block for synchronizing with the provided master APU and to evaluate the current engine position.
Injection Out	The Injection Out function block type lets you generate injection pulses for a real piston engine.
Ignition Out	The Ignition Out function block type lets you generate ignition pulses for a real piston engine.
Engine Angular Pulse Out	The Engine Angular Pulse Out function block generates a periodic pulse pattern based on engine position data retrieved from an assigned master APU provider. The generated pulse pattern can be used, for example, for starting A/D conversions.
Knock In	The Knock In function block lets you analyze the noise of a piston engine to avoid/minimize preignitions caused by improper ignition timing.
Lambda Probe In	The Lambda Probe In function block lets you control LSU 4.9 and LSU ADV wideband lambda probes.
Engine Simulation Setup	The Engine Simulation Setup function block type defines the basic design of virtual four-stroke piston engines, for example, number of cylinders.
Angular Clock Setup	The Angular Clock Setup function block configures and initializes the access to an angular processing unit (APU). The function block works as a provider: Other function blocks can use it to access an APU.

Function Block Type	Purpose
Crank/Cam Voltage Out	The Crank/Cam Current Sink function block type lets you output current sink signals to simulate digital crankshaft, direction sensitive (reverse) crankshaft, and camshaft sensors.
	The Crank/Cam Digital Out function block type lets you output digital signals to simulate digital crankshaft, direction-sensitive crankshaft, and camshaft sensors.
	The Crank/Cam Voltage Out function block type lets you output voltage signals to simulate inductive crankshaft and camshaft sensors.
Injection/Ignition Voltage In	The Injection/Ignition Current In function block type reads the positions and durations of the injection pulses that were generated by the ECU on the basis of a current input signal.
	The Injection/Ignition Voltage In function block type reads the positions and durations of the injection pulses that were generated by the ECU on the basis of a voltage input signal.
Knock Signal Out	The Knock Signal Out function block type outputs voltage signals to simulate automotive knock sensor signals.
Lambda Probe Simulation	With the Lambda DCR function block, you can simulate wide-band lambda probes. The functionality of the function block depends on the direct current regulation (DCR) method.
	With the Lambda NCCR function block, you can simulate wide-band lambda probes. The function block type depends on the current regulation method, i.e., Nernst-controlled current regulation (NCCR).

E-Drive I/O

Function Block Type	Purpose
Block-Commutated PWM Out	The Block-Commutated PWM Out function block generates block-commutated PWM signals to control three-phase brushless DC (BLDC) motors.
Field-Oriented Control In/Out	The Field-Oriented Control In/Out function block generates PWM signals to drive, for example, permanent magnet synchronous motors. The function block provides all the processing required for controlling three-phase motors via the field-oriented control method, such as transforming voltage and current vectors to space vectors and vice versa.

Communication

Function Block Type	Purpose
Bus Configuration	The Bus Configuration function block type is used to implement CAN and LIN bus communication on the basis of imported communication matrix files. The function block type is an element of ConfigurationDesk's Bus Manager. For further information, refer to ConfigurationDesk Bus Manager Implementation Guide .
CAN	The CAN function block type is one part of implementing CAN communication in real-time applications. It lets you specify the hardware access for CAN communication. For example, you can configure transceiver settings for the assigned bus channel. The CAN communication itself must be modeled and supplied via one of the following providers: <ul style="list-style-type: none"> ▪ RTI CAN MultiMessage Blockset (refer to Modeling a CAN Bus Interface (Model Interface Package for Simulink - Modeling Guide)) ▪ Bus Manager (refer to Introduction to the Bus Manager (ConfigurationDesk Bus Manager Implementation Guide)) ▪ V-ECU implementations (refer to Special Aspects of V-ECU Implementations Containing CAN Controllers (ConfigurationDesk Real-Time Implementation Guide)) ▪ Bus simulation containers (refer to Working with Bus Simulation Containers (ConfigurationDesk Real-Time Implementation Guide)) ▪ ECU interface containers (refer to ECU Interfacing with SCALEXIO or MicroAutoBox III Systems (ConfigurationDesk Real-Time Implementation Guide)) In addition, you can use the CAN function block to initialize bus channels that are not involved in CAN communication. These channels can then be accessed during run time via ControlDesk or Real-Time Testing (RTT) (e.g., for monitoring purposes).
FlexRay	The FlexRay function block type is one part of implementing FlexRay communication in real-time applications. It lets you specify the hardware access for FlexRay communication and control the communication of the FlexRay network separately for each FlexRay controller and FlexRay channel (A and/or B). The FlexRay communication itself must be modeled and supplied via the dSPACE FlexRay Configuration Package (refer to Modeling a FlexRay Bus Interface (Model Interface Package for Simulink - Modeling Guide)).
LIN	The LIN function block type is one part of implementing LIN communication in real-time applications. It lets you specify the hardware access for LIN communication. For example, you can configure transceiver settings for the assigned bus channel. The LIN

Function Block Type	Purpose
	<p>communication itself must be modeled and supplied via one of the following providers:</p> <ul style="list-style-type: none"> ▪ RTI LIN MultiMessage Blockset (refer to Modeling a LIN Bus Interface (Model Interface Package for Simulink - Modeling Guide)) ▪ Bus Manager (refer to Introduction to the Bus Manager (ConfigurationDesk Bus Manager Implementation Guide)) ▪ V-ECU implementations (refer to Special Aspects of V-ECU Implementations Containing LIN Controllers (ConfigurationDesk Real-Time Implementation Guide)) ▪ Bus simulation containers (refer to Working with Bus Simulation Containers (ConfigurationDesk Real-Time Implementation Guide)) <p>In addition, you can use the LIN function block to initialize bus channels that are not involved in LIN communication. These channels can then be accessed during run time via ControlDesk or Real-Time Testing (RTT) (e.g., for monitoring purposes).</p>
Ethernet Setup	<p>With the Ethernet Setup function block type, you can configure and initialize the access to an Ethernet controller of your dSPACE real-time hardware. The function block works as a provider: Other function blocks can use it to access the configured Ethernet controller.</p>
Virtual Ethernet Setup	<p>With the Virtual Ethernet Setup function block type, you can configure a virtual Ethernet controller to access an Ethernet network, including VLAN. The function block works as a provider: Other function blocks can use it to access the configured Ethernet controller.</p>
UDP Receive	<p>The UDP Receive function block type receives data bytes via UDP messages. The access to the Ethernet controller must be provided by an Ethernet Setup or Virtual Ethernet Setup function block.</p>
UDP Transmit	<p>The UDP Transmit function block type transmits data bytes via UDP messages. The access to the Ethernet controller must be provided by an Ethernet Setup or Virtual Ethernet Setup function block.</p>
TCP	<p>The TCP function block type can transmit and receive data bytes via TCP messages. The access to the Ethernet controller must be provided by an Ethernet Setup or Virtual Ethernet Setup function block.</p>
PTP Master	<p>The PTP Master function block provides a master time to which Ethernet devices can synchronize their clocks via a precision time protocol (PTP).</p>
PTP Slave	<p>The PTP Slave function block provides time values to the behavior model that are synchronized via a precision time protocol (PTP) to an Ethernet time master.</p>

Function Block Type	Purpose
Ethernet Switch	The Ethernet Switch function block lets you configure the switching of Ethernet traffic and the characteristics of the physical layer transceivers (PHYs).
SENT In	The SENT In function block receives SENT messages. SENT is a protocol used between sensors and ECUs to transmit data of high-resolution sensors as an alternative to an analog interface. The sensors are typically throttle position sensors or mass air flow sensors.
SENT Out	The SENT Out function block type transmits SENT messages. SENT is a protocol used between sensors and ECUs to transmit data of high-resolution sensors as an alternative to an analog interface. The sensors are typically throttle position sensors or mass air flow sensors.
SPI Master	The SPI Master function block controls and performs a short-distance communication via the serial peripheral interface (SPI). SPI communication is a master-slave architecture with a single master.
Gigalink	The Gigalink function block type realizes communication between a SCALEXIO system and a PHS-bus based system (DS1006 or DS1007 modular system) or between two SCALEXIO systems.

System

Function Block Type	Purpose
Power Supply Control	The Power Supply Control function block type gives you access to the battery simulation power supply controller.
Power Switch	The Power Switch function block type provides switched battery simulation supply outputs for connected external devices (e.g., ECU).
Domain Clock	The Domain Clock function block represents a domain clock. The base time can be set and is updated by the processing hardware.
Power On Signal In	The Power On Signal In function block type monitors hardware based shutdown requests for the SCALEXIO LabBox (with DS6001 Processor Board), SCALEXIO AutoBox (with DS6001 Processor Board) and MicroAutoBox III. The request state can be provided to the behavior model and/or can be accessed in the experiment software.
System Shutdown	The System Shutdown function block type lets you control the shutdown of a SCALEXIO LabBox or SCALEXIO AutoBox (both with a DS6001 Processor Board installed) or a MicroAutoBox III from within the behavior model and/or the experiment software. Hardware based shutdown requests can be ignored or delayed.
System Temperature Monitoring	With the System Temperature Monitoring function block type, you can monitor the internal temperature of MicroAutoBox III via a real-time application.

Function Block Type	Purpose
USB Eject	The USB Eject function block provides a trigger from the behavior model to eject (unmount) a USB mass storage device from the dSPACE system. Removing a USB device from the USB port without unmounting can result in data loss.
FuSa Setup	The FuSa Setup function block provides the basic functionality for implementing functional safety in your system. The function block triggers basic error responses and lets you enable additional error responses.
FuSa Response Trigger	Each FuSa Response Trigger function block works as a user-configurable software monitor to implement functional safety in your application. If you use this function block, a Functional Safety (FuSa) error can be triggered directly by software from within the behavior model.
FuSa Challenge-Response Monitoring	Each FuSa Challenge-Response Monitoring function block works as a challenge-response monitor to support functional safety in your application. Based on the challenge and response principle, you can implement advanced monitoring services, for example, monitor periodic tasks or the correct execution of subsystems.
FuSa System Monitoring	The FuSa System Monitoring function block works as a hardware-related monitor to enhance functional safety in your application. If you use this function block, a Functional Safety (FuSa) error can be triggered directly by any activated monitor, for example, if the internal operating temperature of the MicroAutoBox III exceeds a defined upper limit or the operating voltage is out of a defined range.
Non-Volatile Memory Access	The Non-Volatile Memory Access function block provides access to the non-volatile memory of the processing hardware. The function block creates a data set in the non-volatile memory and handles the data transfer between the behavior model and the non-volatile memory for this data set.
Acceleration In	The Acceleration In function block type reads the measured acceleration and angular velocity values on three axes (x, y, z) from an onboard acceleration sensor and provides the values to the behavior model.
Atmospheric Pressure In	The Atmospheric Pressure In function block type reads the measured atmospheric pressure value from an onboard pressure sensor and provides the value to the behavior model.
LED Out	Each LED Out function block type provides access to one of the 4 user LEDs on the MicroAutoBox III. The LEDs can be controlled from the behavior model, for example, to display status information on the real-time application.

Custom Functions

Function Block Type	Purpose
FPGA Custom Function	FPGA custom function blocks contain the functionality of an FPGA application that must be defined with the RTI FPGA Programming Blockset.

Allocating SCALEXIO Channel Types to Function Block Types

Introduction

Every function block type supports one or more suitable channel types.

The following gives you an overview on the allocation of the function block types to SCALEXIO channel types and shows you the efficiency of each channel type.

I/O

The following tables list the function block types for I/O functionality.

Function Block Type		Channel Type																		
		Flexible In 1	Flexible In 2	Flexible In 3	Digital In 1	Digital In 2	Digital In 3	Trigger In 1	Trigger In 2	Analog In 1	Analog In 2	Analog In 4	Analog In 5	Analog In 6	Analog In 16	Flexible Out 1	Digital Out 1	Digital Out 2	Digital Out 3	Digital Out 8
Analog Input	Current In	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	Triggered Current In	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	Voltage In	✓	—	✓	—	—	—	—	—	✓	—	✓	—	✓	✓	—	—	—	—	
Analog Output	Current Sink	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	
	Voltage Out	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	
Digital Input	Multi Bit In	✓	✓	✓	✓	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—	
	Trigger In	✓	✓	✓	✓	✓	—	✓	✓	✓	—	—	—	—	—	—	—	—	—	
Digital Output	Multi Bit Out	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	✓	✓	
	Digital Pulse Out	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	—	✓	
Resistor Simulation	Potentiometer Out	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	
	Resistance Out	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	

Function Block Type		Channel Type																	
		Flexible In 1	Flexible In 2	Flexible In 3	Digital In 1	Digital In 2	Digital In 3	Trigger In 1	Trigger In 2	Analog In 1	Analog In 2	Analog In 4	Analog In 5	Analog In 6	Analog In 16	Flexible Out 1	Digital Out 1	Digital Out 2	Digital Out 3
Pulse Pattern Generation	PWM/PFM Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	✓	✓	✓
	Multi-channel PWM Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
	Block-Commutated PWM Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
	Field-Oriented Control In/Out	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	✓
Pulse Pattern Measurement	Digital Pulse Capture	✓	✓	✓	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-
	PWM/PFM In	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-
Engine Simulation	Crank/Cam Voltage Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Crank/Cam Current Sink	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Crank/Cam Digital Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	-
	Injection/Ignition Voltage In	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Injection/Ignition Current In	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Knock Signal Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Lambda NCCR	-	-	-	-	-	-	-	-	-	✓	-	✓	-	-	-	-	-	-
Wavetable Generation	Wavetable Current Sink	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Wavetable Digital Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	-
	Wavetable Voltage Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Angular Wavetable Voltage Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	-
	Angular Wavetable Digital Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-

Function Block Type		Channel Type																	
		Flexible In 1	Flexible In 2	Flexible In 3	Digital In 1	Digital In 2	Digital In 3	Trigger In 1	Trigger In 2	Analog In 1	Analog In 2	Analog In 4	Analog In 5	Analog In 6	Analog In 16	Flexible Out 1	Digital Out 1	Digital Out 2	Digital Out 3
Waveform Generation	Waveform Current Sink	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Waveform Digital Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	-
	Waveform Voltage Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
Signal Capturing	Current Signal Capture	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Voltage Signal Capture	✓	-	✓	-	-	-	-	-	✓	-	✓	-	✓	✓	-	-	-	-
Extended Sensor Signal Capture	Digital Incremental Encoder In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Sine Encoder In	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Hall Encoder In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Resolver In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	EnDat Master	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	SSI Master	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Extended Sensor Simulation	Digital Incremental Encoder Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	-
	Wheelspeed Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
SENT Protocol	SENT In	✓	✓	✓	✓	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-
	SENT Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	-

Function Block Type		Channel Type																
		Flexible In/Out 1	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9	Analog Out 1	Analog Out 2	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 7	Analog Out 8	Analog Out 9	Analog Out 10	Resistance Out 1	Resistance Out 2	Load 1
Analog Input	Current In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Triggered Current In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Voltage In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Analog Output	Current Sink	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-
	Voltage Out	-	-	-	-	-	✓	✓	-	✓	✓	✓	-	✓	✓	-	-	-
Digital Input	Multi Bit In	-	✓	✓	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-
	Trigger In	-	-	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-

Function Block Type		Channel Type																	
		Flexible In/Out 1	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9	Analog Out 1	Analog Out 2	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 7	Analog Out 8	Analog Out 9	Analog Out 10	Resistance Out 1	Resistance Out 2	Load 1	Resolver In 2
Digital Output	Multi Bit Out	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
	Digital Pulse Out	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Resistor Simulation	Potentiometer Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-
	Resistance Out	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-
Pulse Pattern Generation	PWM/PFM Out	-	✓	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
	Multi-channel PWM Out	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
	Block-Commutated PWM Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Field-Oriented Control In/Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Pulse Pattern Measurement	Digital Pulse Capture	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	PWM/PFM In	-	✓	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Engine Simulation	Crank/Cam Voltage Out	-	-	-	-	-	✓	-	✓	✓	✓	-	✓	✓	-	-	-	-	-
	Crank/Cam Current Sink	-	-	-	-	-	-	-	-	✓	-	-	✓	-	-	-	-	-	-
	Crank/Cam Digital Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Injection/Ignition Voltage In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Injection/Ignition Current In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Knock Signal Out	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-	-
	Lambda NCCR	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	✓	✓	✓	-
	Lambda DCR	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	✓	✓	✓	-
Wavetable Generation	Wavetable Current Sink	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-
	Wavetable Digital Out	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Wavetable Voltage Out	-	-	-	-	-	✓	-	✓	✓	✓	-	✓	✓	✓	-	-	-	-
	Angular Wavetable Voltage Out	-	-	-	-	-	✓	-	✓	✓	✓	-	✓	✓	✓	-	-	-	-
	Angular Wavetable Digital Out	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Function Block Type		Channel Type																	
		Flexible In/Out 1	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9	Analog Out 1	Analog Out 2	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 7	Analog Out 8	Analog Out 9	Analog Out 10	Resistance Out 1	Resistance Out 2	Load 1	Resolver In 2
Waveform Generation	Waveform Current Sink	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
	Waveform Digital Out	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Waveform Voltage Out	-	-	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-	-
Signal Capturing	Current Signal Capture	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Voltage Signal Capture	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Extended Sensor Signal Capture	Digital Incremental Encoder In	✓	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
	Sine Encoder In	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Hall Encoder In	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
	Resolver In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
	EnDat Master	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	SSI Master	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Extended Sensor Simulation	Digital Incremental Encoder Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Wheelspeed Out	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-
SENT Protocol	SENT In	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	SENT Out	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Communication and miscellaneous purposes

The following table lists the function block types for communication and miscellaneous purposes.

Function Block Type		Channel Type																	
		Bus 1	CAN 1	CAN 2	LIN 1	LIN 2	FlexRay 1	FlexRay 2	Ethernet Adapter 1	Ethernet Adapter 2	Ethernet Switch 1	Power Control 1	Power Switch 1	Power Switch 2					
Bus	CAN	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	FlexRay	✓	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-
	LIN	✓	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-

Function Block Type		Channel Type												
		Bus 1	CAN 1	CAN 2	LIN 1	LIN 2	FlexRay 1	FlexRay 2	Ethernet Adapter 1	Ethernet Adapter 2	Ethernet Switch 1	Power Control 1	Power Switch 1	Power Switch 2
Ethernet	Ethernet Setup	-	-	-	-	-	-	-	✓	✓	-	-	-	-
	Virtual Ethernet Setup	-	-	-	-	-	-	-	✓	✓	-	-	-	-
	UDP Transmit	-	-	-	-	-	-	-	✓	✓	-	-	-	-
	UDP Receive	-	-	-	-	-	-	-	✓	✓	-	-	-	-
	TCP	-	-	-	-	-	-	-	✓	✓	-	-	-	-
	PTP Master	-	-	-	-	-	-	-	✓	✓	-	-	-	-
	PTP Slave	-	-	-	-	-	-	-	✓	✓	-	-	-	-
	Ethernet Switch	-	-	-	-	-	-	-	-	✓	-	-	-	-
Battery Simulation	Power Supply Control	-	-	-	-	-	-	-	-	-	✓	-	-	-
	Power Switch	-	-	-	-	-	-	-	-	-	-	✓	✓	✓
UART	UART RS232 FlowControl	✓	-	-	-	-	-	-	-	-	-	-	-	-
	UART	-	-	-	✓	-	-	-	-	-	-	-	-	-

Allocating MicroAutoBox III Channel Types to Function Block Types

Introduction

Every function block type supports one or more suitable channel types.

The following gives you an overview on the allocation of the function block types to MicroAutoBox III channel types and shows you the efficiency of each channel type.

I/O

The following tables list the function block types for I/O functionality.

Function Block Type		Channel Type													
		Digital In 4	Digital In 5	Digital In 9	Digital In 10	Trigger In 3	Analog In 7	Analog In 8	Analog In 9	Analog In 10	Analog In 11	Analog In 12	Analog In 14	Analog In 15	Analog In 17
Analog Input	Voltage In	-	-	-	-	-	✓	✓	✓	✓	✓	✓	-	✓	-
Analog Output	Voltage Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Digital Input	Multi Bit In	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-
	Trigger In	-	✓	-	-	✓	-	-	-	-	-	-	-	-	-

Function Block Type		Channel Type													
		Digital In 4	Digital In 5	Digital In 9	Digital In 10	Trigger In 3	Analog In 7	Analog In 8	Analog In 9	Analog In 10	Analog In 11	Analog In 12	Analog In 14	Analog In 15	Analog In 17
Digital Output	Multi Bit Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Digital Pulse Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Pulse Pattern Generation	PWM/PFM Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Multi-channel PWM Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Pulse Pattern Measurement	Digital Pulse In	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
	PWM/PFM In	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-
Signal Capturing	Voltage Signal Capture (ADC Type 4)	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-
Extended Sensor Signal Capture	Digital Incremental Encoder In	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Engine Control	Crank In	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-
	Cam In	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-
	Injection Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Ignition Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Engine Angular Pulse Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Knock In	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	Lambda Probe In	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
SENT Protocol	SENT In	✓	-	-	-	-	-	-	-	-	-	-	-	-	-

Function Block Type		Channel Type												
		Digital Out 4	Digital Out 5	Digital Out 7	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10	Analog Out 11	Analog Out 12	Analog Out 13	-	-	-	-
Analog Input	Voltage In	-	-	-	-	-	-	-	-	-	-	-	-	-
Analog Output	Voltage Out	-	-	-	-	-	-	-	-	✓	✓	✓	✓	-
Digital Input	Multi Bit In	-	-	-	✓	✓	✓	-	-	-	-	-	-	-
	Trigger In	-	-	-	-	-	-	-	-	-	-	-	-	-
Digital Output	Multi Bit Out	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-
	Digital Pulse Out	✓	-	-	-	-	-	-	-	-	-	-	-	-
Pulse Pattern Generation	PWM/PFM Out	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-
	Multi-channel PWM Out	✓	-	-	-	-	-	-	-	-	-	-	-	-
Pulse Pattern Measurement	Digital Pulse In	-	-	-	-	-	-	-	-	-	-	-	-	-
	Digital Pulse Capture	-	-	-	-	-	-	-	✓	-	-	-	-	-
	PWM/PFM In	-	-	-	✓	✓	✓	-	-	-	-	-	-	-
Signal Capturing	Voltage Signal Capture (ADC Type 4)	-	-	-	-	-	-	-	-	-	-	-	-	-

Function Block Type		Channel Type																						
		Digital In 4	Digital Out 4	Digital In/Out 6	CAN 3	CAN 4	CAN 5	CAN 6	LIN 3	LIN 4	FlexRay 3	FlexRay 4	Ethernet Adapter 2	Ethernet Adapter 3	Ethernet Switch 2	UART 2	UART 3	FuSa Challenge-Response Unit 1	FuSa Unit 1	Analog Out 11	Digital In/Out 8	Digital In/Out 10	Digital In/Out 12	Analog Out 13
Extended Sensor Signal Capture		Digital Incremental Encoder In													-	-	-	-	-	-	-	-	-	-
Engine Control		Crank In													-	-	-	-	-	-	-	-	-	-
		Cam In													-	-	-	-	-	-	-	-	-	-
		Injection Out													-	-	✓	-	✓	-	-	-	-	-
		Ignition Out													-	-	✓	-	✓	-	-	-	-	-
		Engine Angular Pulse Out													-	-	✓	-	✓	-	-	-	-	-
		Knock In													-	-	-	-	-	-	-	-	-	-
		Lambda Probe In													-	-	✓	-	✓	-	-	-	-	-
SENT Protocol		SENT In													-	-	-	-	-	-	-	-	-	-

Communication and miscellaneous purposes

The following table lists the function block types for communication and miscellaneous purposes.

Function Block Type		Channel Type																					
Bus	CAN	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	FlexRay	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	LIN	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-

Function Block Type		Channel Type																				
		Digital In 4	Digital Out 4	Digital In/Out 6	CAN 3	CAN 4	CAN 5	CAN 6	LIN 3	LIN 4	FlexRay 3	FlexRay 4	Ethernet Adapter 2	Ethernet Adapter 3	Ethernet Switch 2	UART 2	UART 3	FuSa Challenge-Response Unit 1	FuSa Unit 1	Acceleration Sensor Unit 1	Pressure Sensor Unit 1	LED Out 1
Ethernet	Ethernet Setup	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	
	Virtual Ethernet Setup	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	
	UDP Transmit	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	
	UDP receive	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	
	TCP	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	
	PTP Master	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	
	PTP Slave	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	
	Ethernet Switch	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	
Serial	SPI Master	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
FuSa	FuSa Setup	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	FuSa Challenge-Response Monitoring	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
System	Acceleration In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-
	Atmospheric Pressure In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
	LED Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-

Implementing I/O Functionality

Motivation

Implementing and configuring I/O functionality is one of the main work steps in implementing a real-time application. In ConfigurationDesk, I/O functionality is added to the signal chain via function blocks.

Where to go from here

Information in this section

Basics on Implementing I/O Functionality.....	62
Configuring Function Blocks.....	77

Information in other sections

Adding Function Blocks to the Signal Chain (ConfigurationDesk Real-Time Implementation Guide )
Model Port Mapping (ConfigurationDesk Real-Time Implementation Guide )
Device Port Mapping (ConfigurationDesk Real-Time Implementation Guide )

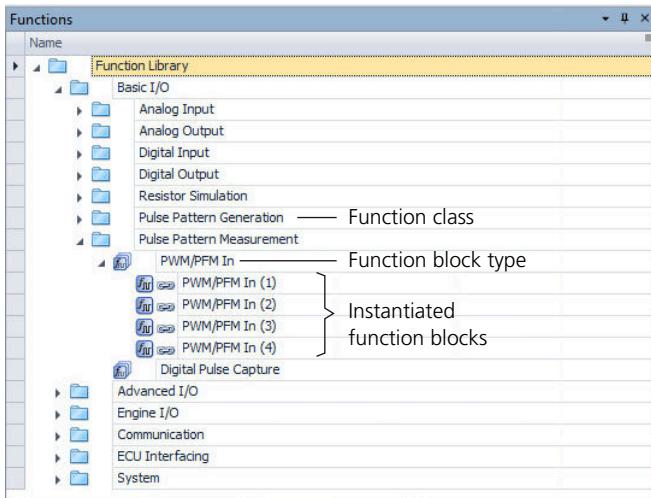
Basics on Implementing I/O Functionality

Objective	To implement I/O functionality, you have to instantiate a function block from the function library. Every function block has ports, which provide the interfaces to the neighboring blocks in the signal chain.
------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Where to go from here	Information in this section
	Basics on Instantiating Function Blocks..... 62
	Basics on Function Blocks..... 67
	Characteristics of Signal Ports..... 71
	Characteristics of Function Ports..... 73
	Characteristics of Event Ports..... 75

Basics on Instantiating Function Blocks

Function block types and function library	<p>The basis for implementing I/O functionality are function block types. Every function block type has unique features which are different from other function block types.</p> <p>To use a function block type in your application, you have to create an instance of it. This instance is called a <i>function block</i>. Function block types can be instantiated multiple times in a ConfigurationDesk application. The types and their instantiated function blocks are displayed in the function library of the Function Browser.</p> <p>The illustration below shows the structure of the function library. Function block types are grouped in function classes. Instantiated blocks are added below the corresponding function block type.</p>
--------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



All displayed function blocks are used in your active ConfigurationDesk application.

Note

- The function library allows access to the I/O functionality provided by the software installation on your host PC. Thus, the availability of function block types depends only on the installed plug-ins and not on the configuration of your ConfigurationDesk application.
- A filter for function block types supported by the current hardware topology is active by default. Thus, the displayed function block types depend on the elements of the hardware topology in the Hardware Resource Browser. You can deactivate the filter in the context menu of the Function Browser or on the Home – Filters ribbon. If there are no hardware topology elements, all function block types are displayed.

Methods for instantiating function blocks

To instantiate function blocks, you need a function block license. This license is purchased in different sizes that allow you to use a certain number of instantiated function blocks in your active ConfigurationDesk application. There are licenses for the usage of 100, 200, 300, 1000 or an unlimited number of function blocks. For details on function block licenses, refer to [Details on Function Block Licenses \(ConfigurationDesk Real-Time Implementation Guide\)](#).

There are several methods to instantiate function blocks:

- Via Function Browser
- Via device ports
- Via copy and paste of existing function blocks
- Via drag & drop of a single channel from the hardware topology
- Via drag & drop to the Model-Function Mapping Browser

Instantiating via Function Browser You can choose a functionality from the function library and add it to the signal chain via drag and drop. You have to map the ports of the function block to other blocks in the signal chain afterwards. For instructions, refer to [How to Add Function Blocks to the Signal Chain via Function Browser \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Instantiating via device ports If you already added device ports to the signal chain you can extend them with a suitable function block. ConfigurationDesk suggests one or more functions block types which match the configuration settings of the corresponding device port(s).

After you have selected a functionality, ConfigurationDesk automatically:

- Instantiates new function blocks.
- Maps the device port(s) to the first suitable signal port of the corresponding function block(s).
- Transfers several configuration settings (for example, the allowed failure classes) from a device port to the mapped signal port.

Note

This method is only useful if you configured the device port(s) beforehand. The suggested functionality depends only on the settings of the device port. For example: If you have configured a device port as an inport, ConfigurationDesk suggest all functions, which can output signals from the function block to an external device.

For instructions, refer to [How to Add Function Blocks to the Signal Chain via Device Ports \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Instantiating via copy and paste of existing function blocks It is useful to copy and paste an already configured function block, if you need several function blocks of the same type and with a similar configuration. You can do this via the Function Browser or via working views.

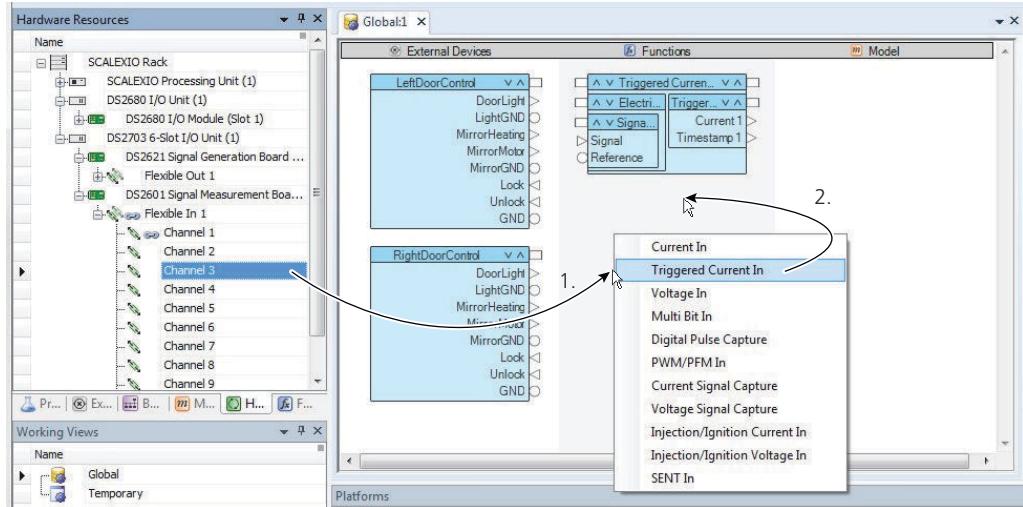
During the paste action, ConfigurationDesk:

- Instantiates the function blocks.
 - Adds the suffix Copy to the function block name.
 - Copies the configuration of the function block, except for assigned channels.
- Each function block must be assigned to a specific hardware resource. From this it follows that the assignment cannot be shared with other function blocks.

For instructions, refer to [How to Add Function Blocks via Copy and Paste \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Instantiating via drag and drop to a working view You can instantiate a function block by dragging a single channel from the available hardware topology to a working view and drop it at the desired position.

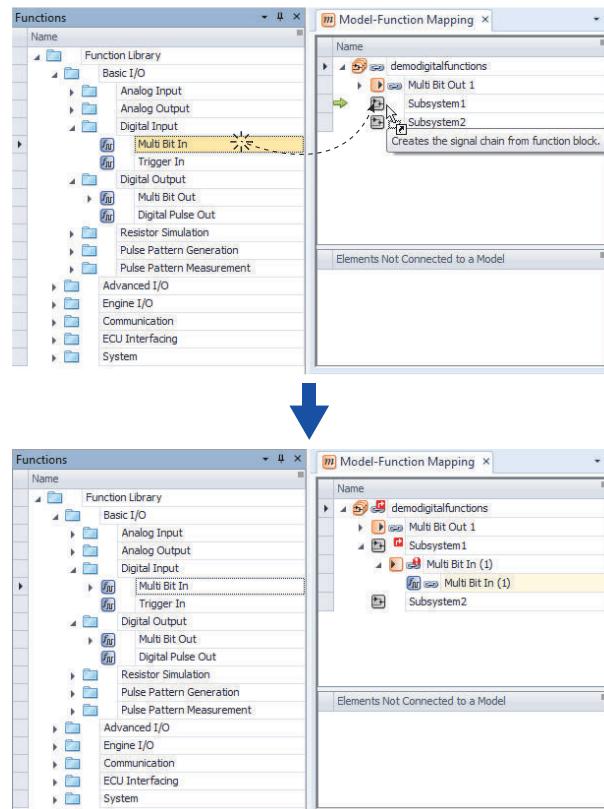
A context menu opens and displays a list of the function block types that support the hardware characteristics of the channel. You can select a function block type from the list to add a function block to the signal chain. See illustration below.



If the function block has more than one channel request, you have to complete the hardware resource assignment after instantiating the function block. For details, refer to [Methods for Assigning Hardware Resources \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Instantiating via drag and drop to the Model-Function Mapping Browser

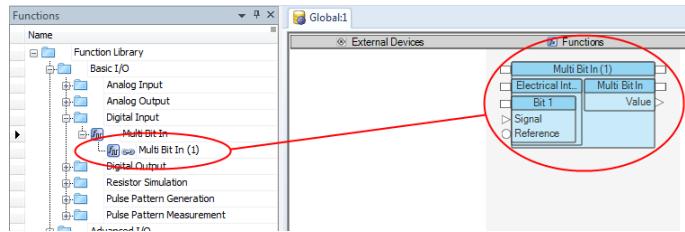
Browser The Model-Function Mapping Browser lets you instantiate a function block and create the signal chain for the work with Simulink behavior models via drag & drop.



For more information, refer to [Creating Signal Chains via the Model-Function Mapping Browser \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Results of instantiating

After you have instantiated a function block, it is displayed on the active working view and in the Function Browser (below the corresponding function block type) as shown below.



Every function block which is instantiated from the Function Browser or via device ports is given a unique identification name consisting of the function block type and a unique index. You can change this default name as required. For instructions, refer to [How to Rename Function Blocks](#) on page 83.

Each instantiated function block has a state which is determined by the system. For details, refer to [Basics on Function Blocks](#) on page 67.

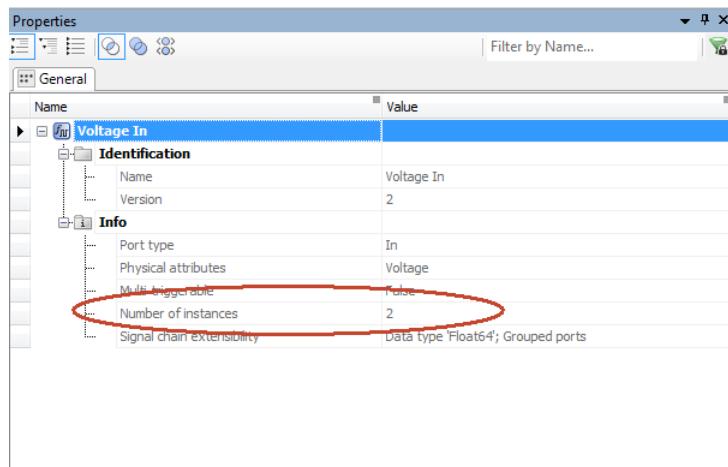
Displaying in working views If you have instantiated a function block via copying, it is always displayed in the Global working view and in the Function Browser, and if you pasted it to a specific working view, it is also displayed in that.

Tip

You can add an instantiated function block to several working views. This does not mean that the function block is used several times in your application. It is used only once, but can be displayed on several working views. For information on using several working views, refer to [Using Working Views \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Counting instantiated function blocks ConfigurationDesk also counts the number of instantiated function blocks for each function block type. So you can find out how many instances of a specific function block type, for example, **Voltage In**, are used in your active ConfigurationDesk application.

The number is displayed in the Properties Browser as shown in the following screenshot.



To access the Number of instances property, select the desired function block type in the Function Browser.

Showing only instantiated function blocks

By default, all the elements of the function library are displayed in the Function Browser. For a clearer view, you can filter for the instantiated functions blocks from the function library via a context menu command. Then only the instantiated function blocks (with their subelements) and their higher-level elements (for example, the function block types) in the function library hierarchy are displayed.

Deleting function blocks

You can delete a function block from the signal chain of your ConfigurationDesk application via a context menu command available in the Function Browser and the working area.

Basics on Function Blocks

Objective

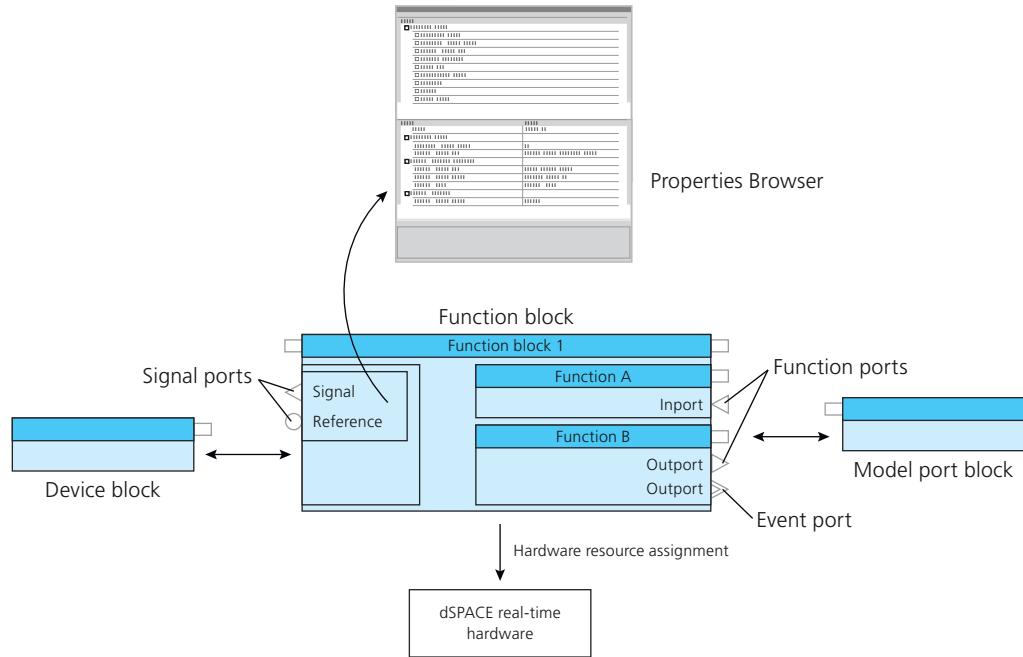
Function blocks are the central components in the signal chain. Every function block has ports, which provide the interfaces to the neighboring blocks in the signal chain.

Purpose of function blocks

A function block provides:

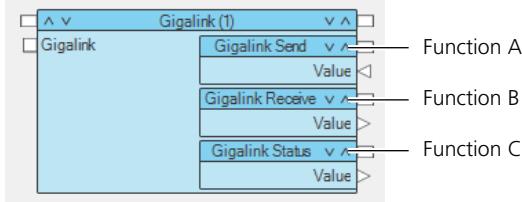
- Specific functionality which is inherited from the corresponding function block type
- Access to the properties for configuring the block properties
- The interface to the real-time hardware (= hardware resource assignment)

- The interfaces to the neighboring blocks in the signal chain:
- Signal ports for mapping the function block to device ports or to other signal ports (device port mapping)
- Function ports and event ports for mapping the function block to model ports (model port mapping)



Container for functions

A function block serves as a container for functions. Each function block contains at least one function. If two or more functions are combined in one block, their features are closely related. They usually have common properties and they usually need the same types of hardware resources. The following illustration shows an example of a function block which contains three functions.



Tip

The complete display of functions in the function block, as shown in the example, depends on the layout of the signal chain in a working view. Functions are only displayed if the block structure is not collapsed. For details, refer to [Collapsing and Expanding Blocks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

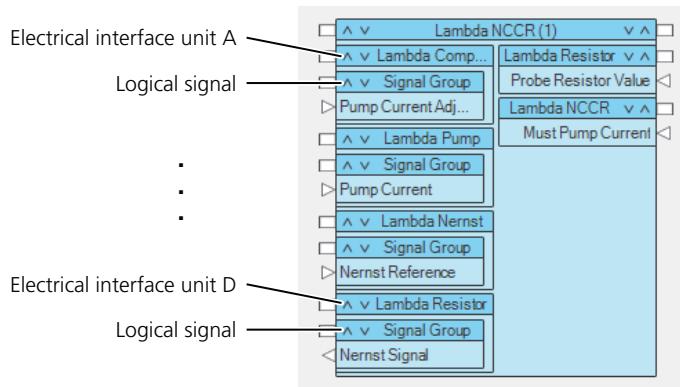
Container for electrical interface units and their logical signals

A function block also serves as a container for electrical interface units and their logical signals.

An electrical interface unit provides the interface of the function block to the external devices and to the real-time hardware (via hardware resource assignment). Each electrical interface unit of a function block usually needs a different channel set to be assigned to. It also provides properties to configure the characteristics of the hardware.

A logical signal combines all the signal ports which belong together to provide the functionality of the signal.

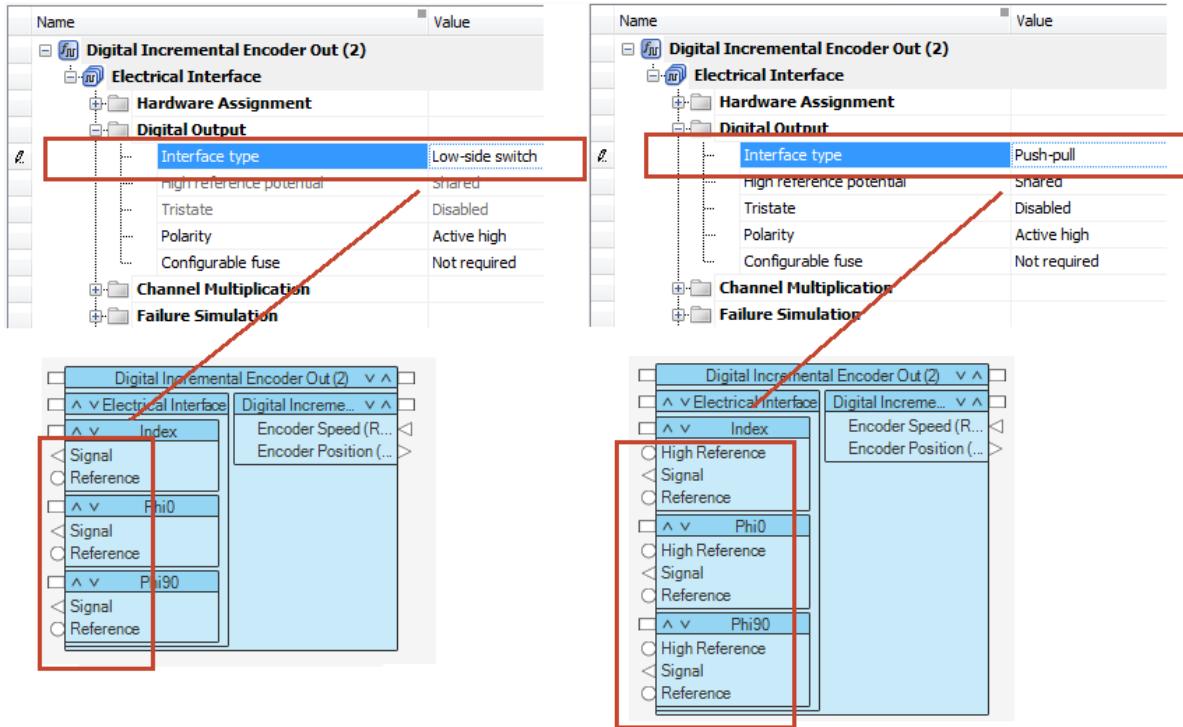
Each function block can contain several electrical interface units, each with several logical signals. The following illustration shows an example of a function block which contains four electrical interface units, each with one logical signal.



Dependencies between property settings and ports

The setting of function block properties can affect the number and availability of ports at the function block.

The following example shows that changing the **Interface type** property from **Low-side switch** to **Push-pull** affects the appearance of additional signal ports.



You have to note the following behavior, if ports are added or deleted caused by configuration changes: In the example above, three new signal ports (reference ports) are needed after reconfiguration. ConfigurationDesk therefore adds them to the function block. The electrical interface of the function block is changed completely.

All new ports are added with their default configurations. You might have to reconfigure the port properties. However, if ports are already mapped, they are not deleted. A mapped port continues to exist, next to the new ports, until you remove its mapping line(s). In this way ConfigurationDesk supports you in remapping the ports.

Possible states of function blocks

ConfigurationDesk determines the status of each function block after every configuration change. It is displayed graphically and in a tooltip (in case of conflicts). The following table shows the possible states, how they are displayed and their effects on code generation during the build process.

State	Graphical Display		Description	Effect on Code Generation
	Working View 1)	Function Browser		
OK	Block frame is black.	No display.	The function block can be used in the application without any restriction.	Code is generated.

State	Graphical Display		Description	Effect on Code Generation
	Working View 1)	Function Browser		
Conflict	Block frame or elements of the block are highlighted (orange). This indicates a warning conflict.	No display.	The required hardware resource(s) is/are not yet assigned to the function block. Assignment Conflicts and Their Effects on Code Generation (ConfigurationDesk Real-Time Implementation Guide) .	The effect depends on the assignment as follows: <ul style="list-style-type: none"> ▪ Channel set not assigned: The function block is not considered during code generation. ▪ Channel not assigned: The I/O access is simulated. In this case, the initial values configured at function outports, are input into the behavior model.
			The settings for the load rejection behavior in the signal chain differ. For details, refer to Basics on Load Rejection (ConfigurationDesk Real-Time Implementation Guide) .	No effect on code generation.
			The block configuration has a failure. For example: Some external configuration elements are not added, such as the wavetable files for configuration of wavetable functionality.	The effect depends on the specific conflict source. For example, default code is generated if wavetable files are missing for wavetable function blocks. In the Conflicts Viewer you can see the effects on code generation (for example, generate no code, generate default code, warning). Warnings are created during the build process and displayed in the Build Log Viewer.
Unresolved ²⁾	Block frame is highlighted (orange). This indicates a conflict of the warning category.	No display	The required user files for the corresponding function block type are not available. In this case you cannot: <ul style="list-style-type: none"> ▪ Instantiate further function blocks ▪ Change the configuration settings 	The function block is not considered during code generation.

1) To highlight conflicts, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts \(ConfigurationDesk Real-Time Implementation Guide\)](#).

2) Only possible for custom function blocks.

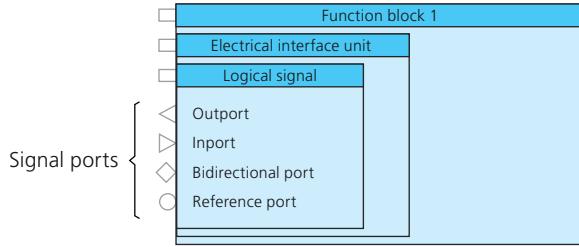
Characteristics of Signal Ports

Objective

Signal ports provide the interface to external devices (for example ECUs) via device blocks. They represent the electrical connection points of a function block.

Port types

The illustration below shows the different types and their graphical display. The presence of specific port types depends on the function block's functionality.



Signal output Represents an electrical connection point of a function block which provides signal generation (= output) functionality.

Signal import Represents an electrical connection point of a function block which provides signal measurement (= input) functionality.

Bidirectional signal port This port is independent of a data direction or current flow. Bidirectional ports are used to implement bus communication, etc.

Signal reference port A specific port which represents the connection point for the reference potential of imports, exports and bidirectional ports. For example: With differential signals, this is a reference signal, with single-ended signals, it is the ground signal (GND).

Port characteristics

The following table lists some general characteristics of signal ports.

Characteristics	Description
Port name	<ul style="list-style-type: none"> ▪ Defined by the system, unique per function block ▪ Not user-configurable
Port type	<ul style="list-style-type: none"> ▪ Defined by the system ▪ Not user-configurable ▪ Defines the signal direction of a port and is the basis for device port mapping ▪ Possible port types: In, Out, Bidirectional, Reference
Role	<ul style="list-style-type: none"> ▪ Defined by the system ▪ Not user-configurable ▪ Displays the role of the signal port. Most settings are configurable for signal ports with the Signal role ▪ Examples of roles: Signal, Low reference, High reference, Load signal, Load reference

Possible port states

The following table shows the possible states, how they are displayed and their effects on code generation during the build process. The state of the port is

determined after every configuration change. It is displayed graphically and in a tooltip.

The port states do not affect the mapping of the signal ports. You can add or delete mapping lines regardless of port state.

State	Display of Port Symbol ¹⁾	Description	Effect on Code Generation
OK	Black	There are no conflicts.	Code is generated.
Obsolete	Orange. This indicates a warning conflict.	The port is mapped, but due to configuration changes to the function block, the port is no longer used or required. If you remove the mapping line from the obsolete port, this port is deleted. Background: The mapping is not deleted immediately, because this simplifies the remapping of the corresponding device block.	The port is not considered during code generation.

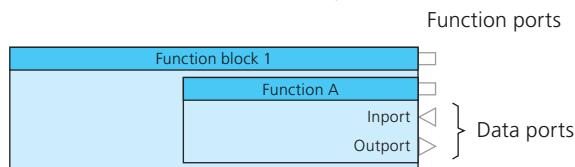
¹⁾ To highlight obsolete ports, you have to set the highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts \(ConfigurationDesk Real-Time Implementation Guide\)](#).

The device port mapping is not considered during the build process, so the different mapping states do not affect code generation. However, device port mapping affects the calculation of the external cable harness.

Characteristics of Function Ports

Objective Function ports provide the data interface to the behavior model via model port blocks.

Port types The illustration below shows the different types of function ports. Every function has at least one function port. The presence of specific port types depends on the function block's functionality.



Function import At this data port the values from the behavior model are input to be processed by the function.

Function outport This data port outputs the value of a function, to be used in the behavior model.

The values available at the data ports can be observed via experiment software, only if you have enabled test automation support in ConfigurationDesk. Then the corresponding variables are written to the variable description file during the build process.

Function port characteristics

The following table lists some general characteristics of function ports.

Characteristics	Description
Port name	<ul style="list-style-type: none"> ▪ Defined by the system ▪ Not user-configurable ▪ If test automation support is enabled in ConfigurationDesk, the port name is also used in the variable description file (*.TRC). The name serves as a structure element which groups the variables for the function port values below it.
Range/value	<ul style="list-style-type: none"> ▪ Calculated by the system (depends on the assigned real-time hardware, etc.) ▪ System ranges/values are not user-configurable ▪ For certain function ports you can reduce the min/max values provided by the system to user-configured values. The min/max values are displayed in the function block properties.
Unit	<ul style="list-style-type: none"> ▪ Defined by the system ▪ Not user-configurable ▪ The unit is displayed in the function block properties.
Data width	Vector and scalar data are supported.

Possible function port states

The following table shows the possible states, how they are displayed and their effects on code generation during the build process. The status of each port is determined after every configuration change. It is displayed graphically and in a tooltip.

The port states do not affect the mapping of the function ports. You can add or delete mapping lines regardless of port status.

State	Display of Port Symbol ¹⁾	Description	Effect on Code Generation
OK	Black	There are no conflicts.	Code is generated.
Obsolete	Orange. This indicates a warning conflict.	The port is mapped, but due to configuration changes to the function block, the port is no longer required. If you remove the mapping line from the obsolete port, this port is deleted. Background: The mapping is not deleted immediately, because this simplifies the remapping of the corresponding model port block.	The port is not considered during code generation.
Not mappable	<ul style="list-style-type: none"> ▪ Function import:  ▪ Function output:  	The function port cannot be mapped to model port blocks, because the port's model access is disabled. The port	The port is not considered during code generation.

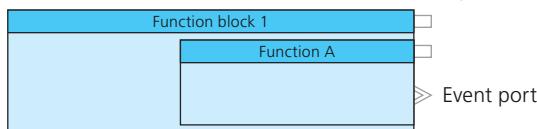
State	Display of Port Symbol ¹⁾	Description	Effect on Code Generation
		symbols are displayed in the signal chain only if the Filter for Mappable Ports filter is disabled. You can enable model access and therefore reset this state via the Model access property that is available for each function port.	

¹⁾ To highlight obsolete ports, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Characteristics of Event Ports

Objective	Event ports provide the interface to trigger asynchronous task of the behavior model via model port blocks.
------------------	-------------------------------------------------------------------------------------------------------------

Port type	The illustration below shows the type of event ports. The presence of event ports depends on the function block's functionality.
------------------	----------------------------------------------------------------------------------------------------------------------------------



Event ports output I/O events to trigger an asynchronous task in the behavior model. The event is triggered by the assigned real-time hardware. Each event port can trigger only one event in the behavior model.

Event port characteristics	The following table lists some general characteristics of event ports.
-----------------------------------	------------------------------------------------------------------------

Characteristics	Description
Port name	<ul style="list-style-type: none"> ▪ Defined by the system ▪ Not user-configurable
Event type	I/O event

Possible event port states	The following table shows the possible states, how they are displayed and their effects on code generation during the build process. The status of each port is determined after every configuration change. It is displayed graphically and in a tooltip.
-----------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The port states do not affect the mapping of the event ports. You can add or delete mapping lines regardless of port status.

State	Display of Port Symbol ¹⁾	Description	Effect on Code Generation
OK	Black	There are no conflicts.	Code is generated.
Obsolete	Orange. This indicates a warning conflict.	The port is mapped, but due to configuration changes to the function block, the port is no longer required. If you remove the mapping line from the obsolete port, this port is deleted. Background: The mapping is not deleted immediately, because this simplifies the remapping of the corresponding model port block.	The port is not considered during code generation.

¹⁾ To highlight obsolete ports, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts \(ConfigurationDesk Real-Time Implementation Guide](#) .

Configuring Function Blocks

Objective	You can configure the properties of the function blocks according to your needs. ConfigurationDesk's Properties Browser gives you access to the block and port properties.
------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Where to go from here	Information in this section
	Basics on Function Block Configuration 77
	Categories of Configurable Parameters 79
	Basics on Hardware Resources and Channel Types 80
	How to Rename Function Blocks 83
	How to View Circuit Diagrams of Hardware Resources 84

Basics on Function Block Configuration

Objective	Several properties of a function block are user-configurable. Some general aspects of configuring and handling configuration data are described here.
Dependencies to model and hardware	<p>ConfigurationDesk lets you configure I/O functionality independently of the behavior model and the real-time hardware:</p> <ul style="list-style-type: none"> ▪ You can configure properties without changing the behavior model. However, in some cases reconfiguration changes the existence of function ports. In this case you have to check the effects on the corresponding model ports and their representation in the behavior model. ▪ You can configure properties with and without connected real-time hardware, and even without a hardware topology.
Access to function block properties	<p>As a precondition for accessing the function block properties, at least one function block must be instantiated and therefore available in the signal chain.</p> <p>Every function block and its elements (for example, the ports) have properties. Some of them are user-configurable, others are only for information purposes.</p> <p>The Properties Browser shows you all the properties of the function block element(s) you have selected in a working view or in a table. If a property is configurable, you can change the setting there.</p>

If you have selected several elements, the Properties Browser displays all the properties available for them. However, only the values which are identical for all selected elements are shown.

Tip

You can change the value of a property separately for one element or for several selected elements at once.

For details on the Properties Browser, refer to [Configuring Elements with the Properties Browser \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Flexible configuration settings

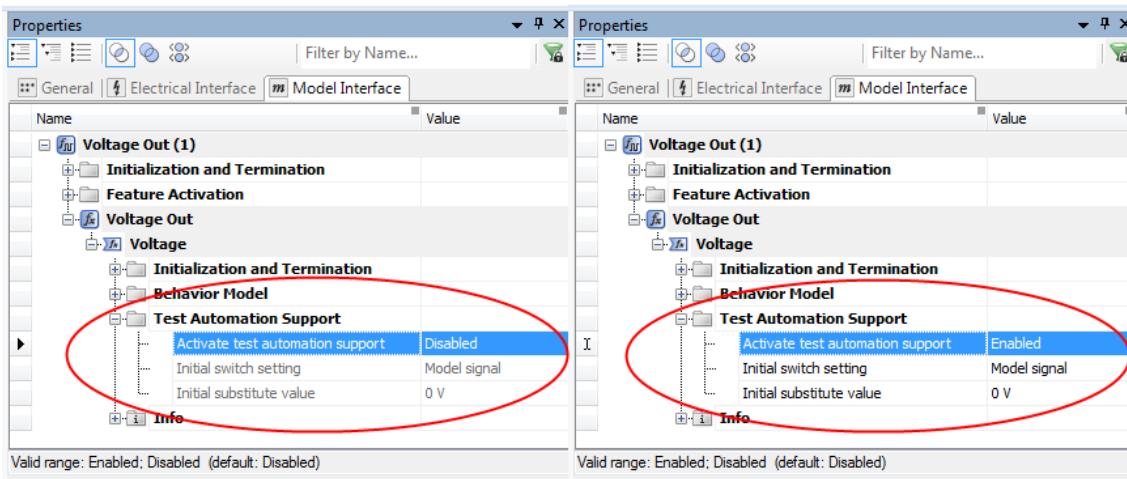
The concept of ConfigurationDesk allows very flexible configuration settings, with the effect that one setting might not match the setting of another function block property. These configuration conflicts are allowed at first. The conflicts are displayed in the Conflicts Viewer. However, before you build a real-time application, you have to resolve the conflicts to get proper build results.

The property which might cause the most conflicts is the **Assigned channel set** property (available for most function block types) for the following reason:

Every function block type supports one or more suitable channel sets (each based on a specific channel type). This means that you can use a function block in combination with hardware resources which have different electrical characteristics (for example, more or less functionality). After you select a channel set, the function block supports only settings and value ranges which are provided by the hardware resources of the selected channel set. All unsupported settings (= mismatches) are then displayed in the Conflicts Viewer.

Dependencies between different properties

For several properties it might not be possible to change the values or settings, because their features are disabled by another property. In this case the property and the current values are displayed in gray, and cannot be changed. For example, the test automation support is configurable only if the **Activate test automation support** property is set to **Enabled** as shown below.



Managing configuration data

The configuration data (property settings etc.) of a function block and its usage in the signal chain is saved automatically together with the ConfigurationDesk application to which it belongs.

Exporting an application allows you to save the contents of one application including the function block configuration in a dSPACE archive file (DSA file) and transfer the application to another project.

You can import and export an application via context menu commands in the Project Manager. For details on handling applications in ConfigurationDesk, refer to [Managing ConfigurationDesk Applications \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Categories of Configurable Parameters

Objective

The configurable parameters of a function block can be categorized according to their characteristics. You cannot change these characteristics.

Inline parameter

The inline parameter type is the standard parameter category for function blocks in ConfigurationDesk. You cannot access them via other software tools such as the experiment software.

Inline parameters are not specially marked either in the properties or in the user documentation.

Note

Parameters which affect the behavior of several function blocks, for example all function blocks instantiated from a specific function block type, do not exist.

Tunable parameter

Unlike inline parameters, you can change the value of tunable parameters via your experiment software. For access of these parameters from outside ConfigurationDesk, they are available in the variable description file which is generated during the build process.

There are two types of tunable parameters:

- Stop-Run-Tunable parameter

You can change this parameter in the experiment tool, when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

- Run-Time-Tunable parameter

You can change this parameter in the experiment tool, when the real-time application is running or stopped. The changes take effect immediately.

Tunable parameters are not specially labeled or marked in the properties. However, they are indicated in the user documentation. Refer to the online help via **F1** key, for example.

Basics on Hardware Resources and Channel Types

Hardware resource

A hardware resource is a hardware element (normally a channel on an I/O board or I/O unit) which is required to execute a function block. A hardware resource can be localized unambiguously in a hardware system. Every hardware resource has specific characteristics. A function block therefore needs a hardware resource that matches the requirements of its functionality.

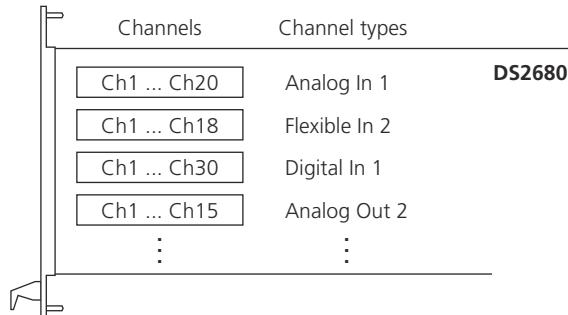
This means that not every function block can be executed on every hardware resource. There could be limitations on a function block's features and/or the physical ranges.

Channel types

To identify all the hardware resources (channels) in the hardware system that provide exactly the same characteristics, ConfigurationDesk provides channel types, for example, the **Flexible In 1** channel type.

An I/O board in a hardware system can have channels of several channel types. Channels of one channel type can be available on different I/O boards. The

The illustration below shows (as an example) the DS2680 I/O Unit with a selection of available channel types and the number of channels related to these channel types.

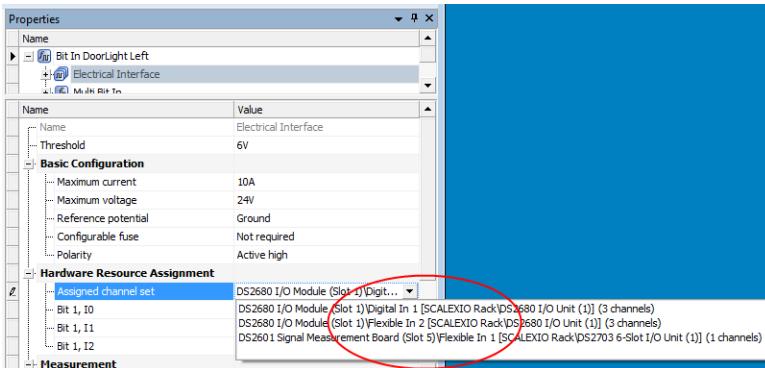


Definition of channel set

A channel set is a number of channels of the same channel type located on the same I/O board (or I/O unit). Channels in a channel set can be combined, for example, to provide a signal with channel multiplication.

Function block support of channel types

Every function block type supports one or more suitable channel types. If there are several channel types, you can select the one you need by assigning a channel set which supports the channel type as shown below:



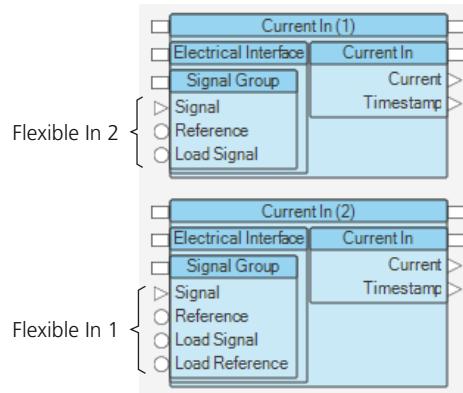
Support of several channel types lets you use a function block in combination with hardware resources which have different electrical characteristics (for example, more or less functionality). After you assign a channel set which is based on a specific channel type, the function block supports only the settings and value ranges which are provided by that channel type. All unsupported settings (= mismatches) are then displayed in the Conflicts Viewer.

Assigning a channel set You can assign a channel set if the hardware topology of the ConfigurationDesk application contains hardware which provides at least one channel set which is suitable for the function block. You can change this assignment at any time during configuration. However, to avoid conflicts and

to reduce conflict resolutions, you should assign a channel set as early as possible.

Using a channel set based on a different channel type Changing the channel type (via channel set assignment) affects the function block as follows:

- The functionality of the function block changes. For example, channel type A features noise generation, channel type B does not. The extent of the differences varies and depends on the specific function block type and the selected channel type.
- The signal ports providing the electrical interface of the function block are firmly tied to the characteristics of the I/O circuits of the channel type. The number and type of signal ports that are provided therefore depend on the selected channel set (based on a specific channel type) as shown below. Note that sometimes the availability of signal ports depends on the settings of several other properties such as the Reference potential property.



The circuit diagrams are available in the software and in the documentation. Refer to [Circuit Diagrams of Channel Types](#) on page 1541.

Aspects on choosing a channel type

Note the following points when choosing a channel type (via channel set assignment):

- You should use dSPACE real-time hardware resources that are as suitable and economical as possible for the function block functionality you require. This means:
 - Use only tailored channels, i.e., ones that are not oversized for the required functionality.
 - Use channel sets and channels of hardware that is already present and available.
- Specify the maximum values of the expected electrical signal (for signal inports) or the required maximum values (for signal outports) via the Required voltage and Required current properties according to the required values of the external devices, not to the values the hardware supports. For basics, refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95.

- Your electrical devices and/or the wiring of these devices may require specific I/O characteristics, for example, differential signals or a specific reference potential. One channel type might be more suitable than others for a specific use case.

The circuit diagrams of the channel types will help you choose a suitable type. These circuit diagrams are available in the software and in the user documentation. Refer to [Circuit Diagrams of Channel Types](#) on page 1541.

To help you choose a suitable channel type, the documentation contains tables showing the differences between the channel types in detail. Refer to the *Hardware Dependencies (<Function Block Name>)* chapters (in [ConfigurationDesk I/O Function Implementation Guide](#) on page 1) which are available for each function block type.

Related topics

Basics

[Assigning Hardware Resources to Function Blocks \(ConfigurationDesk Real-Time Implementation Guide\)](#)

How to Rename Function Blocks

Objective

Each function block has a unique default name defined by the system. You can change this name to comply with your project definitions, etc.

Display and use of function block names

Function block names are used for block identification. They are displayed:

- In the graphical user interface (Function Browser, working views, tables, Properties Browser, Conflicts Viewer)
- In error, warning and information messages
- In exported ConfigurationDesk files, for example in the variable description file (*.TRC)

Allowed characters

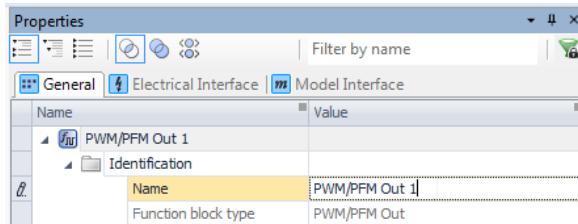
- All characters except for the following ones are allowed:
 - The single characters " and \
 - The character strings */ and /*
- Number of characters: 1 ... 127

Precondition

A function block is instantiated.

Method**To rename function blocks**

- 1 Select the function block that you want to rename.
- 2 In the Properties Browser, click the Name edit field as shown in the following example.

**Tip**

In most panes, you can also double-click the function block or select it and press F2 to make the name editable.

- 3 Enter a new name and press Enter or click anywhere outside the edit field to confirm.

Result

You renamed a function block. If the new name is already used for a function block in the application, a conflict is generated and displayed in the Conflicts Viewer.

How to View Circuit Diagrams of Hardware Resources

Objective

Circuit diagrams provide details of the electrical characteristics of the hardware resources of the real-time hardware. You can access them via instantiated function blocks.

Purpose of circuit diagrams

You should view circuit diagrams if you want:

- To choose a suitable channel set which is based on a specific channel type. Your electrical devices and/or the wiring of these devices may require specific I/O characteristics. One channel type might be more suitable than others for a specific use case.
- To get information on the possible hardware configurations. Some hardware resources are configurable via jumper settings, for example, you have to specify internal and/or external loads.
- To get details on the mapping of signal ports to device ports and on the wiring to external devices, and to view the differences between mapping and wiring.

Available diagram types

ConfigurationDesk provides the following diagram types:

- At least one use-case-specific circuit diagram is available for each channel type. For example, for the Flexible In 1 channel type, there is a circuit diagram showing the Current and voltage measurement of low-side controlled load use case.
- General diagrams for channel multiplication
- Diagrams for complex function block types where channels of different channel types are combined to get the required functionality, for example, for the Lambda DCR function block type.

Tip

The circuit diagrams shown in ConfigurationDesk are also available in the user documentation. For diagrams on the channel types, refer to [Circuit Diagrams of Channel Types](#) on page 1541.

Contents

Circuit diagrams can provide the following hardware details:

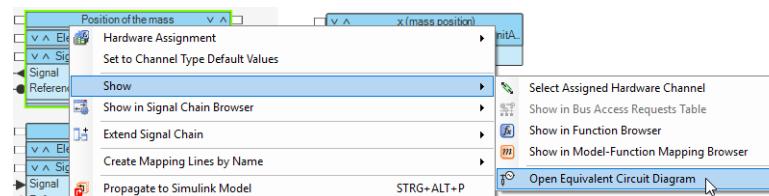
- The connection to external devices, such as ECUs and loads. This includes the mapping between signal ports and device ports as well as the wiring.
- Configuration settings for internal and/or external loads with explanations
- Switch setting with explanations of their effects
- Locations and trigger levels of fuses

Precondition

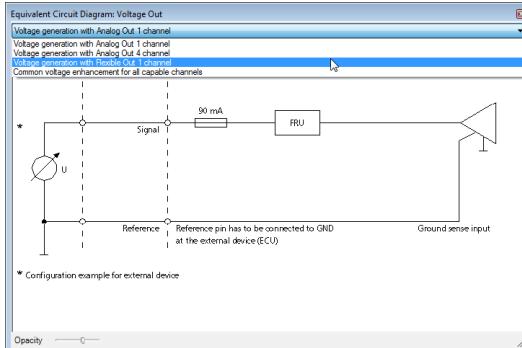
A least one function block is instantiated.

Method**To view circuit diagrams of hardware resources**

- 1 In a working view, right-click the function block and select **Show - Open Equivalent Circuit Diagram** from the context menu as shown below.



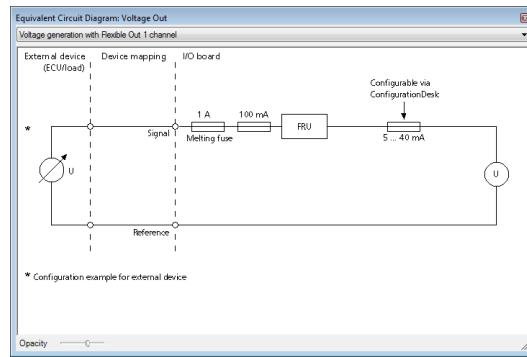
A dialog with the first available circuit diagram opens.



- From the list at the top of the dialog, select the desired circuit diagram.

Result

ConfigurationDesk shows the circuit diagram that you selected.



Related topics

Basics

Basics on Hardware Resources and Channel Types.....	80
-----------------------------------------------------	----

Configuring Standard Features

Objective

Every function block provides a selection of standard configuration features according to its function block type. Some standard features are available for a few function block types, others such as specifying initial values for function ports are available for every function block type.

Where to go from here**Information in this section**

Configuring Standard Features of Function Ports.....	88
Configuring Standard Features of the Electrical Interface.....	95
Configuring Standard Features of Signal Ports.....	121
Configuring Standard Functionalities of Function Blocks.....	126

Configuring Standard Features of Function Ports

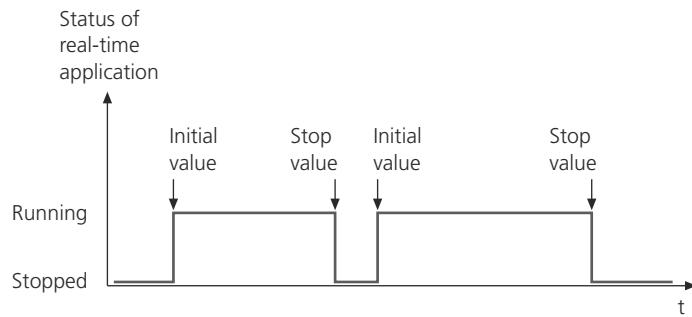
Objective	The function ports provide the interface to the behavior model via model port blocks. The function blocks provide standard features to configure the properties of these interfaces.
------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Where to go from here	Information in this section
	Specifying Initialization and Stop Behavior..... 88
	Specifying User Saturation..... 90
	Configuring Test Automation Support..... 92

Specifying Initialization and Stop Behavior

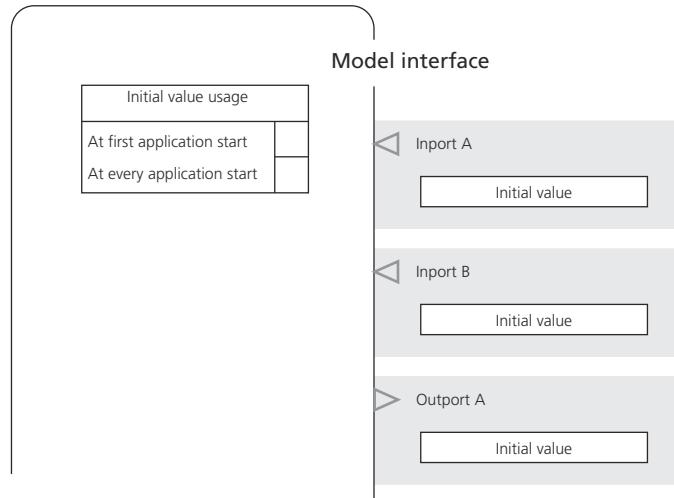
Context	Every function block provides configuration features that let you influence the behavior of function ports when starting and stopping the real-time application on the dSPACE hardware.
----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Statuses of a real-time application	The illustration below shows the different statuses of a real-time application.
--------------------------------------------	---------------------------------------------------------------------------------



Specifying initialization behavior	For each function port, you can specify an initial value which is set during system initialization and used directly after the real-time application starts running on the hardware. The initial value is used as the default value until new values are available at the function port, for example, new measurement values or new values provided from the behavior model. The initial value specified for function outports is also
-------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

used if the function block is not assigned to a hardware resource. Then the input access is simulated and the initial values are input into the behavior model.

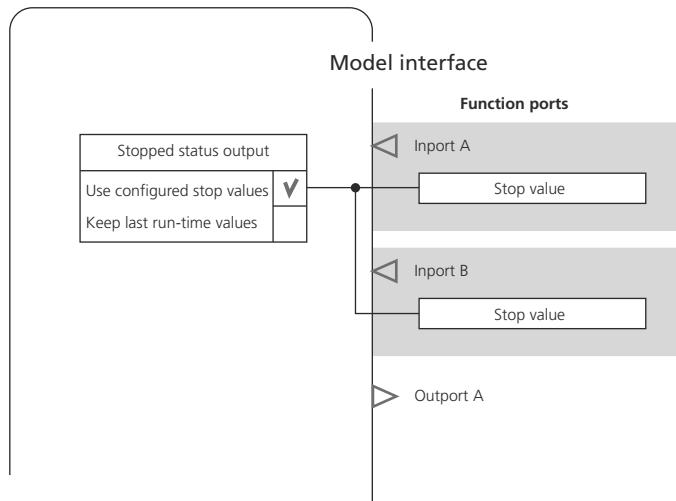


With the **Initial value usage** property, you can specify whether the initial value is set only once at first application start after downloading the real-time application or every time the application starts, including when the application status switches from stopped to running.

For function blocks which do not provide the **Initial value usage** property, the initial value is used only for the first application start. For further status switches (from stopped to running), the last measurement value is used as the initial value.

Specifying stop behavior

For function imports only, you can specify a stop value which is set when the application status changes from running to stopped. The **Stopped status output** property lets you specify whether the function outputs this user-configured stop value or keeps and outputs the last run-time value.

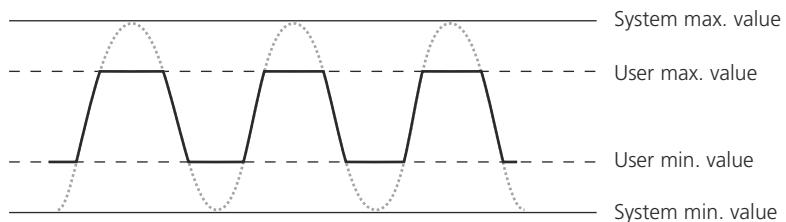


Specifying User Saturation

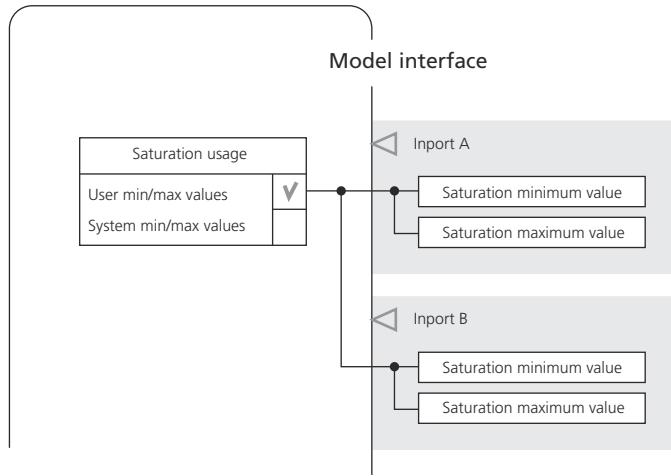
Objective	Saturation means that a signal is limited to minimum/maximum values.
------------------	----------------------------------------------------------------------

Reducing system values For most function imports, you can reduce the system's minimum and maximum values by specifying your own minimums and maximums. This may be useful if you want to protect your external devices against damage.

The following illustration shows an analog signal which is saturated to user-defined values.



With the **Saturation usage** property, you can define whether the signal is saturated to the user-defined minimum/maximum values or to the values provided by the system.



User saturation combined with signal offset and/or noise generation

Some function block types, such as the *Voltage Out* function block type, have function imports where voltage or current values are provided directly by the behavior model. If you want to add a signal offset and/or noise to the provided signal, you should note the behavior described below.

The system adds the specified signal offset and/or the specified noise to the values available at the function block's function port. The resulting value will be saturated against the maximum system values. If you reduce the maximum system value by your own values to protect your connected device against damage, only the value at the function port (coming from the behavior model) is saturated against this reduced user maximum value. However, the signal offset and/or the noise is/are added to this reduced user maximum values. As a result, the maximum value which is available at the signal port can be higher than the specified user maximum value.

⚠ CAUTION

If you use user saturation in combination with signal offset and/or noise generation, the maximum value (voltage or current) available at the signal port can exceed the specified value for user saturation.

Risk of damage to connected devices!

- Ensure that the sum of the specified values (user saturation value, the signal offset, and the noise amplitude) does not exceed the voltage limit of the connected device.
- If necessary, reduce the maximum user saturation value.

Saturation rules for Extend signal chain	There are specific saturation rules that apply when you use the Extend Signal Chain - Create Suitable Model Port Block command. Refer to Specifying Options for Creating Model Port Blocks (ConfigurationDesk Real-Time Implementation Guide) .
-------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Configuring Test Automation Support

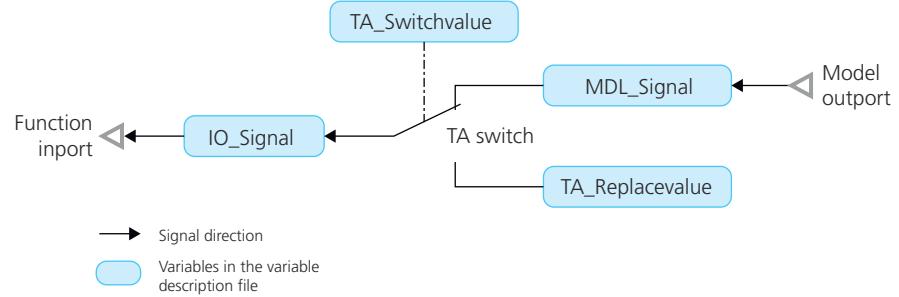
Objective	Each function port features an intervention point for test automation support. You can enable this support and configure its initial behavior in ConfigurationDesk.
------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Use scenarios	Using the test automation support in ConfigurationDesk, you can carry out automation tasks via your experiment software. For example, you can: <ul style="list-style-type: none">▪ Test the real-time hardware using stimulus signals on specific I/O channels.▪ Stimulate specific inputs of the external device using a constant value (which is adjustable via experiment software) during the initial operating phase.▪ Test the external device (for example the ECU) by feeding its inputs with specific erroneous signals.
----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

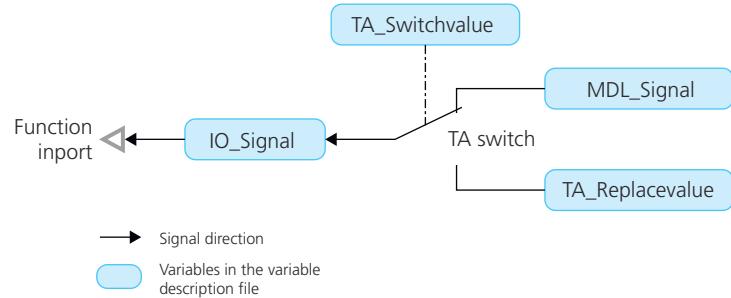
Principles	If you activate test automation support, you can use a test automation (TA) switch in your experiment software to switch between the original signal and a substitute value. Therefore, ConfigurationDesk generates variables to the variable description file (TRC) for each function port with activated test automation support during the build process. The following variables support the TA switch in your experiment software: <ul style="list-style-type: none">▪ IO_Signal This variable provides the signal that is used as input or output for the function block.▪ MDL_Signal This variable provides the signal that is used as input or output for the behavior model.▪ TA_Replacevalue This variable provides the substitute value that can be used to carry out the automation tasks as described for the use cases above. It can be provided, for example, by AutomationDesk from dSPACE.▪ TA_Switchvalue This variable lets you switch between the original signal and the substitute value.
-------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

You can change the values of the TA_Switchvalue variable and the TA_Switchvalue variable via experiment software (for example, ControlDesk) when the real-time application is running or stopped. The changes take effect immediately.

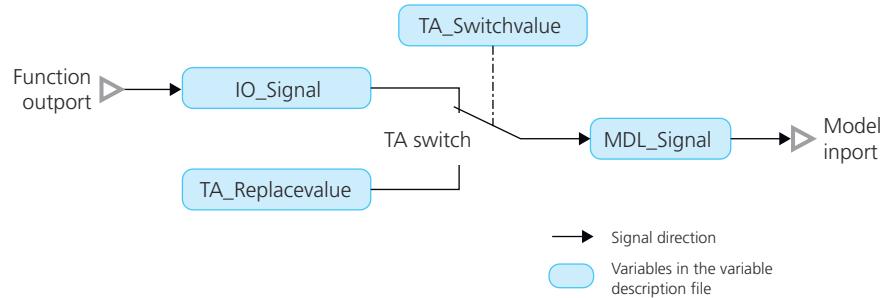
Test automation support for function imports When you activate test automation support for function imports, you can provide substitute values to the function import as shown in the following illustration.



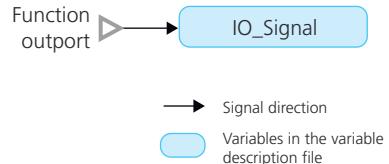
Even if the function port is not mapped to a model port, you can switch between the value of the `MDL_Signal` variable and the `TA_Replacevalue` variable. The value of the function port's `Initial value` property is used as the default value for the `MDL_Signal` variable. The following illustration shows the provided variables.



Test automation support for function outports When you activate test automation support for function outports, you can provide substitute values to the mapped model import as shown in the following illustration.



If the function port is not mapped to a model port, ConfigurationDesk adds only the `IO_Signal` variable to the variable description file as shown in the following illustration.



Specifying the initial behavior

In ConfigurationDesk, you can specify the initial behavior of the test automation support for each function port via the Properties Browser:

- The Initial switch setting property lets you specify the initial setting of the TA_Switchvalue variable.
- The Initial substitute value property lets you specify the initial value of the TA_Replacevalue variable.

Configuring Standard Features of the Electrical Interface

Objective	The electrical interface provides the interface to the hardware (real-time hardware and external devices via signal ports). The function blocks offer standard configuration features which influence this interface.
------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Where to go from here	Information in this section
	Specifying Current and Voltage Values for Channel Multiplication..... 95 Specifying Digital Output Signals (Flexible Out 1)..... 98 Specifying Digital Output Signals (Digital Out 1)..... 102 Specifying Digital Output Signals (Digital Out 2, Digital In/Out 1)..... 106 Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3)..... 109 Specifying Digital Output Signals (Digital In/Out 5)..... 112 Specifying the Circuit Type for Voltage Input Signals..... 116 Specifying the Circuit Type for Analog Output Signals..... 118 Specifying the Measurement Point for Input Signals..... 119

Specifying Current and Voltage Values for Channel Multiplication

Context	For most function blocks, you can specify the maximum values of the expected electrical signal (for signal inputs) or the required maximum values (for signal outputs) via the Required voltage or the Required current properties.
Purpose	<p>A hardware channel is able to provide a certain voltage and current range depending on the channel type. ConfigurationDesk uses the specified value to calculate the number of hardware channels needed.</p> <p>With this <i>channel multiplication</i> feature, the max. current or max. voltage of a single hardware channel can be enhanced.</p>
Effects on the specified values	Based on the number of channels and the channel type, the system min./max. values arise (system saturation range). Example: Suppose one channel can generate 0 ... 10 V. If you specify 15 V as the required voltage, 2 channels are allocated by ConfigurationDesk. In consequence, these two channels could

provide voltage signals in the range 0 ... 20 V, so the system saturation range is also extended to 0 ... 20 V.

Note

You cannot use the specified values for safety purposes, for example, to saturate output signals to protect your external device. The maximum voltage and current values which are available at a signal port depend only on the maximum system values (system saturation). Therefore the specified Required voltage and the Required current values do not define a user saturation range. User saturation is possible at each function import. There you can specify user saturation min/max values. For details, refer to [Specifying User Saturation](#) on page 90.

Specified values are absolute values

The values you edit via the Required voltage or the Required current properties are always treated as absolute values. However, the signal ports provide/support unipolar signals as well as bipolar signals, depending on the electrical characteristics of the channel type.

Limitations for channel multiplication

- Various function block types do not support channel multiplication. In this case the voltage and current ranges of a single hardware channel apply. Affected function blocks: SENT In, SENT Out, Knock Signal Out, Crank/Cam Current Sink, Crank/Cam Digital Out, Current Signal Capture.
- Some hardware resources, such as the DS6101 Multi-I/O Board do not support channel multiplication in general.
- Channel multiplication across several I/O boards and several channels sets is not supported.

Current or voltage enhancement

Depending on the function block type, channel multiplication is provided either for current enhancement or for voltage enhancement. For example, the Multi Bit Out function block type provides channel multiplication for current enhancement. The Voltage Out function block type (if the Flexible Out 1 channel type is selected) provides channel multiplication for voltage enhancement.

Examples for current enhancement

- One channel of the DS2621 Signal Generation Board provides a max. current of 40 mA. If you specify 60 mA, ConfigurationDesk determines two channels which must be assigned to the function block. 10 channels of the DS2621 sink a maximum current of 400 mA.
- One channel of the DS2601 Signal Measurement Board processes a maximum current of 10 A. If you specify 24 A, ConfigurationDesk determines three channels which must be assigned to the function block. 10 channels of the DS2601 can process a maximum input current of 80 A.

Note

Note that for input channels a safety reserve due to current imbalance effects is calculated when channel multiplication is used.

Identification You can identify the type of channel multiplication via the channel request entries for the hardware resource assignment. The following illustration shows two examples:

Hardware Resource Assignment	
... Assigned channel set	DS2621 Signal Ge...
... Voltage Out, U0	
... Voltage Out, U1	
... Voltage Out, U2	

Hardware Resource Assignment	
... Assigned channel set	DS2680 I/O Modu...
... Current Sink, I0	
... Current Sink, I1	
... Current Sink, I2	
... Current Sink, I3	

As shown above, channel requests with the syntax *<Function block type name> - Ux* identify channel multiplication for voltage enhancement and channel requests with the syntax *<Function block type name> - Ix* identify channel multiplication for current enhancement.

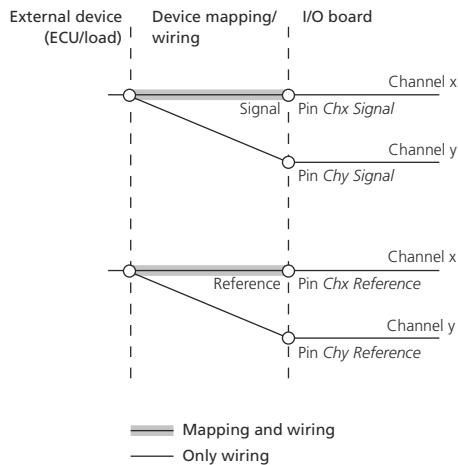
Electrical connection Channel multiplication means electrically:

- For current enhancement: Two or more channels are connected in parallel.
- For voltage enhancement: Two or more channels are connected in series.

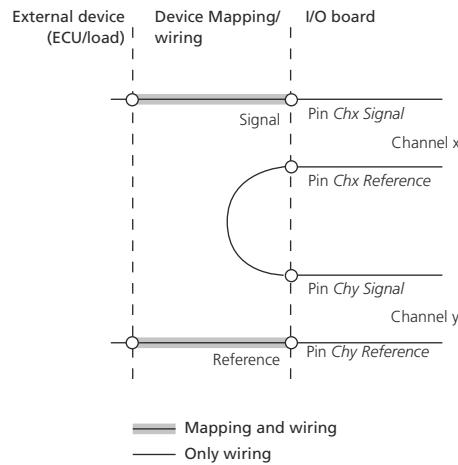
Mapping and wiring

If channel multiplication is performed, note that the device port mapping displayed in the signal chain shows only the logical signal connections. The signal chain does not display the complete external wiring in the cable harness. The illustrations below show these differences.

Channel multiplication for current enhancement The illustration below shows two channels connected in parallel.



Channel multiplication for voltage enhancement The illustration below shows two channels connected in series.



Specifying Digital Output Signals (Flexible Out 1)

Context

You can specify the electrical characteristics of the outputs of function blocks which generate digital output signals, for example, the Multi Bit Out or the Digital Incremental Encoder Out function blocks.

The characteristics and the configuration features described below are valid for the Flexible Out 1 channel type.

Configuration features

For the Flexible Out 1 channel type, the relevant function blocks provide the following features to configure the required output signal (binary 0 or binary 1):

- Specifying the polarity of the output signal
- Specifying the interface type (galvanically isolated switch, high-side switch, low-side switch, push-pull configuration)
- Setting digital output signals to tristate (available only for the push-pull configuration)

Polarity of digital output signals

You can adapt the polarity of the digital output signals to different requirements without the need to use inverters in the behavior model.

With the **Polarity** property, you can set the polarity of the output signals to **Active high** or to **Active low**:

- Active high: The higher voltage of the signal represents a binary 1, and the lower voltage represents a binary 0.
- Active low: The lower voltage of the signal represents a binary 1, and the higher voltage represents a binary 0.

This property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

Specifying the interface type

The digital outputs can be operated as high-side switch, low-side switch, galvanically isolated switch (all use one channel), and in push-pull configuration (uses two channels) to get the required output signal (binary 0 or binary 1).

The specified interface type may require the configuration of a high reference potential as shown in the following table:

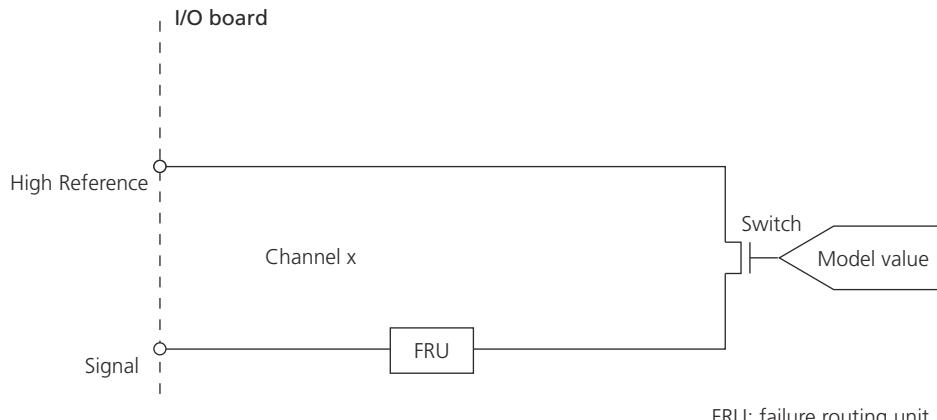
Property Setting of ...		Available Signal Ports
Interface Type	High Reference Potential 1) ¹⁾	
High-side switch	<ul style="list-style-type: none"> ▪ Individual ▪ VBat ▪ Shared ▪ Shared 2 	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal
Low-side switch	–	<ul style="list-style-type: none"> ▪ Signal ▪ Reference
Galvanically isolated switch	–	<ul style="list-style-type: none"> ▪ Signal ▪ Reference
Push-pull	<ul style="list-style-type: none"> ▪ Individual ▪ VBat ▪ Shared ▪ Shared 2 	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal ▪ Reference

¹⁾ You can connect any potential to the High Reference and Reference signal ports. The channels of Flexible Out 1 channel type are galvanically isolated to the ground potential of the dSPACE real-time hardware. This allows the connection of a potential shifted signal.

Note

The table above shows applicable settings for the High Reference Potential property. It is your responsibility to connect suitable signals to the signal ports. Conflicts are only generated and displayed if the settings do not match the functionality of the assigned hardware resources in general.

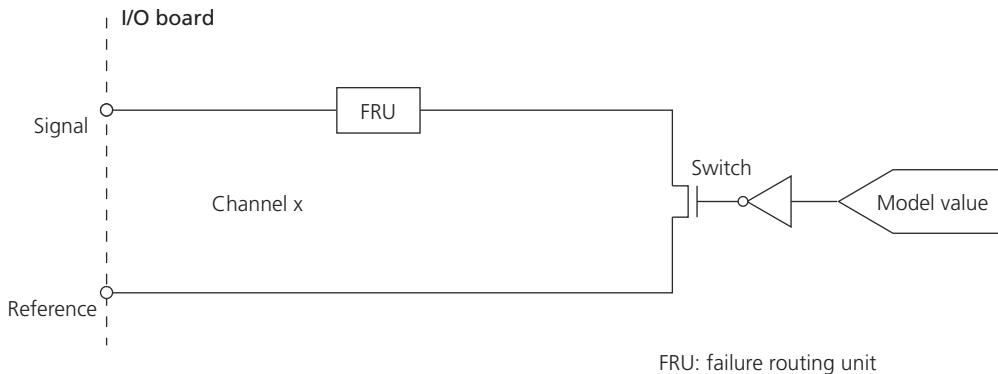
High-side switch For the high-side switch interface type a single channel is used as a simple switch. The signal is connected/disconnected to/from a reference potential that is higher than the signal itself.



The following table shows the output behavior according to the polarity setting and the model value.

Output Behavior	Polarity Setting	Model Value
Signal is connected to High Reference.	Active high	1
	Active low	0
Signal is not connected.	Active high	0
	Active low	1

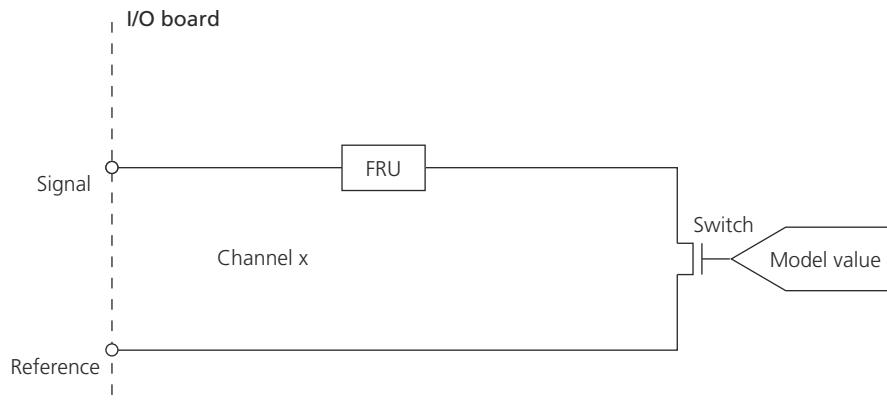
Low-side switch For the low-side switch interface type a single channel is used as a simple switch. The signal is connected/disconnected to/from a reference potential that is lower than the signal itself.



The following table shows the output behavior according to the polarity setting and the model value.

Output Behavior	Polarity Setting	Model Value
Signal is connected to Reference.	Active high	0
	Active low	1
Signal is not connected.	Active high	1
	Active low	0

Galvanically isolated switch For the galvanically isolated switch interface type a single channel is used as a simple switch without any potential reference. The signal is connected/disconnected to/from an individual reference potential which only is limited by the min./max. values of the assigned hardware resource.

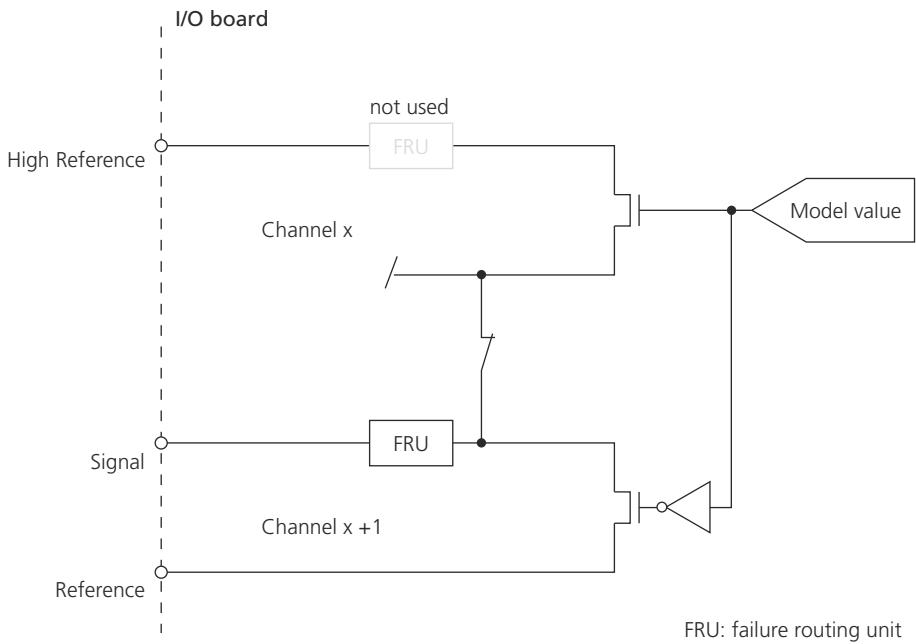


FRU: failure routing unit

The following table shows the output behavior according to the polarity setting and the model value.

Output Behavior	Polarity Setting	Model Value
Signal is connected to Reference.	Active high	1
	Active low	0
Signal is not connected.	Active high	0
	Active low	1

Push-pull Push-pull configuration requires two neighboring channels. These channels are connected together in order to switch the signal between two different potentials (High Reference and Reference), for example, VBAT and ground.



The following table shows the output behavior according to the polarity setting and the model value.

Output Behavior	Polarity Setting	Model Value
Signal is connected to High Reference.	Active high	1
	Active low	0
Signal is connected to Reference.	Active high	0
	Active low	1

Enabling tristate

In push-pull configuration, the digital outputs can be disconnected completely by setting the channel to high impedance (tristate) from the behavior model during run time. This is done independently of the current states of the function's inputs and internal conditions. If you enable the Tristate property, the function block provides a separate function port (Output Enable) to control this feature from the behavior model.

Specifying Digital Output Signals (Digital Out 1)

Context

You can specify the electrical characteristics of the outputs of function blocks which generate digital output signals, for example, the Multi Bit Out or the PWM/PFM Out function blocks.

The characteristics and the configuration features described below are valid for the Digital Out 1 channel type.

Configuration features

For the Digital Out 1 channel type, the relevant function block types provide the following features to configure the required output signal (binary 0 or binary 1):

- Specifying the polarity of the output signal
- Specifying the interface type (high-side switch, low-side switch, push-pull configuration)
- Setting digital output signals to tristate (available only for the push-pull configuration)

Polarity of digital output signals

You can adapt the polarity of the digital output signals to different requirements without the need to use inverters in the behavior model.

With the **Polarity** property, you can set the polarity of the output signals to **Active high** or to **Active low**:

- Active high: The higher voltage of the signal represents a binary 1, and the lower voltage represents a binary 0.
- Active low: The lower voltage of the signal represents a binary 1, and the higher voltage represents a binary 0.

This property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

Specifying the interface type

The digital outputs can be operated as low-side switch, high-side switch, and in push-pull configuration to get the required output signal (binary 0 or binary 1).

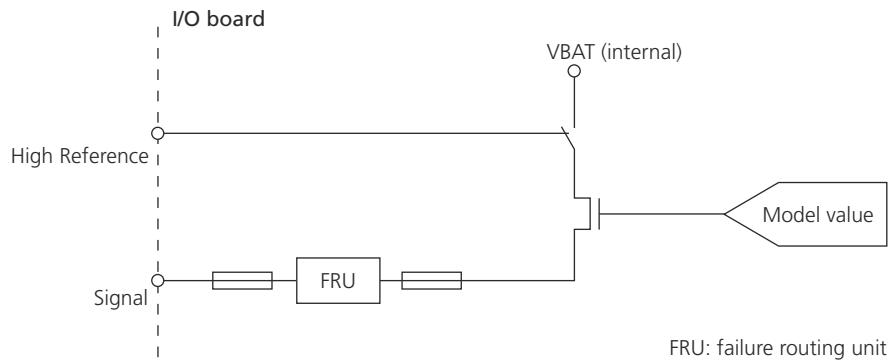
The specified interface type may require the configuration of a high reference potential as shown in the following table:

Property Setting of ...		Available Signal Ports
Interface Type	High Reference Potential	
High-side switch	Shared ¹⁾	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal
	VBat ²⁾	<ul style="list-style-type: none"> ▪ Signal
Low-side switch	–	<ul style="list-style-type: none"> ▪ Signal ▪ Reference
Push-pull	Shared ¹⁾	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal ▪ Reference
	VBat ²⁾	<ul style="list-style-type: none"> ▪ Signal ▪ Reference

¹⁾ You can connect any potential to the High Reference signal port. The Digital Out 1 channel type shares the connected potential with all other digital output channels that belong to a channel set. The potential is therefore available on a single pin of the I/O connector.

²⁾ The internal VBat of the dSPACE real-time hardware is used as high-reference potential. The High Reference signal port is not available.

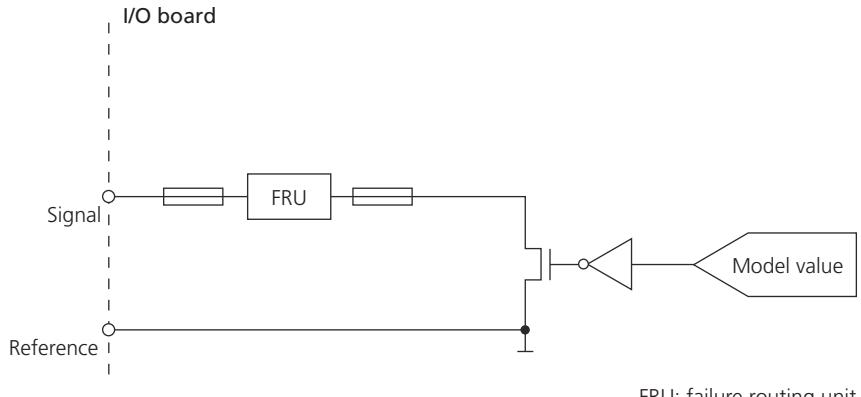
High-side switch For the high-side switch interface type, a single channel is used as a switch. The signal is connected/disconnected to/from a reference potential that is higher than the signal itself. This reference can be an internal reference (VBAT) of the SCALEXIO system or an external reference, for example, the 5 V reference output from your ECU.



The following table shows the output according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to VBAT (internal) or to an external potential via the High Reference signal port. The connected external potential is shared with all other digital output channels that belong to a channel set.	Active high	1
	Active low	0
Signal is not connected.	Active high	0
	Active low	1

Low-side switch For the low-side switch interface type, a single channel is used as a switch. The signal is connected/disconnected to/from ground potential (Reference).

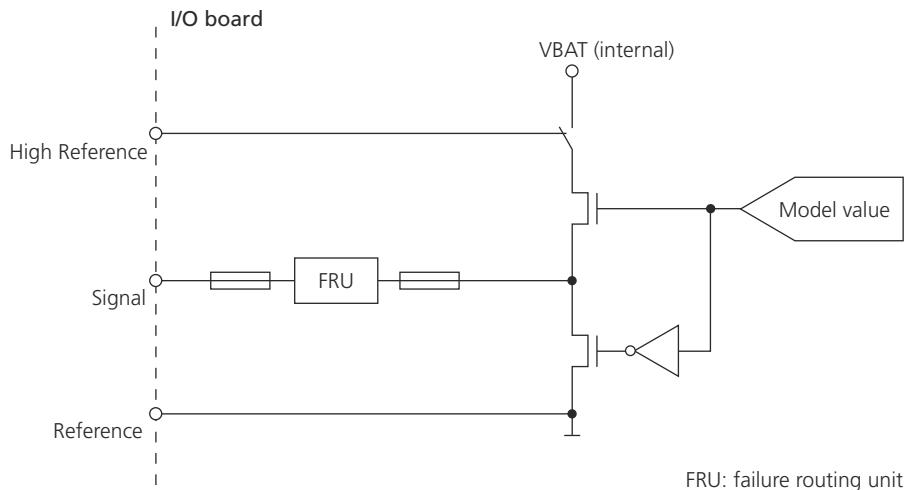


The following table shows the output according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to Reference.	Active high	0
	Active low	1

Output Behavior	Polarity Setting	Model Value
Signal is not connected.	Active high	1
	Active low	0

Push-pull configuration In push-pull configuration, a single channel is used to switch the signal between GND and another potential (internal or external reference), for example, VBAT of the SCALEXIO system.



The following table shows the output according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to VBAT (internal) or to an external potential via the High Reference signal port. The connected external potential is shared with all other digital output channels that belong to a channel set.	Active high	1
	Active low	0
Signal is connected to Reference.	Active high	0
	Active low	1

Enabling tristate

In push-pull configuration, the digital outputs can be disconnected completely by setting the channel to high impedance (tristate) from the behavior model during run time. This is done independently of the current states of the function's inputs and internal conditions. If you enable the Tristate property, the function block provides a separate function port (Output Enable) to control this feature from the behavior model.

Specifying Digital Output Signals (Digital Out 2, Digital In/Out 1)

Context

You can specify the electrical characteristics of the outputs of function blocks which generate digital output signals, for example, the Multi Bit Out or the PWM/PFM Out function blocks.

The characteristics and the configuration features described below are valid for the Digital Out 2 and the Digital In/Out 1 channel types.

Configuration features

For the Digital Out 2 and Digital In/Out 1 channel types, the relevant function blocks provide the following features to configure the required output signal (binary 0 or binary 1):

- Specifying the polarity of the output signal
- Specifying the interface type (high-side switch, low-side switch, push-pull configuration)
- Setting digital output signals to tristate (available only for the push-pull configuration)

Polarity of digital output signals

You can adapt the polarity of the digital output signals to different requirements without the need to use inverters in the behavior model.

With the **Polarity** property, you can set the polarity of the output signals to Active high or to Active low:

- Active high: The higher voltage of the signal represents a binary 1, and the lower voltage represents a binary 0.
- Active low: The lower voltage of the signal represents a binary 1, and the higher voltage represents a binary 0.

This property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

Specifying the interface type

The digital outputs can be operated as low-side switch, high-side switch, and in push-pull configuration to get the required output signal (binary 0 or binary 1).

The specified interface type may require the configuration of a high reference potential as shown in the following table:

Property Setting of ...		Available Signal Ports
Interface Type	High Reference Potential ¹⁾	
High-side switch	<ul style="list-style-type: none"> ▪ Individual ▪ VBat ▪ Shared ▪ Shared 2 	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal
Low-side switch	–	▪ Signal

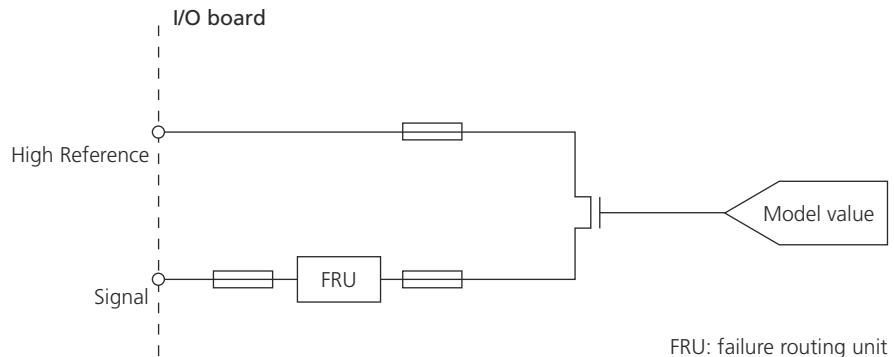
Property Setting of ...		Available Signal Ports
Interface Type	High Reference Potential ¹⁾	
Push-pull	<ul style="list-style-type: none"> ▪ Individual ▪ VBat ▪ Shared ▪ Shared 2 	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal

¹⁾ You can connect any potential to the High Reference signal port. Each channel of the channel types provides its own electrical high reference potential. Therefore, each channel is independent of the other channels. This allows the connection of an individual arbitrary potential.

Note

The table above shows applicable settings for the High Reference Potential property. It is your responsibility to connect suitable signals to the signal ports. Conflicts are only generated and displayed if the settings do not match the functionality of the assigned hardware resources in general.

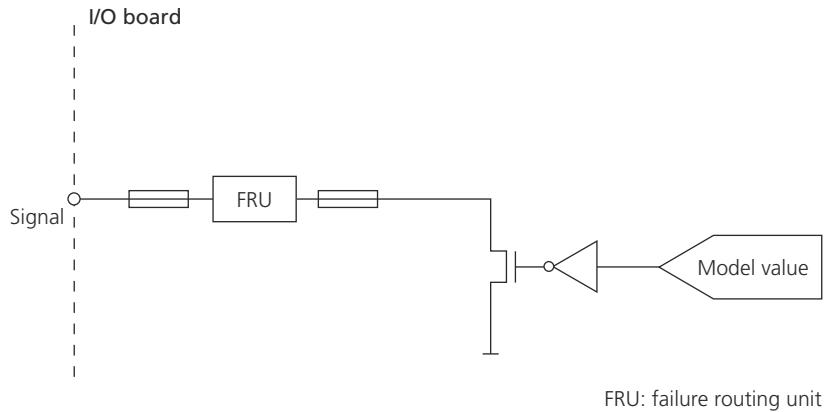
High-side switch For the high-side switch interface type, a single channel is used as a switch. The signal is connected/disconnected to/from a reference potential that is higher than the signal itself.



The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to High Reference. Each channel provides its own electrical high reference potential. Therefore, each channel is independent of the other channels.	Active high	1
	Active low	0
Signal is not connected.	Active high	0
	Active low	1

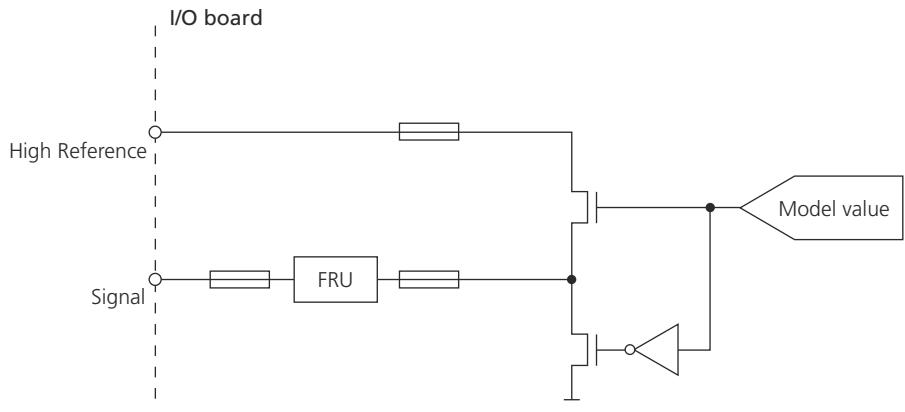
Low-side switch For the low-side switch interface type, a single channel is used as a switch. The signal is connected/disconnected to/from a ground potential of the dSPACE real-time hardware.



The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to the ground potential of the dSPACE real-time hardware.	Active high	0
	Active low	1
Signal is not connected.	Active high	1
	Active low	0

Push-pull configuration In push-pull configuration, a single channel is used to switch the signal between the ground potential of the dSPACE real-time hardware and an external reference, for example, VBAT of the SCALEXIO system.



The following table shows the output according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to High Reference. Each channel provides its own electrical high reference potential. Therefore, each channel is independent of the other channels.	Active high	1
	Active low	0
Signal is connected to the ground potential of the dSPACE real-time hardware.	Active high	0
	Active low	1

Enabling tristate	In push-pull configuration, the digital outputs can be disconnected completely by setting the channel to high impedance (tristate) from the behavior model during run time. This is done independently of the current states of the function's inputs and internal conditions. If you enable the Tristate property, the function block provides a separate function port (Output Enable) to control this feature from the behavior model.
--------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3)

Context	You can specify the electrical characteristics of the outputs of function blocks which generate digital output signals, for example, the Multi Bit Out or the PWM/PFM Out function block. The characteristics and the configuration features described below are valid for the Digital Out 3 and Digital In/Out 3 channel types.
Configuration features	For the Digital Out 3 and Digital In/Out 3 channel types, the relevant function blocks provide the following features to configure the required output signal (binary 0 or binary 1): <ul style="list-style-type: none"> ▪ Specifying the polarity of the output signal ▪ Specifying the interface type (high-side switch, low-side switch, push-pull configuration) ▪ Setting digital output signals to tristate (available only for the push-pull configuration)
Polarity of digital output signals	You can adapt the polarity of the digital output signals to different requirements without the need to use inverters in the behavior model. With the Polarity property, you can set the polarity of the output signals to Active high or to Active low: <ul style="list-style-type: none"> ▪ Active high: The higher voltage of the signal represents a binary 1, and the lower voltage represents a binary 0. ▪ Active low: The lower voltage of the signal represents a binary 1, and the higher voltage represents a binary 0. This property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.
Specifying the interface type	The digital outputs can be operated as low-side switch, high-side switch, and in push-pull configuration to get the required output signal (binary 0 or binary 1).

The specified interface type may require the configuration of a high reference potential as shown in the following table:

Property Setting of ...		Available Signal Ports
Interface Type	High Reference Potential	
High-side switch	<ul style="list-style-type: none"> ▪ VBat¹⁾ ▪ Shared¹⁾ ▪ Shared 2^{1), 2)} 	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal
Low-side switch	–	<ul style="list-style-type: none"> ▪ Signal ▪ Reference
Push-pull	<ul style="list-style-type: none"> ▪ VBat¹⁾ ▪ Shared¹⁾ ▪ Shared 2^{1) 2)} 	<ul style="list-style-type: none"> ▪ High Reference ▪ Signal ▪ Reference

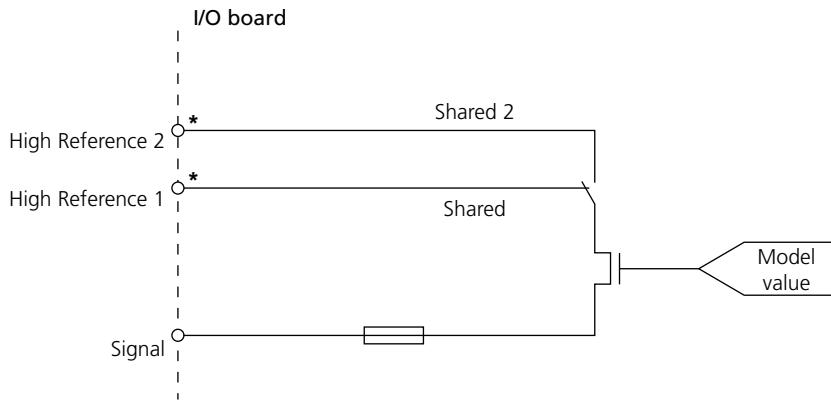
¹⁾ You can connect any potential to the High Reference signal port. The Digital Out 3 channel type shares the connected potential with all other digital output channels that belong to a channel set. The potential is therefore available on a single pin of the I/O connector.

²⁾ The connector pin that shares this potential (Shared 2) is not the same as the pin that is used for the Shared setting.

Note

The table above shows applicable settings for the High Reference Potential property. It is your responsibility to connect suitable signals to the signal ports. Conflicts are only generated and displayed if the settings do not match the functionality of the assigned hardware resources in general.

High-side switch For the high-side switch interface type, a single channel is used as a switch. The signal is connected/disconnected to/from a reference potential that is higher than the signal itself. You can switch between two different external reference potentials via the Shared and Shared 2 settings.

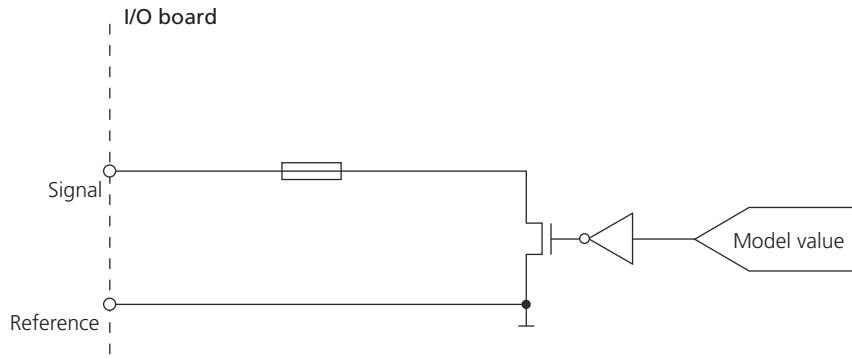


* Only one „High Reference“ port is available and can be used per channel

The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to High Reference. This port is electrically connected to one of the external reference potentials, which can be switched via the High Reference Potential property (to Shared or to Shared 2).	Active high	1
	Active low	0
The connected potential is shared with all other digital output channels that belong to a channel set.	Active high	0
	Active low	1
Signal is not connected.		

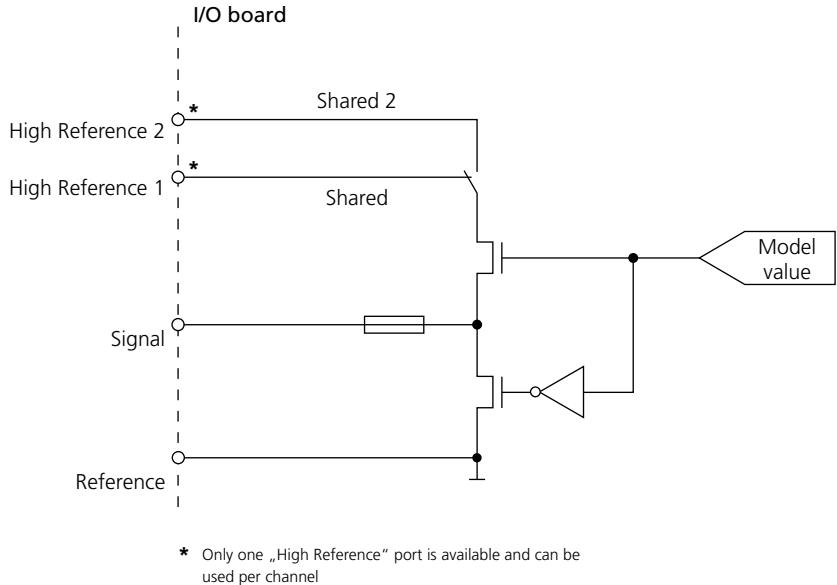
Low-side switch For the low-side switch interface type, a single channel is used as a switch. The signal is connected/disconnected to/from ground potential (Reference).



The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to Reference.	Active high	0
	Active low	1
Signal is not connected.	Active high	1
	Active low	0

Push-pull mode In push-pull configuration, a single channel is used to switch the signal between GND and another potential (internal or external reference), for example, VBAT of the SCALEXIO system.



The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to High Reference. This port is electrically connected to one of the external reference potentials, which can be switched via the High Reference Potential property (to Shared or to Shared 2). The connected potential is shared with all other digital output channels that belong to a channel set.	Active high	1
	Active low	0
Signal is connected to Reference.	Active high	0
	Active low	1

Enabling tristate

In push-pull configuration, the digital outputs can be disconnected completely by setting the channel to high impedance (tristate) from the behavior model during run time. This is done independently of the current states of the function's inputs and internal conditions. If you enable the Tristate property, the function block provides a separate function port (Output Enable) to control this feature from the behavior model.

Specifying Digital Output Signals (Digital In/Out 5)

Context

You can specify the electrical characteristics of the outputs of function blocks which generate digital output signals, for example, the Multi Bit Out or the PWM/PFM Out function block.

The characteristics and the configuration features described below are valid for the Digital In/Out 5 channel type.

Configuration features

For the Digital In/Out 5 channel type, the relevant function blocks provide the following features to configure the required output signal (binary 0 or binary 1):

- Specifying the polarity of the output signal
- Specifying the interface type (high-side switch, low-side switch, push-pull configuration)
- Setting digital output signals to tristate (available only for the push-pull configuration)

Polarity of digital output signals

You can adapt the polarity of the digital output signals to different requirements without the need to use inverters in the behavior model.

With the **Polarity** property, you can set the polarity of the output signals to **Active high** or to **Active low**:

- Active high: The higher voltage of the signal represents a binary 1, and the lower voltage represents a binary 0.
- Active low: The lower voltage of the signal represents a binary 1, and the higher voltage represents a binary 0.

This property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

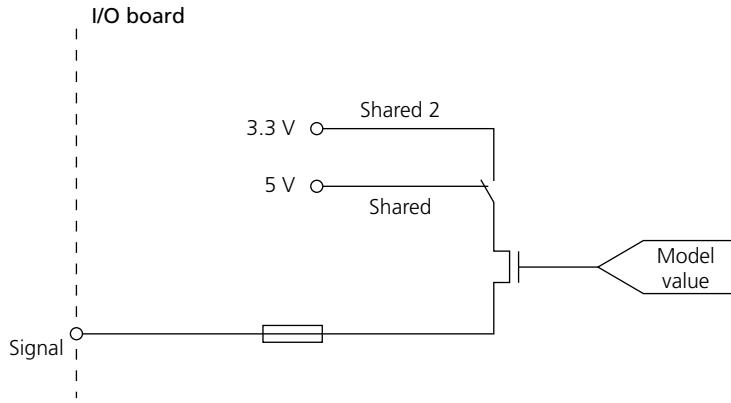
Specifying the interface type

The digital outputs can be operated as low-side switch, high-side switch, and in push-pull configuration to get the required output signal.

The specified interface type may require the configuration of a high-reference potential as shown in the following table:

Property Setting of ...		Available Signal Ports
Interface Type	High Reference Potential	
High-side switch	<ul style="list-style-type: none"> ▪ Shared ▪ Shared 2 	<ul style="list-style-type: none"> ▪ Signal
Low-side switch	–	<ul style="list-style-type: none"> ▪ Signal ▪ Reference
Push-pull	<ul style="list-style-type: none"> ▪ Shared ▪ Shared 2 	<ul style="list-style-type: none"> ▪ Signal ▪ Reference

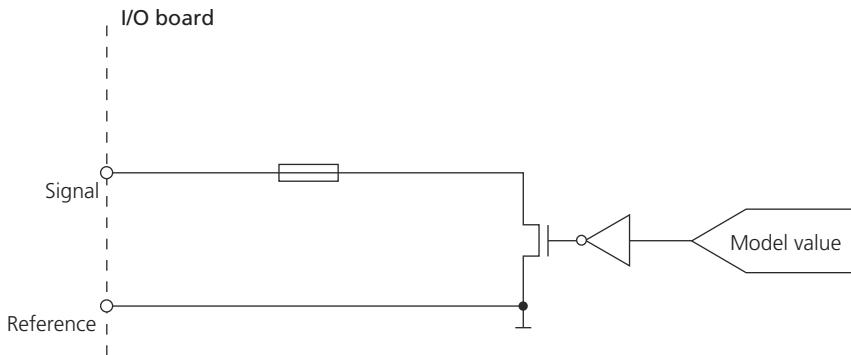
High-side switch For the high-side switch interface type, a single channel is used as a switch. The signal is connected to or disconnected from an internal high-reference potential. You can switch the internal reference to 5 V or 3.3 V via the **High Reference Potential** property.



The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to an internal high-reference potential. Each channel can be switched separately to an internal high-reference potential. Therefore, you can set the High Reference Potential property of each function block to Shared or to Shared 2 independent of other function blocks.	Active high	1
	Active low	0
Signal is not connected.	Active high	0
	Active low	1

Low-side switch For the low-side switch interface type, a single channel is used as a switch. The signal is connected to or disconnected from ground potential (Reference).

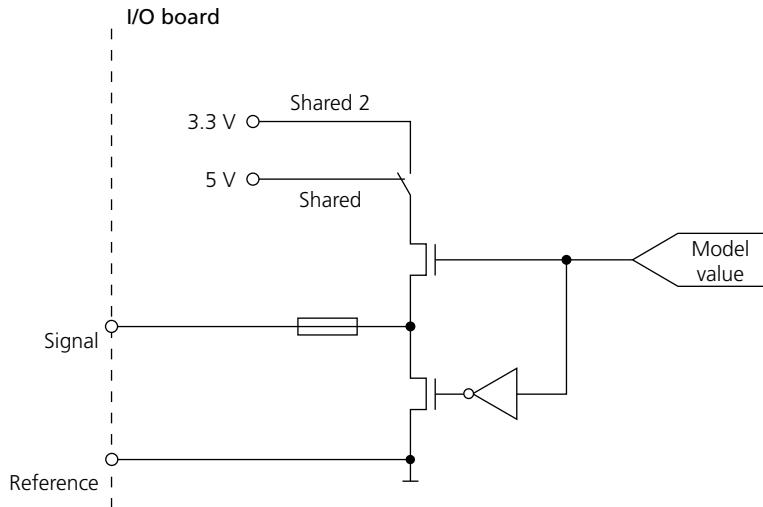


The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to Reference.	Active high	0
	Active low	1

Output Behavior	Polarity Setting	Model Value
Signal is not connected.	Active high	1
	Active low	0

Push-pull mode In the push-pull configuration, a single channel is used to switch the signal between GND and an internal high-reference potential. You can switch the internal high-reference to 5 V or 3.3 V via the High Reference Potential property.



The following table shows the output behavior according to the polarity setting and the model value:

Output Behavior	Polarity Setting	Model Value
Signal is connected to an internal high-reference potential. Each channel can be switched separately to an internal high-reference potential. Therefore, you can set the High Reference Potential property of each function block to Shared or to Shared 2 independent of other function blocks.	Active high	1
	Active low	0
Signal is connected to Reference.	Active high	0
	Active low	1

Enabling tristate

In push-pull configuration, the digital outputs can be disconnected completely by setting the channel to high impedance (tristate) from the behavior model during run time. This is done independently of the current states of the function's inputs and internal conditions. If you enable the Tristate property, the function block provides a separate function port (Output Enable) to control this feature from the behavior model.

Specifying the Circuit Type for Voltage Input Signals

Context Each I/O board and I/O unit has specific input circuits for the voltage input signals provided from the external devices that are connected to the dSPACE hardware.

Specifying the input circuit type You have to specify the required input circuit type via the **Interface type** property for the signal that is connected to the electrical interface of a function block. This helps you to assign a suitable channel set of the dSPACE hardware, because the associated channel types supports only a specific interface type:

- Select **Ground-based** if you want to connect the Reference signal port to the internal ground of the dSPACE hardware.
- Select **Galvanically isolated** if you want to connect an individual arbitrary potential other than ground potential to the Reference signal port.
- Select **Differential with ground sense** if you want to connect the Reference signal port to ground potential of the external device. This setting is provided only for analog voltage inputs.
- Select **Differential** if you want to connect the input signal to a fully differential electrical interface.

Interface type settings and supported channel types The table below shows the available interface types and the channel types which provide them.

Interface Type	Supported Channel Type	Available on	Description
Ground-based	Flexible In 2	DS2680 DS6101 DS1511 DS1511B1 DS1513 DS1552 DS1552B1 DS6201 DS6202 DS1554 DS6121 DS1521 DS1511	For these channel types, the Reference signal port is hard-wired to the internal ground of the dSPACE hardware.
	Digital In 1		
	Digital In 3		
	Digital In 4		
	Trigger In 3		
	Digital In 5		
	Digital In/Out 6		
	Digital In/Out 3		
	Digital In/Out 5		
	Digital In/Out 8		
	Digital In/Out 9		
	Digital In/Out 10		
	Analog In 7		
	Analog In 8		

Interface Type	Supported Channel Type	Available on	Description
		▪ DS1513	These channel types are referenced to the internal ground of the dSPACE hardware. To the Reference signal port you can connect an individual higher reference potential that is required for the signal.
	Analog In 9	DS1513	
	Analog In 12	▪ DS1552 ▪ DS1552B1	
	Analog In 17	DS1521	
	Digital In 2	DS2690	
	Digital In/Out 1		
	Digital In 9	DS1554	
Differential with ground sense	Analog In 1	DS2680	To the Reference signal port you can connect an external ground potential (for example, at the external device). This ground potential must be in a small voltage range and is limited by the min./max. values of the assigned hardware resource.
	Flexible In 3	DS6101	
	Analog In 4		
	Analog In 10	DS1552	
	Analog In 11	DS1552B1	
	Analog In 16	DS6121	
Differential	Analog In 6	DS6221	This channel type supports the connection of two complementary signals carrying data in the voltage difference between the Signal + and Signal - signal ports. Both signal ports have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.
	Analog In 14	DS1554	
	Analog In 15		
	Digital In/Out 5	DS6202	This channel type supports the connection of two complementary signals carrying data in the voltage difference between the Signal and Inverted Signal signal ports. Both signal ports have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.
	Flexible In/Out 1	DS6121	
Galvanically isolated	Flexible In 1	DS2601	This channel type is galvanically isolated to the ground potential of the dSPACE real-time hardware. To the Reference signal port you can connect an individual arbitrary potential which only is limited by the min./max. values of the assigned hardware resource.
	Digital In 10	DS1554	

The internal and external wiring of the different interface types is shown in channel-type specific circuit diagrams. Refer to [Circuit Diagrams of Channel Types](#) on page 1541.

Specifying the Circuit Type for Analog Output Signals

Context	Each I/O board and I/O unit has specific output circuits for the voltage output signals that are connected to the external devices.
Specifying the output circuit type	<p>You have to specify the required output circuit type via the Interface type property (Differential with ground sense, Galvanically isolated) for the signal that is connected to the external devices. This helps you to assign a suitable channel set of the dSPACE real-time hardware, because the associated channel types supports only a specific interface:</p> <ul style="list-style-type: none"> ▪ Select Ground-based if you want to connect the Reference signal port to the internal ground of the dSPACE hardware. ▪ Select Differential with ground sense if you want to connect the Reference signal port to ground potential of the external device. ▪ Select Galvanically isolated if you want to connect an individual arbitrary potential other than ground potential to the Reference signal port.
Reference potential settings and supported channel types	The table below shows the different reference potentials and the channel types which provide the corresponding potential.

Interface Type	Supported Channel Type	Available on	Description	
Ground-based	Analog Out 11	▪ DS1511 ▪ DS1511B1	For these channel types, the Reference signal port is hard-wired to the internal ground of the dSPACE hardware.	
	Analog Out 12			
	Analog Out 13			
Differential with ground sense	Analog Out 1	DS2680	To the Reference signal port (reference pin) you can connect an external ground potential (for example, at the external device). This ground potential must be in a small voltage range and is limited by the min./max. values of the assigned hardware resource.	
	Analog Out 2			
	Analog Out 4			
	Analog Out 6	DS6101		
	Analog Out 9			
	Analog Out 10			
Galvanically isolated	Flexible Out 1	DS2621	These channel types are galvanically isolated to the ground potential of the dSPACE real-time hardware. To the Reference signal port you can connect an individual arbitrary potential which only is limited by the min./max. values of the assigned hardware resource.	
	Analog Out 2	DS2680		
	Analog Out 3			
	Analog Out 7	DS6101		
	Analog Out 8			

The internal and external wiring of the different interface types is shown in channel-type specific circuit diagrams. Refer to [Circuit Diagrams of Channel Types](#) on page 1541.

Specifying the Measurement Point for Input Signals

Context

Depending on the signal type (voltage or current) and the channel type, either the measurement point can be defined or it is fixed and cannot be changed.

Hardware limitation

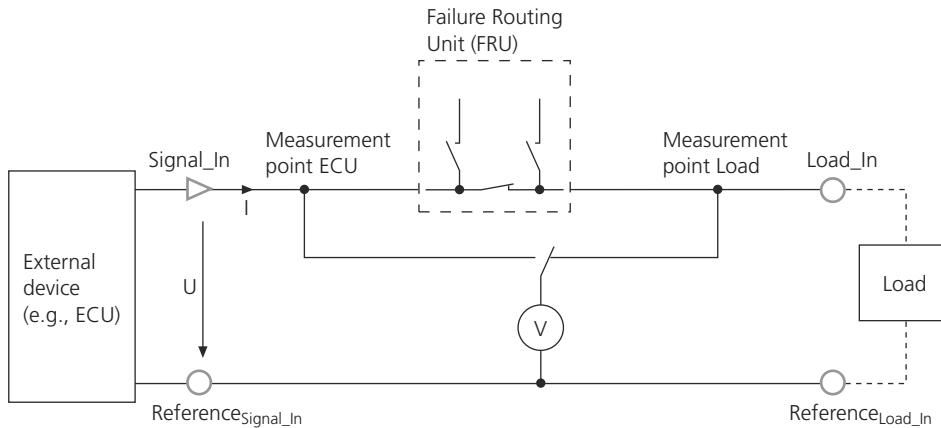
Note

The measurement point setting is only relevant for I/O boards which support failure simulation. Therefore the setting is only configurable for the following channel types:

- Flexible In 1 (available on DS2601)
- Flexible In 2, Digital In 1, Analog In 1, Analog In 2 (all available on DS2680)
- Digital In 2, Digital In/Out 1 (all available on DS2690)

Voltage measurement

You can define whether the voltage measurement is executed on the ECU side or on the load side. The signal can be interrupted or short-circuited by an electrical failure routing unit (FRU). If the measurement is performed on the load side and load rejection is enabled, the signal cannot be measured when the failure is activated.



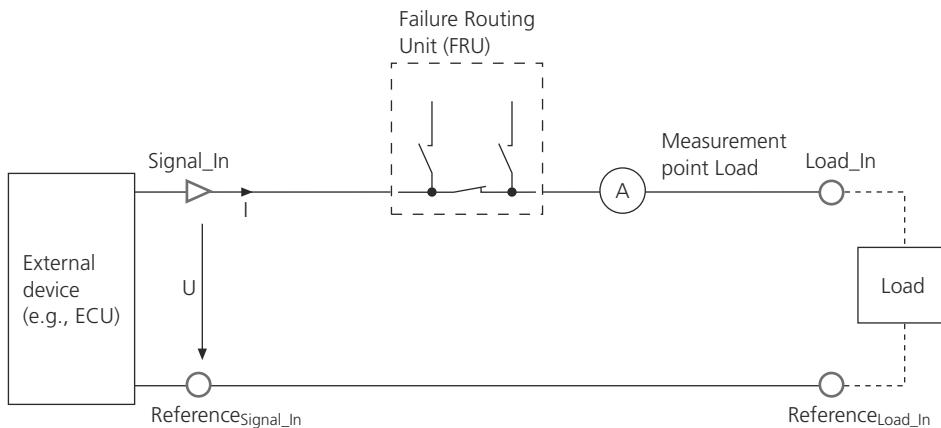
The table below shows which measurement points for voltage measurement are used for each channel type.

Channel Type	Available on	Measurement Point
Flexible In 1	DS2601	The measurement point can be chosen on either the: ▪ Load side ▪ ECU side

Channel Type	Available on	Measurement Point
Flexible In 2	DS2680	The measurement point is on the load side and cannot be changed. Measuring on the ECU side is not supported.
Digital In 1		
Analog In 1		
Analog In 2		
Digital In 2	DS2690	
Digital In/Out 1		

Current measurement

Current measurement always is provided on the load side. A load must be connected to measure currents. Measuring on the ECU side is not supported. The signal can be interrupted or short-circuited by an electrical failure routing unit (FRU). If load rejection is enabled, the signal cannot be measured when the failure is activated.

**Details on failure simulation**

For further information on the failure routing unit, refer to [Basics of Electrical Errors Simulation in a SCALEXIO System \(SCALEXIO – Hardware and Software Overview\)](#).

Configuring Standard Features of Signal Ports

Objective	The signal ports provide the interface to the external devices (e.g., ECU). The function block offers standard features to configure the properties of these interfaces.
------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Where to go from here	Information in this section
	Specifying Load Settings..... 121
	Specifying Fuse Settings..... 122
	Specifying Settings for Failure Simulation..... 123

Specifying Load Settings

Objective	To measure actuator signals, you have to use electrical loads such as resistors or real actuators such as electric motors. These loads can be represented in your ConfigurationDesk application. You can specify load settings for each signal import.
------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Hardware limitation	Note
	<p>Load settings are not relevant and therefore not configurable for the following channel types supporting input signals:</p> <ul style="list-style-type: none"> ▪ Analog In 4, Analog In 5, Digital In 3, Flexible In 3 (all available on DS6101) ▪ Digital InOut 3 (available on DS6201)

Specifying load location	You can specify whether the real load is inserted directly on your real-time hardware (= internal load) or connected to your hardware via I/O connector (= external load). For further information on handling loads in the signal chain, refer to Handling Loads (ConfigurationDesk Real-Time Implementation Guide  .
---------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Enforcing load rejection	You can enforce load rejection at each signal port. Load rejection is used to protect sensitive loads against damage. If load rejection is enforced, the loads are
---------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------

automatically disconnected by the system when a failure such as a short circuit is simulated.

You can enforce load rejection at different positions in the signal chain. If load rejection is enforced at one of these positions, other settings are ignored. For further information on configuring load rejection in the signal chain, refer to [Basics on Load Rejection \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Load compare check

For using internal loads, ConfigurationDesk provides a load compare check (during configuration and during download) to ensure that each required load matches the load mounted on the hardware.

To support the load compare check, you can enter a load description for each signal port of the function block and a load description (to be stored on the hardware) for every hardware channel.

When a channel is assigned to a function block, the function block's load description is checked against the load description defined for the hardware. This check can be performed as follows:

- Compare mode is set to Load description is compared by system
ConfigurationDesk considers entries to be identical only if they contain the same characters (= string-based comparison). A failed check causes a conflict which is displayed in the Conflicts Viewer. The build process generates a warning message.
- Compare mode is set to Load description is compared by user
You have to verify that the descriptions are compatible yourself.
ConfigurationDesk does not perform a check, so different load descriptions do not cause conflicts and generate warnings.

Specifying Fuse Settings

Objective

Several hardware resources of the SCALEXIO system provide fuses which are configurable via ConfigurationDesk.

Hardware limitation

Note

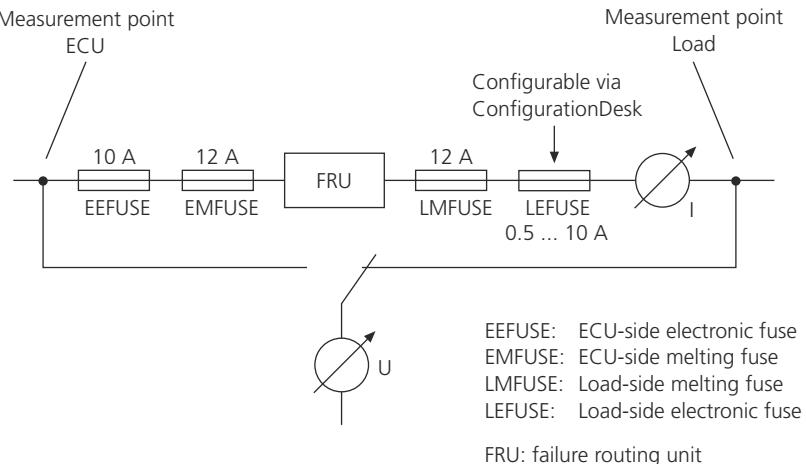
Only the following channel types provide configurable fuses:

- Flexible In 1 (available on DS2601)
- Flexible Out 1 (available on DS2621)

Fuse concept

The SCALEXIO hardware provides different types of fuses: software-configurable electronic fuses, non-configurable electronic fuses and melting fuses.

The types used and their locations in the signal path are channel-type-dependent. The following illustration shows the available fuses and their locations in the signal path of the Flexible In 1 channel type as an example. This channel type is available on the DS2601 Signal Measurement Board.



For the Flexible In 1 channel type, the trigger level (rated current) for the LEFUSE is configurable for each channel at the signal ports of the function blocks.

For more details on the fuse concept, refer to [Fuse Concept \(SCALEXIO Hardware Installation and Configuration\)](#).

Monitoring and resetting fuses

The state of the electronic fuses can be monitored and the electronic fuses can be reset (after removing the failure) in your experiment software. Two global variables are available for this in the TRC file (variable description file).

For further information on the TRC file, refer to [Diagnostics Group \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Specifying Settings for Failure Simulation

Objective

By using a failure simulation system, you can simulate failures in the cable harness of your ECU. To control the failure simulation with your experiment software, you have to specify the allowed failure classes in ConfigurationDesk. You have to specify these settings for each signal separately at the function blocks.

Supported hardware	Only the channel types available on the following I/O boards do support failure simulation: <ul style="list-style-type: none">▪ DS2601 Signal Measurement Board▪ DS2680 I/O Unit▪ DS2690 Digital I/O Board
Allowed failure classes	<p>Each kind of failure which can be simulated is defined as a failure class. These failure classes can then be simulated via the experiment software.</p> <p>The following failure classes are provided by the SCALEXIO system and are supported by the software tools involved:</p> <ul style="list-style-type: none">▪ Open circuit▪ Short to GND▪ Short to VBAT▪ Short to signal measurement channel▪ Short to signal generation channel▪ Short to bus channel
Load rejection or bus signal dropping (optional)	<p>Effects on the settings Failure classes which are set to Allowed at the signal port of a function block can be used in your experiment software. The settings for each signal are analyzed during the build process.</p> <p>A conflict is shown in the Conflicts Viewer if a failure class is allowed at a signal port but not allowed in the mapped device port.</p> <p>Not allowed settings override the Allowed settings in a signal chain. This means that if a class is not allowed at one function block in the signal chain, failure simulation for that class is not possible for the signal. If a signal port is mapped multiple times, the common subset is found.</p> <p>You can use load rejection or bus signal dropping to protect hardware such as sensitive loads or bus transceivers against damage during failure simulation. Enforcing load rejection or bus signal dropping for a signal in ConfigurationDesk means that you cannot use failure simulation without load rejection or bus signal dropping for that signal in the experiment software.</p> <p>For details and restrictions on load rejection and signal dropping, see Load Rejection and Signal Dropping (ConfigurationDesk Real-Time Implementation Guide).</p>
Multiple failure support (optional)	Using the Failure Simulation module of your experiment software, you can simulate failures for multiple signals simultaneously. For example, you can simulate a situation where a short to GND for one signal and an open circuit for a different signal happen at the same time. Multiple failures are not restricted to the same I/O unit: You can simulate multiple failures simultaneously for signals assigned to channels on different I/O units or even from different SCALEXIO racks.

As a precondition, the Activation by FRU relay property must be set to Allowed in ConfigurationDesk at all signal ports that are part of a multiple failure scenario. This way, the SCALEXIO system can use the electromechanical relays of the failure routing unit (FRU) to activate the failure directly by switching the relays under load instead of just switching the signal to the failplane. For details on failure simulation with a SCALEXIO system, refer to [Hardware for Electrical Error Simulation on SCALEXIO Systems \(SCALEXIO – Hardware and Software Overview\)](#).

NOTICE**Risk of increased wear and permanent damage**

When you use the “Activation by FRU relay” feature, there is a risk of increased wear and permanent damage to the relays of the dSPACE failure simulation hardware. For details, refer to [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System \(SCALEXIO – Hardware and Software Overview\)](#).

Further information

For information on the settings for failure simulation in the signal chain, refer to [Specifying Failure Simulation in ConfigurationDesk \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Functionalities of Function Blocks

Context

Basic configuration features influence the behavior of a function block highly. Some functionalities are treated as standard, because they are available at several function blocks and work in the same manner.

Using Angular Processing Units (APUs)

Basic principle of angular processing units

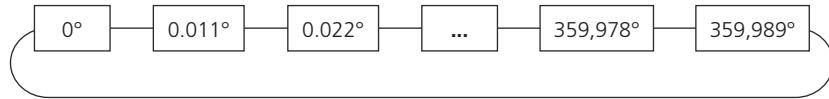
The angular processing unit (APU) is the functional core unit to simulate virtual piston engines and electric motors and to control real piston engines and electric motors. Additionally, it can also be used to simulate or control any other rotating device.

APUs calculate current angle positions as follows:

- For virtual piston engines and electric motors as a function of the rotation speed.
- For real piston engines and electric motors from the evaluated values of connected position sensors, e.g., encoders, or from the relative positions of crankshaft and camshaft.

The angular processing unit is implemented as an angle counter. The APU must be set to the angle range of the virtual or real piston engine or motor, which is usually 360° for e-drives and 720° for four-stroke piston engines.

The step size of the angle counter - rounded to 3 decimal digits - is 0.011°. Its exact step size is $360^\circ / 2^{15}$ and the angle range is 0° ... 359.989° for the 360° angle range or 0° ... 719.989° for the 720° angle range. After the upper range limit is reached, the counter is reset to 0°. The following illustration shows the operating principle of a 360° angle counter.



Note

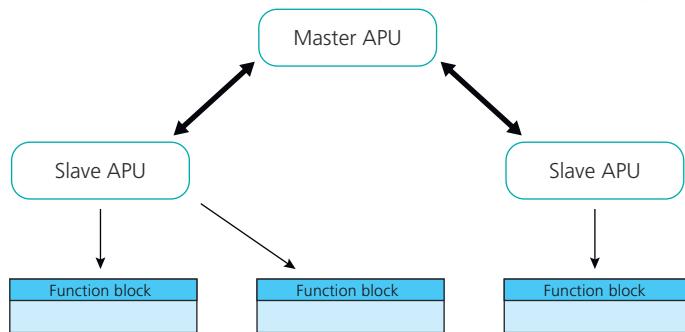
Deviating from the previous description, the step size of an angle counter on a DS1514 FPGA Base Board with DS1554 Engine Control I/O Module is 0.1°.

Forward and reverse rotation The angular processing unit counts forward or backward depending on the rotation of the encoder or engine. Counting forward means forward rotation, and vice versa.

Distribution of the angle values

Angle values are calculated by a master APU that is executed by an angle unit. Angle units are provided by Processing Units/processor boards, certain I/O boards, and FPGA boards (refer to [Hardware resources with angle units](#) on page 127).

Function blocks do not directly access the master APU. On the hardware resources that are assigned to function blocks, slave APUs are executed synchronously to the master APU and provide the angle values for the function blocks. The following illustration shows the distribution of the angle values to the function blocks that require the position of an encoder or piston engine.



Hardware resources with angle units The following hardware resources provide angle units to execute master APUs:

- Processing Units/processor boards
 - SCALEXIO Processing Unit
 - DS6001 Processor Board
- I/O boards:
 - DS6121 Multi-I/O Board
 - DS6202 Digital I/O Board
- FPGA boards (APUs implemented by FPGA custom function blocks):
 - DS2655 FPGA Base Board
 - DS6601 FPGA Base Board
 - DS6602 FPGA Base Board
 - DS1514 FPGA Base Board with DS1554 Engine Control I/O Board (APU implemented by a Crank In function block)

Configuring the access to master APUs

Certain function blocks can access a master APU and therefore serve as master APU providers.

Encoder evaluating function blocks For applications, such as electric drives, encoders are used as angle sources.

Therefore, the following function blocks can, for example, access the angle units of the DS6121 Multi-I/O Board, and then work as master APU providers:

- Sine Encoder In
- Hall Encoder In
- Resolver In

- SSI Master
- EnDat Master
- Digital Incremental Encoder In

If the master APU provider functionality of these function blocks is enabled, other function blocks, such as Block-Commutated PWM Out, can use the measured/calculated angle values.

Angular Clock Setup For engine simulation applications, the Angular Clock Setup function block configures and initializes the access to a master APU that is executed by an angle unit of a processing unit/processor board. It must be assigned to an Engine Simulation Setup function block. Refer to [Basics on Engine Simulation with SCALEXIO](#) on page 364.

Crank In For engine control applications, the Crank In function block configures and initializes the access to a master APU that is executed on the DS1514 FPGA Base Board with DS1554 Engine Control I/O Board of a MicroAutoBox III. The Crank In function block is the only available master APU provider for engine control applications. It must be assigned directly or via an Engine Control Setup function block to all the related function blocks of an engine control application. Refer to [Basics on Engine Control with MicroAutoBox III](#) on page 550.

FPGA custom function blocks Master APUs of FPGA custom function blocks are configured in the FPGA application. If enabled, you can configure the angle range of the master APU. Refer to [Configuring the Basic Functionality \(FPGA\)](#) on page 1525.

Using the provided angle value

To use the provided angle value of a specific master APU, you have to assign a master APU provider to the function block that resides in your active ConfigurationDesk application. ConfigurationDesk automatically assigns a suitable slave APU to the function block.

The following rules apply:

- Function blocks can use only slave APUs of the hardware resource to which they are assigned.
- All function blocks that use a specific slave APU must be assigned to the same master APU provider. If, for example, the hardware resource provides one slave APU, all function blocks using that slave APU must reference the same master APU provider.

The actual number of slave APUs depends on the hardware resource that is assigned to the function block. For example, if the hardware resource provides six slave APUs, each function block can reference one of six different master APU providers.

Angle range of encoder evaluating function blocks Function blocks, such as Digital Incremental Encoder In (refer to [Encoder evaluating function blocks](#) on page 127), have a fixed angle range of 360° when they serve as master APU providers. The angle range of the consuming function blocks, such as Block-

Commutated PWM Out, is also 360°. Therefore, the angle values of the consuming function blocks are equal to the measured encoder position.

Angle range of piston engines One engine cycle of a piston engine can cover a 360° or a 720° angle range. The engine cycle of a four-stroke piston engine, for example, usually covers 720°. Therefore, the angle range of master APU providers (Angular Clock Setup for engine simulation on SCALEXIO hardware or Crank In for engine control on MicroAutoBox III hardware) can be set to 360° or 720°.

The angle range of the function blocks to which the master APU providers are assigned can be specified as follows:

- **SCADEXO hardware:**

For function blocks and FPGA custom function blocks as of RTI FPGA Programming Blockset 3.6, you can inherit the angle range of the assigned master APU provider or you specify a fixed angle range. The following table shows the results for the different angle range settings.

Angle Range Setting		Resulting Angle Range of the Function Block
Master APU	Function Block	
720°	Inherit	720°
	360°	360° One APU cycle runs twice through the 360° angle range of the function block. If you simulate a four-stroke piston engine, for example, the function block only depends on the crankshaft position.
	720°	720°
360°	Inherit	360°
	360°	360°
	720°	720° Two APU cycles are required to run through the 720° angle range. If you simulate a four-stroke piston engine, for example, the angle values of the function block are not clearly related to the camshaft position.

- **MicroAutoBox III hardware:**

The angle range of function blocks cannot be specified. It is always inherited from the assigned master APU provider.

Slave APUs of FPGA custom function blocks If you assign a master APU provider to a slave APU of an FPGA custom function block, you can select one of the following master APU providers:

- APU master signal provided by an Angular Clock Setup function block.
- APU master signal provided by another FPGA custom function block. The assigned master APU provider must not be a master APU of the same FPGA custom function block.

Current In

Where to go from here

Information in this section

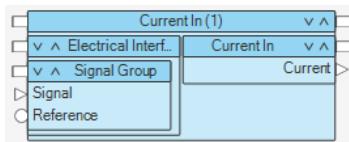
Introduction (Current In).....	132
Overviews (Current In).....	133
Configuring the Function Block (Current In).....	137
Hardware Dependencies (Current In).....	140

Introduction (Current In)

Introduction to the Function Block (Current In)

Function block purpose The Current In function block type digitizes analog current signals coming from an external device.

Default display The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features These are the main features:

- Measuring analog current signals and providing the digitized values to the behavior model.
- Capturing timestamps and angle positions at each measurement.
- Using analog and/or digital filters for the input signal.
- Generating I/O events and providing them to the behavior model.

Supported channel types The Current In function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	Flexible In 1	Flexible In 2
Hardware	DS2601	DS2680

Overviews (Current In)

Where to go from here

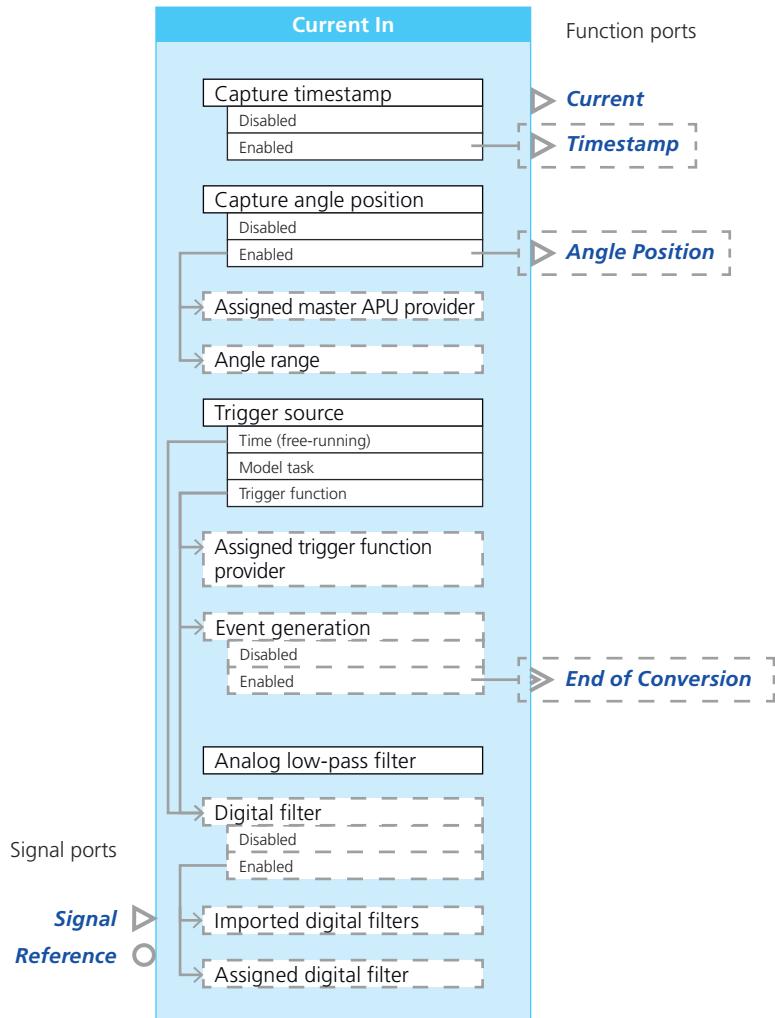
Information in this section

Overview of Ports and Basic Properties (Current In).....	133
Overview of Tunable Properties (Current In).....	136

Overview of Ports and Basic Properties (Current In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



The function block provides function outports that write data to the behavior model. To save computation time, you can disable some function ports if you do not need their values.

Current

This function outport writes the measured current value to the behavior model.

Value range	<ul style="list-style-type: none"> $I_{\min} \dots I_{\max}$ (in Ampere) Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current In) on page 140.
Dependencies	-

Timestamp This function outport writes the time the current value is measured to the behavior model.

Value range	0 ... 612,489,500 s
Dependencies	Available only if the Capture timestamp property is set to Enabled .

Angle Position This function outport writes the angle position the current value is measured to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0° ... 360° or 0° ... 720° ▪ The range depends on the Angle range property setting of the function block.
Dependencies	Available only if the Capture angle position property is set to Enabled .

End of Conversion This event port provides an I/O event each time the current measurement is completed (end of A/D conversion).

Value range	–
Dependencies	Available only if both of the following settings match: <ul style="list-style-type: none"> ▪ The Trigger source property is set to Trigger function. ▪ The Event generation property is set to Enabled.

Signal This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current In) on page 140.
Dependencies	–

Reference This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current In) on page 140.
Dependencies	–

Load Signal This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Supported only for the Flexible In 1 channel type.

Overview of Tunable Properties (Current In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Trigger source	✓	–
Assigned digital filter	✓	–
Analog low-pass filter	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Current In)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Current In)..... | 137 |
| Configuring Standard Features (Current In)..... | 139 |

Configuring the Basic Functionality (Current In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the trigger source.
- Enabling or disabling the capturing of timestamps and angle position data to save computation time.
- Using digital and/or analog filters for the input signal.
- Specifying the generation of I/O events.

Specifying the trigger source

You can choose between different trigger sources:

- If you select the **Time (free-running)** trigger source, the measurement runs independently from a trigger. It runs repeatedly at a fixed sample frequency, which depends on the assigned hardware.
The value of the latest measured signal is provided to the behavior model.
- If you select the **Model task** trigger source, the measurement is triggered synchronously to the model task in which the mapped model port block is executed. The function block starts to measure if an event triggers the model task to execute the next model step, for example, a timer event. The trigger is ignored if a measurement is still running.
- If you use a **Trigger function** as trigger source, the measurement starts with an external trigger that is provided by a specific trigger function. This can be, for example, the **Trigger In** function block. Trigger function providers have their own defined trigger conditions.

Using the **Trigger function** trigger source, you can enable the generation of an I/O event each time the measurement is completed (end of A/D conversion). These I/O events (interrupts) can be used as trigger source for runnable functions in the behavior model.

To use the I/O events in the behavior model, you have to assign them to a task via the **End of Conversion** property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

You can change the trigger source via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

Limitations Only the Flexible In 1 channel type supports the Model task trigger source.

Enabling or disabling the capturing of timestamps and angle positions

Depending on your requests, you can capture timestamps and angle positions and write them to the behavior model via specific function ports. Timestamp and angle position values are captured synchronously to the measured current values. To save computation time, you are recommended to disable capturing for data that is not required.

As a precondition for capturing angle positions you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Using digital filters

The function block supports the usage of digital filters, for example, to smooth the input signal or to reduce signal noise, or to adapt the signal bandwidth to a specific model sample frequency in order to avoid aliasing problems. However, the digital filter properties are only supported for the Time (free-running) trigger source. This is because the calculation of the filter coefficients is based on a specific sample frequency.

The filter coefficients are stored in digital filter files which you can import and use them afterwards. The supported filters and filter types depend on the assigned hardware resources. For example, the DS2601 Signal Measurement Board supports various digital filter types, such as, Butterworth filter.

You can change the assigned digital filter file via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

Limitation Importing and using digital filters is only supported for the Flexible In 1 channel type.

Using analog low-pass filter

You can use the analog low-pass filter, for example, to reduce signal noise. The filter characteristics depend on the assigned hardware resource. For details, refer to [Hardware Dependencies \(Current In\)](#) on page 140.

For further basic information on using the filter, refer to [Analog and Digital Filtering with the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Limitation The analog low-pass filter is only supported for the Flexible In 1 channel type.

Providing I/O events

If the Trigger source property is set to Trigger function and the Event generation property is set to Enabled, the function block generates I/O events to trigger runnable functions in the behavior model.

The Current In function block can generate an I/O event at the end of each current measurement.

To use the I/O events in the behavior model, you have to assign them to a task via the End of Conversion property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide](#) .

Configuring Standard Features (Current In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SCALEXIO			
Configuration Feature	Flexible In 2	Flexible In 1	Further Information
Channel multiplication	✓	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports			
Trigger level of electronic fuse	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Flexible In 2](#) on page 1595
- [Flexible In 1](#) on page 1593

Model interface	The interface to the behavior model of the function block provides the following standard features.
------------------------	-----------------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Current In)

SCALEXIO Hardware Dependencies (Current In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Flexible In 2	Flexible In 1
Hardware		DS2680 I/O Unit	DS2601 Signal Measurement Board
Event generation		✓	✓
Input current range		<ul style="list-style-type: none"> ▪ 0 ... 6 A_{RMS} for 1 channel ▪ 86.4 A_{RMS}¹⁾ for 18 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... 10 A_{RMS} for 1 channel ▪ 80 A_{RMS} for 10 channels (channel multiplication)
Input voltage range		0 ... 60 V	-60 V ... +60 V
Peak current per channel		±18 A	±30 A
Sample rate		85.7 kS/s (fixed, multiplexed)	0 ... 250 kS/s
Resolution		16 bit	16 bit
Offset error		±25 mA (typ.)	±5 mA (typ.)
Gain error		±1.0 % (typ.)	±0.1% (typ.)
Analog low-pass filter	Filter type	–	2 nd -order Bessel filter
	Edge frequency (-3dB cutoff frequency)		20 kHz
Trigger source		<ul style="list-style-type: none"> ▪ Time (free-running), with a sample frequency of 41.6 kHz 	<ul style="list-style-type: none"> ▪ Time (free-running), with a sample frequency of 250 kHz ▪ Model task
Using digital filter files		–	✓ ²⁾
Angular processing unit	Number of slaves	6	1
	Maximum speed	168,000 °/s	
Measurement point		Load side	
Configurable fuse		–	<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS}
Circuit diagrams		Flexible In 2 on page 1595	Flexible In 1 on page 1593
Required channels		1 (additional channels are required for current enhancements.)	

¹⁾ This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

²⁾ You only can use digital filter files if the Trigger source is set to Time (free-running) or Trigger function.

General limitations

Channel multiplication across several I/O boards is not supported.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Triggered Current In

Where to go from here

Information in this section

Introduction (Triggered Current In).....	144
Overviews (Triggered Current In).....	145
Configuring the Function Block (Triggered Current In).....	149
Hardware Dependencies (Triggered Current In).....	155

Introduction (Triggered Current In)

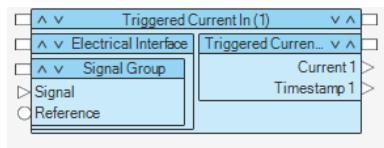
Introduction to the Function Block (Triggered Current In)

Function block purpose

The Triggered Current In function block type features the digitization of PWM-synchronous current signals coming from an external device, for example, to measure average currents of PWM driven valves and motors.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Measuring analog currents at specified trigger conditions and providing the digitized values to the behavior model.
- Capturing timestamps at each measurement and providing them to the behavior model.
- Using analog and/or digital filters for the input signal.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Triggered Current In function block type supports the following channel types:

	SCALEXIO		MicroAutoBox III
Channel type	Flexible In 1	Flexible In 2	–
Hardware	DS2601	DS2680	–

Overviews (Triggered Current In)

Where to go from here

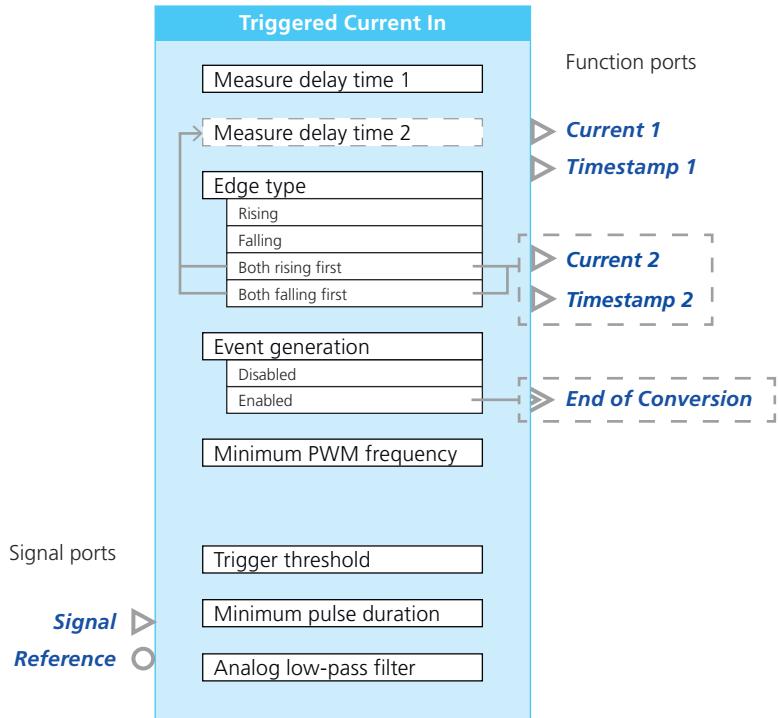
Information in this section

- [Overview of Ports and Basic Properties \(Triggered Current In\)..... 145](#)
- [Overview of Tunable Properties \(Triggered Current In\)..... 148](#)

Overview of Ports and Basic Properties (Triggered Current In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Current 1

This function outport writes the current value of the input signal that is first triggered on the input voltage signal to the behavior model.

Note

It depends on the Trigger edge type property whether the current value is measured on the rising or the falling edge of the input voltage signal.

Value range	<ul style="list-style-type: none"> ▪ $I_{\min} \dots I_{\max}$ (in Ampere). ▪ Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Triggered Current In) on page 155.
Dependencies	–

Current 2

This function outport writes the current value of the input signal that is second triggered on the input voltage signal to the behavior model.

Note

It depends on the Trigger edge type property whether the current value is measured on the rising or the falling edge of the input voltage signal.

Value range	<ul style="list-style-type: none"> ▪ $I_{\min} \dots I_{\max}$ (in Ampere). ▪ Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Triggered Current In) on page 155.
Dependencies	Available only if the Trigger edge type property is set to Both rising first or Both falling first.

Timestamp 1

This function outport writes the timestamp corresponding to the current value on the Current 1 function port to the behavior model.

Value range	0 ... 612,489,500 s
Dependencies	–

Timestamp 2

This function outport writes the timestamp corresponding to the current value on the Current 2 function port to the behavior model.

Value range	0 ... 612,489,500 s
Dependencies	Available only if the Trigger edge type property is set to Both rising first or Both falling first.

End of Conversion

This event port provides an I/O event each time the current measurement is completed (end of A/D conversion).

Value range	—
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Triggered Current In) on page 155.
Dependencies	—

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Triggered Current In) on page 155.
Dependencies	—

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Supported only for the Flexible In 1 channel type.

Overview of Tunable Properties (Triggered Current In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Measure delay time 1	✓	–
Measure delay time 2	✓	–
Minimum PWM frequency	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Trigger threshold	✓	–
Minimum pulse duration	✓	–
Analog low-pass filter	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Triggered Current In)

Where to go from here

Information in this section

- | | |
|---------------------------------------------------------------------------------|---------------------|
| Configuring the Basic Functionality (Triggered Current In)..... | 149 |
| Configuring Standard Features (Triggered Current In)..... | 153 |

Configuring the Basic Functionality (Triggered Current In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the trigger conditions (edge type, trigger threshold, delay time).
- Specifying a minimum frequency (PWM timeout).
- Using filters for the input signal.
- Specifying the generation of I/O events.

Selecting the trigger edge type

The input voltage signal triggers the analog current measurement. You can select the trigger on the rising edge or on the falling edge of the signal. Furthermore, you can trigger on both edges:

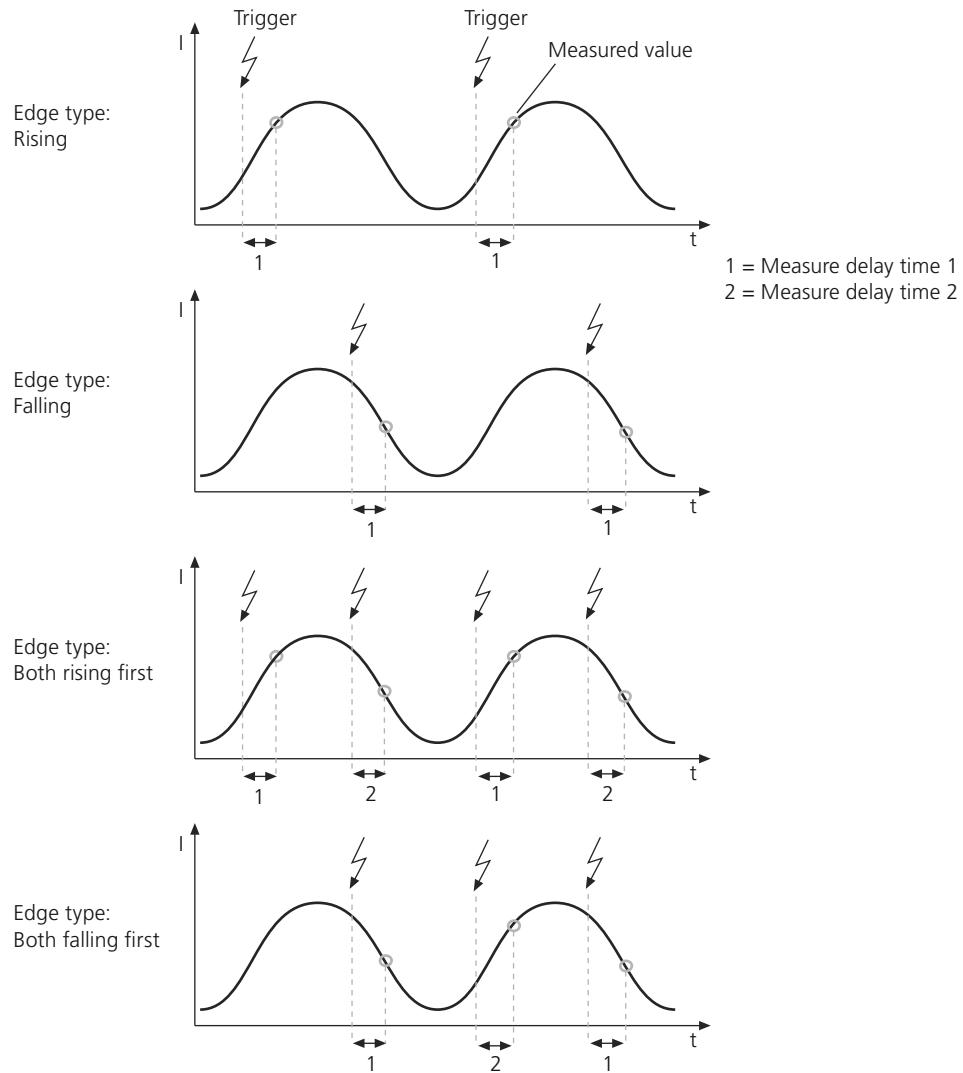
- Both rising first means that the **Current 1** function port and the related **Timestamp 1** function port represent the measurement result of the trigger on the *rising* edge. The function ports **Current 2** and **Timestamp 2** represent the measurement result of the trigger on the *falling* edge.
- The meaning of Both falling first is vice versa according to the description above. The function ports **Current 1** and **Timestamp 1** represent the measurement result of the trigger on the *falling* edge. The function ports **Current 2** and **Timestamp 2** represent the measurement result of the trigger on the *rising* edge.

Ignored triggers If a trigger edge occurs during an A/D conversion and its related measure delay time, the trigger is ignored. If the Edge type property is set to Both rising first or Both falling first, the subsequent trigger edge is also ignored. The A/D conversion time depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Triggered Current In\)](#) on page 155.

Specifying measure delay times

You can specify a delay time for the measurement to avoid, for example, switching noise. The two different measure delay times are related to the trigger

of the rising or falling edge of the input voltage signal depending on the setting of the **Edge type** property. Refer to the following illustration:



Note

When you specify the measure delay times, you must consider the setting of the **Minimum PWM frequency** property. The measure delay times must be at least 2.3 μ s shorter than the PWM timeout ($1/\text{Minimum PWM frequency}$) for a proper PWM timeout detection. The following rule applies:

$$\text{Measure delay time} < \frac{1}{\text{Minimum PWM frequency}} - 2.3 \mu\text{s}$$

Specifying a minimum frequency (PWM timeout)

You can specify a minimum frequency at the Minimum PWM frequency property to detect the absence of the input signal (PWM timeout). Frequencies lower than specified are evaluated as 0 Hz and initiate more current measurements independently from a PWM voltage signal.

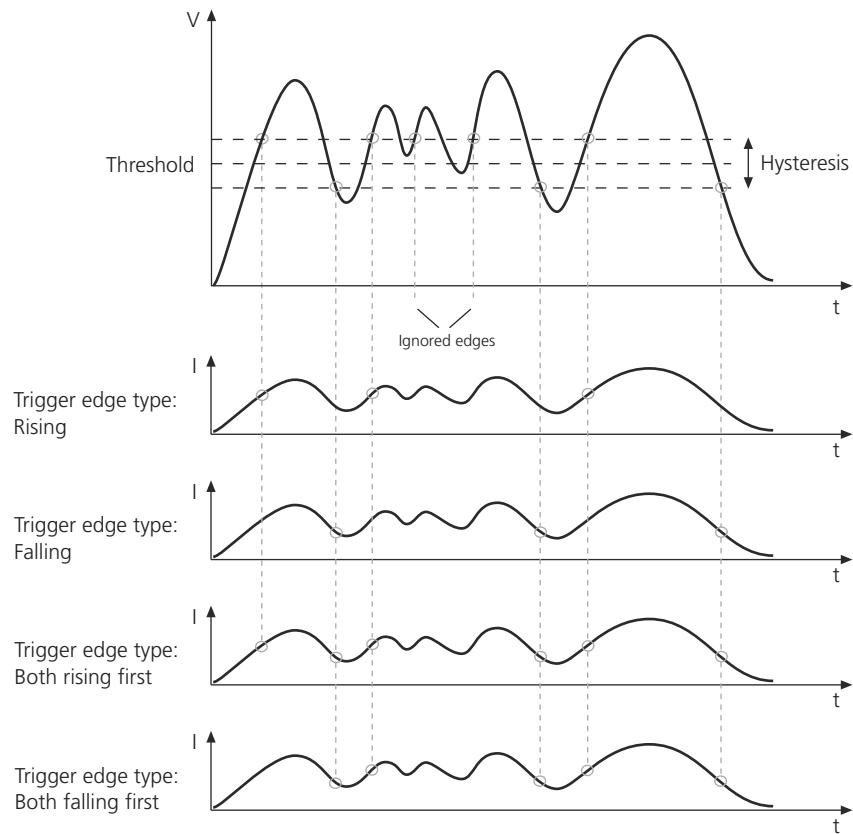
Note

When you specify the minimum PWM frequency, you must consider the settings of the Measure delay time properties. The PWM timeout (1/Minimum PWM frequency) must be at least 2.3 μ s longer than the measure delay times for a proper PWM timeout detection. The following rule applies:

$$\text{Minimum PWM frequency} < \frac{1}{\text{Measure delay time} + 2.3 \mu\text{s}}$$

Specifying the trigger threshold

The adjustable trigger threshold lets you configure the level of the triggering voltage input signal, to avoid, for example, the measurement being triggered by noise. A rising edge of the voltage input signal must exceed the threshold plus half the hysteresis value to trigger the measurement. A falling edge of the input signal must fall below the threshold minus half the hysteresis value. Refer to the following illustration, which also shows the effects of the specified trigger edge type:



Edges of the voltage input signal can be ignored for the following reasons:

- The signal does not reach the threshold plus half the hysteresis
- The signal does not reach the threshold minus half the hysteresis

The value range of the threshold and the hysteresis depend on the assigned hardware. For specific values, refer to [Hardware Dependencies \(Triggered Current In\)](#) on page 155.

Using a pulse filter

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the minimum pulse duration depends on the assigned hardware. For specific values, refer to [Hardware Dependencies \(Triggered Current In\)](#) on page 155.

Using the analog low-pass filter

You can use the analog low-pass filter, for example, to reduce signal noise. The filter characteristics depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Triggered Current In\)](#) on page 155.

For basic information on using the filter, refer to [Analog and Digital Filtering with the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration](#) ().

Limitation The analog low-pass filter is supported only for the Flexible In 1 channel type.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Triggered Current In function block can generate an I/O event each time a measurement of the pump current and the probe resistance is completed.

To use the I/O events in the behavior model, you have to assign them to a task via the End of Conversion property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide](#) ()

Configuring Standard Features (Triggered Current In)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Flexible In 1	Flexible In 2	Further Information
Channel multiplication	✓	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports			
Trigger level of electronic fuse	✓	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Flexible In 1](#) on page 1593
- [Flexible In 2](#) on page 1595

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Triggered Current In)

SCALEXIO Hardware Dependencies (Triggered Current In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Flexible In 1	Flexible In 2
Hardware		DS2601 Signal Measurement Board	DS2680 I/O Unit
Event generation		✓	✓
Input current range		<ul style="list-style-type: none"> ▪ 0 ... 10 A_{RMS} for 1 channel ▪ 80 A_{RMS} for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... 6 A_{RMS} for 1 channel ▪ 86.4 A_{RMS}¹⁾ for 18 channels (channel multiplication)
Input voltage range		-60 V ... +60 V	0 ... +60 V
Peak current per channel		±30 A	±18 A
Sample rate		0 ... 250 kS/s	85.7 kS/s (fixed, multiplexed)
Resolution		16 bit	16 bit
Offset error		±5 mA (typ.)	±25 mA (typ.)
Gain error		±1.0% (typ.)	±0.1% (typ.)
Analog low-pass filter	Filter type	2 nd -order Bessel filter	—
	Edge frequency (-3dB cutoff frequency)	20 kHz	—
Measure delay time		918 ns ... 292.058 s	14 µs ... 292.058 s
Minimum PWM frequency		0.00342498 Hz ... 240 kHz	0.00342498 Hz ... 80 kHz
Trigger threshold	Input threshold voltage	-60 V ... +60 V	0 ... +24 V
	Input hysteresis voltage	250 mV (typ.)	200 mV (typ.)
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs	0.264 µs ... 131 µs
Measurement point		Load side	Load side
Configurable fuse		<ul style="list-style-type: none"> ▪ On load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	—
Circuit diagram		Flexible In 1 on page 1593	Flexible In 2 on page 1595
Required channels		1 (additional channels are required for current enhancements)	1 (additional channels are required for current enhancements)

¹⁾ This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

General limitations

- Channel multiplication across several I/O boards is not supported.
 - The measurement of current signals on the ECU side is not supported.
-

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Voltage In

Where to go from here

Information in this section

Introduction (Voltage In).....	158
Overviews (Voltage In).....	159
Configuring the Function Block (Voltage In).....	164
Hardware Dependencies (Voltage In)	170

Introduction (Voltage In)

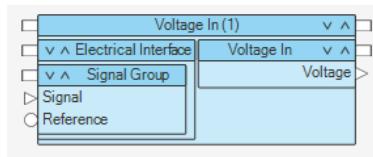
Introduction to the Function Block (Voltage In)

Function block purpose

The Voltage In function block type digitizes analog voltage signals coming from an external device.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Measuring analog voltage signals and providing the digitized values to the behavior model.
- Capturing timestamps and angle positions at each measurement.
- Using analog and/or digital filters for the input signal.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Voltage In function block supports the following channel types:

SCALEXIO								
Channel type	Analog In 1	Analog In 2	Analog In 4	Analog In 5	Flexible In 3	Flexible In 1	Analog In 6	Analog In 16
Hardware	DS2680		DS6101			DS2601	DS6221	DS6121

MicroAutoBox III								
Channel type	Analog In 7	Analog In 8	Analog In 9	Analog In 10	Analog In 11	Analog In 12	Analog In 14	Analog In 17
Hardware	DS1511	<ul style="list-style-type: none"> ▪ DS1511B1 ▪ DS1513 	DS1513	DS1552	DS1552B1	<ul style="list-style-type: none"> ▪ DS1552 ▪ DS1552B1 	DS1554	DS1521

Overviews (Voltage In)

Where to go from here

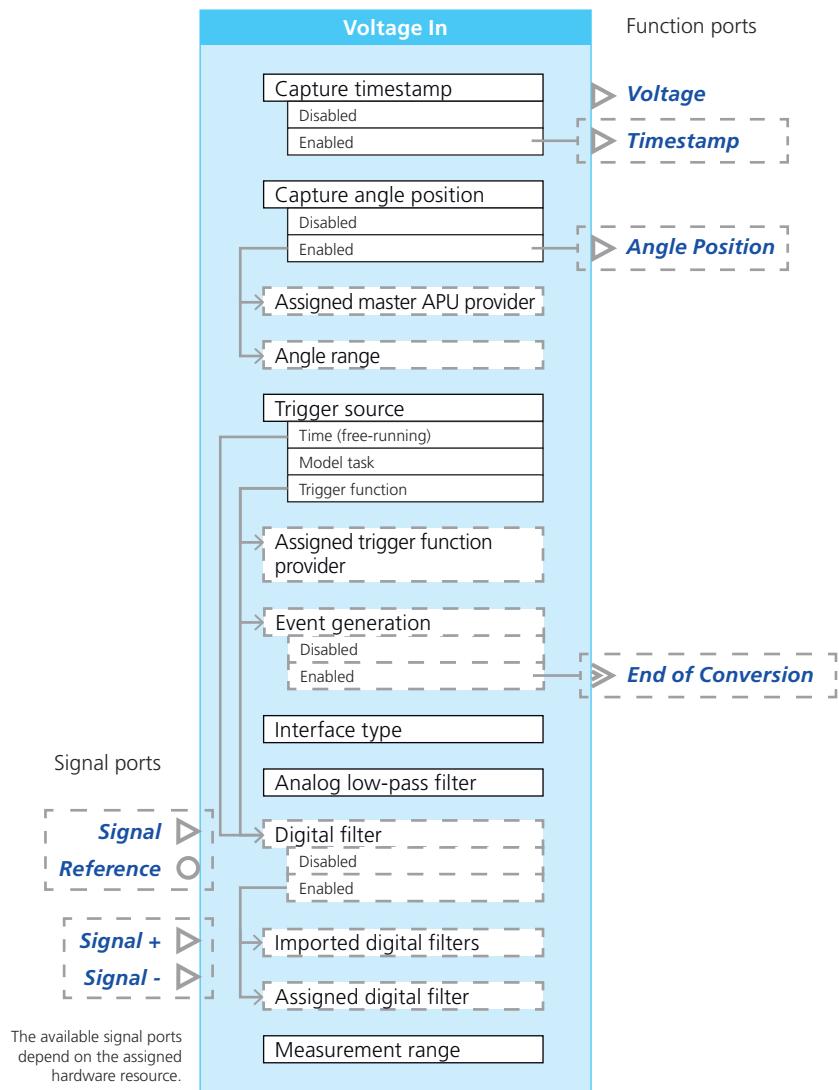
Information in this section

Overview of Ports and Basic Properties (Voltage In).....	159
Overview of Tunable Properties (Voltage In).....	163

Overview of Ports and Basic Properties (Voltage In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Voltage**

This function outport writes the measured voltage value to the behavior model.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt) Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage In) on page 170.
Dependencies	-

Timestamp

This function outport writes the time the voltage value is measured to the behavior model.

Value range	0 ... 612,489,500 s
Dependencies	<ul style="list-style-type: none"> SCALEXIO hardware:

- | | |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none"> ▪ Available only if the Capture timestamp property is set to Enabled. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported. |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Angle Position

This function outport writes the angle position to the behavior model each time the voltage value is measured.

Value range	<ul style="list-style-type: none"> ▪ 0° ... 360° or 0° ... 720° ▪ The range depends on the Angle range property setting of the function block.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Available only if the Capture angle position property is set to Enabled. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported.

End of Conversion

This event port provides an I/O event each time the voltage measurement is completed (end of A/D conversion).

Value range	–
Dependencies	<ul style="list-style-type: none"> Available only if both of the following settings match: ▪ The Trigger source property is set to Trigger function. ▪ The Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage In) on page 170.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Not supported for the Analog In 6 channel type. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported for the Analog In 14 channel type.

Signal + / Signal -

These signal ports represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (Signal + and Signal -) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Voltage In

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage In) on page 170.
Dependencies	<ul style="list-style-type: none">▪ SCALEXIO hardware:<ul style="list-style-type: none">▪ Supported only for the Analog In 6 channel type.▪ MicroAutoBox III hardware:<ul style="list-style-type: none">▪ Supported only for the Analog In 14 channel type.

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage In) on page 170.
Dependencies	<ul style="list-style-type: none">▪ SCALEXIO hardware:<ul style="list-style-type: none">▪ Not supported for the Analog In 6 channel type.▪ MicroAutoBox III hardware:<ul style="list-style-type: none">▪ Not supported for the Analog In 14 channel type.

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none">▪ SCALEXIO hardware:<ul style="list-style-type: none">▪ Supported only for Analog In 1 and Flexible In 1 channel types.▪ Available only if the Connected load property is set to Use external load.▪ MicroAutoBox III hardware:<ul style="list-style-type: none">▪ Not supported.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none">▪ SCALEXIO hardware:<ul style="list-style-type: none">▪ Supported only for the Flexible In 1 channel type.▪ Available only if the Connected load property is set to Use external load.▪ MicroAutoBox III hardware:<ul style="list-style-type: none">▪ Not supported.

Overview of Tunable Properties (Voltage In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Measurement point	✓	–
Trigger source	✓	–
Assigned digital filter	✓	–
Analog low-pass filter	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Voltage In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Voltage In).....	164
Configuring Standard Features (Voltage In).....	167

Configuring the Basic Functionality (Voltage In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the trigger source.
- Enabling or disabling the capturing of timestamps and angle position data to save computation time.
- Using digital and/or analog filters for the input signal.
- Selecting the voltage measurement range.
- Specifying the generation of I/O events.

Specifying the trigger source

You can specify the trigger source that starts the measurement:

- If you select the **Time** (free-running) trigger source, the measurement runs independently from a trigger. It runs repeatedly at a fixed sample frequency, which depends on the assigned hardware.
The value of the latest measured signal is provided to the behavior model.
- If you select the **Model task** trigger source, the measurement is triggered synchronously to the model task in which the mapped model port block is executed. The function block starts to measure if an event triggers the model task to execute the next model step, for example, a timer event. The trigger is ignored if a measurement is still running.
- If you use a **Trigger function** as trigger source, the measurement starts with an external trigger that is provided by a specific trigger function. This can be, for example, the **Trigger In** function block. Trigger function providers have their own defined trigger conditions.

Using the **Trigger function** trigger source, you can enable the generation of an I/O event each time the measurement is completed (end of A/D conversion). These I/O events (interrupts) can be used as trigger source for runnable functions in the behavior model.

To use the I/O events in the behavior model, you have to assign them to a task via the **End of Conversion** property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

You can change the trigger source via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

Limitations

- The Model task trigger source is supported only by the following channel types:
 - SCALEXIO hardware: Flexible In 1, Analog In 6, Analog In 16
 - MicroAutoBox III hardware: Analog In 7, Analog In 8, Analog In 10, Analog In 11
- The Trigger function trigger source is not supported by the following channel types:
 - MicroAutoBox III hardware: Analog In 9, Analog In 12, Analog In 14, Analog In 17

Enabling or disabling the capturing of timestamps and angle positions

Depending on your requests, you can capture timestamps and angle positions and write them to the behavior model via specific function ports. Timestamp and angle position values are captured synchronously to the measured voltage values. To save computation time, you are recommended to disable capturing for data that is not required.

As a precondition for capturing angle positions you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Limitation

- MicroAutoBox III hardware: Capturing timestamps and angle positions is not supported. If you enable one of these features and assign a MicroAutoBox III hardware resource to the function block, a conflict is generated and displayed in the Conflicts Viewer.

Using digital filters

The function block supports the usage of digital filters, for example, to smooth the input signal or to reduce signal noise, or to adapt the signal bandwidth to a specific model sample frequency in order to avoid aliasing problems. However, the digital filter properties are only supported for the Time (free-running) and the Trigger function trigger source. This is because the calculation of the filter coefficients is based on a specific sample frequency.

The filter coefficients are stored in digital filter files which you can import and use them afterwards. The supported filters and filter types depend on the assigned hardware resources. For example, the DS2601 Signal Measurement Board supports various digital filter types, such as, Butterworth filter.

You can change the assigned digital filter file via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

Limitations

- SCALEXIO hardware: Importing and using digital filters is only supported for the Flexible In 1 channel type.
- MicroAutoBox III hardware: Importing and using digital filters is not supported.

Using the analog low-pass filter

You can use the analog low-pass filter, for example, to reduce signal noise. The filter characteristics depend on the assigned hardware resource. For details, refer to [Hardware Dependencies \(Voltage In\)](#) on page 170.

You can change the usage of the analog low-pass filter via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

For further basic information on using the filter, refer to [Analog and Digital Filtering with the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Limitations

- SCALEXIO hardware: The analog low-pass filter is only supported for the Flexible In 1 channel type.
 - MicroAutoBox III hardware: An analog low-pass filter is not supported.
-

Selecting the measurement range

You have to select the measurement range for the voltage input signals, so that the range covers the expected input voltage level.

The voltage values for the low and high ranges depend on the hardware assigned to the function block. For specific values, refer to [Hardware Dependencies \(Voltage In\)](#) on page 170.

Limitations

- SCALEXIO hardware: Selecting the measurement range is only supported by the Analog In 2, Analog In 5, and Analog In 16 channel types.
 - MicroAutoBox III hardware: Selecting the measurement range is not supported.
-

Providing I/O events

If the Trigger source property is set to Trigger function and the Event generation property is set to Enabled, the function block generates I/O events to trigger runnable functions in the behavior model.

The Voltage In function block can generate an I/O event at the end of each voltage measurement.

To use the I/O events in the behavior model, you have to assign them to a task via the End of Conversion property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Limitation Event generation is not supported for the following channel types:

- MicroAutoBox III hardware: Analog In 9, Analog In 12, Analog In 14, Analog In 17.

Configuring Standard Features (Voltage In)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO					
Configuration Feature	Analog In 1	Analog In 2	Analog In 4	Analog In 5	Further Information
Measurement point	Load side	Load side	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Differential with ground sense	Specifying the Circuit Type for Voltage Input Signals on page 116			
Channel multiplication	✓	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Measurement range	–	✓	–	✓	–

Signal Ports					
Trigger level of electronic fuse	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	–	–	–	Specifying Load Settings on page 121

SCALEXIO					
Configuration Feature	Analog In 6	Analog In 16	Flexible In 1	Flexible In 3	Further Information
Measurement point	–	–	▪ Load side ▪ ECU side	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	▪ Differential with ground sense ▪ Differential	Differential with ground sense	▪ Differential with ground sense ▪ Galvanically isolated	Differential with ground sense	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	✓	–	Specifying Current and Voltage Values for Channel Multiplication on page 95

SCALEXIO									
Configuration Feature	Analog In 6	Analog In 16	Flexible In 1	Flexible In 3	Further Information				
Measurement range	–	✓	–	–	–				
Signal Ports									
Trigger level of electronic fuse	–	–	✓	–	Specifying Fuse Settings on page 122				
Failure simulation support	–	–	✓	–	Specifying Settings for Failure Simulation on page 123				
Usage of loads	–	–	✓	–	Specifying Load Settings on page 121				
MicroAutoBox III									
Configuration Feature	Analog In 7	Analog In 8	Analog In 9	Analog In 10	Analog In 11	Analog In 12	Analog In 14	Analog In 17	Further Information
Measurement point	–	–	–	–	–	–	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Ground-based	Ground-based	Differential with ground sense	Differential with ground sense	Ground-based	Differential	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	–	–	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Measurement range	–	–	–	–	–	–	–	–	–
Signal Ports									
Trigger level of electronic fuse	–	–	–	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	–	–	–	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	–	–	–	–	–	–	–	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MABX III
<ul style="list-style-type: none"> ▪ Analog In 1 on page 1544 ▪ Analog In 2 on page 1546 ▪ Analog In 4 on page 1546 ▪ Analog In 5 on page 1547 ▪ Analog In 6 on page 1547 ▪ Analog In 16 on page 1552 ▪ Flexible In 1 on page 1593 ▪ Flexible In 3 on page 1603 	<ul style="list-style-type: none"> ▪ Analog In 7 on page 1548 ▪ Analog In 8 on page 1548 ▪ Analog In 9 on page 1549 ▪ Analog In 10 on page 1549 ▪ Analog In 11 on page 1550 ▪ Analog In 12 on page 1550 ▪ Analog In 14 on page 1551 ▪ Analog In 17 on page 1552

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Voltage In)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Voltage In).....	170
MicroAutoBox III Hardware Dependencies (Voltage In).....	173

SCALEXIO Hardware Dependencies (Voltage In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog In 1	Analog In 2	Analog In 4	Analog In 5	Flexible In 3
Hardware	DS2680 I/O Unit		DS6101 Multi-I/O Board		
Event generation	✓		✓		
Timestamp and angle position capturing	✓		✓		
Input current range	<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 96 A_{RMS}¹⁾ for 20 channels (channel multiplication) 	0 ... +0.3 A	-	0 ... +0.3 A	-
Input voltage range	0 ... +60 V	-60 V ... +60 V	0 ... +60 V	-60 V ... +60 V	
Measurement range	0 ... +60 V	<ul style="list-style-type: none"> ▪ Low: -2 V ... +2 V ▪ High: -10 V ... +10 V 	0 ... +60 V	<ul style="list-style-type: none"> ▪ Low: -2 V ... +2 V ▪ High: -10 V ... +10 V 	-10 V ... +10 V
Resolution	16 bit	12 bit	16 bit	12 bit	16 bit
Sample rate	85.7 kS/s (fixed, multiplexed)	1.8 MS/s	85.7 kS/s (fixed, multiplexed)	1.8 MS/s	85.7 kS/s (fixed, multiplexed)
Offset error	±20 mV (typ.)	Depending on the selected measurement range: <ul style="list-style-type: none"> ▪ Low: ±2 mV (typ.) ▪ High: ±10 mV (typ.) 	±5 mV (typ.)	Depending on the selected measurement range: <ul style="list-style-type: none"> ▪ Low: ±2 mV (typ.) ▪ High: ±10 mV (typ.) 	±2 mV (typ.)
Gain error	±0.5 % (typ.)	±0.1 % (typ.)	±0.1 % (typ.)	±0.2 % (typ.)	±0.1 % (typ.)

Channel Type		Analog In 1	Analog In 2	Analog In 4	Analog In 5	Flexible In 3		
Analog low-pass filter	Filter type	–	–		–			
	Edge frequency (-3dB cutoff frequency)	–	–		150 kHz			
Using digital filter files		–	–		–			
Trigger source		<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 41.6 kHz ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 1.8 MHz ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 41.6 kHz ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 1.8 MHz ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 41.6 kHz ▪ Trigger function 		
Angular processing unit	Number of slaves	6	6		6			
	Maximum speed	168,000 °/s	168,000 °/s		168,000 °/s			
Supported interface type	Differential with ground sense		Differential with ground sense					
Measurement point	Load side		–					
Configurable fuse	–		–					
Circuit diagrams	Analog In 1 on page 1544	Analog In 2 on page 1546	Analog In 4 on page 1546	Analog In 5 on page 1547	Flexible In 3 on page 1603			
Required channels	1 (additional channels are required for current enhancements.)	1	1	1				

¹⁾ This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

Channel Type	Flexible In 1	Analog In 6	Analog In 16
Hardware	DS2601 Signal Measurement Board	DS6221 A/D Board	DS6121 Multi-I/O Board
Event generation	✓	✓	✓
Timestamp and angle position capturing	✓	✓	✓
Input current range	<ul style="list-style-type: none"> ▪ 0 ... +10 A_{RMS} for 1 channel ▪ 80 A_{RMS} for 10 channels (channel multiplication) 	–	–
Input voltage range	-60 V ... +60 V	-60 V ... +60 V	-60 V ... +60 V
Measurement range	-60 V ... +60 V	-10 V ... +10 V	<ul style="list-style-type: none"> ▪ Low: -10 V ... +10 V ▪ High: -60 V ... +60 V
Resolution	16 bit	16 bit	16 bit
Sample rate	0 ... 250 kS/s	0 ... 5 MS/s	0 ... 2 MS/s
Offset error	±10 mV (typ.)	±1 mV	Depending on the selected measurement range: <ul style="list-style-type: none"> ▪ Low: ±2 mV (typ.)

Channel Type		Flexible In 1	Analog In 6	Analog In 16
Gain error		±0.1 % (typ.)	±0.1 %	▪ High: ±5 mV (typ.) ±0.1 %
Analog low-pass filter	Filter type	2 nd -order Bessel filter	2 nd -order (always active)	–
	Edge frequency (-3dB cutoff frequency)	20 kHz	1.9 MHz	3.5 MHz
Trigger source		<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 250 kHz ▪ Model task ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 2.1 MHz ▪ Model task ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 2 MHz ▪ Model task ▪ Trigger function
Using digital filter files		✓ ¹⁾	–	–
Angular processing unit	Number of slaves	1	6	6
	Maximum speed	168,000 °/s	168,000 °/s	1,200,000 °/s
Supported interface type		<ul style="list-style-type: none"> ▪ Differential with ground sense ▪ Galvanically isolated 	<ul style="list-style-type: none"> ▪ Differential with ground sense ▪ Differential 	Differential with ground sense
Measurement point		<ul style="list-style-type: none"> ▪ Load side ▪ ECU side 	–	–
Configurable fuse		<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	–	–
Circuit diagrams		Flexible In 1 on page 1593	Analog In 6 on page 1547	Analog In 16 on page 1552
Required channels		1 (additional channels are required for current enhancements.)	1	1

¹⁾ You only can use digital filter files if the Trigger source is set to Time (free-running).

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6121 Multi-I/O Board Channel multiplication is not supported in general.

DS6221 A/D Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6221 A/D Board For more board-specific data, refer to [Data Sheet of the DS6221 A/D Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (Voltage In)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog In 7	Analog In 8	Analog In 9	Analog In 10
Hardware	DS1511 Multi-I/O Board	<ul style="list-style-type: none"> ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	DS1513 Multi-I/O Board	DS1552 Multi-I/O Module
Event generation	✓	✓	–	✓
Timestamp and angle position capturing	–	–	–	–
Input voltage range	0 ... +5 V	-10 V ... +10 V	-10 V ... +10 V	0 ... +5 V
Resolution	16 bit	16 bit	16 bit	16 bit
Sample rate	0 ... 1 MS/s	0 ... 1 MS/s	0 ... 200 kS/s	0 ... 1 MS/s
Offset error	±0.5 mV (typ.)	±3 mV (typ.)	±2 mV (typ.)	±0.5 mV (typ.)
Gain error	±0.25 % (typ.)	±0.25 % (typ.)	±1 % (typ.)	±0.25 % (typ.)
Edge frequency (-3dB cutoff frequency)	400 kHz	400 kHz	23 kHz	400 kHz
Trigger source	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 500 kHz ▪ Model task ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 500 kHz ▪ Model task ▪ Trigger function 	Time (free-running) with a sample frequency of 200 kHz	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 500 kHz ▪ Model task ▪ Trigger function
Supported interface type	Ground-based	Ground-based	Ground-based	Differential with ground sense
Circuit diagrams	Analog In 7 on page 1548	Analog In 8 on page 1548	Analog In 9 on page 1549	Analog In 10 on page 1549
Required channels	1	1	1	1

Channel Type	Analog In 11	Analog In 12	Analog In 14	Analog In 17
Hardware	DS1552B1 Multi-I/O Module	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module 	DS1554 Engine Control I/O Module	DS1521 Bus Board

Channel Type	Analog In 11	Analog In 12	Analog In 14	Analog In 17
Event generation	✓	–	–	–
Input voltage range	-10 V ... +10 V	-10 V ... +10 V	-10 V ... +10 V	-10 V ... +10 V
Timestamp and angle position capturing	–	–	–	–
ADC	Resolution	16 bit	16 bit	16 bit
	Sample time	1 µs	5 µs	3 µs
Offset error	±3 mV (typ.)	±2 mV (typ.)	±3 mV (typ.)	±2 mV (typ.)
Gain error	±0.25 % (typ.)	±1 % (typ.)	±0.25 % (typ.)	±1 % (typ.)
Edge frequency (-3dB cutoff frequency)	400 kHz	23 kHz	400 kHz	24 kHz
Trigger source	<ul style="list-style-type: none"> ▪ Time (free-running) with a sample frequency of 500 kHz ▪ Model task ▪ Trigger function 	Time (free-running) with a sample frequency of 200 kHz	Time (free-running) with a sample frequency of 1 MHz	Time (free-running) with a sample frequency of 333.333 kHz
Supported interface types	Differential with ground sense	Ground-based	Differential	Ground-based
Circuit diagrams	Analog In 11 on page 1550	Analog In 12 on page 1550	Analog In 14 on page 1551	Analog In 17 on page 1552
Required channels	1	1	1	1

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to [DS1552 Multi-I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1554 Engine Control I/O Module For more module-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more module-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Current Sink

Where to go from here

Information in this section

Introduction (Current Sink).....	176
Overviews (Current Sink).....	177
Configuring the Function Block (Current Sink).....	180
Hardware Dependencies (Current Sink).....	184

Introduction (Current Sink)

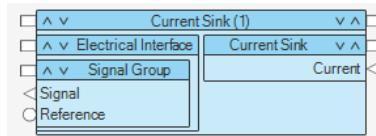
Introduction to the Function Block (Current Sink)

Function block purpose

The Current Sink function block type lets you simulate sensors with a current interface like most Hall sensors.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Sinking a current with a value provided from the behavior model.
- Adding noise to the output signal via a noise generator.

Supported channel types

The Current Sink function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Analog Out 4	Analog Out 9	Flexible Out 1	–
Hardware	DS2680	DS6101	DS2621	–

Overviews (Current Sink)

Where to go from here

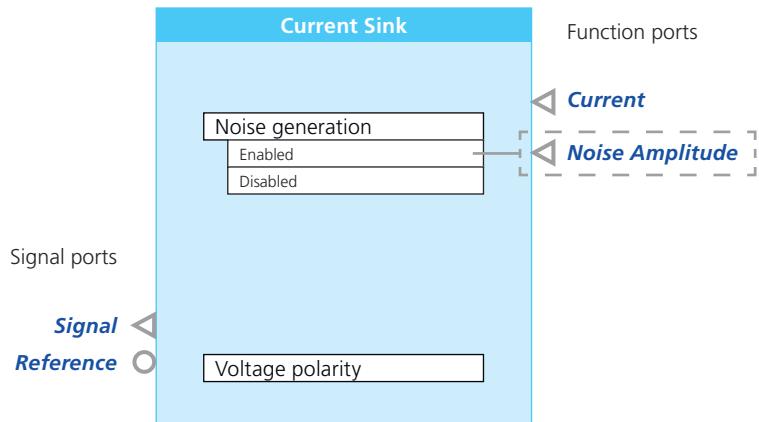
Information in this section

- | | |
|----------------------------------------------------------------------|-----|
| Overview of Ports and Properties (Current Sink)..... | 177 |
| Overview of Tunable Properties (Current Sink)..... | 179 |

Overview of Ports and Properties (Current Sink)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Current

This function import lets you define the output current from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $I_{min} \dots I_{max}$ (in Ampere). ▪ The range depends on the follows: <ul style="list-style-type: none"> ▪ It depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Sink) on page 184. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	–

Noise Amplitude This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none">▪ 0 ... I_{max} (in Ampere).▪ The range depends on the follows:<ul style="list-style-type: none">▪ It depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Sink) on page 184.▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.▪ 0 A: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none">▪ Available only if the Noise generation property is set to Enabled.▪ Supported only for Flexible Out 1 channel type.

Signal This signal port sinks current. It represents the electrical connection point of the logical signal of the function block.

Note

The hardware is passive and adjusts its internal resistance so that the defined current can flow at the signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Sink) on page 184.
Dependencies	–

Reference This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Sink) on page 184.
Dependencies	–

Overview of Tunable Properties (Current Sink)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–	–
Function Ports			
Initial switch setting (Test Automation)	–	✓	
Initial substitute value (Test Automation)	–	✓	
Electrical Interface			
Voltage Polarity	✓	–	

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Current Sink)

Where to go from here

Information in this section

Configuring the Basic Functionality (Current Sink).....	180
Configuring Standard Features (Current Sink).....	182

Configuring the Basic Functionality (Current Sink)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Adding noise to the output signal
- Specifying the signal polarity

Adding noise to the output signal

The Current Sink function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

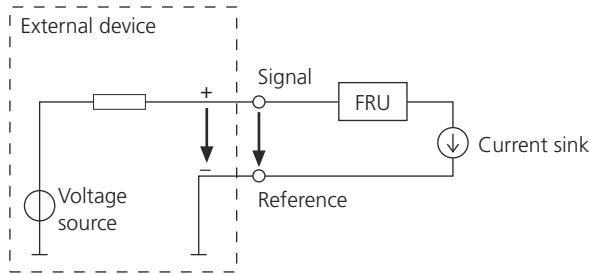
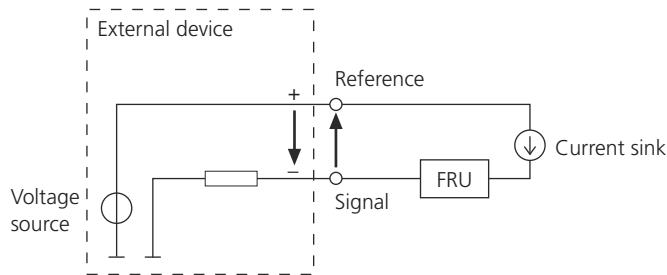
Limitation Noise generation is supported only for the Flexible Out 1 channel type.

Specifying the signal polarity

The Current Sink function block must be fed by an external voltage source. Depending on the selected channel type, the voltage polarity affects the functionality of the function block.

Flexible Out 1 channel type You must specify the voltage polarity of the signal ports with the Voltage polarity property to ensure the correct behavior of the function block.

Note that changing the Voltage polarity property also changes the signal path that the failure routing unit (FRU) is used on. This means that you can use the FRU on either the one or the other signal path by switching the polarity of the voltage. The following illustration shows a connection example with an FRU on the signal path.

Positive polarity**Negative polarity**

For details on the FRU, refer to [Electrical Error Simulation Concept \(SCALEXIO Hardware Installation and Configuration\)](#).

Note

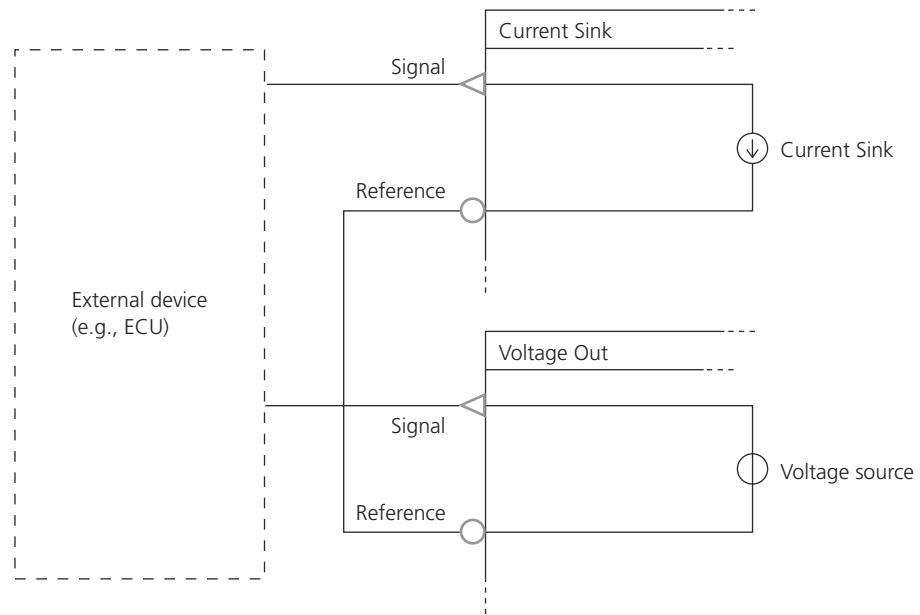
The Current Sink function block adjusts only the current. The feed voltage must come from an external source as shown in the illustrations above.

The Voltage polarity property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

Analog Out 4, Analog Out 9 channel types The voltage polarity does not affect the functionality of the function block. The behavior is correct if the voltage polarity of the signal is negative as well as positive. Therefore the Voltage polarity property is not configurable for these channel types.

Application example: Current Sink function block as current source

You can use the Current Sink function block as a current source if you connect it, for example, to a Voltage Out function block as shown in the illustration below. This example is valid only if you use the Flexible Out 1 channel type for the Voltage Out function block.



Configuring Standard Features (Current Sink)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the **Conflicts Viewer**. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Analog Out 4	Analog Out 9	Flexible Out 1	Further Information
Channel multiplication	✓	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 4](#) on page 1555
- [Analog Out 9](#) on page 1557
- [Flexible Out 1](#) on page 1604

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Current Sink)

SCALEXIO Hardware Dependencies (Current Sink)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 4	Analog Out 9	Flexible Out 1
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board	DS2621 Signal Generation Board
Output current range	<ul style="list-style-type: none"> ▪ +0.1 mA ... +30 mA for 1 channel ▪ +0.24 A for 8 channels (channel multiplication) 	+0.1 mA ... +30 mA	<ul style="list-style-type: none"> ▪ +20 µA ... +40 mA for 1 channel ▪ 0.4 A for 10 channels (channel multiplication)
Output voltage range	-2 V ... +18 V	<p>The following conditions must be met:</p> <ul style="list-style-type: none"> ▪ Range for the lower potential of signal and reference: -2 V ... +18 V ▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V 	-60 V ... +60 V
Update rate	382.3 kS/s (fixed, multiplexed)	382.3 kS/s (fixed, multiplexed)	0 ... 1 MS/s
Resolution	14 bit	14 bit	15 bit
Offset error	±20 µA (typ.)	±30 µA (typ.)	±10 µA (typ.)
Gain error	±0.2 % (typ.)	±0.2 % (typ.)	±0.1 % (typ.)
Noise generation	–	–	✓
Configurable fuse	–	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Circuit diagrams	Analog Out 4 on page 1555	Analog Out 9 on page 1557	Flexible Out 1 on page 1604
Required channels	1 (additional channels are required for current enhancements)	1	1 (additional channels are required for current enhancements)

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101](#) on page 1539.

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Voltage Out

Where to go from here

Information in this section

Introduction (Voltage Out).....	188
Overviews (Voltage Out).....	189
Configuring the Function Block (Voltage Out).....	192
Hardware Dependencies (Voltage Out).....	195

Introduction (Voltage Out)

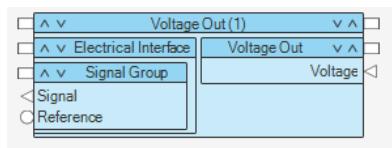
Introduction to the Function Block (Voltage Out)

Function block purpose

The Voltage Out function block provides the possibility to output analog voltages.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Output of an analog voltage with a value provided from the behavior model.
- Adding noise to the output signal via a noise generator.

Supported channel types

The Voltage Out function block type supports the following channel types:

	SCALEXIO								MicroAutoBox III		
Channel type	Analog Out 1	Analog Out 2	Analog Out 4	Analog Out 6	Analog Out 7	Analog Out 9	Analog Out 10	Flexible Out 1	Analog Out 11	Analog Out 12	Analog Out 13
Hardware	DS2680			DS6101			DS6241	DS2621	▪ DS1511 ▪ DS1511B1	DS1513	▪ DS1552 ▪ DS1552B1

Overviews (Voltage Out)

Where to go from here

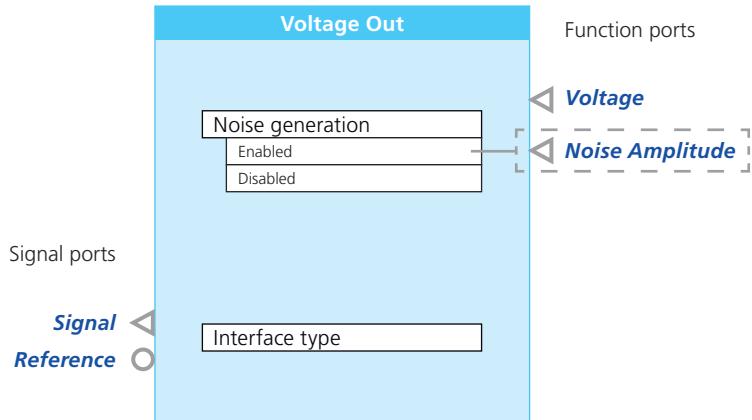
Information in this section

- | | |
|---------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Voltage Out)..... | 189 |
| Overview of Tunable Properties (Voltage Out)..... | 191 |

Overview of Ports and Basic Properties (Voltage Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Voltage

This function import lets you define the output voltage from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $U_{\min} \dots U_{\max}$ (in Volt) ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ It depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Out) on page 195. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	–

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none">▪ 0 V ... U_{\max} (in Volt)▪ The range depends on the following:<ul style="list-style-type: none">▪ It depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Out) on page 195.▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.▪ 0 V: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none">▪ SCALEXIO hardware:<ul style="list-style-type: none">▪ Supported only for the Flexible Out 1 and Analog Out 10 channel types.▪ Available only if the Noise generation property is set to Enabled.▪ MicroAutoBox III hardware:<ul style="list-style-type: none">▪ Not supported.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Out) on page 195.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Out) on page 195.
Dependencies	–

Overview of Tunable Properties (Voltage Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Voltage Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (Voltage Out).....	192
Configuring Standard Features (Voltage Out).....	192

Configuring the Basic Functionality (Voltage Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Adding noise to the output signal

Adding noise to the output signal

The Voltage Out function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Limitations

- SCALEXIO hardware:
 - Noise generation is supported only for the Flexible Out 1 and Analog Out 10 channel types.
- MicroAutoBox III hardware:
 - Noise generation is not supported.

Configuring Standard Features (Voltage Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer.

You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO							
Configuration Feature	Analog Out 1	Analog Out 2	Analog Out 4	Analog Out 6	Analog Out 7	Analog Out 9	Further Information
Interface type	Differential with ground sense	<ul style="list-style-type: none"> ▪ Differential with ground sense ▪ Galvanically isolated 	Differential with ground sense	Differential with ground sense	Galvanically isolated	Differential with ground sense	Specifying the Circuit Type for Analog Output Signals on page 118
Reference signal	–	<ul style="list-style-type: none"> ▪ System ground ▪ Galvanically isolated 	–	–	Galvanically isolated	–	–
Channel multiplication	–	–	–	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports							
Trigger level of electronic fuse	–	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	✓	–	–	–	Specifying Settings for Failure Simulation on page 123

SCALEXIO				
Configuration Feature	Analog Out 10	Flexible Out 1	Further Information	
Interface type	Differential with ground sense	Galvanically isolated	Specifying the Circuit Type for Analog Output Signals on page 118	
Channel multiplication	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95	
Signal Ports				
Trigger level of electronic fuse	–	✓	Specifying Fuse Settings on page 122	
Failure simulation support	–	✓	Specifying Settings for Failure Simulation on page 123	

MicroAutoBox III				
Configuration Feature	Analog Out 11	Analog Out 12	Analog Out 13	Further Information
Interface type	Ground-based	Ground-based	Ground-based	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95

MicroAutoBox III				
Configuration Feature	Analog Out 11	Analog Out 12	Analog Out 13	Further Information
Signal Ports				
Trigger level of electronic fuse	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Analog Out 1 on page 1553 ▪ Analog Out 2 on page 1554 ▪ Analog Out 4 on page 1555 ▪ Analog Out 6 on page 1556 ▪ Analog Out 7 on page 1556 ▪ Analog Out 9 on page 1557 ▪ Analog Out 10 on page 1558 ▪ Flexible Out 1 on page 1604 	<ul style="list-style-type: none"> ▪ Analog Out 11 on page 1559 ▪ Analog Out 12 on page 1559 ▪ Analog Out 13 on page 1560

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Voltage Out)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Voltage Out).....	195
MicroAutoBox III Hardware Dependencies (Voltage Out).....	197

SCALEXIO Hardware Dependencies (Voltage Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 1	Analog Out 2	Analog Out 4	Flexible Out 1
Hardware	DS2680 I/O Unit			DS2621
Output current range	0 ... +5 mA _{RMS}			0 ... +40 mA _{RMS}
Output voltage range	0 ... +10 V	-10 V ... +10 V	0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication)
Update rate	382.3 kS/s (fixed, multiplexed)	0 ... 1.953 MS/s	382.3 kS/s (fixed, multiplexed)	0 ... 1 MS/s
Resolution	14 bit			16 bit
Offset error	±5 mV (typ.)	±3.5 mV (typ.)	±5 mV (typ.)	±2 mV (typ.)
Gain error	±0.1 % (typ.)	±0.5 % (typ.)	±0.1 % (typ.)	±0.1 % (typ.)
Noise generation	–			✓
Configurable fuse	–			<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Circuit diagrams	Analog Out 1 on page 1553	Analog Out 2 on page 1554	Analog Out 4 on page 1555	Flexible Out 1 on page 1604
Required channels	1			1 (additional channels are required for voltage enhancements.)

Channel Type	Analog Out 6	Analog Out 7	Analog Out 9	Analog Out 10
Hardware	DS6101 Multi-I/O Board			DS6241 D/A Board
Current range	0 ... +5 mA _{RMS}	-5 mA ... +5 mA	0 ... +5 mA _{RMS}	-5 mA ... +5 mA (min.)
Voltage range	0 ... +10 V	-10 V ... +10 V	0 ... +10 V	-10 V ... +10 V

Channel Type	Analog Out 6	Analog Out 7	Analog Out 9	Analog Out 10
Update rate	382.3 kS/s (fixed, multiplexed)	0 ... 1.953 MS/s	382.3 kS/s (fixed, multiplexed)	0 ... 500 kS/s
Resolution	14 bit			16 bit
Offset error	± 5 mV (typ.)	± 3.5 mV (typ.)	± 5 mV (typ.)	± 1 mV (typ.)
Gain error	± 0.1 % (typ.)	± 0.5 % (typ.)	± 0.1 % (typ.)	± 0.03 % (typ.)
Noise generation	–			✓
Configurable fuse	–			–
Circuit diagrams	Analog Out 6 on page 1556	Analog Out 7 on page 1556	Analog Out 9 on page 1557	Analog Out 10 on page 1558
Required channels	1			1

Synchronous signal update on DS2680 and DS6101 If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101 on page 1539](#).

General limitations

Channel multiplication across several I/O boards is not supported.

DS2680 I/O Unit The Voltage Out function block does not support channel multiplication on the DS2680.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6241 D/A Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6241 D/A Board For more board-specific data, refer to [Data Sheet of the DS6241 D/A Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (Voltage Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 11	Analog Out 12	Analog Out 13
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board 	DS1513 Multi-I/O Board	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module
Output current range	-5 mA ... +5 mA	-8 mA ... +8 mA	-8 mA ... +8 mA
Output voltage range	0 ... +4.5 V	-10 V ... +10 V	0 ... +5 V
Resolution	12 bit	16 bit	16 bit
Offset error	±11 mV	±4 mV	±2 mV
Gain error	±0.5 %	±0.25 %	±0.25 %
Circuit diagrams	Analog Out 11 on page 1559	Analog Out 12 on page 1559	Analog Out 13 on page 1560
Required channels	1	1	1

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to [DS1552 Multi-I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Multi Bit In

Where to go from here

Information in this section

Introduction (Multi Bit In).....	200
Overviews (Multi Bit In).....	201
Configuring the Function Block (Multi Bit In).....	206
Hardware Dependencies (Multi Bit In).....	213

Introduction (Multi Bit In)

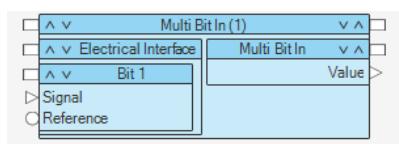
Introduction to the Function Block (Multi Bit In)

Function block purpose

The Multi Bit In function block type lets you measure digital signals coming from an external device.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Providing digital signals from external devices to the behavior model.
- Generating and providing I/O events to the behavior model when a specified number of rising or falling edges is detected in the input signal.

Supported channel types

The Multi Bit In function block type supports the following channel types:

	SCALEXIO									
Channel type	Digital In 1	Flexible In 2	Flexible In 1	Digital In 2	Digital In/Out 1	Digital In 3	Flexible In 3	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9
Hardware	DS2680	DS2601	DS2690	DS6101		DS6201	DS6202	DS6121		

	MicroAutoBox III				
Channel type	Digital In 4	Digital In 5	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10
Hardware	<ul style="list-style-type: none"> ▪ DS1511 ▪ DS1511B1 ▪ DS1513 	<ul style="list-style-type: none"> ▪ DS1552 ▪ DS1552B1 	<ul style="list-style-type: none"> ▪ DS1552 ▪ DS1552B1 	DS1554	DS1521

Overviews (Multi Bit In)

Where to go from here

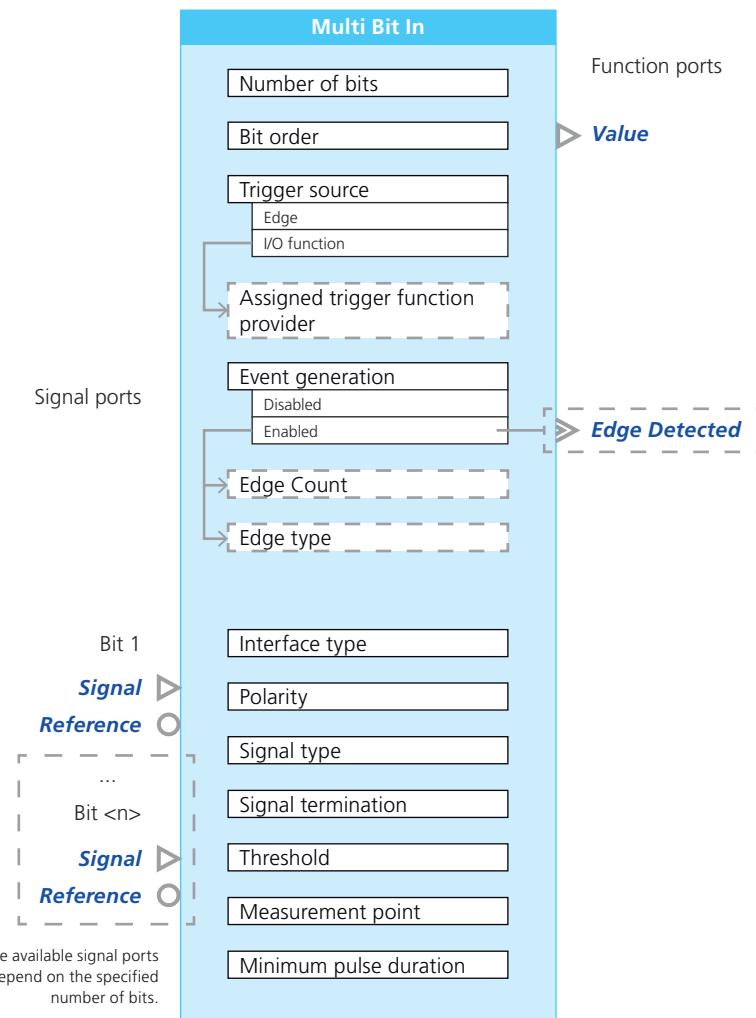
Information in this section

Overview of Ports and Basic Properties (Multi Bit In).....	201
Overview of Tunable Properties (Multi Bit In).....	204

Overview of Ports and Basic Properties (Multi Bit In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Value**

This function outport writes data coming from the external device's output to the behavior model. The incoming data is written to an array.

Value range	<ul style="list-style-type: none"> 0 or 1 for each entry in the array. The number of entries in the array depends on the value specified at the Number of bits property. The maximum number of entries depends on the assigned hardware. For specific values, refer to Hardware Dependencies (Multi Bit In) on page 213.
Dependencies	-

Edge Detected

This event port provides an I/O event each time an edge is detected during signal capturing and this edge matches the specified trigger condition. Only signals at the signal port of the logical signal Bit 1 are evaluated for generating I/O events.

You have to specify the following settings for the trigger condition:

- Which type of edge (rising or falling or both) of the input signal is relevant for event generation (via Edge type property).
- The number of detected edges after which an event is generated (via Edge count property).

Value range	-
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Event generation property is set to Enabled. ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Not supported for the Digital In 2 and Digital In/Out 1 channel types. ▪ The number of detected edges is not evaluated for event generation for the Digital In/Out 3 channel type (Edge count = 1). ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ The number of detected edges is not evaluated for event generation for the following channel types: Digital In 4, Digital In 5, Digital In/Out 6, Digital In/Out 8 (Edge count = 1).

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit In) on page 213.
Dependencies	The number of Signal ports depends on the value specified at the Number of bits property. The maximum number of ports depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit In) on page 213.

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit In) on page 213.
Dependencies	<ul style="list-style-type: none"> ▪ The number of Reference ports depends on the value specified at the Number of bits property. The maximum number of ports depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit In) on page 213. ▪ SCALEXIO hardware: Not supported for Digital In 2 and Digital In/Out 1 channel types.

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Always available for Digital In 2 and Digital In/Out 1 channel types. ▪ Not supported for Digital In 3, Digital In/Out 3, Digital In/Out 5, Digital In/Out 9, and Flexible In 3 channel types. ▪ For all the other channel types available only if the Connected load property is set to Use external load. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Supported only for the Flexible In 1 channel type. ▪ Available only if the Connected load property is set to Use external load. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported.

Overview of Tunable Properties (Multi Bit In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Bit order	✓	-
Edge count ³⁾	✓	-
Edge type ³⁾	✓	-

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Polarity	✓	-
Signal termination	✓	-
Threshold	✓	-
Measurement point	✓	-
Minimum pulse duration	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ This parameter is available only if the Event generation parameter is enabled.

Configuring the Function Block (Multi Bit In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Multi Bit In).....	206
Configuring Standard Features (Multi Bit In).....	210

Configuring the Basic Functionality (Multi Bit In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the number of bits and the bit order
- Specifying the signal type (voltage or current)
- Specifying the trigger threshold
- Terminating the input signal
- Specifying the trigger source for signal measurement
- Specifying pulse filter for the input signal
- Specifying the zero detection frequency
- Specifying the polarity of the binary signal provided to the behavior model
- Providing I/O events

Specifying the number of bits (vector size)

You can define the number of bits/digital input lines representing the input signal (= vector size) by specifying the Number of bits property.

The maximum possible range depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Multi Bit In\)](#) on page 213.

Specifying the bit order

You can change the Bit order of a vector delivered by the external device (e.g., ECU):

- Normal mode:
Bit 1 of the vector (delivered to the behavior model) is mapped to the lowest channel number (LSB) and bit n of the vector is mapped to the highest channel number (MSB) of the relevant channels.
- Inverse mode:
Bit 1 of the vector (delivered to the behavior model) is mapped to the highest channel number (MSB) and bit n of the vector is mapped to the lowest channel number (LSB) of the relevant channels.

Specifying the signal type

You must select the Signal type according to the connected external device that delivers the input signal:

- Voltage

The input signal is delivered as a voltage signal, for example, a square-wave signal in the range 0 ... 5 V.

- Current

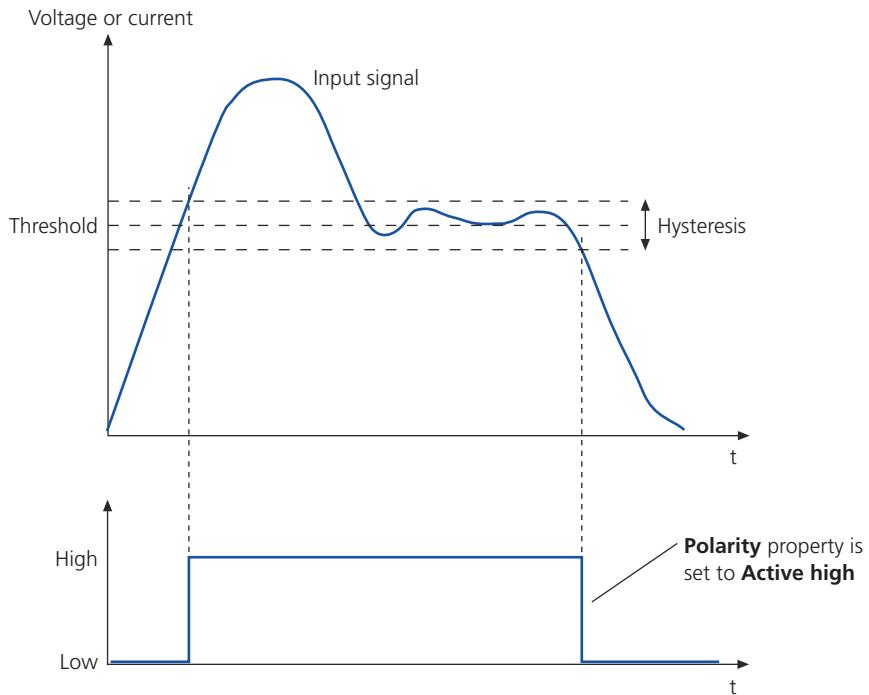
The input signal is delivered as a current signal, for example, a square-wave signal in the range 0 ... 500 mA.

Limitation The Current signal type is only supported by the following channel types:

- SCALEXIO hardware: Flexible In 1, Flexible In 2

Specifying the trigger threshold

You can specify a threshold value to transform the input signal into binary values. The hysteresis value is fixed and depends on the assigned hardware resource. A rising edge of the input signal must exceed the threshold plus half the hysteresis value to be detected as high, and a falling edge of the input signal must fall below the threshold minus the half hysteresis value to be detected as low.



The value range of the threshold and the hysteresis constant depend on the assigned hardware resource and the selected signal type. For specific values, refer to [Hardware Dependencies \(Multi Bit In\)](#) on page 213.

Limitation The threshold value is fixed and cannot be changed for the following channel types:

- MicroAutoBox III hardware: Digital In 4, Digital In 5, Digital In/Out 10

Terminating the input signal

You can activate signal termination for differential input signals to prevent signal reflections at the line end.

Note

You cannot activate a signal termination for ground-based input signals.

Therefore, signal termination is supported only by the Digital In/Out 5 channel type. A $120\ \Omega$ resistor is connected in series to a 5 nF capacitor between the input lines. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying the trigger source for signal measurement

You can trigger the signal measurement with each detected edge of a digital input line or via trigger events of a trigger function provider.

Triggering via detected edges A detected edge of a digital input line triggers the signal measurement to update the value of the bit that represents the digital input line.

Triggering via trigger function Each trigger event that is provided by the assigned trigger function provider triggers the signal measurement of all digital input lines, e.g., trigger events of the Trigger In function block. All bit values are synchronously updated.

Using MicroAutoBox III hardware Signal measurement is accomplished with each model step. Therefore, a trigger source cannot be specified for MicroAutoBox III hardware.

Filtering the input signal

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the pulse duration depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Multi Bit In\)](#) on page 213 .

You can also change the value of the **Minimum pulse duration** property via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Limitation The pulse filter cannot be configured for the following channel types:

- SCALEXIO hardware: Digital In 2, Digital In/Out 1, Digital In/Out 3

Specifying polarity of the signal

To adapt the characteristic of the input signal to the requirements of your behavior model, you can switch the **Polarity** of the signal available at the function port. With this you can handle signals of different polarity without the need to perform signal inversion in the behavior model or use inverters on the hardware side.

- Active high

The higher value (voltage/current) of the signal represents a binary 1, and the lower value represents a binary 0.

- Active low

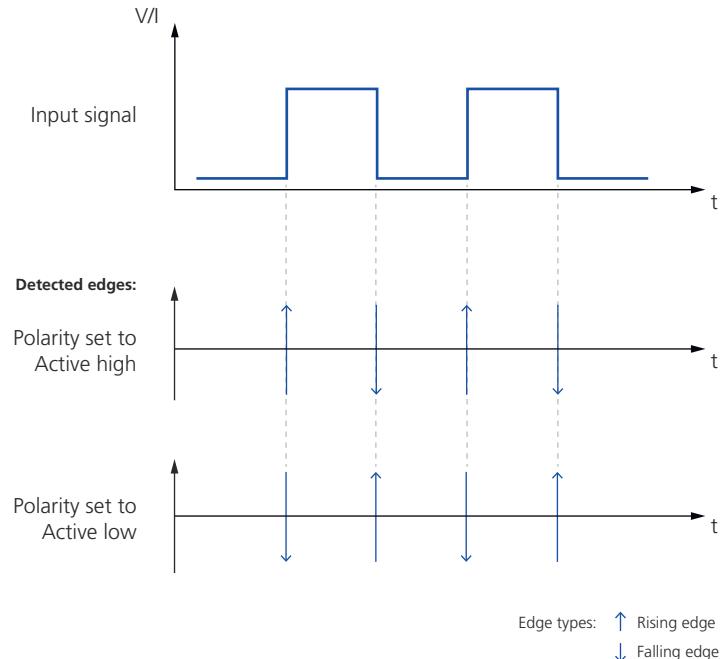
The lower value (voltage/current) of the signal represents a binary 1, and the higher value represents a binary 0.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

You can specify the number of detected edges after which an event is generated via the **Edge count** property. The maximum number depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Multi Bit In\)](#) on page 213. The **Edge type** property lets you specify if all edges are counted or only rising/falling edges are counted.

Note that if you invert the input signal by setting the **Polarity** property to Active low, the edge type of the detected edges differs from the original input signal as shown in the following illustration:



To use the I/O events in the behavior model, you have to assign them to a task via the **Edge Detected** property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Limitations

- SCALEXIO hardware:
 - Event generation is not supported for the Digital In 2 and Digital In/Out 1 channel types.
 - The number of detected edges is not evaluated for event generation for the Digital In/Out 3 channel type (Edge count = 1).
- MicroAutoBox III hardware:
 - The number of detected edges is not evaluated for event generation for the following channel types: Digital In 4, Digital In 5, Digital In/Out 6, Digital In/Out 8 (Edge count = 1).

Configuring Standard Features (Multi Bit In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SCALEXIO							
Configuration Feature	Digital In 1	Digital In 2	Digital In 3	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Further Information
Measurement point	Load side	Load side	–	Load side	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Ground-based	Ground-based	Ground-based	Ground-based	▪ Ground-based ▪ Differential	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	✓	✓	–	✓	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports							
Trigger level of electronic fuse	–	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	✓	–	–	Specifying Settings for Failure Simulation on page 123

SCALEXIO							
Configuration Feature	Digital In 1	Digital In 2	Digital In 3	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Further Information
Usage of loads	✓	✓	–	✓	–	–	Specifying Load Settings on page 121

SCALEXIO						
Configuration Feature	Digital In/Out 9	Flexible In 1	Flexible In 2	Flexible In 3	Further Information	
Measurement point	–	▪ Load side ▪ ECU side ¹⁾	Load side	–	Specifying the Measurement Point for Input Signals on page 119	
Interface type	Ground-based	▪ Galvanically isolated ▪ Ground-based	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116	
Channel multiplication	–	✓	✓	–	Specifying Current and Voltage Values for Channel Multiplication on page 95	
Signal Ports						
Trigger level of electronic fuse	–	✓	–	–	Specifying Fuse Settings on page 122	
Failure simulation support	–	✓	✓	–	Specifying Settings for Failure Simulation on page 123	
Usage of loads	–	✓	✓	–	Specifying Load Settings on page 121	

¹⁾ Supported only if you measure voltages (Signal type property is set to Voltage).

MicroAutoBox III						
Configuration Feature	Digital In 4	Digital In 5	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10	Further Information
Measurement point	–	–	–	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Ground-based	Ground-based	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	–	–	–	–	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Digital In 1 on page 1569 ▪ Digital In 2 on page 1571 ▪ Digital In 3 on page 1572 ▪ Digital In/Out 1 on page 1582 ▪ Digital In/Out 3 on page 1584 ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 ▪ Flexible In 1 on page 1593 ▪ Flexible In 2 on page 1595 ▪ Flexible In 3 on page 1603 	<ul style="list-style-type: none"> ▪ Digital In 4 on page 1572 ▪ Digital In 5 on page 1573 ▪ Digital In/Out 6 on page 1588 ▪ Digital In/Out 8 on page 1589 ▪ Digital In/Out 10 on page 1591

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Multi Bit In)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Multi Bit In).....	213
MicroAutoBox III Hardware Dependencies (Multi Bit In).....	216

SCALEXIO Hardware Dependencies (Multi Bit In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital In 1	Flexible In 2	Flexible In 1	Digital In 2	Digital In/Out 1
Hardware	DS2680 I/O Unit		DS2601 Signal Measurement Board	DS2690 Digital I/O Board	
Number of bits	1 ... 30	1 ... 18	1 ... 10	1 ... 10	
Event generation	✓		✓	–	
Maximum number of edges after which an event is generated	65,535		65,535	–	
Input current range	<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 144 A_{RMS}¹⁾ for 30 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 86.4 A_{RMS}¹⁾ for 18 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +10 A_{RMS} for 1 channel ▪ 80 A_{RMS} for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 48 A_{RMS} for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +100 mA_{RMS} for 1 channel ▪ 0.8 A_{RMS} for 10 channels (channel multiplication)
Input voltage range	0 ... +60 V		-60 V ... +60 V	0 ... +60 V	
Signal type	Voltage	<ul style="list-style-type: none"> ▪ Voltage ▪ Current 	<ul style="list-style-type: none"> ▪ Voltage ▪ Current 	Voltage	
Threshold	Input threshold voltage	0 ... +24 V	<ul style="list-style-type: none"> ▪ Voltage signal type: 0 ... +24 V ▪ Current signal type: -18 A ... +18 A 	<ul style="list-style-type: none"> ▪ Voltage signal type: -60 V ... +60 V ▪ Current signal type: -30 A ... +30 A 	+1 V ... +23.8 V
	Input hysteresis voltage	200 mV (typ.)	<ul style="list-style-type: none"> ▪ Voltage signal type: 210 mV (typ.) ▪ Current signal type: 	<ul style="list-style-type: none"> ▪ Voltage signal type: 250 mV (typ.) ▪ Current signal type: 	200 mV (typ.)

Channel Type		Digital In 1	Flexible In 2	Flexible In 1	Digital In 2	Digital In/Out 1		
		120 mA (typ.)		125 mA (typ.)				
DAC resolution		8 bit		14 bit	8 bit			
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs		0.264 µs ... 131 µs	80 ns			
Supported interface type		Ground-based		<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Ground-based 	Ground-based			
Measurement point		Load side		Voltage signal type: <ul style="list-style-type: none"> ▪ Load side ▪ ECU side Current signal type: <ul style="list-style-type: none"> ▪ Load side 	Load side			
Configurable fuse	–		<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	–				
Signal termination	–		–	–				
Circuit diagrams	Digital In 1 on page 1569	Flexible In 2 on page 1595	Flexible In 1 on page 1593	Digital In 2 on page 1571	Digital In/Out 1 on page 1582			
Required channels	1 for each bit (Additional channels are required for current enhancements.)							

¹⁾ This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

Channel Type	Digital In 3	Flexible In 3	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9
Hardware	DS6101 Multi-I/O Board		DS6201 Digital I/O Board	DS6202 Digital I/O Board	DS6121 Multi-I/O Board
Number of bits	1 ... 12	1 ... 10	<ul style="list-style-type: none"> ▪ 1 ... 32 (each bank) ▪ 3 banks available, each with 32 channels 	1 ... 32	1 ... 4
Event generation	✓	✓ ¹⁾		✓	✓
Maximum number of edges after which an event is generated	65,535		1	256	256
Input current range	–		–	–	–
Input voltage range	0 ... +60 V		0 ... +60 V	0 ... +30 V	0 ... +60 V
Signal type	Voltage		Voltage	Voltage	Voltage

Channel Type		Digital In 3	Flexible In 3	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9
Threshold	Input threshold voltage	0 ... +24 V	-10 V ... +10 V	+1 V... +23 V	0 ... +12 V	0 ... +12 V
	Input hysteresis voltage	200 mV (typ.)		230 mV (typ.)	600 mV (typ.)	600 mV (typ.)
	DAC resolution	8 bit		8 bit	8 bit	8 bit
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs		80 ns	0 ... 10 ms	0 ... 10 ms
Supported interface type		Ground-based		Ground-based	<ul style="list-style-type: none"> ▪ Ground-based ▪ Differential 	Ground-based
Measurement point		–		–	–	–
Configurable fuse		–		–	–	–
Signal termination		–		–	<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (parallel termination, only for differential input signals) ▪ Switchable 	–
Circuit diagrams		Digital In 3 on page 1572	Flexible In 3 on page 1603	Digital In/Out 3 on page 1584	Digital In/Out 5 on page 1586	Digital In/Out 9 on page 1590
Required channels		1 for each bit		1 for each bit	1 for each bit	1 for each bit

¹⁾ For this channel type, the number of detected edges is not evaluated for event generation (Edge count = 1).

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6121 Multi I/O Board Channel multiplication is not supported in general.

DS6201 Digital I/O Board

- Channel multiplication is not supported in general.
- The DS6201 allows the generation of max. eight events on the board.

DS6202 Digital I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2690 Digital I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS2690 Digital I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6101 Multi-I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6121 Multi-I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6201 Digital I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6201 Digital I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6202 Digital I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

MicroAutoBox III Hardware Dependencies (Multi Bit In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital In 4	Digital In 5	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module 		DS1554 Engine Control I/O Module	DS1521 Bus Board
Number of bits	1 ... 16	1 ... 16	1 ... 8	1 ... 8	1 ... 6
Event generation	✓ ¹⁾	✓ ¹⁾		✓ ¹⁾	✓
Maximum number of edges after which an event is generated	1	1		1	256
Input voltage range	0 ... +40 V	0 ... +40 V	0 ... +15 V	0 ... +15 V	0 ... +60 V
Signal type	Voltage	Voltage		Voltage	Voltage
Threshold	Input threshold voltage	+2 V (typ.)	+2 V (typ.)	+1 V ... +7.5 V	+1 V ... +7.5 V
	Input hysteresis voltage	1 V (typ.)	1 V (typ.)	0.7 V (typ.)	0.6 V (typ.)

Channel Type		Digital In 4	Digital In 5	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10
Pulse filter	Minimum pulse duration	0 ... 12.75 µs	0 ... 12.75 µs		0 ... 12.75 µs	0 ... 0.01 s
Supported interface type		Ground-based	Ground-based		Ground-based	Ground-based
Circuit diagram		Digital In 4 on page 1572	Digital In 5 on page 1573	Digital In/Out 6 on page 1588	Digital In/Out 8 on page 1589	Digital In/Out 10 on page 1591
Required channels		1 for each bit	1 for each bit		1 for each bit	1 for each bit

¹⁾ For this channel type, the number of detected edges is not evaluated for event generation (Edge count = 1).

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to [DS1552 Multi-I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1554 Engine Control I/O Module For more module-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more module-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Trigger In

Where to go from here

Information in this section

Introduction (Trigger In).....	220
Overviews (Trigger In).....	221
Configuring the Function Block (Trigger In).....	224
Hardware Dependencies (Trigger In).....	228

Introduction (Trigger In)

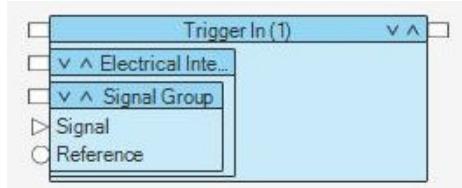
Introduction to the Function Block (Trigger In)

Function block purpose

The Trigger In function block type generates a trigger signal each time the external input signal matches the defined triggering conditions. The function block works as a provider: Other function blocks can use the generated trigger signal as trigger source.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating trigger signals from an external input signal for specified trigger conditions.
- Providing the triggers to be used by other function blocks.

Supported channel types

The Trigger In function block type supports the following channel types:

	SCALEXIO									MicroAutoBox III	
Channel type	Digital In 1	Flexible In 2	Digital In 3	Flexible In 3	Flexible In 1	Digital In/Out 5	Digital In/Out 9	Trigger In 1	Trigger In 2	Digital In 5	Trigger In 3
Hardware	DS2680		DS6101		DS2601	DS6202	DS6121	DS6221	DS6241	▪ DS1552 ▪ DS1552B1	▪ DS1511 ▪ DS1511B1 ▪ DS1513

Overviews (Trigger In)

Where to go from here

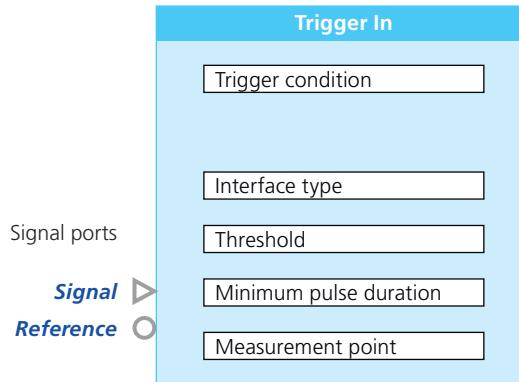
Information in this section

- | | |
|--------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Trigger In)..... | 221 |
| Overview of Tunable Properties (Trigger In)..... | 223 |

Overview of Ports and Basic Properties (Trigger In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



The Trigger In function block does not provide function ports.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Trigger In) on page 228.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Trigger In) on page 228.
Dependencies	–

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none">▪ SCALEXIO hardware:<ul style="list-style-type: none">▪ Supported only for Flexible In 1 and Flexible In 2 channel types.▪ Available only if the Connected load property is set to Use external load.▪ MicroAutoBox III hardware:<ul style="list-style-type: none">▪ Not supported.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none">▪ SCALEXIO hardware:<ul style="list-style-type: none">▪ Supported only for the Flexible In 1 channel type.▪ Available only if the Connected load property is set to Use external load.▪ MicroAutoBox III hardware:<ul style="list-style-type: none">▪ Not supported.

Overview of Tunable Properties (Trigger In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports	–	–
Electrical Interface		
Threshold	✓	–
Minimum pulse duration	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Trigger In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Trigger In).....	224
Configuring Standard Features (Trigger In).....	226

Configuring the Basic Functionality (Trigger In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

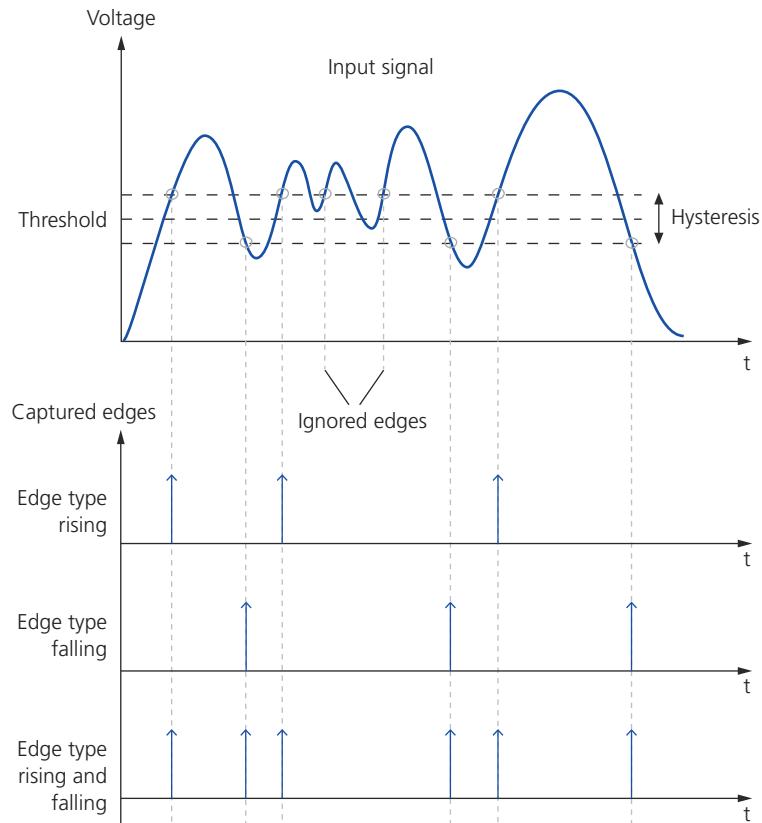
- Specifying trigger conditions (edge type, threshold)

Specifying the trigger conditions

Via the Trigger condition property, you can specify on which edge of the input signal a trigger is generated: On the rising edge, on the falling edge, or on both edges.

The adjustable threshold (via Threshold property) lets you configure the level of the triggering voltage input signal, to avoid, for example, that a trigger is generated by noise.

A rising edge of the voltage input signal must exceed the threshold plus half the hysteresis value to trigger the capture. A falling edge of the input signal must fall below the threshold minus half the hysteresis value. The following illustration shows the capturing of the different edge types depending on threshold and hysteresis.



Edges of the input signal are ignored for the following reasons:

- The signal does not pass through the threshold.
- The signal does not reach the threshold plus half the hysteresis.
- The signal does not reach the threshold minus half the hysteresis.

You can change the threshold via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

The value range of the threshold and the hysteresis depend on the assigned hardware. For specific values, refer to [Hardware Dependencies \(Trigger In\)](#) on page 228.

Limitations

- MicroAutoBox III hardware:
 - Only rising edges or falling edges are supported trigger conditions.
 - The threshold value is fixed.

Configuring Standard Features (Trigger In)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO						
Configuration Feature	Digital In 1	Digital In 3	Digital In/Out 5	Digital In/Out 9	Flexible In 1	Further Information
Measurement point	Load side	–	–	–	▪ Load side ▪ ECU side	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Ground-based	Ground-based	Ground-based	▪ Ground-based ▪ Galvanically isolated	Specifying the Circuit Type for Voltage Input Signals on page 116
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	–	✓	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	–	–	–	✓	Specifying Load Settings on page 121

SCALEXIO					
Configuration Feature	Flexible In 2	Flexible In 3	Trigger In 1	Trigger In 2	Further Information
Measurement point	Load side	–	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Ground-based	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Signal Ports					
Trigger level of electronic fuse	–	–	–	–	Specifying Fuse Settings on page 122

SCALEXIO					
Configuration Feature	Flexible In 2	Flexible In 3	Trigger In 1	Trigger In 2	Further Information
Failure simulation support	✓	–	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	–	–	–	Specifying Load Settings on page 121

MicroAutoBox III			
Configuration Feature	Digital In 5	Trigger In 3	Further Information
Measurement point	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Signal Ports			
Trigger level of electronic fuse	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	–	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Digital In 1 on page 1569 ▪ Digital In 3 on page 1572 ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 ▪ Flexible In 1 on page 1593 ▪ Flexible In 2 on page 1595 ▪ Flexible In 3 on page 1603 ▪ Trigger In 1 on page 1621 ▪ Trigger In 2 on page 1622 	<ul style="list-style-type: none"> ▪ Digital In 5 on page 1573 ▪ Trigger In 3 on page 1622

Model interface

The Trigger In function block type does not provide a model interface.

Hardware Dependencies (Trigger In)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Trigger In).....	228
MicroAutoBox III Hardware Dependencies (Trigger In).....	230

SCALEXIO Hardware Dependencies (Trigger In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital In 1	Flexible In 2	Flexible In 1	Digital In 3	Flexible In 3
Hardware		DS2680 I/O Unit		DS2601 Signal Measurement Board	DS6101 Multi-I/O Board	
Input voltage range		0 ... +60 V		-60 V ... +60 V	0 ... +60 V	
Threshold	Input threshold voltage	0 ... +24 V		-60 V ... +60 V	0 ... +24 V	-10 V ... +10 V
	Input hysteresis voltage	200 mV (typ.)	210 mV (typ.)	250 mV (typ.)	200 mV (typ.)	
	DAC resolution	8 bit		14 bit	8 bit	
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs		0.264 µs ... 131 µs	0.264 µs ... 131 µs	
Supported interface type		Ground-based		<ul style="list-style-type: none"> ▪ Ground-based ▪ Galvanically isolated 	Ground-based	
Measurement point		Load side		<ul style="list-style-type: none"> ▪ Load side ▪ ECU side 	ECU side	
Configurable fuse		–		<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	–	
Circuit diagrams		Digital In 1 on page 1569	Flexible In 2 on page 1595	Flexible In 1 on page 1593	Digital In 3 on page 1572	Flexible In 3 on page 1603
Required channels		1		1	1	

Channel Type	Digital In/Out 5	Digital In/Out 9	Trigger In 1	Trigger In 2
Hardware	DS6202 Digital I/O Board	DS6121 Multi-I/O Board	DS6221 A/D Board	DS6241 D/A Board
Input voltage range	0 ... +30 V	0 ... +60 V	0 ... +60 V	0 V ... +60 V

Channel Type		Digital In/Out 5	Digital In/Out 9	Trigger In 1	Trigger In 2
Threshold	Input threshold voltage	0 ... +12 V	0 ... +12 V	0 ... +24 V	0 V ... +24 V
	Input threshold hysteresis	600 mV (typ.)	600 mV (typ.)	600 mV (typ.)	200 mV (typ.)
	DAC resolution	8 bit	8 bit	8 bit	8 bit
Pulse filter	Minimum pulse duration	0 ... 10 ms	0 ... 10 ms	–	–
Supported interface type	Ground-based	Ground-based	Ground-based	Ground-based	Ground-based
Measurement point	–	–	–	–	–
Configurable fuse	–	–	–	–	–
Circuit diagrams	Digital In/Out 5 on page 1586	Digital In/Out 9 on page 1590	Trigger In 1 on page 1621	Trigger In 2 on page 1622	
Required channels	1	1	1	1	

General limitations

The hardware resources assigned to the Trigger In function block and the hardware resources assigned to the function block using the Trigger In function block as trigger source must be located on the same I/O board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6221 A/D Board For more board-specific data, refer to [Data Sheet of the DS6221 A/D Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6241 D/A Board For more board-specific data, refer to [Data Sheet of the DS6241 D/A Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (Trigger In)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital In 5	Trigger In 3
Hardware	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module 	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board
Input voltage range	0 ... +40 V	0 ... +40 V
Supported interface type	Ground-based	Ground-based
Circuit diagram	Digital In 5 on page 1573	Trigger In 3 on page 1622
Required channels	1	1

General limitations

The hardware resources assigned to the Trigger In function block and the hardware resources assigned to the function block using the Trigger In function block as trigger source must be located on the same I/O board.

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to [DS1552 Multi-I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Multi Bit Out

Where to go from here

Information in this section

Introduction (Multi Bit Out).....	232
Overviews (Multi Bit Out).....	233
Configuring the Function Block (Multi Bit Out).....	237
Hardware Dependencies (Multi Bit Out).....	242

Introduction (Multi Bit Out)

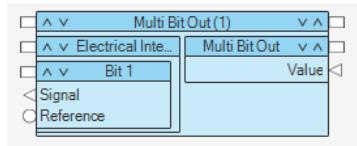
Introduction to the Function Block (Multi Bit Out)

Function block purpose

The Multi Bit Out function block type lets you stimulate digital inputs of an external device.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating digital signals from data received from the behavior model.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Multi Bit Out function block type supports the following channel types:

	SCALEXIO									
Channel type	Digital Out 1	Digital Out 2	Digital In/Out 1	Digital Out 3	Digital In/Out 3	Digital In/Out 5	Digital Out 8	Digital In/Out 9	Flexible Out 1	
Hardware	DS2680	DS2690		DS6101	DS6201	DS6202	DS6121		DS2621	

	MicroAutoBox III						
Channel type	Digital Out 4	Digital Out 5	Digital In/Out 6	Digital Out 7	Digital In/Out 8	Digital In/Out 10	
Hardware	▪ DS1511 ▪ DS1511B1 ▪ DS1513	▪ DS1552 ▪ DS1552B1		DS1554		DS1521	

Overviews (Multi Bit Out)

Where to go from here

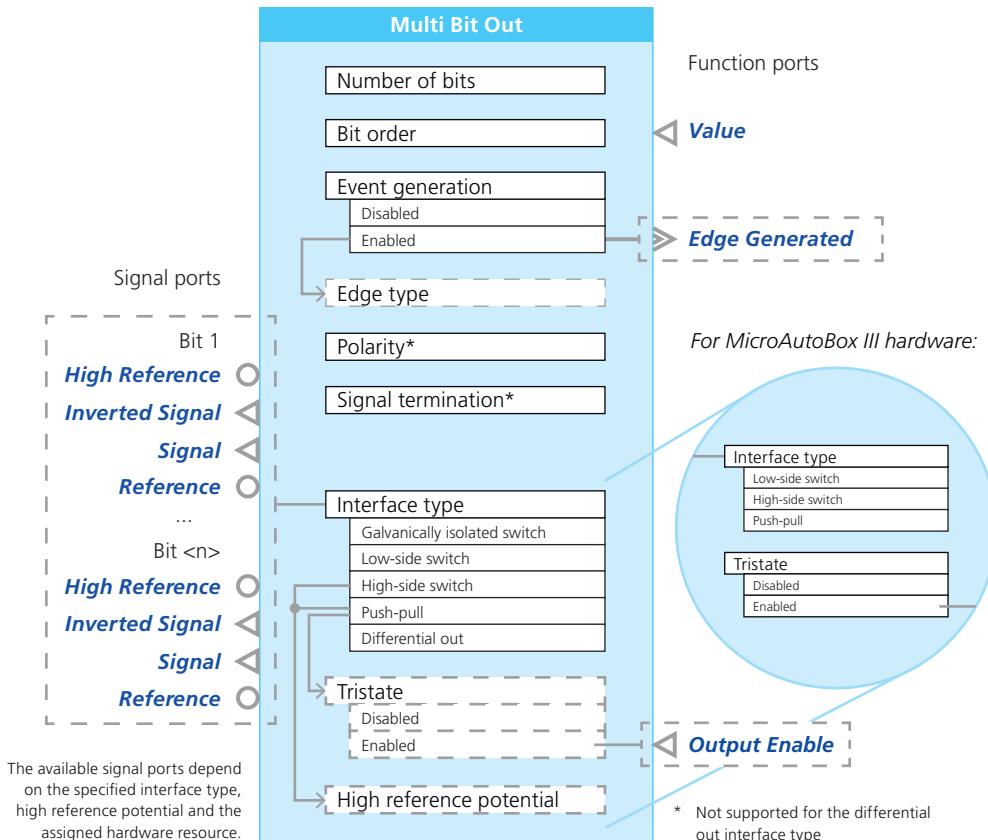
Information in this section

- [Overview of Ports and Basic Properties \(Multi Bit Out\).....](#) 233
- [Overview of Tunable Properties \(Multi Bit Out\).....](#) 236

Overview of Ports and Basic Properties (Multi Bit Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Value

This function import lets you define data of a logical signal going to the external device's input from within the behavior model. The outgoing data is read from an array.

Value range	<ul style="list-style-type: none"> ▪ 0 or 1 for each entry in the array ▪ The number of entries in the array depends on the value specified at the Number of bits property. The maximum number of entries depend on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242.
Dependencies	–

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Available only if the Tristate property is set to Enabled.

Edge Generated

This event port provides an I/O event each time an edge is generated during signal generation and this edge matches the trigger condition. Only signals at the signal port of the logical signal Bit 1 are evaluated for generating I/O events.

You have to specify which type of edge of the output signal (rising and/or falling) is relevant for event generation (via the Edge type property).

Value range	–
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Not supported ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Available only if the Event generation property is set to Enabled. ▪ Not supported for the Digital In/Out 10 channel type.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource. ▪ The number of High Reference ports depends on the value specified at the Number of bits property. The maximum number of ports depend on the assigned hardware. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242
Dependencies	The number of Signal ports depends on the value specified at the Number of bits property. The maximum number of ports depend on the assigned hardware. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242.

Inverted Signal

This signal port and the Signal signal port together represent the electrical connection points of a logical signal with two complementary potentials carrying the information in the voltage difference between these signals. Both signal ports (Signal and Inverted Signal) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242
Dependencies	<ul style="list-style-type: none"> ▪ The number of Inverted Signal ports depends on the value specified at the Number of bits property. The maximum number of ports depend on the assigned hardware. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242. ▪ Available only if the Interface type property is set to Differential out.

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource. ▪ The number of Reference ports depends on the value specified at the Number of bits property. The maximum number of ports depend on the assigned hardware. For specific values, refer to Hardware Dependencies (Multi Bit Out) on page 242.

Overview of Tunable Properties (Multi Bit Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Parameter	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Bit order	✓	–
Edge type	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Polarity	✓	–
Signal termination	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Multi Bit Out)

Where to go from here

Information in this section

- | | | |
|---------------------------------------------------------------------|-------|-----|
| Configuring the Basic Functionality (Multi Bit Out) | | 237 |
| Configuring Standard Features (Multi Bit Out) | | 238 |

Configuring the Basic Functionality (Multi Bit Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Defining the number of bits and the bit order.
- Specifying the generation of I/O events.
- Specifying signal termination.

Defining the number of bits (vector size)

You can define the number of bits/digital input lines representing the output signal (= vector size) by specifying the [Number of bits](#) property.

The maximum possible range depends on the assigned hardware. For further information, refer to [Hardware Dependencies \(Multi Bit Out\)](#) on page 242.

Specifying the bit order

You can change the Bit order of a vector provided to the external device (e.g., ECU):

- Normal mode:
Bit 1 of the vector (delivered from the behavior model) is mapped to the lowest channel number (LSB) and bit n of the vector is mapped to the highest channel number (MSB) of the relevant channels.
- Inverse mode:
Bit 1 of the vector (delivered from the behavior model) is mapped to the highest channel number (MSB) and bit n of the vector is mapped to the lowest channel number (LSB) of the relevant channels.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Multi Bit Out function block can generate an I/O event each time an edge is generated during signal generation and this edge matches the setting of the Edge type property (rising or falling edge or both edges).

To use the I/O events in the behavior model, you have to assign them to a task via the Edge Generated property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Limitation Event generation is not supported for the following channel types:

- SCALEXIO hardware: All channel types
- MicroAutoBox III hardware: Digital In/Out 10

Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

Circuit diagrams For circuit diagrams of the channel types that support signal termination, refer to:

- [Digital Out 8](#) on page 1581
- [Digital In/Out 5](#) on page 1586
- [Digital In/Out 9](#) on page 1590

Limitation Signal termination is supported by the following channel types:

- SCALEXIO hardware: Digital Out 8, Digital In/Out 5 (not supported for the differential out interface type), Digital In/Out 9
- MicroAutoBox III hardware: Signal termination is not supported.

Configuring Standard Features (Multi Bit Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO					
Configuration Feature	Digital Out 1	Digital Out 2	Digital Out 3	Digital Out 8	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<p>Depending on the channel type, refer to:</p> <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 2, Digital In/Out 1) on page 106 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.				–
Polarity of output signals	✓	✓	✓	✓	–
Channel multiplication	✓	✓	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports					
Trigger level of electronic fuse	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	–	Specifying Settings for Failure Simulation on page 123

SCALEXIO						
Configuration Feature	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<p>Depending on the channel type, refer to:</p> <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 2, Digital In/Out 1) on page 106 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109 ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112 ▪ Specifying Digital Output Signals

SCALEXIO						
Configuration Feature	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9	Flexible Out 1	Further Information
						(Flexible Out 1) on page 98
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.					–
Polarity of output signals	✓	✓	<ul style="list-style-type: none"> ▪ ✓ ▪ Not supported for the differential out interface type. 	✓	✓	–
Channel multiplication	✓	–	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	–	✓	Specifying Settings for Failure Simulation on page 123

MicroAutoBox III							
Configuration Feature	Digital Out 4	Digital Out 5	Digital Out 7	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–
High impedance (tristate)	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.					High impedance (tristate) is supported only for the push-pull interface type.	–
Polarity of output signals	✓	✓	✓	✓	✓	✓	–
Channel multiplication	–	–	–	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95

MicroAutoBox III							
Configuration Feature	Digital Out 4	Digital Out 5	Digital Out 7	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10	Further Information
Signal Ports							
Trigger level of electronic fuse	–	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	–	–	–	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Digital Out 1 on page 1575 ▪ Digital Out 2 on page 1576 ▪ Digital Out 3 on page 1577 ▪ Digital Out 8 on page 1581 ▪ Digital In/Out 1 on page 1582 ▪ Digital In/Out 3 on page 1584 ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 ▪ Flexible Out 1 on page 1604 	<ul style="list-style-type: none"> ▪ Digital Out 4 on page 1578 ▪ Digital Out 5 on page 1579 ▪ Digital Out 7 on page 1580 ▪ Digital In/Out 6 on page 1588 ▪ Digital In/Out 8 on page 1589 ▪ Digital In/Out 10 on page 1591

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Multi Bit Out)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Multi Bit Out).....	242
MicroAutoBox III Hardware Dependencies (Multi Bit Out).....	245

SCALEXIO Hardware Dependencies (Multi Bit Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 2	Digital In/Out 1	Digital Out 3
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS2690 Digital I/O Board		DS6101 Multi-I/O Board
Number of bits	1 ... 28	1 ... 10	1 ... 10		1 ... 14
Event generation	–	–	–		–
Output current range	<ul style="list-style-type: none"> ▪ 0 ... +80 mA_{RMS} for 1 channel ▪ 1.792 A_{RMS} for 28 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +40 mA_{RMS} (1 channel) ▪ 0.32 A_{RMS} for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +80 mA_{RMS} for 1 channel ▪ 0.64 A_{RMS} for 10 channels (channel multiplication) 		<ul style="list-style-type: none"> ▪ 0 ... +140 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}.
High side reference voltage	+5 V ... +60 V	-60 V ... +60 V	+5 V ... +60 V		+5 V ... +60 V
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull
In push-pull configuration the outputs can be set to high impedance (tristate) at run time.					
Configurable fuse	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–		–
Signal termination	–	–	–		–

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 2	Digital In/Out 1	Digital Out 3
Circuit diagrams	Digital Out 1 on page 1575	Flexible Out 1 on page 1604	Digital Out 2 on page 1576	Digital In/Out 1 on page 1582	Digital Out 3 on page 1577
Required channels	1 for each bit (additional channels are required for current enhancement.)	1 for each bit (additional channels are required for current enhancement and operation in push/pull mode.)	1 for each bit (additional channels are required for current enhancement.)		1
Channel Type	Digital In/Out 3	Digital In/Out 5	Digital Out 8	Digital In/Out 9	
Hardware	DS6201 Digital I/O Board	DS6202 Digital I/O Board	DS6121 Multi-I/O Board		
Number of bits	<ul style="list-style-type: none"> ▪ 1 ... 32 (each bank) ▪ 3 banks available, each with 32 channels 	1 ... 32	1 ... 12	1 ... 4	
Event generation	–	–	–	–	
Output current range	<ul style="list-style-type: none"> ▪ 0 ... +150 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT (VREF) pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	0 ... ±40 mA (each channel)	0 ... ±40 mA (each channel)		
High side reference voltage	+3.3 V ... +60 V	<ul style="list-style-type: none"> ▪ +3.3 V and +5 V (switchable) ▪ +3.3 V (fix) for the differential out interface type 	+3.3 V and +5 V (switchable)		
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull 		
	In push-pull configuration the outputs can be set to high impedance (tristate) at run time.				
Configurable fuse	–	–	–	–	
Signal termination	–	<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable ▪ Not supported for the differential out interface type 	<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable 		
Circuit diagrams	Digital In/Out 3 on page 1584	Digital In/Out 5 on page 1586	Digital Out 8 on page 1581	Digital In/Out 9 on page 1590	
Required channels	1 for each bit	1 for each bit (2 channels are required for each bit to operate the outputs using the differential out interface type.)	1 for each bit		

General limitations	Channel multiplication across several I/O boards is not supported.
DS2621 Signal Generation Board	The following restrictions apply to the DS2621 Signal Generation Board when the push/pull mode is used: <ul style="list-style-type: none">▪ Every logical signal requests two channels (primary and auxiliary) on the real-time hardware to be assigned to the function block. These channels must be adjacent to one another on an I/O board. Example: primary channel = n, auxiliary channel = n + 1.▪ Channel 10 cannot be used as the primary channel because using channel 1 as the next adjacent channel is not supported.
DS6101 Multi-I/O Board	Channel multiplication is not supported in general.
DS6121 Multi-I/O Board	Channel multiplication is not supported in general.
DS6201 Digital I/O Board	Channel multiplication is not supported in general.
DS6202 Digital I/O Board	Channel multiplication is not supported in general.
More hardware data	DS2680 I/O Unit For more board-specific data, refer to Data Sheet of the DS2680 I/O Unit (SCALEXIO Hardware Installation and Configuration) .
	DS2621 Signal Generation Board For more board-specific data, refer to Data Sheet of the DS2621 Signal Generation Board (SCALEXIO Hardware Installation and Configuration) .
	DS2690 Digital I/O Board For more board-specific data, refer to Data Sheet of the DS2690 Digital I/O Board (SCALEXIO Hardware Installation and Configuration) .
	DS6101 Multi-I/O Board For more board-specific data, refer to Data Sheet of the DS6101 Multi-I/O Board (SCALEXIO Hardware Installation and Configuration) .
	DS6121 Multi-I/O Board For more board-specific data, refer to Data Sheet of the DS6121 Multi-I/O Board (SCALEXIO Hardware Installation and Configuration) .
	DS6201 Digital I/O Board For more board-specific data, refer to Data Sheet of the DS6201 Digital I/O Board (SCALEXIO Hardware Installation and Configuration) .
	DS6202 Digital I/O Board For more board-specific data, refer to Data Sheet of the DS6202 Digital I/O Board (SCALEXIO Hardware Installation and Configuration) .

MicroAutoBox III Hardware Dependencies (Multi Bit Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 4	Digital Out 5	Digital In/Out 6	Digital Out 7	Digital In/Out 8	Digital In/Out 10
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module 	DS1554 Engine Control I/O Module			DS1521 Bus Board
Number of bits	1 ... 16	1 ... 16	1 ... 8	1 ... 16	1 ... 8	1 ... 6
Event generation	✓	✓		✓		–
Output current range	0 ... ±5 mA	0 ... ±5 mA	0 ... ±10 mA	0 ... ±5 mA	0 ... ±10 mA	-100 mA ...+50 mA
High side reference voltage	+4.5 V ... +40 V	+4.5 V ... +40 V	+5 V	+4.5 V ... +40 V	+5 V	<ul style="list-style-type: none"> ▪ +5 V ▪ VBAT
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 			<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull
	The outputs can be set to high impedance (tristate) at run time.					High impedance (tristate) is supported only for the push-pull interface type.
Configurable fuse	–	–	–	–	–	–
Circuit diagrams	Digital Out 4 on page 1578	Digital Out 5 on page 1579	Digital In/Out 6 on page 1588	Digital Out 7 on page 1580	Digital In/Out 8 on page 1589	Digital In/Out 10 on page 1591
Required channels	1	1		1		1

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to [DS1552 Multi-I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1554 Engine Control I/O Module For more module-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more board-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Digital Pulse Out

Where to go from here

Information in this section

Introduction (Digital Pulse Out).....	248
Overviews (Digital Pulse Out).....	249
Configuring the Function Block (Digital Pulse Out).....	252
Hardware Dependencies (Digital Pulse Out).....	256

Introduction (Digital Pulse Out)

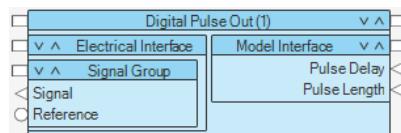
Introduction to the Function Block (Digital Pulse Out)

Function block purpose

The Digital Pulse Out function block generates a digital pulse with each model step of the behavior model or with each trigger event of another function block.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

This is the main feature:

- Generating digital pulses based on values for pulse length and a pulse delay provided from the behavior model.

Supported channel types

The Digital Pulse Out function block type supports the following channel types:

	SCALEXIO						MicroAutoBox III
Channel type	Digital Out 1	Digital Out 3	Digital Out 8	Digital In/Out 9	Digital In/Out 5	Flexible Out 1	Digital Out 4
Hardware	DS2680	DS6101	DS6121		DS6202	DS2621	<ul style="list-style-type: none">▪ DS1511▪ DS1511B1▪ DS1513

Overviews (Digital Pulse Out)

Where to go from here

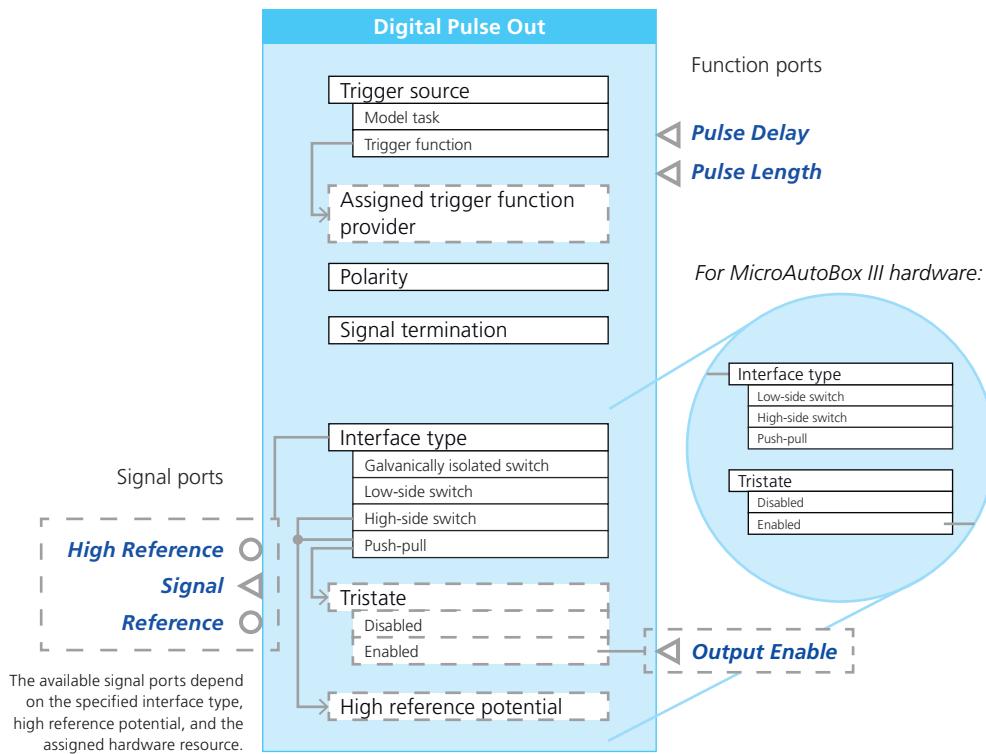
Information in this section

- | | |
|-----------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Digital Pulse Out)..... | 249 |
| Overview of Tunable Properties (Digital Pulse Out)..... | 251 |

Overview of Ports and Basic Properties (Digital Pulse Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Pulse Delay

This function import provides a time from within the behavior model to delay the generation of a digital pulse. If a trigger occurs, the pulse generation is delayed until the specified time is elapsed.

Value range	<ul style="list-style-type: none"> ▪ $T_{\min} \dots T_{\max}$ (in seconds) ▪ The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse Out) on page 256.
Dependencies	–

Pulse Length

This function import provides the pulse length of the generated digital pulses from within the behavior model.

Value range	0 s ... 1 s
Dependencies	–

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Available only if the Tristate property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switches or are in push-pull mode.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse Out) on page 256.
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse Out) on page 256 .
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse Out) on page 256 .
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Digital Pulse Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓
Electrical Interface			
	Polarity	✓	–
	Signal termination	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Digital Pulse Out)

Where to go from here

Information in this section

- | | |
|------------------------------------------------------------------------------|---------------------|
| Configuring the Basic Functionality (Digital Pulse Out)..... | 252 |
| Configuring Standard Features (Digital Pulse Out)..... | 253 |

Configuring the Basic Functionality (Digital Pulse Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

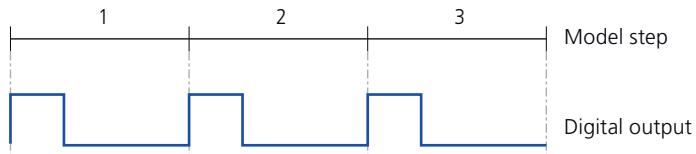
- Specifying the trigger source for pulse generation.
- Specifying signal termination.

Specifying the trigger source

You can trigger the pulse generation with the sample rate of the behavior model or via trigger events of a trigger function provider. Trigger function providers have their own defined trigger conditions.

The pulse length and a time delay of the pulse generation are provided by function ports from within the behavior model.

Triggering via model task A pulse is generated if an event triggers the model task to execute the next model step, for example, a timer event.



Triggering via trigger function A pulse is generated with each trigger event that is provided by the assigned trigger function provider.



Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the

connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

Circuit diagrams For circuit diagrams of the channel types that support signal termination, refer to:

- [Digital Out 8](#) on page 1581
- [Digital In/Out 5](#) on page 1586
- [Digital In/Out 9](#) on page 1590

Limitation Signal termination is supported by the following channel types:

- SCALEXIO hardware: Digital Out 8, Digital In/Out 5 (not supported for the differential out interface type), Digital In/Out 9
- MicroAutoBox III hardware: Signal termination is not supported.

Configuring Standard Features (Digital Pulse Out)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer . You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SCALEXIO				
Configuration Feature	Digital Out 1	Digital Out 3	Digital Out 8	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	Depending on the channel type, refer to: <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.			-
Polarity of output signals	✓	✓	✓	-

SCALEXIO				
Configuration Feature	Digital Out 1	Digital Out 3	Digital Out 8	Further Information
Signal Ports				
Trigger level of electronic fuse	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	Specifying Settings for Failure Simulation on page 123

SCALEXIO				
Configuration Feature	Digital In/Out 5	Digital In/Out 9	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	Depending on the channel type, refer to: <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112 ▪ Specifying Digital Output Signals (Flexible Out 1) on page 98
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.			–
Polarity of output signals	✓	✓	✓	–
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	–	–	✓	Specifying Settings for Failure Simulation on page 123

MicroAutoBox III		
Configuration Feature	Digital Out 4	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–
High impedance (tristate)	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.	–
Polarity of output signals	✓	–
Signal Ports		
Trigger level of electronic fuse	–	Specifying Fuse Settings on page 122
Failure simulation support	–	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Digital Out 1 on page 1575 ▪ Digital Out 3 on page 1577 ▪ Digital Out 8 on page 1581 ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 ▪ Flexible Out 1 on page 1604 	<ul style="list-style-type: none"> ▪ Digital Out 4 on page 1578

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Digital Pulse Out)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Digital Pulse Out).....	256
MicroAutoBox III Hardware Dependencies (Digital Pulse Out).....	257

SCALEXIO Hardware Dependencies (Digital Pulse Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 1	Digital Out 3	Flexible Out 1
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board	DS2621 Signal Generation Board
Pulse delay	0 ... 1 s	0 ... 1 s	0 ... 1 s
Output current range	0 ... 80 mA _{RMS}	0 ... 140 mA _{RMS}	0 ... 40 mA
High-side reference voltage	+5 V ... +60 V	+5 V ... +60 V	-60 V ... +60 V
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ Low-side switch ▪ High-side switch ▪ Push-pull
	In push-pull configuration the outputs can be set to high impedance (tristate) at run time.		
Configurable fuse	–	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Signal termination	–	–	–
Circuit diagram	Digital Out 1 on page 1575	Digital Out 3 on page 1577	Flexible Out 1 on page 1604
Required channels	1	1	1 (2 channels in push-pull mode)

Channel Type	Digital Out 8	Digital In/Out 9	Digital In/Out 5
Hardware	DS6121 Multi-I/O Board		DS6202 Digital I/O Board
Pulse delay range	0 ... 1 s		0 ... 1 s
Output current range	0 ... ±40 mA		0 ... ±40 mA
High-side reference voltage	+3.3 V and +5 V		+3.3 V and +5 V
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull
	In push-pull configuration the outputs can be set to high impedance (tristate) at run time.		
Configurable fuse	–		–

Channel Type	Digital Out 8	Digital In/Out 9	Digital In/Out 5
Signal termination	<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable 		<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable
Circuit diagram	Digital Out 8 on page 1581	Digital In/Out 9 on page 1590	Digital In/Out 5 on page 1586
Required channels	1		1

General limitations

Channel multiplication is not supported in general.

DS2621 Signal Generation Board The following restrictions apply to the DS2621 Signal Generation Board when the push/pull mode is used:

- Every logical signal requests two channels (primary and auxiliary) on the real-time hardware to be assigned to the function block. These channels must be adjacent to one another on an I/O board. Example: primary channel = n, auxiliary channel = n + 1.
- Channel 10 cannot be used as the primary channel because using channel 1 as the next adjacent channel is not supported.

DS6202 Digital I/O Board Only 4 Digital Pulse Out function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (Digital Pulse Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 4
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board
Pulse delay	0 ... 0.5 s
Output current range	0 ... ±5 mA
High side reference voltage	+4.5 V ... +40 V
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull <p>The outputs can be set to high impedance (tristate) at run time.</p>
Circuit diagrams	Digital Out 4 on page 1578
Required channels	1

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Potentiometer Out

Where to go from here

Information in this section

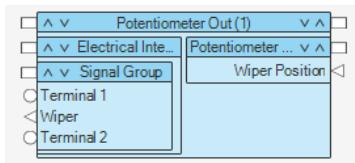
Introduction (Potentiometer Out).....	260
Overviews (Potentiometer Out).....	261
Configuring the Function Block (Potentiometer Out).....	263
Hardware Dependencies (Potentiometer Out).....	266

Introduction (Potentiometer Out)

Introduction to the Function Block (Potentiometer Out)

Function block purpose The Potentiometer Out function block type allows adjustable three-terminal resistances (potentiometers) to be simulated.

Default display The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features These are the main features:

- Simulation of a potentiometer providing the wiper position value from the behavior model.

Supported channel types The Potentiometer Out function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Resistance Out 1	Resistance Out 2	Flexible Out 1	-
Hardware	DS2680	DS6101	DS2621	-

Overviews (Potentiometer Out)

Where to go from here

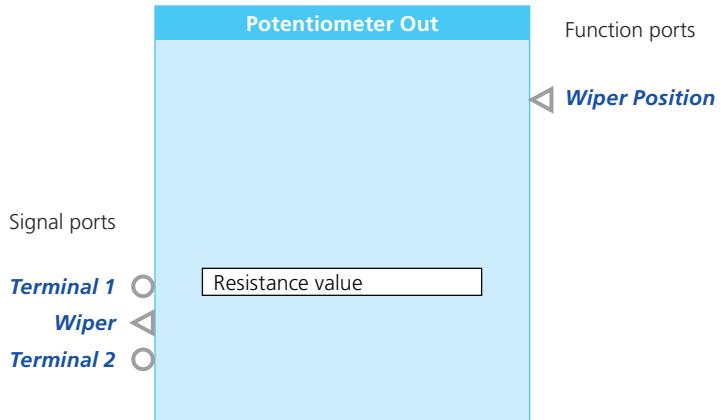
Information in this section

- [Overview of Ports and Basic Properties \(Potentiometer Out\).....](#) 261
- [Overview of Tunable Properties \(Potentiometer Out\).....](#) 262

Overview of Ports and Basic Properties (Potentiometer Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Wiper Position

This function import lets you define the output signal at the Wiper signal port from within the behavior model. You can set a percentage value relating to the Resistance value that is available between the Terminal 1 and Terminal 2 signal ports.

Value range	<ul style="list-style-type: none"> ▪ 0 % ... 100 % ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	-

Wiper

This signal port provides the percentage value for the resistance value which is set via the Resistance value property related to the Terminal 2 signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Potentiometer Out) on page 266 .
Dependencies	–

Terminal 1/Terminal 2

These signal ports are reference ports and provide the references for the Wiper signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Potentiometer Out) on page 266
Dependencies	–

Overview of Tunable Properties (Potentiometer Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Resistance value	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Potentiometer Out)

Where to go from here

Information in this section

- | | |
|------------------------------------------------------------------------------|---------------------|
| Configuring the Basic Functionality (Potentiometer Out)..... | 263 |
| Configuring Standard Features (Potentiometer Out)..... | 264 |

Configuring the Basic Functionality (Potentiometer Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

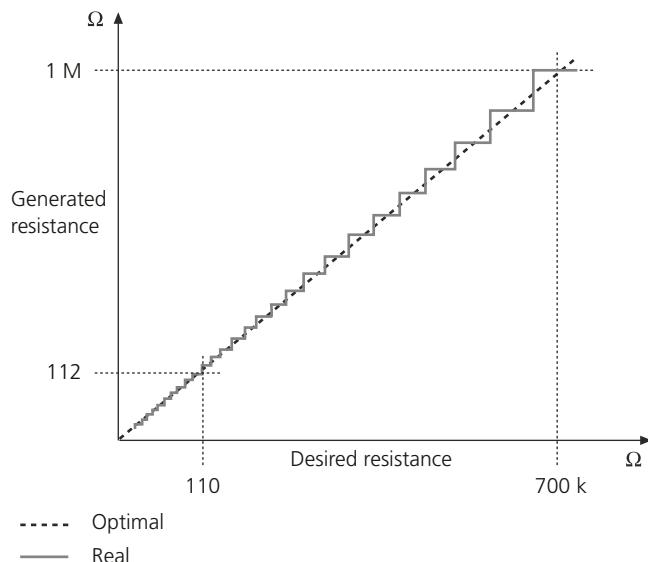
- Specifying the resistance value

Specifying resistance value

You can specify the resistance of the function block in Ohm. The possible resistance values and the accuracy depend on the assigned hardware resource.

Example The DS2621 Signal Generation Board internally connects single selected resistors in parallel to get the desired resistance output.

This can cause quantization effects as shown in the simplified illustration below. Thus, the value you set can differ from the real resistance value the hardware is able to provide.



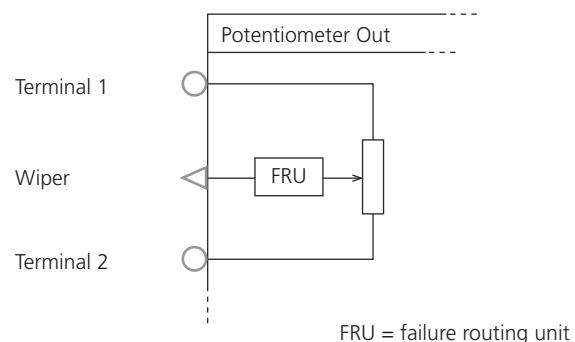
Configuring Standard Features (Potentiometer Out)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.			
Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.			
Configuration Feature				
Channel multiplication	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Resistance Out 1 on page 1619](#)
- [Resistance Out 2 on page 1620](#)
- [Flexible Out 1 on page 1604](#)

Internal connection of signal ports The following illustration shows the internal connection of signal ports and the failure routing unit (FRU) in the Wiper branch.



Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Potentiometer Out)

SCALEXIO Hardware Dependencies (Potentiometer Out)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Resistance Out 1	Resistance Out 2	Flexible Out 1
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board	DS2621 Signal Generation Board
Resistance range	34 Ω ... 1 MΩ	34 Ω ... 1 MΩ	36 Ω ... 1 MΩ
Resolution of conductance	16 bit	16 bit	18 bit ¹⁾
Accuracy	±2 % (typ.)	±2 % (typ.)	±1 % (typ.)
Input current range	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW}/R)}$	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW}/R)}$	0 ... +40 mA _{RMS}
Input voltage range	-3 V ... +18 V	-3 V ... +18 V	-20 V ... +20 V for 2 channels -60 V ... +60 V for 6 channels (channel multiplication)
Configurable fuse	–	–	▪ Load side ▪ 5 mA _{RMS} ... 40 mA _{RMS}
Circuit diagrams	Resistance Out 1 on page 1619	Resistance Out 2 on page 1620	Flexible Out 1 on page 1604
Required channels	2	2	2 (additional channels are required for voltage enhancements.)

¹⁾ (1/4 MΩ) per bit

General limitations

Channel multiplication across several I/O boards is not supported.

DS2680 I/O Unit The Potentiometer Out function block does not support channel multiplication on the DS2680.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Resistance Out

Where to go from here

Information in this section

Introduction (Resistance Out).....	268
Overviews (Resistance Out).....	269
Configuring the Function Block (Resistance Out).....	272
Hardware Dependencies (Resistance Out).....	274

Introduction (Resistance Out)

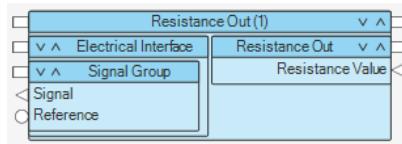
Introduction to the Function Block (Resistance Out)

Function block purpose

The Resistance Out function block provides the possibility to simulate resistances. The resistance value is provided from within the behavior model.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main feature

This is the main feature:

- Simulation of a resistance value provided from the behavior model.

Supported channel types

The Resistance Out function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Resistance Out 1	Resistance Out 2	Flexible Out 1	–
Hardware	DS2680	DS6101	DS2621	–

Overviews (Resistance Out)

Where to go from here

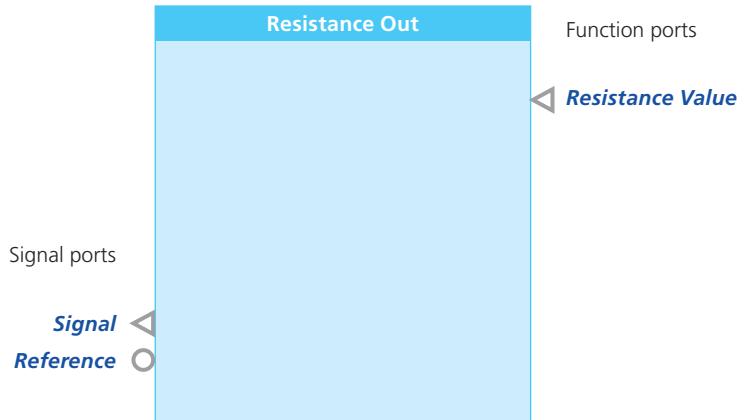
Information in this section

- | | |
|------------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Resistance Out)..... | 269 |
| Overview of Tunable Properties (Resistance Out)..... | 271 |

Overview of Ports and Basic Properties (Resistance Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



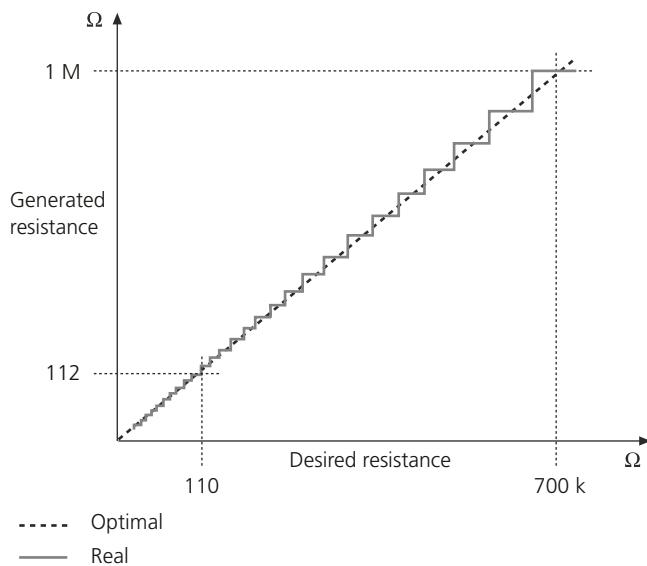
For this function block, there are no properties that influence the basic functionality.

Resistance Value

This function import lets you define the resistance value from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $R_{min} \dots R_{max}$ (in Ohm). ▪ Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Resistance Out) on page 274.
Dependencies	–

Signal	This signal port represents the electrical connection point of the logical signal of the function block.
Value range	<p>The generated resistance values and the accuracy depend on the assigned hardware resource.</p> <p>For example: The DS2621 Signal Generation Board internally connects single selected resistors in parallel to get the desired resistance output. This can cause quantization effects as shown in the following simplified illustration. Thus, the value provided from the behavior model might differ from the real resistance value the hardware provides.</p> <p>For specific values, refer to Hardware Dependencies (Resistance Out) on page 274.</p>
Dependencies	–



Reference	This signal port is a reference port and provides the reference signal for the Signal signal port.
Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Resistance Out) on page 274.
Dependencies	–

Overview of Tunable Properties (Resistance Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Resistance Out)

Configuring Standard Features (Resistance Out)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

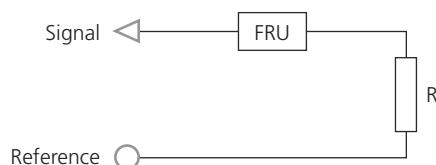
Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Configuration Feature	Resistance Out 1	Resistance Out 2	Flexible Out 1	Further Information
Channel multiplication	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Resistance Out 1 on page 1619](#)
- [Resistance Out 2 on page 1620](#)
- [Flexible Out 1 on page 1604](#)

Internal connection of signal ports The following illustration shows the internal connection of signal ports and the failure routing unit (FRU) in the Signal branch.



Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Resistance Out)

SCALEXIO Hardware Dependencies (Resistance Out)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Resistance Out 1	Resistance Out 2	Flexible Out 1
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board	DS2621 Signal Generation Board
Resistance range	17 Ω ... 1 MΩ	17 Ω ... 1 MΩ	18 Ω ... 1 MΩ
Resolution of conductance	16 bit	16 bit	18 bit ¹⁾
Accuracy	±2 % (typ.)	±2 % (typ.)	±1 % (typ.)
Input current range	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW}/R)}$	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW}/R)}$	0 ... +40 mA _{RMS}
Input voltage range	-3 V ... +18 V	-3 V ... +18 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication)
Configurable fuse	–	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Circuit diagrams	Resistance Out 1 on page 1619	Resistance Out 2 on page 1620	Flexible Out 1 on page 1604
Required channels	1	1	1 (additional channels are required for voltage enhancements.)

¹⁾ (1/4 MΩ) per bit

General limitations

Channel multiplication across several I/O boards is not supported.

DS2680 I/O Unit The Resistance Out function block does not support channel multiplication on the DS2680.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

PWM/PFM Out

Where to go from here

Information in this section

Introduction (PWM/PFM Out).....	276
Overviews (PWM/PFM Out).....	279
Configuring the Function Block (PWM/PFM Out).....	283
Hardware Dependencies (PWM/PFM Out).....	288

Introduction (PWM/PFM Out)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------------------|---------------------|
| Introduction to the Function Block (PWM/PFM Out)..... | 276 |
| Basics on PWM/PFM Signals..... | 277 |

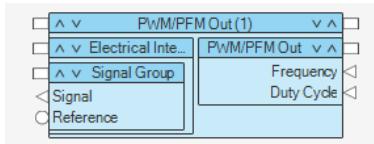
Introduction to the Function Block (PWM/PFM Out)

Function block purpose

The PWM/PFM Out function block type can be used to generate one-phase pulse-width-modulated signals.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating a PWM or PFM signal based on values for frequency and duty cycle provided from the behavior model.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The PWM/PFM Out function block type supports the following channel types:

	SCALEXIO								
Channel type	Digital Out 1	Digital Out 2	Digital In/Out 1	Digital Out 3	Digital In/Out 3	Digital In/Out 5	Digital Out 8	Digital In/Out 9	Flexible Out 1
Hardware	DS2680	DS2690		DS6101	DS6201	DS6202	DS6121		DS2621

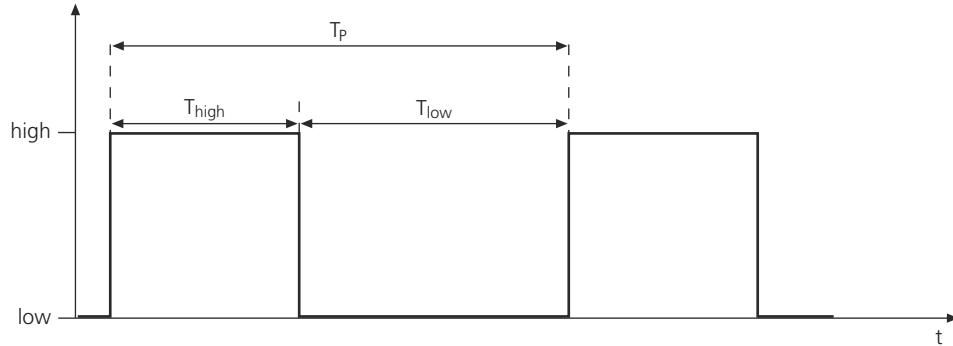
	MicroAutoBox III						
Channel type	Digital Out 4	Digital Out 5	Digital In/Out 6	Digital Out 7	Digital In/Out 8	Digital In/Out 10	
Hardware	▪ DS1511 ▪ DS1552 ▪ DS1511B1			DS1554		DS1521	

	MicroAutoBox III			
	▪ DS1513			

Basics on PWM/PFM Signals

Purpose	You can generate pulse-width modulated (PWM) or pulse-frequency modulated (PFM) signals with the PWM/PFM Out function block.
----------------	------------------------------------------------------------------------------------------------------------------------------

Terms and definitions	A square-wave signal is specified by the following properties: <ul style="list-style-type: none"> ▪ T_{high}: Duration of a pulse that starts with a rising edge ▪ T_{low}: Duration of a pulse that starts with a falling edge ▪ T_{active}: Duration of the pulse with which the period starts. If the period starts with a rising edge, the signal has an active high pulse, if it starts with a falling edge, it has an active low pulse. ▪ T_p: Duration of a period as the sum of the high and the low pulses $T_p = T_{high} + T_{low}$
------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



Calculation formulas	The PWM/PFM Out function block generates the frequency and the pulse-to-period ratio:
-----------------------------	---------------------------------------------------------------------------------------

- Frequency (f): The frequency is the inverse of the period.

$$f = 1 / T_p$$

- Duty cycle (d): The duty cycle is the pulse-to-period ratio of a period.

$$d = T_{active} / T_p$$

The signal characteristics are as follows:

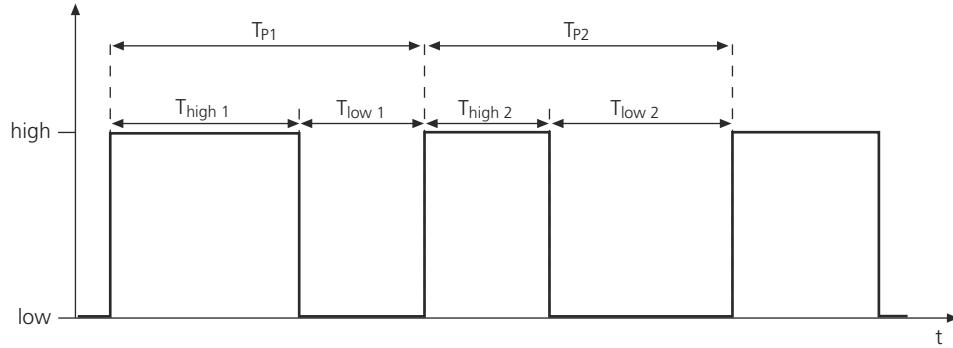
- Pulse-width modulated (PWM) signals: Signals with a constant frequency ($f_2 = f_1$)

- Pulse-frequency modulated (PFM) signals: Signals with a constant duty cycle ($d_2 = d_1$)
- Irregular square-wave signals with no constant frequency or duty cycle

The signal characteristics are described in detail below.

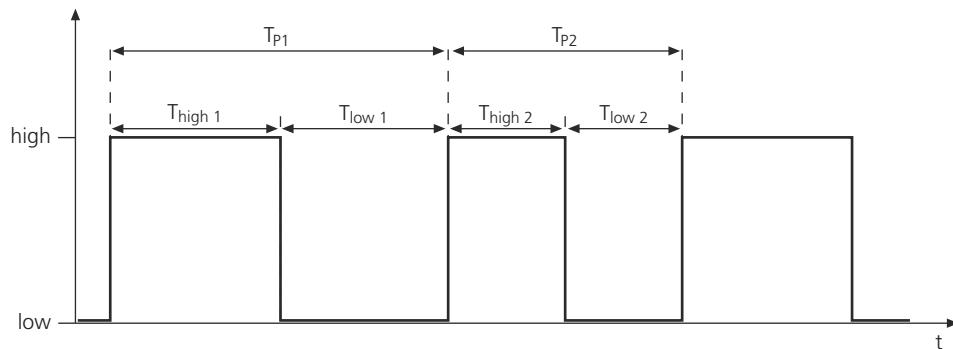
Characteristics of a PWM signal

PWM signals have a constant frequency, while the duty cycle can vary to control the signal information. This means that $1/T_{P2} = 1/T_{P1}$, in other words, the period duration T_p as the sum of $T_{high} + T_{low}$ is constant in the signal shape.



Characteristics of a PFM signal

PFM signals have a constant duty cycle, while the frequency can vary to control the signal information. This means that $d_2 = d_1$, in other words, the ratio of T_{active} / T_p is constant in the signal shape. For example, if the period starts with a rising edge, the ratio of $T_{high} / (T_{high} + T_{low})$ is constant in the signal shape.



Overviews (PWM/PFM Out)

Where to go from here

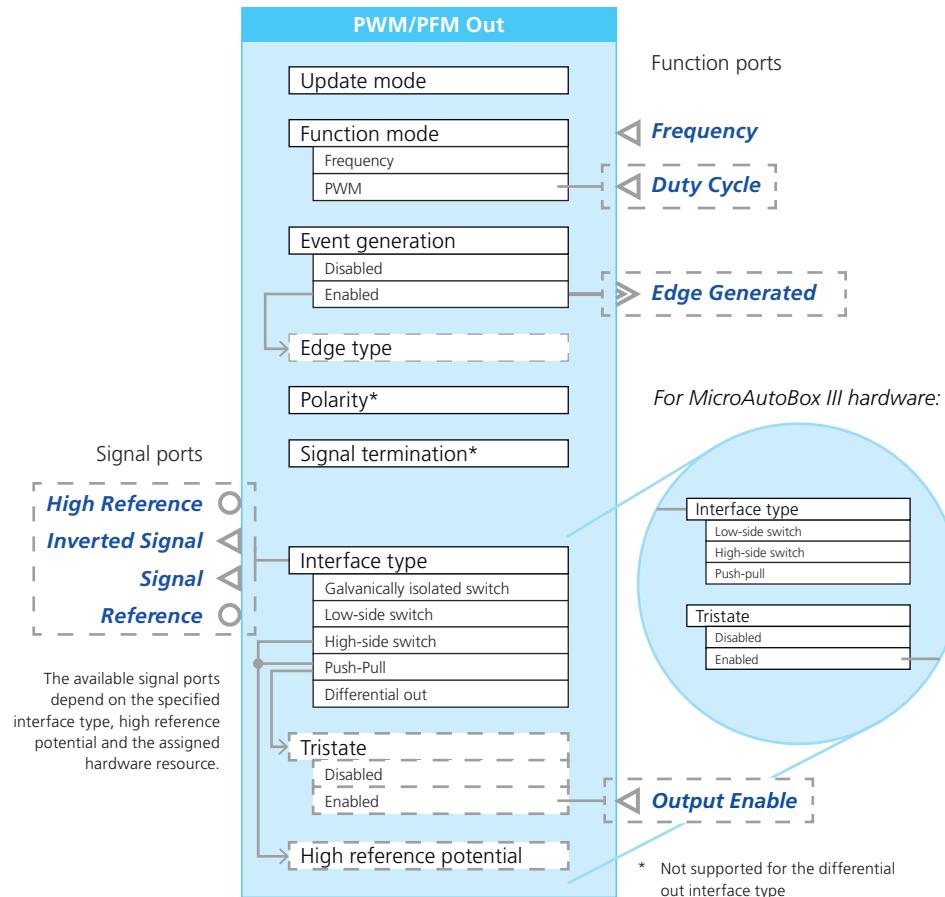
Information in this section

- | | |
|-----------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (PWM/PFM Out)..... | 279 |
| Tunable Properties (PWM/PFM Out)..... | 282 |

Overview of Ports and Basic Properties (PWM/PFM Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Frequency

This function import lets you define the frequency of the output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $f_{\min} \dots f_{\max}$ (in Hertz). ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM Out) on page 288. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range. ▪ 0 Hz: The signal generation stops. ▪ In synchronous update mode: After the current period ends, the output signal is set to the inactive level. ▪ In asynchronous update mode: The output signal keeps the current level. <p>Note that the signal saturation does not stop if the user saturation saturates the minimum frequency to a value greater than 0 Hz.</p>
Dependencies	–

Duty Cycle

This function import lets you define the duty-cycle value of the PWM signal from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0% ... 100% ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	Available only if the Function mode property is set to PWM .

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Available only if the Tristate property is set to Enabled.

Edge Generated

This event port provides an I/O event each time an edge is generated during signal generation and this edge matches the trigger condition.

You have to specify which type of edge of the output signal (rising or falling edge or both) is relevant for event generation (via the Edge type property).

Value range	–
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Not supported ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Available only if the Event generation property is set to Enabled. ▪ Not supported for the Digital In/Out 10 channel type.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM Out) on page 288.
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM Out) on page 288.
Dependencies	–

Inverted Signal

This signal port and the Signal signal port together represent the electrical connection points of a logical signal with two complementary potentials carrying the information in the voltage difference between these signals. Both signal ports (Signal and Inverted Signal) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM Out) on page 288.
Dependencies	Available only if the Interface type property is set to Differential out.

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM Out) on page 288.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Tunable Properties (PWM/PFM Out)

Frequency

This function import lets you define the frequency of the output signal from within the behavior model.

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Properties	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Update mode	✓	-
Edge type	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Polarity	✓	-
Signal termination	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (PWM/PFM Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (PWM/PFM Out).....	283
Configuring Standard Features (PWM/PFM Out).....	284

Configuring the Basic Functionality (PWM/PFM Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Choosing PWM or square-wave output.
- Specifying the update mode for generated values (synchronous or asynchronous).
- Specifying the generation of I/O events.
- Specifying signal termination.

Choosing PWM or square-wave output

Via the Function mode property, you can specify whether you can influence the frequency and the duty cycle of your output signal or just the frequency from within the behavior model. In the latter case the PWM/PFM Out function block generates a pure square-wave signal, because the duty cycle is fixed to 50%.

Specifying the update mode for generated values

The function provides the Update mode property to specify if the generated values are written synchronously or asynchronously:

- In synchronous update mode, the new frequency and duty-cycle values become valid when a new PWM period starts.
- In asynchronous update mode, the new frequency and duty-cycle values immediately become valid during execution of a PWM period.

Note

The asynchronous update of the duty cycle (with constant frequency) can cause a single PWM period with a duty cycle between the old and the new duty-cycle value and briefly causes undefined states of the PWM signal.

Providing I/O events	<p>If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.</p> <p>The PWM/PFM Out function block can generate an I/O event each time an edge is generated during signal generation and this edge matches the setting of the Edge type property (rising or falling edge or both edges).</p> <p>To use the I/O events in the behavior model, you have to assign them to a task via the Edge Generated property. For more information, refer to Modeling Asynchronous Tasks (ConfigurationDesk Real-Time Implementation Guide).</p> <p>Limitation Event generation is not supported for the following channel types:</p> <ul style="list-style-type: none"> ▪ SCALEXIO hardware: All channel types ▪ MicroAutoBox III hardware: Digital In/Out 10
Enabling signal termination	<p>To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.</p> <p>Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.</p> <p>Circuit diagrams For circuit diagrams of the channel types that support signal termination, refer to:</p> <ul style="list-style-type: none"> ▪ Digital Out 8 on page 1581 ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 <p>Limitation Signal termination is supported by the following channel types:</p> <ul style="list-style-type: none"> ▪ SCALEXIO hardware: Digital Out 8, Digital In/Out 5 (not supported for the differential out interface type), Digital In/Out 9 ▪ MicroAutoBox III hardware: Signal termination is not supported.

Configuring Standard Features (PWM/PFM Out)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows

only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO					
Configuration Feature	Digital Out 1	Digital Out 2	Digital Out 3	Digital Out 8	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	Depending on the channel type, refer to: <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 2, Digital In/Out 1) on page 106 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.				–
Polarity of output signals	✓	✓	✓	✓	–
Channel multiplication	✓	✓	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports					
Trigger level of electronic fuse	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	–	Specifying Settings for Failure Simulation on page 123

SCALEXIO						
Configuration Feature	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	Depending on the channel type, refer to: <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 2, Digital In/Out 1) on page 106 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109 ▪ Specifying Digital Output Signals

SCALEXIO						
Configuration Feature	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9	Flexible Out 1	Further Information
						(Digital In/Out 5) on page 112 ▪ Specifying Digital Output Signals (Flexible Out 1) on page 98
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.					-
Polarity of output signals	✓	✓	▪ ✓ ▪ Not supported for the differential out interface type.	✓	✓	-
Channel multiplication	✓	-	-	-	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	-	-	-	-	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	-	-	-	✓	Specifying Settings for Failure Simulation on page 123

MicroAutoBox III							
Configuration Feature	Digital Out 4	Digital Out 5	Digital Out 7	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10	Further Information
Interface type	▪ High-side switch ▪ Low-side switch ▪ Push-pull	▪ High-side switch ▪ Low-side switch ▪ Push-pull	▪ High-side switch ▪ Low-side switch ▪ Push-pull	▪ High-side switch ▪ Low-side switch ▪ Push-pull	▪ High-side switch ▪ Low-side switch ▪ Push-pull	▪ High-side switch ▪ Low-side switch ▪ Push-pull	-
High impedance (tristate)	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.					High impedance (tristate) is supported only for the push-pull interface type.	-
Polarity of output signals	✓	✓	✓	✓	✓	✓	-
Channel multiplication	-	-	-	-	-	-	Specifying Current and Voltage Values for Channel

MicroAutoBox III							
Configuration Feature	Digital Out 4	Digital Out 5	Digital Out 7	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10	Further Information
							Multiplication on page 95
Signal Ports							
Trigger level of electronic fuse	–	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	–	–	–	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Digital Out 1 on page 1575 ▪ Digital Out 2 on page 1576 ▪ Digital Out 3 on page 1577 ▪ Digital Out 8 on page 1581 ▪ Digital In/Out 1 on page 1582 ▪ Digital In/Out 3 on page 1584 ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 ▪ Flexible Out 1 on page 1604 	<ul style="list-style-type: none"> ▪ Digital Out 4 on page 1578 ▪ Digital Out 5 on page 1579 ▪ Digital Out 7 on page 1580 ▪ Digital In/Out 6 on page 1588 ▪ Digital In/Out 8 on page 1589 ▪ Digital In/Out 10 on page 1591

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (PWM/PFM Out)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (PWM/PFM Out).....	288
MicroAutoBox III Hardware Dependencies (PWM/PFM Out).....	291

SCALEXIO Hardware Dependencies (PWM/PFM Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital Out 1	Flexible Out 1	Digital Out 2	Digital In/Out 1	Digital Out 3
Hardware		DS2680 I/O Unit	DS2621 Signal Generation Board	DS2690 Digital I/O Board	DS6101 Multi-I/O Board	
Event generation		–	–	–	–	
Frequency	Range	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 150 kHz ▪ Frequency mode: 0 ... 1 MHz 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 150 kHz ▪ Frequency mode: 0 ... 1.838235 MHz 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 125 kHz ▪ Frequency mode: 0 ... 1 MHz 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 150 kHz ▪ Frequency mode: 0 ... 500 kHz ¹⁾ 	
	Time resolution	68 ns	68 ns	80 ns	68 ns	
Interface type for digital outputs		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	
In push-pull configuration the outputs can be set to high impedance (tristate) at run time.						
Output current range		<ul style="list-style-type: none"> ▪ 0 ... +80 mA_{RMS} for 1 channel ▪ 1.792 A_{RMS} for 28 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +40 mA_{RMS} for 1 channel ▪ 0.32 A_{RMS} for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +80 mA_{RMS} for 1 channel ▪ 0.64 A_{RMS} for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +150 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 2	Digital In/Out 1	Digital Out 3
High side reference voltage	+5 V ... +60 V	-60 V ... +60 V	+5 V ... +60 V		+5 V ... +60 V
Configurable fuse	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–		–
Signal termination	–	–	–		–
Circuit diagrams	Digital Out 1 on page 1575	Flexible Out 1 on page 1604	Digital Out 2 on page 1576	Digital In/Out 1 on page 1582	Digital Out 3 on page 1577
Required channels	1 (additional channels are required for current enhancement.)	1 (additional channels are required for current enhancement and operation in push/pull mode.)	1 (additional channels are required for current enhancement.)	1	

¹⁾ For frequencies above 125 kHz the maximum output voltage has to be limited to 30 V.

Channel Type	Digital In/Out 3	Digital In/Out 5	Digital Out 8	Digital In/Out 9
Hardware	DS6201 Digital I/O Board	DS6202 Digital I/O Board	DS6121 Multi-I/O Board	
Event generation	–	–	–	
Frequency	Range	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 125 kHz ▪ Frequency mode: 0 ... 500 kHz ¹⁾ 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 1.25 MHz ▪ Frequency mode: 0 ... 20 MHz 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 1.25 MHz ▪ Frequency mode: 0 ... 20 MHz
	Time resolution	80 ns	8 ns	8 ns
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull 	In push-pull configuration the outputs can be set to high impedance (tristate) at run time.
Output current range	<ul style="list-style-type: none"> ▪ 0 ... +140 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT (VREF) pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	0 ... ±40 mA (each channel)	0 ... ±40 mA	
High side reference voltage	+3.3 V ... +60 V	<ul style="list-style-type: none"> ▪ +3.3 V and +5 V (switchable) ▪ +3.3 V (fix) for the differential out interface type 	+3.3 V and +5 V (switchable)	

Channel Type	Digital In/Out 3	Digital In/Out 5	Digital Out 8	Digital In/Out 9
Configurable fuse	–	–	–	
Signal termination	–	<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable ▪ Not supported for the differential out interface type 	<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable 	
Circuit diagrams	Digital In/Out 3 on page 1584	Digital In/Out 5 on page 1586	Digital Out 8 on page 1581	Digital In/Out 9 on page 1590
Required channels	1	1 (2 channels are required to operate the outputs using the differential out interface type.)	1	

¹⁾ For frequencies above 125 kHz the maximum output voltage has to be limited to 30 V.

General limitations

Channel multiplication across several I/O boards is not supported.

DS2621 Signal Generation Board The following restrictions apply to the DS2621 Signal Generation Board when the push/pull mode is used:

- Every logical signal requests two channels (primary and auxiliary) on the real-time hardware to be assigned to the function block. These channels must be adjacent to one another on an I/O board. Example: primary channel = n, auxiliary channel = n + 1.
- Channel 10 cannot be used as the primary channel because using channel 1 as the next adjacent channel is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6121 Multi-I/O Board Channel multiplication is not supported in general.

DS6201 Digital I/O Board Channel multiplication is not supported in general.

DS6202 Digital I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2690 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS2690 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6101 Multi-I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6121 Multi-I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6201 Digital I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6201 Digital I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6202 Digital I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

MicroAutoBox III Hardware Dependencies (PWM/PFM Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 4	Digital Out 5	Digital In/Out 6	Digital Out 7	Digital In/Out 8	Digital In/Out 10
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module 	DS1554 Engine Control I/O Module		DS1521 Bus Board	
Event generation	✓	✓		✓		–
Frequency	Range	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 150 kHz ▪ Frequency mode: 0 ... 150 kHz 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 150 kHz ▪ Frequency mode: 0 ... 150 kHz 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 150 kHz ▪ Frequency mode: 0 ... 150 kHz 	<ul style="list-style-type: none"> ▪ PWM mode: 0 ... 20 kHz ▪ Frequency mode: 0 ... 250 kHz 	
	Time resolution	12.5 ns	12.5 ns	12.5 ns	8 ns	
Interface type for digital outputs		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	
		The outputs can be set to high impedance (tristate) at run time.			High impedance (tristate) is supported only for the push-pull interface type.	

Channel Type	Digital Out 4	Digital Out 5	Digital In/Out 6	Digital Out 7	Digital In/Out 8	Digital In/Out 10
Output current range	0 ... ±5 mA	0 ... ±5 mA	0 ... ±10 mA	0 ... ±5 mA	0 ... ±10 mA	-100 mA ...+50 mA
High side reference voltage	+4.5 V ... +40 V	+4.5 V ... +40 V	+5 V	+4.5 V ... +40 V	+5 V	▪ +5 V ▪ VBAT
Configurable fuse	–	–	–	–	–	–
Circuit diagrams	Digital Out 4 on page 1578	Digital Out 5 on page 1579	Digital In/Out 6 on page 1588	Digital Out 7 on page 1580	Digital In/Out 8 on page 1589	Digital In/Out 10 on page 1591
Required channels	1	1	1	1	1	1

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to [DS1552 Multi-I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1554 Engine Control I/O Module For more module-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more board-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Multi-Channel PWM Out

Where to go from here

Information in this section

Introduction to the Functionality (Multi-Channel PWM Out).....	294
Overviews (Multi-Channel PWM Out).....	298
Configuring the Function Block (Multi-Channel PWM Out).....	304
Hardware Dependencies (Multi-Channel PWM Out).....	311

Introduction to the Functionality (Multi-Channel PWM Out)

Where to go from here

Information in this section

Introduction to the Function Block (Multi-Channel PWM Out).....	294
Basics on Multi-Channel PWM Signals.....	295

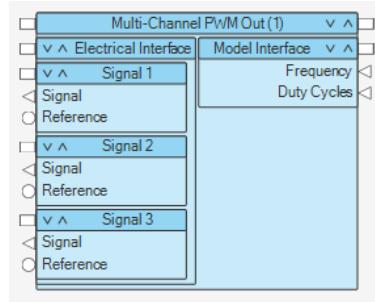
Introduction to the Function Block (Multi-Channel PWM Out)

Function block purpose

The Multi-Channel PWM Out function block type synchronously generates multiple PWM signals with a common frequency. The function block can work as a provider: Other function blocks can use the generated trigger signal as a trigger source.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating PWM signals on multiple channels.
- Receiving values for duty cycles and frequency from the behavior model.
- Providing trigger events to be used as trigger source by other function blocks.
- Providing I/O events and function triggers.

Available demo project

ConfigurationDesk provides the *EDrivesControlDemo* project. This project is an example for electric motor control using the DS6121 Multi-I/O Board. You can use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to [EDrivesControlDemo Project: Example of](#)

[Electric Motor Control Using the DS6121 Multi-I/O Board \(ConfigurationDesk Demo Projects\)](#).

Supported channel types

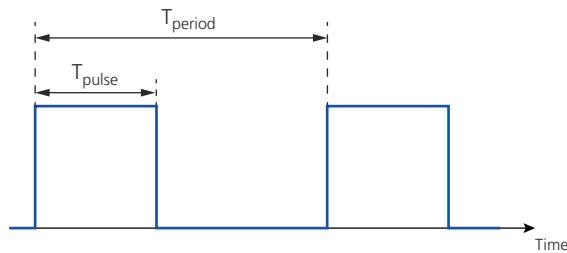
The Sent In function block supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Digital Out 8	Digital In/Out 9	Digital In/Out 5	Digital Out 4
Hardware	DS6121		DS6202	<ul style="list-style-type: none"> ▪ DS1511 ▪ DS1511B1 ▪ DS1513

Basics on Multi-Channel PWM Signals

PWM signals in general

PWM signals are square-wave signals with a constant frequency. The main use of PWM signals is the power control of motors and inverters. A PWM signal is specified by the following characteristics:



Symbol	Definition
T_{pulse}	Pulse width
T_{period}	Period time

Modulation technique PWM signals are pulse width modulated signals. The pulse width is expressed by the duty cycle, which is independent of the period time. The duty cycle is the ratio of the pulse duration to the period time:

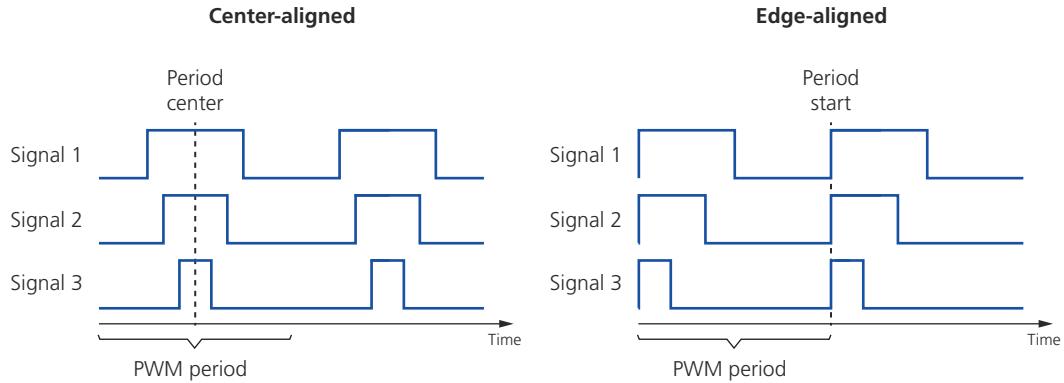
Symbol	Definition
D	$D = T_{\text{pulse}} / T_{\text{period}}$

Particular characteristics of multi-channel PWM signals

Particular characteristics of the the PWM signals generated with the Multi-Channel PWM Out function block:

- All PWM signals have the same period time/frequency.
- All PWM signals start at the same point in time.

These characteristics allow the alignment of the PWM signal as shown below.



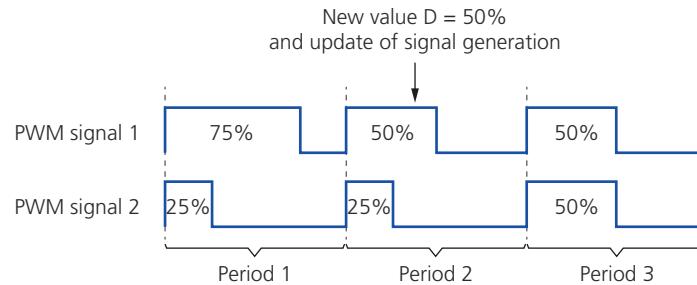
Center-aligned PWM signals are commonly used in multi-phase motor applications because this alignment reduces the generation of harmonics.

Updating the duty cycle of PWM signals

The point in time of a duty cycle update is essential for the generated PWM signals. The following illustrations show the basic principle and the resulting PWM signals of different update strategies.

- Strategy 1:

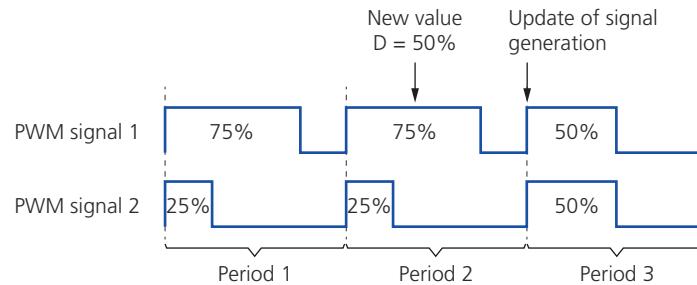
The signal values are updated immediately.



Disadvantage: PWM signals might be updated within different signal periods.
For example: The pulse of PWM signal 2 is already generated. The new duty cycle value can be generated only with the next signal period.

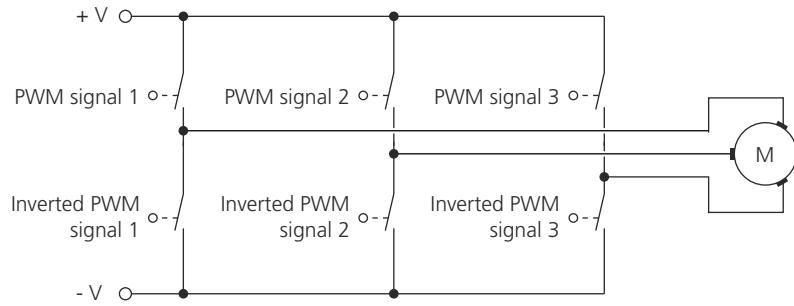
- Strategy 2:

The signal values are updated at a certain point in time when all PWM signals can be updated, for example, at the beginning of the next signal period.



Use scenario: Controlling half bridges

Multi-channel PWM signals can be used, for example, to control half bridges that drive electric motors. Each half bridge is controlled by a PWM signal and an inverted PWM signal, as shown in the following example of a three-phase motor driver with three half bridges.



Oversviews (Multi-Channel PWM Out)

Where to go from here

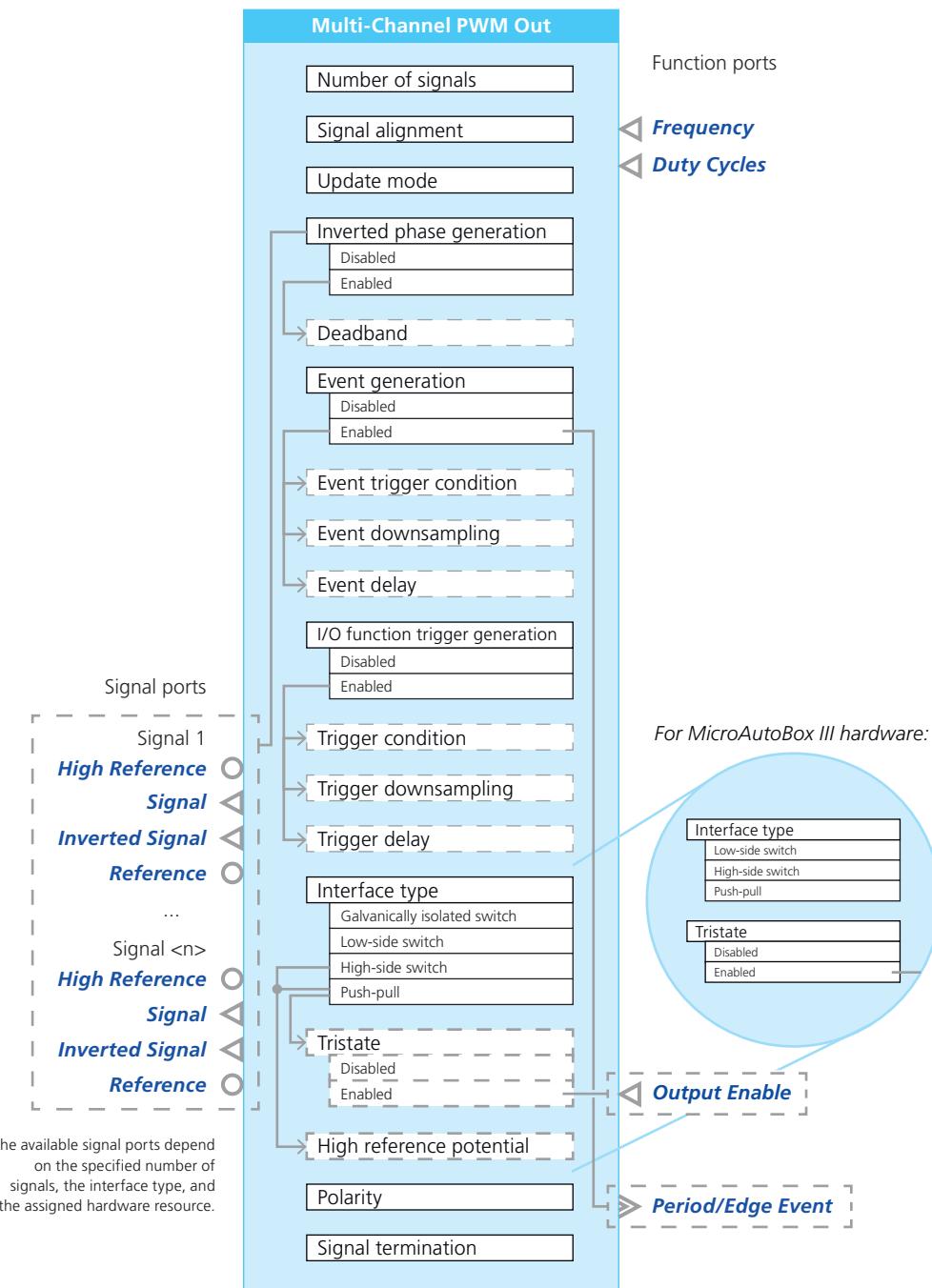
Information in this section

Overview of Ports and Basic Properties (Multi-Channel PWM Out)	298
Overview of Tunable Properties (Multi-Channel PWM Out).....	303

Overview of Ports and Basic Properties (Multi-Channel PWM Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Frequency**

This function import reads the common frequency value for all PWM signals from within the behavior model to use it for the generation of the PWM signals.

Value range	<ul style="list-style-type: none"> ▪ $f_{\min} \dots f_{\max}$ (in Hertz). ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi-Channel PWM Out) on page 311. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ 0 Hz: The signal generation will stop after the next update. Center-aligned signals keep the updated levels. The levels of edge-aligned signals depend on the update mode. Note that the signal generation does not stop if the user saturation saturates the minimum frequency to a value greater than 0 Hz. <p>For the generated output signals, refer to the following table.</p>
Dependencies	–

The following table shows the output signals if the signal generation stops:

Signal Alignment	Update Mode	Generated Output Signals at 0 Hz
Center-aligned	Period start	After the current period ends, the signals are set to the following levels: <ul style="list-style-type: none"> ▪ Duty cycle < 100 %: The non-inverted signals are set to the inactive level and inverted signals to the active level. ▪ Duty cycle = 100 %: The non-inverted signals are set to the active level and inverted signals to the inactive level.
	Period center	At the next period center, the signals are set to the following levels: <ul style="list-style-type: none"> ▪ Duty cycle > 0 %: The non-inverted signals are set to the active level and inverted signals to the inactive level. ▪ Duty cycle = 0 %: The non-inverted signals are set to the inactive level and inverted signals to the active level.
	Period start and center	Depends on the moment the next update of the signal generation takes place. Refer to <i>period start</i> and <i>period center</i> .
Edge-aligned	Synchronous	After the current period ends, the non-inverted signals are set to the inactive level and inverted signals are set to the active level.
	Asynchronous	All signals keep the current levels.

Duty Cycles

This function import reads a vector with duty-cycle values from within the behavior model to use it for the generation of PWM signals.

Value range	▪ 0 % ... 100 % for each entry of the vector.
-------------	-----------------------------------------------

	<ul style="list-style-type: none"> The vector size depends on the number specified at the Number of signals property. If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> 0: Disabled 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled .

Period/Edge Event

This event port provides an I/O event each time at period start/center or at a certain edge of the first PWM signal.

You have to specify the following settings for the trigger condition:

- The trigger point (via **Event trigger condition** property).
- The downsampling of occurred triggers (via **Event downsampling** property).
- The delaying of the I/O event generation (via **Event delay** property).

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled .

High Reference

This signal port is a reference port and provides the high-side reference for the **Signal** and the **Inverted Signal** signal ports when the digital outputs are operating as high-side switches or are in push-pull mode.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi-Channel PWM Out) on page 311.
Dependencies	<ul style="list-style-type: none"> Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource. Not available for Digital InOut 5 channel type. This channel type provides internal high reference voltages. The number of High Reference ports depends on the value specified at the Number of signals property.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi-Channel PWM Out) on page 311.
Dependencies	The number of Signal ports depends on the value specified at the Number of signals property.

Inverted Signal

This signal port represents the electrical connection point of the logical inverted signal of the function block. The inverted signal is the inverted signal to the Signal signal port and provides the complementary signal to control an inverter bridge, for example.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi-Channel PWM Out) on page 311.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Inverted phase generation property is set to Enabled. ▪ The number of Inverted Signal ports depends on the value specified at the Number of signals property.

Reference

This signal port is a reference port and provides the low-side reference signal for the Signal and the Inverted Signal signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Multi-Channel PWM Out) on page 311.
Dependencies	The number of Reference ports depends on the value specified at the Number of signals property.

Overview of Tunable Properties (Multi-Channel PWM Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Parameter	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Update mode	✓	–
	Event trigger condition	✓	–
	Event downsampling	✓	–
	Event delay	✓	–
	Trigger condition	✓	–
	Trigger downsampling	✓	–
	Trigger delay	✓	–
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓
Electrical Interface			
	Deadband	✓	–
	Signal termination	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Multi-Channel PWM Out)

Where to go from here

Information in this section

- | | |
|------------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Multi-Channel PWM Out)..... | 304 |
| Configuring Standard Features (Multi-Channel PWM Out)..... | 308 |

Configuring the Basic Functionality (Multi-Channel PWM Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the number of PWM signals.
- Providing inverted signals.
- Specifying the alignment and update mode of PWM signals.
- Providing I/O events and function triggers.
- Enabling signal termination.

Specifying the number of PWM signals

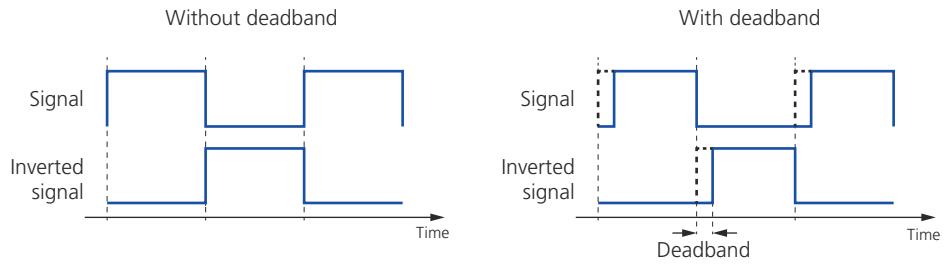
You have to specify the number of non-inverted PWM signals with the Number of signals property. If you enable generating inverted signals, they are automatically added.

The maximum number of all the PWM signals that the function block can generate depends on the assigned hardware resource. For specific maximum values, refer to [Hardware Dependencies \(Multi-Channel PWM Out\)](#) on page 311.

Providing inverted signals

You can enable the generation of an additional inverted signal for each generated non-inverted signal with the Inverted phase generation property. In this case the maximum number of non-inverted signals that can be generated is cut by half.

Avoiding shoot-through currents To avoid shoot-through currents on a connected driver, you can specify a deadband (time gap) between the pulses of the non-inverted signal and its inverted signal with the Deadband property. The deadband delays the edge generation of new pulses, as shown in the following illustration:



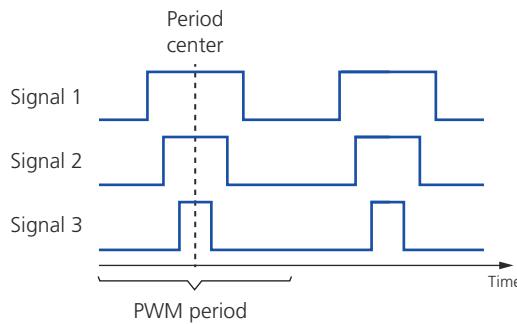
The value range of the deadband depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Multi-Channel PWM Out\)](#) on page 311.

Aligning and updating the PWM signals

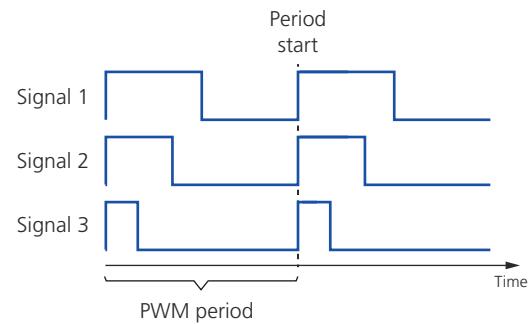
Depending on the alignment of the PWM signals to each other, the Multi-Channel PWM Out function block provides different modes to update the signal generation.

Specifying the alignment You can choose between center-aligned signal generation and edge-aligned signal generation.

Center-aligned



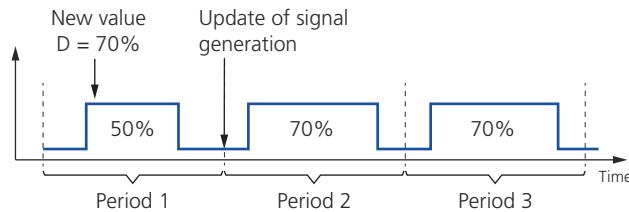
Edge-aligned



Updating center-aligned PWM signals You can specify the timing for updating center-aligned PWM signals with the **Update mode** property as follows:

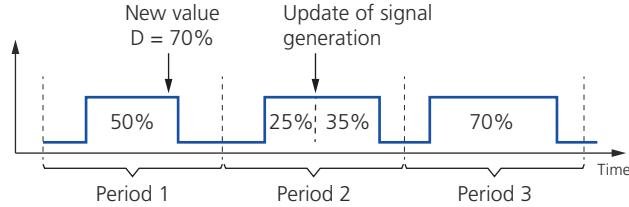
- **Period start:**

In this mode, all PWM signals are updated at period start. This means that new values do not take effect before the current PWM period is completed, as shown in the following illustration:



- **Period center:**

In this mode, all PWM signals are updated at the center of the PWM period. This means that new values do not take effect before the center of the PWM period is generated, as shown in the following illustration:



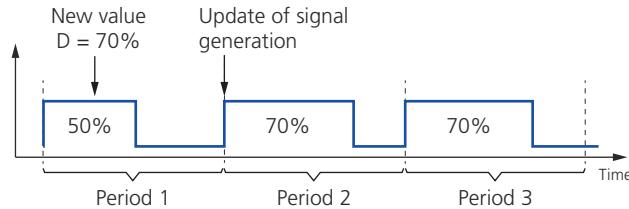
- **Period start and center:**

In this mode, all PWM signals are updated at period start and at the period center.

Updating edge-aligned PWM signals You can specify the timing for updating edge-aligned PWM signals with the **Update mode** property as follows:

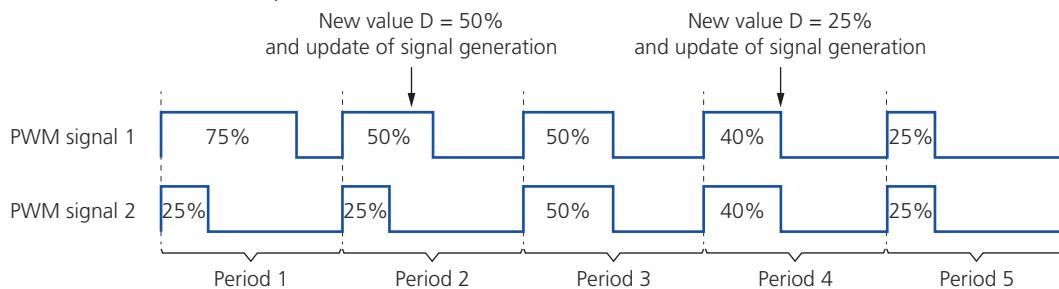
- **Synchronous:**

In this mode, all PWM signals are synchronously updated after every completed period. This means that new values do not take effect before the current PWM period is completed, as shown in the following illustration.



- **Asynchronous:**

In this mode, new values immediately take effect. Pulses of PWM signals are shortened or stretched if they are not already completed. PWM signals with completed pulses are updated with the next signal period, as shown in the following illustration. As a result, the duty cycle values are asynchronously updated.



If the current pulse width already exceeds the pulse width of the new duty cycle, the pulse is cut, as shown in the illustration above at the second update. Then, the resulting duty cycle of the current period lies between the new duty cycle and the old duty cycle.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

For more information on specifying the conditions to generate and provide I/O events, refer to [Specifying the conditions to provide events and triggers](#) on page 307.

To use the I/O events in the behavior model, you have to assign them to a task via the Period/Edge Event property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Providing function triggers

If the I/O function trigger generation property is enabled, the function block generates a trigger signal to use it as a trigger source for other function blocks.

For more information on specifying the conditions to generate and provide function triggers, refer to [Specifying the conditions to provide events and triggers](#) on page 307.

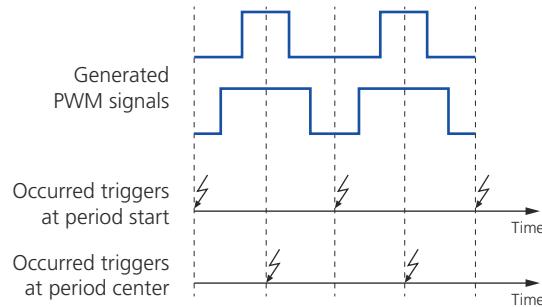
Specifying the conditions to provide events and triggers

The function block provides properties for specifying the trigger point to generate I/O events or function triggers. Furthermore, you can downsample occurred triggers and delay the generation of the I/O events or function triggers.

Specifying the trigger point The available trigger points depend on the specified signal alignment:

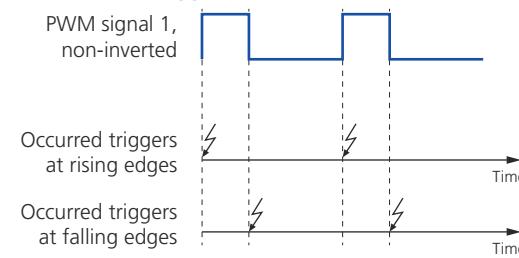
- Center-aligned PWM signals:

You can use period start, period center, or both points in time as a trigger point.



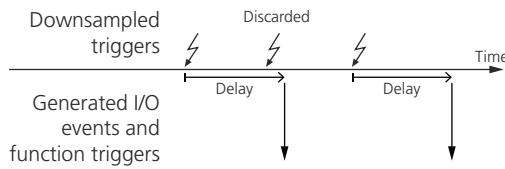
- Edge-aligned PWM signals:

You can use rising, falling, or all edges of the first signal (Signal 1, non-inverted) as a trigger point.



Downsampling occurred triggers The occurred triggers can be downsampled. The downsampling lets you specify whether to generate an I/O event or function trigger on each trigger occurrence or only on each n-th occurrence. The placeholder n is the specified downsampling value.

Delaying the generation of I/O events and function triggers You can specify time intervals to delay the generation of I/O events and function triggers. If a (downsampled) trigger initiates the generation of an I/O event or function trigger, the function block delays the generation until the specified time interval is elapsed. During the delay, all other downsampled triggers are discarded as shown below.



Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

Circuit diagrams For circuit diagrams of the channel types that support signal termination, refer to:

- [Digital Out 8](#) on page 1581
- [Digital In/Out 5](#) on page 1586
- [Digital In/Out 9](#) on page 1590

Limitation Signal termination is supported by the following channel types:

- SCALEXIO hardware: Digital Out 8, Digital In/Out 5 (not supported for the differential out interface type), Digital In/Out 9
- MicroAutoBox III hardware: Signal termination is not supported.

Configuring Standard Features (Multi-Channel PWM Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO				
Configuration Feature	Digital Out 8	Digital In/Out 5	Digital In/Out 9	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	Depending on the channel type, refer to: <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.		–	
Polarity of output signals	✓	✓	✓	–

Signal Ports				
Trigger level of electronic fuse	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	Specifying Settings for Failure Simulation on page 123

MicroAutoBox III				
Configuration Feature	Digital Out 4	Further Information		
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–		
High impedance (tristate)	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.	–		
Polarity of output signals	✓	–		
Channel multiplication	–	Specifying Current and Voltage Values for Channel Multiplication on page 95		

Signal Ports				
Trigger level of electronic fuse	–	Specifying Fuse Settings on page 122		
Failure simulation support	–	Specifying Settings for Failure Simulation on page 123		

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none">▪ Digital Out 8 on page 1581▪ Digital In/Out 5 on page 1586▪ Digital In/Out 9 on page 1590	<ul style="list-style-type: none">▪ Digital Out 4 on page 1578

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Multi-Channel PWM Out)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Multi-Channel PWM Out).....	311
MicroAutoBox III Hardware Dependencies (Multi-Channel PWM Out).....	312

SCALEXIO Hardware Dependencies (Multi-Channel PWM Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital Out 8	Digital In/Out 9	Digital In/Out 5
Hardware		DS6121 Multi-I/O Board		DS6202 Digital I/O Board
Number of PWM signals		1 ... 12	1 ... 4	1 ... 32
Event generation		✓		✓
Frequency	Range	1 Hz ... 1.25 MHz		1 Hz ... 1.25 MHz
	Time resolution	8 ns		8 ns
Minimum pulse duration		24 ns		–
Supported deadband		0 µs ... 400 µs		0 µs ... 400 µs
Interface type for digital outputs		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull
		In push-pull configuration the outputs can be set to high impedance (tristate) at run time.		
High side reference voltage (high-side switch, push-pull)		+3.3 V and +5 V		+3.3 V and +5 V
Output current range		0 mA ... ±40 mA (each channel)		0 mA ... ±40 mA (each channel)
Signal termination		<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable 		<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable
Circuit diagram		Digital Out 8 on page 1581	Digital In/Out 9 on page 1590	Digital In/Out 5 on page 1586
Required channels		1 per PWM signal ¹⁾		1 per PWM signal ¹⁾

¹⁾ One channel for each non-inverted PWM signal and one channel for each inverted signal.

General limitations

- Channel multiplication is not supported in general.
- If the Multi-Channel PWM Out function block works as a trigger function provider:
The hardware resources assigned to the Multi-Channel PWM Out function block and the hardware resources assigned to the function block using the Multi-Channel PWM Out function block as trigger source must be located on the same I/O board.

More hardware data

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (Multi-Channel PWM Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital Out 4
Hardware		<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board
Number of PWM signals		1 ... 16
Event generation		–
Frequency	Range	1 Hz ... 150 kHz
	Time resolution	12.5 ns
Supported deadband		0 ... 12.75 µs
Interface type for digital outputs		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull <p>The outputs can be set to high impedance (tristate) at run time.</p>
Output current range		0 ... ±5 mA
High side reference voltage		+4.5 V ... +40 V
Signal termination		–
Circuit diagrams		Digital Out 4 on page 1578
Required channels		1 per PWM signal ¹⁾

¹⁾ One channel for each non-inverted PWM signal and one channel for each inverted signal.

General limitations

- Channel multiplication is not supported in general.
- If the Multi-Channel PWM Out function block works as a trigger function provider:
For DS1511, DS1511B1, and DS1513 Multi-I/O Boards, the hardware resources assigned to the Multi-Channel PWM Out function block and the hardware resources assigned to the function block using the Multi-Channel PWM Out function block as trigger source must be located on the same *module* of the I/O board.

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

PWM/PFM In

Where to go from here

Information in this section

Introduction (PWM/PFM In).....	316
Overviews (PWM/PFM In).....	320
Configuring the Function Block (PWM/PFM In).....	324
Hardware Dependencies (PWM/PFM In).....	331

Introduction (PWM/PFM In)

Where to go from here

Information in this section

- | | |
|------------------------------------------------------|-----|
| Introduction to the Function Block (PWM/PFM In)..... | 316 |
| Basics on Signal Pattern Measurement..... | 317 |

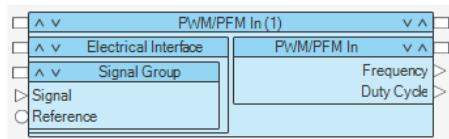
Introduction to the Function Block (PWM/PFM In)

Function block purpose

With the PWM/PFM In function block, you can measure one-phase pulse-width-modulated signal patterns.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Calculating frequency and duty cycle of the input signal and providing the results to the behavior model.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The PWM/PFM In function block type supports the following channel types:

	SCALEXIO									
Channel type	Digital In 1	Flexible In 2	Flexible In 1	Digital In 2	Digital In/Out 1	Digital In 3	Flexible In 3	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9
Hardware	DS2680		DS2601	DS2690		DS6101		DS6201	DS6202	DS6121

	MicroAutoBox III				
Channel type	Digital In 4	Digital In 5	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10
Hardware	<ul style="list-style-type: none"> ▪ DS1511 ▪ DS1511B1 ▪ DS1513 	<ul style="list-style-type: none"> ▪ DS1552 ▪ DS1552B1 	<ul style="list-style-type: none"> ▪ DS1552 ▪ DS1552B1 	DS1554	DS1521

Basics on Signal Pattern Measurement

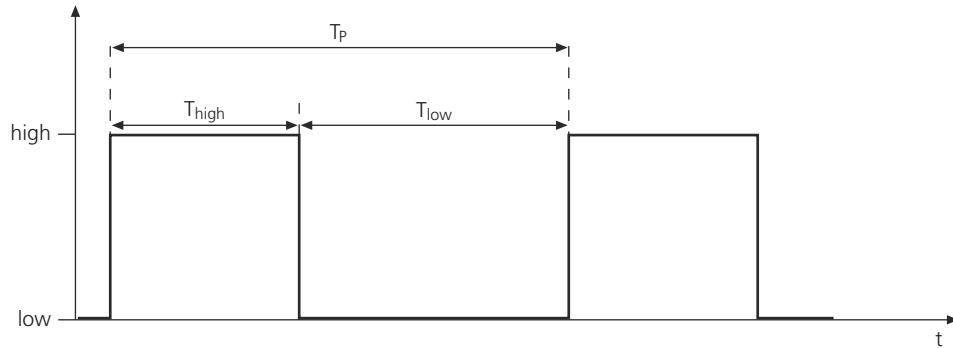
Use cases

Some ECUs deliver control signals for electrical motors (such as a throttle valve motor) as digital signal patterns. The PWM/PFM In function can handle signal patterns in the form of one-phase digital signals. The PWM/PFM In function uses the edges of the measured signal shape to calculate the frequency and/or the duty cycle of the input signal.

Terms and definitions

A digital input signal is specified by the following properties:

- T_{high} : Duration of a pulse that starts with a rising edge
 - T_{low} : Duration of a pulse that starts with a falling edge
 - T_{active} : Duration of the pulse with which the period starts. If the period starts with a rising edge, the signal has an active high pulse, if it starts with a falling edge, it has an active low pulse.
 - T_p : Duration of a period as the sum of the high and the low pulses
- $$T_p = T_{high} + T_{low}$$



Calculation formulas

The PWM/PFM In function block calculates the frequency and the pulse-to-period ratio:

- Frequency (f): The frequency is the inverse of the period.

$$f = 1 / T_p$$
- Duty cycle (d): The duty cycle is the pulse-to-period ratio of a period.

$$d = T_{active} / T_p$$

The signal characteristics are as follows:

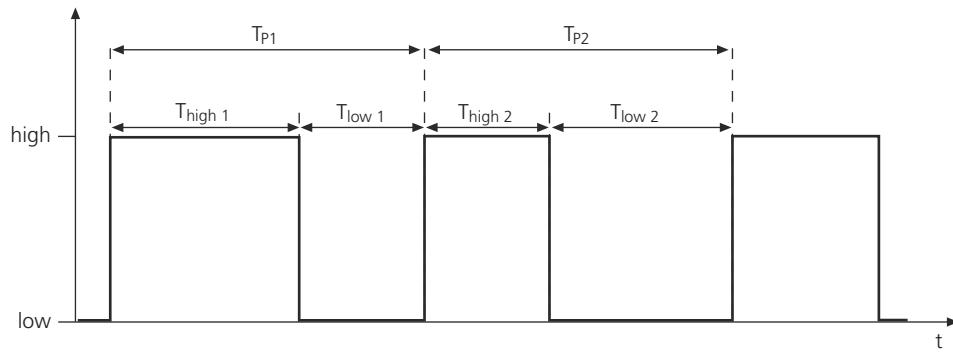
- Pulse-width modulated (PWM) signals: Signals with a constant frequency ($f_2 = f_1$)
- Pulse-frequency modulated (PFM) signals: Signals with a constant duty cycle ($d_2 = d_1$)
- Irregular square-wave signals with no constant frequency or duty cycle

With the Function mode of the PWM/PFM In function, you specify the signal characteristic to be used for calculation. In *Frequency* function mode, the signal is handled as a PFM input signal, and in *Duty Cycle* function mode, it is handled as a PWM input signal. In *Duty Cycle and Frequency* function mode, the aim is to observe variations in signal frequency and duty cycle.

The signal characteristics are described in detail below.

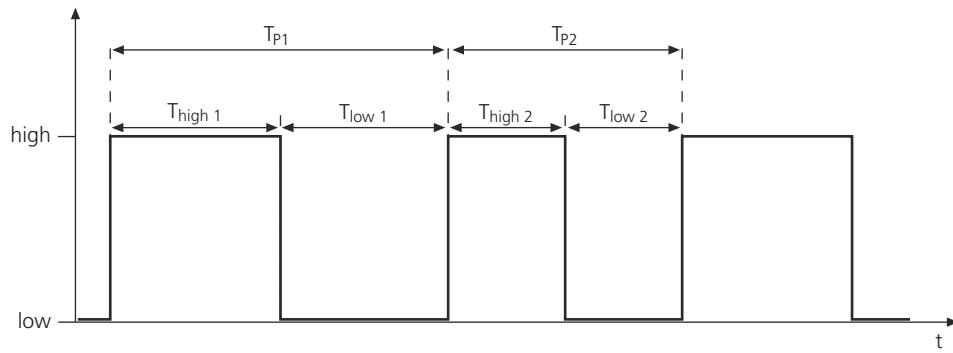
Characteristics of a PWM signal

PWM signals have a constant frequency, while the duty cycle can vary to control the signal information. This means that $1/T_{P2} = 1/T_{P1}$, in other words, the period duration T_p as the sum of $T_{high} + T_{low}$ is constant in the signal shape.



Characteristics of a PFM signal

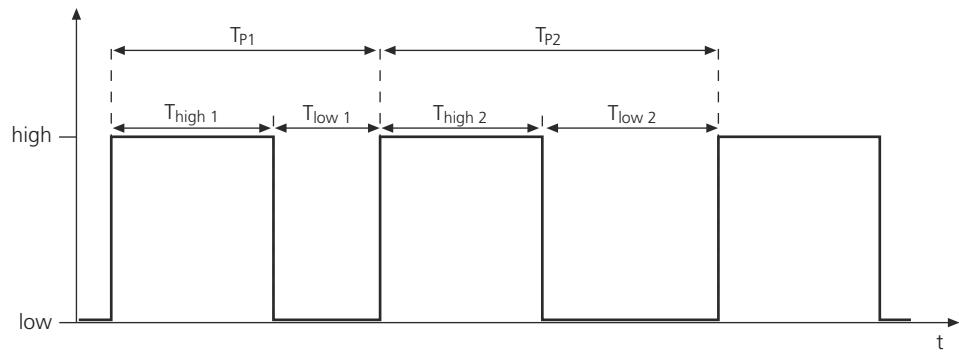
PFM signals have a constant duty cycle, while the frequency can vary to control the signal information. This means that $d_2 = d_1$, in other words, the ratio of T_{active} / T_p is constant in the signal shape. For example, if the period starts with a rising edge, the ratio of $T_{high} / (T_{high} + T_{low})$ is constant in the signal shape.



Characteristics of an irregular input signal

Irregular input signals have no constant frequency or duty cycle. Both properties can vary from period to period. The information of the signal depends on both signal properties.

In the following example, the duration of the high level signal is constant and the duration of the low level signal varies. This results in a variable frequency and duty cycle for each period.



Overviews (PWM/PFM In)

Where to go from here

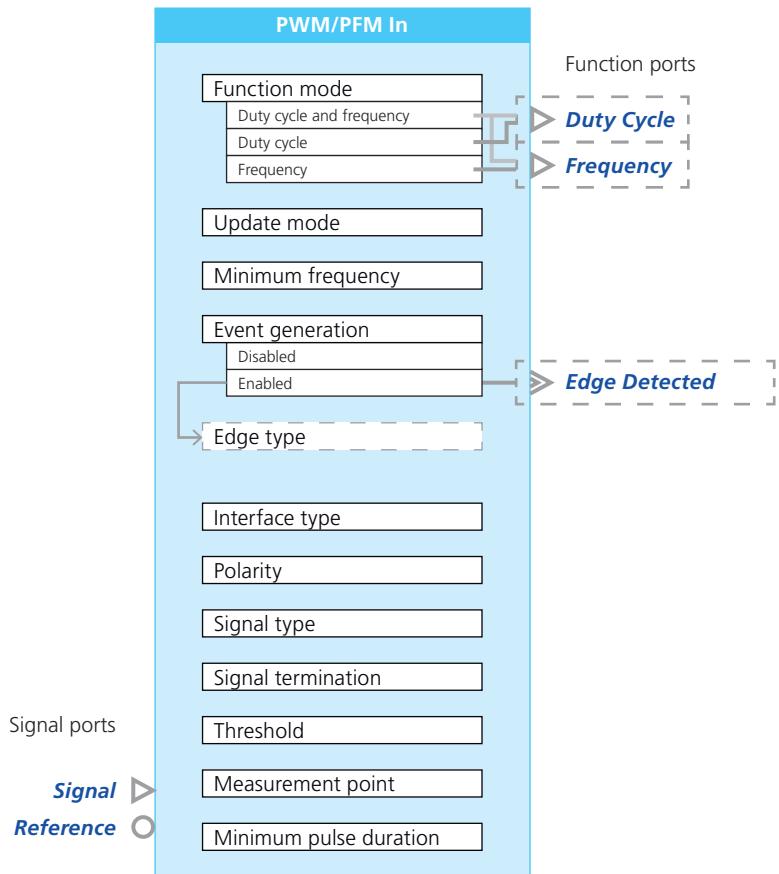
Information in this section

- | | |
|----------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (PWM/PFM In)..... | 320 |
| Overview of Tunable Properties (PWM/PFM In)..... | 323 |

Overview of Ports and Basic Properties (PWM/PFM In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



The function ports provide the processed values for the frequency and the duty cycle. You can select one or both of the function ports by choosing the function mode.

Frequency

This function outport writes the measured frequency of the input signal to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $f_{\min} \dots f_{\max}$ (in Hertz). ▪ Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM In) on page 331.
Dependencies	Available only if the Function mode property is set to Duty cycle and frequency or Frequency.

Duty Cycle

This function outport writes the measured duty cycle of the input signal to the behavior model.

Value range	0 % ... 100 %
Dependencies	Available only if the Function mode property is set to Duty cycle and frequency or Duty.

Edge Detected

This event port provides an I/O event each time an edge is detected at the signal port and this edge matches the specified trigger condition.

You have to specify which type of edge of the input signal (rising and/or falling) is relevant for event generation (via the Edge type property).

Value range	–
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Not supported. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Available only if the Event generation property is set to Enabled. ▪ Not supported for the Digital In/Out 10 channel type.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM In) on page 331.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (PWM/PFM In) on page 331.
Dependencies	Not supported for Digital In 2 and Digital InOut 1 channel type.

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Always available for Digital In 2 and Digital In/Out 1 channel types. ▪ Not supported for Digital In 3, Digital In/Out 3, Digital In/Out 5, Digital In/Out 9, and Flexible In 3 channel types. ▪ For all the other channel types available only if the Connected load property is set to Use external load. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Supported only for the Flexible In 1 channel type. ▪ Available only if the Connected load property is set to Use external load. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported.

Overview of Tunable Properties (PWM/PFM In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Update mode	✓	–
Minimum frequency	✓	–
Edge type	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Polarity	✓	–
Signal termination	✓	–
Threshold	✓	–
Measurement point	✓	–
Minimum pulse duration	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (PWM/PFM In)

Where to go from here

Information in this section

Configuring the Basic Functionality (PWM/PFM In).....	324
Configuring Standard Features (PWM/PFM In).....	328

Configuring the Basic Functionality (PWM/PFM In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the function mode (frequency, duty cycle or both).
- Specifying the update mode for the measured values (synchronous or asynchronous).
- Specifying the signal type (voltage or current).
- Specifying the trigger threshold.
- Specifying the pulse filter for the input signal
- Terminating the input signal..
- Specifying the zero detection frequency.
- Specifying the polarity of the binary signal provided to the behavior model.
- Specifying the generation of I/O events.

Specifying the function mode

You can choose the Function mode to specify the variables to be calculated according to your requirements:

▪ Duty cycle

This mode can be used to measure the duty cycle of the input signal. Only the function outport for the duty cycle is provided.

▪ Frequency

This mode can be used to measure the frequency of the input signal. Only the function outport for the frequency is provided.

▪ Duty cycle and frequency

This mode can be used to measure the frequency and the duty cycle of the input signal. The calculation includes both variables, so both variables are also provided as function outports.

Specifying the update mode for measured values

The function block provides the Update mode property to read the measured values synchronously or asynchronously:

- In synchronous update mode, the values are read after one PWM period is completed.
- In asynchronous update mode, the values are read after each edge.

Note

If both the frequency and the duty cycle of the input signal vary within one period, the calculated values can be erroneous in asynchronous update mode.

Specifying the signal type

You must select the Signal type according to the connected external device that delivers the input signal:

- Voltage

The input signal is delivered as a voltage signal, for example, a square-wave signal in the range 0 ... 5 V.

- Current

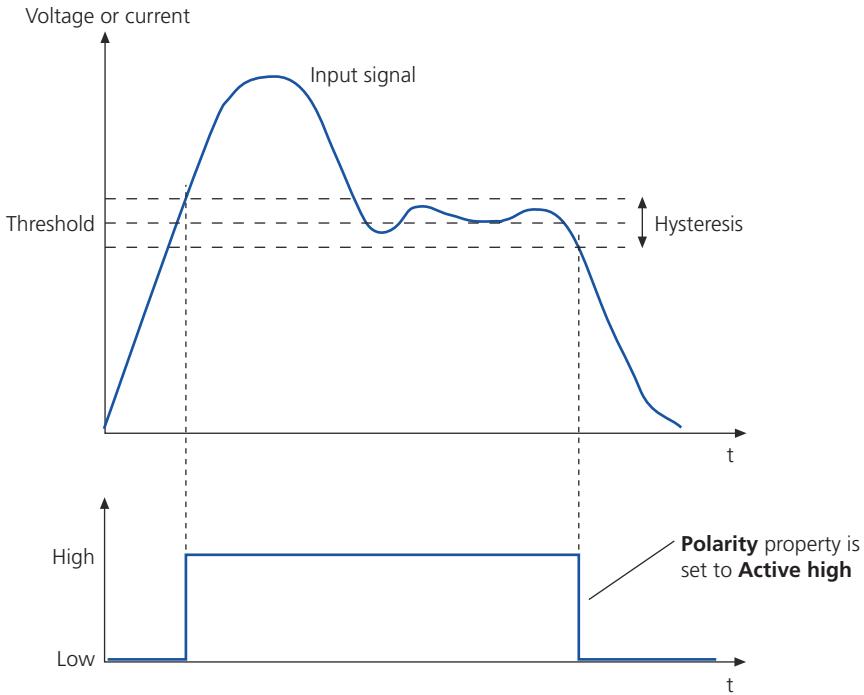
The input signal is delivered as a current signal, for example, a square-wave signal in the range 0 ... 500 mA.

Limitation The Current signal type is only supported by the following channel types:

- SCALEXIO hardware: Flexible In 1, Flexible In 2

Specifying the trigger threshold

You can specify a threshold value to transform the input signal into binary values. The hysteresis value is fixed and depends on the assigned hardware resource. A rising edge of the input signal must exceed the threshold plus half the hysteresis value to be detected as high, and a falling edge of the input signal must fall below the threshold minus the half hysteresis value to be detected as low.



The value range of the threshold and the hysteresis constant depend on the assigned hardware resource and the selected signal type. For specific values, refer to [Hardware Dependencies \(PWM/PFM In\)](#) on page 331.

Limitation The threshold value is fixed and cannot be changed for the following channel types:

- MicroAutoBox III hardware: Digital In 4, Digital In 5, Digital In/Out 10

Filtering the input signal

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the pulse duration depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(PWM/PFM In\)](#) on page 331 .

You can also change the value of the **Minimum pulse duration** property via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Limitation The pulse filter cannot be configured for the following channel types:

- SCALEXIO hardware: Digital In 2, Digital In/Out 1, Digital In/Out 3

Terminating the input signal

You can activate signal termination for differential input signals to prevent signal reflections at the line end.

Note

You cannot activate a signal termination for ground-based input signals.

Therefore, signal termination is supported only by the Digital In/Out 5 channel type. A $120\ \Omega$ resistor is connected in series to a 5 nF capacitor between the input lines. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Zero frequency detection

You can specify the minimum measurable frequency for zero frequency detection of the input signal. If a frequency lower than the specified minimum frequency is detected for one period, the frequency is set to 0.0 Hz. The duty cycle is then set to 0% or 100% according to the measured signal level (active signal: 100%, inactive signal: 0%). This feature detects a zero frequency when the input signal is turned off or disconnected during measurement. The minimum adjustable frequency value depends on the hardware used. For details, refer to [Hardware Dependencies \(PWM/PFM In\)](#) on page 331.

Polarity of the binary signal

To adapt the characteristic of the input signal to the requirements of your behavior model, you can switch the **Polarity** of the signal:

▪ Active high

The higher voltage or current of the signal represents an active signal, and the lower voltage or current represents an inactive signal.

The duty cycle is then calculated as:

$$d_{\text{Active high}} = T_{\text{high}} / (T_{\text{high}} + T_{\text{low}})$$

▪ Active low

The lower voltage or current of the signal represents an active signal, and the higher voltage or current represents an inactive signal.

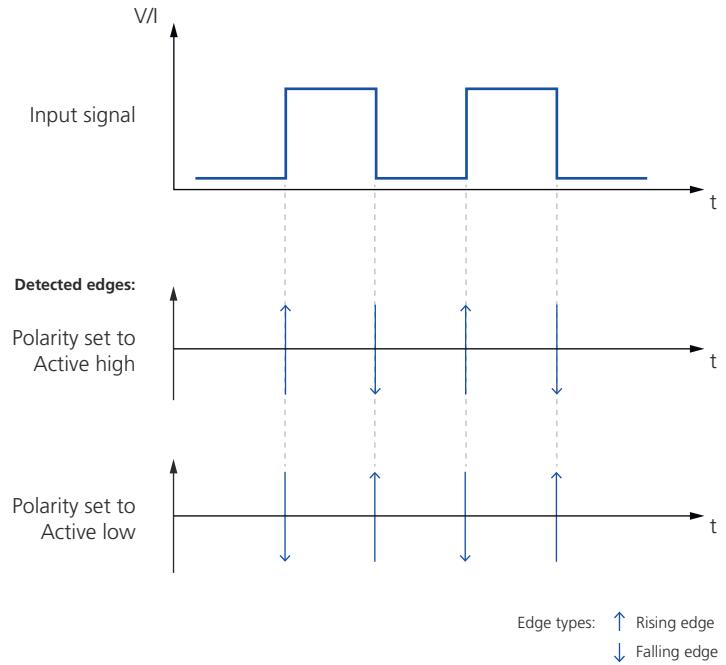
The duty cycle is then calculated as:

$$d_{\text{Active low}} = T_{\text{low}} / (T_{\text{high}} + T_{\text{low}})$$

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

You can specify whether the rising edges and/or the falling edges of the input signal are used to trigger event generation via the **Edge type** property. Note that if you invert the input signal by setting the **Polarity** property to **Active low**, the edge type of the detected edges differs from the original input signal as shown in the following illustration:



To use the I/O events in the behavior model, you have to assign them to a task via the Edge Detected property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Limitation Event generation is not supported for the following channel types:

- SCALEXIO hardware: All channel types.
- MicroAutoBox III hardware: Digital In/Out 10.

Configuring Standard Features (PWM/PFM In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SCALEXIO							
Configuration Feature	Digital In 1	Digital In 2	Digital In 3	Digital In/Out 1	Digital In/Out 3	Digital In/Out 5	Further Information
Measurement point	Load side	Load side	–	Load side	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Ground-based	Ground-based	Ground-based	Ground-based	▪ Ground-based ▪ Differential	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	✓	✓	–	✓	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports							
Trigger level of electronic fuse	–	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	✓	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	–	✓	–	–	Specifying Load Settings on page 121
SCALEXIO							
Configuration Feature	Digital In/Out 9	Flexible In 1		Flexible In 2	Flexible In 3	Further Information	
Measurement point	–	▪ Load side ▪ ECU side ¹⁾		Load side	–	Specifying the Measurement Point for Input Signals on page 119	
Interface type ¹⁾	Ground-based	▪ Galvanically isolated ▪ Ground-based		Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116	
Channel multiplication	–	✓		✓	–	Specifying Current and Voltage Values for Channel Multiplication on page 95	
Signal Ports							
Trigger level of electronic fuse	–	✓		–	–	Specifying Fuse Settings on page 122	
Failure simulation support	–	✓		✓	–	Specifying Settings for Failure Simulation on page 123	
Usage of loads	–	✓		✓	–	Specifying Load Settings on page 121	

¹⁾ Configurable only if you measure voltages (Signal type property is set to Voltage).

MicroAutoBox III						
Configuration Feature	Digital In 4	Digital In 5	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10	Further Information
Measurement point	–	–	–	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type ¹⁾	Ground-based	Ground-based	Ground-based	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	–	–	–	–	Specifying Load Settings on page 121

¹⁾ Configurable only if you measure voltages (Signal type property is set to Voltage).

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Digital In 1 on page 1569 ▪ Digital In 2 on page 1571 ▪ Digital In 3 on page 1572 ▪ Digital In/Out 1 on page 1582 ▪ Digital In/Out 3 on page 1584 ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 ▪ Flexible In 1 on page 1593 ▪ Flexible In 2 on page 1595 ▪ Flexible In 3 on page 1603 	<ul style="list-style-type: none"> ▪ Digital In 4 on page 1572 ▪ Digital In 5 on page 1573 ▪ Digital In/Out 6 on page 1588 ▪ Digital In/Out 8 on page 1589 ▪ Digital In/Out 10 on page 1591

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (PWM/PFM In)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (PWM/PFM In).....	331
MicroAutoBox III Hardware Dependencies (PWM/PFM In).....	334

SCALEXIO Hardware Dependencies (PWM/PFM In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital In 1	Flexible In 2	Flexible In 1	Digital In 2	Digital In/Out 1
Hardware		DS2680 I/O Unit		DS2601 Signal Measurement Board	DS2690 Digital I/O Board	
Frequency	Maximum	330 kHz ¹⁾		1.838 MHz	330 kHz	
	Time resolution	68 ns		68 ns	80 ns	
Input current range		▪ 0 ... +6 A _{RMS} for 1 channel ▪ 144 A _{RMS} ²⁾ for 30 channels (channel multiplication)	▪ 0 ... +6 A _{RMS} for 1 channel ▪ 86.4 A _{RMS} ²⁾ for 18 channels (channel multiplication)	▪ 0 ... +10 A _{RMS} for 1 channel ▪ 80 A _{RMS} for 10 channels (channel multiplication)	▪ 0 ... +6 A _{RMS} for 1 channel ▪ 48 A _{RMS} for 10 channels (channel multiplication)	▪ 0 ... +100 mA _{RMS} for 1 channel ▪ 0.8 A _{RMS} for 10 channels (channel multiplication)
Input voltage range		0 ... +60 V		-60 V ... +60 V	0 ... +60 V	
Signal type		Voltage	▪ Voltage ▪ Current	▪ Voltage ▪ Current	Voltage	
Threshold	Input threshold voltage	0 ... +24 V	▪ Voltage signal type: 0 ... +24 V ▪ Current signal type: -18 A ... +18 A	▪ Voltage signal type: -60 V ... +60 V ▪ Current signal type: -30 A ... +30 A	+1 V ... +23.8 V	
	Input hysteresis voltage	200 mV (typ.)	▪ Voltage signal type: 210 mV (typ.) ▪ Current signal type:	▪ Voltage signal type: 250 mV (typ.) ▪ Current signal type:	200 mV (typ.)	

Channel Type		Digital In 1	Flexible In 2	Flexible In 1	Digital In 2	Digital In/Out 1
			120 mA (typ.)	125 mA (typ.)		
	DAC resolution	8 bit			14 bit	8 bit
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs		0.264 µs ... 131 µs	80 ns (not configurable)	
Supported interface type		Ground-based		<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Ground-based 	Ground-based	
Measurement point		Load side		Voltage signal type: <ul style="list-style-type: none"> ▪ Load side ▪ ECU side Current signal type: <ul style="list-style-type: none"> ▪ Load side 	Load side	
Configurable fuse		–		<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	–	
Signal termination		–		–	–	
Circuit diagrams		Digital In 1 on page 1569	Flexible In 2 on page 1595	Flexible In 1 on page 1593	Digital In 2 on page 1571	Digital In/Out 1 on page 1582
Required channels		1 (additional channels are required for current enhancements.)				

1) For frequencies higher than 150 kHz (-3 db cutoff frequency), the acquisition might not be correct if the configured threshold level is not at 50 % of the input signal range or if the duty cycle of the input signal is not 50 %.

2) This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

Channel Type		Digital In 3	Flexible In 3	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9
Hardware		DS6101 Multi-I/O Board		DS6201 Digital I/O Board	DS6202 Digital I/O Board	DS6121 Multi-I/O Board
Frequency	Maximum	330 kHz ¹⁾		330 kHz ¹⁾	20 MHz	20 MHz
	Time resolution	68 ns		80 ns	8 ns	8 ns
Input current range		–		–	–	–
Input voltage range		0 ... +60 V		0 ... +60 V	0 ... +30 V	0 ... +60 V
Signal type		Voltage		Voltage	Voltage	Voltage
Threshold	Input threshold voltage	0 ... +24 V	-10 V ... +10 V	+1 V ... +23 V	0 ... +12 V	0 ... +12 V
	Input hysteresis voltage	200 mV (typ.)		230 mV (typ.)	600 mV (typ.)	600 mV (typ.)
	DAC resolution	8 bit		8 bit	8 bit	8 bit

Channel Type		Digital In 3	Flexible In 3	Digital In/Out 3	Digital In/Out 5	Digital In/Out 9
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs		80 ns (not configurable)	0 ... 10 ms	0 ... 10 ms
Supported interface type		Ground-based		Ground-based	<ul style="list-style-type: none"> ▪ Ground-based ▪ Differential 	Ground-based
Measurement point		–		–	–	–
Configurable fuse		–		–	–	–
Signal termination		–		–	<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (parallel termination, only for differential input signals) ▪ Switchable 	–
Circuit diagrams		Digital In 3 on page 1572	Flexible In 3 on page 1603	Digital In/Out 3 on page 1584	Digital In/Out 5 on page 1586	Digital In/Out 9 on page 1590
Required channels		1		1	1	1

¹⁾ For frequencies higher than 150 kHz (-3 db cutoff frequency), the acquisition might not be correct if the configured threshold level is not at 50 % of the input signal range or if the duty cycle of the input signal is not 50 %.

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6121 Multi I/O Board Channel multiplication is not supported in general.

DS6201 Digital I/O Board Channel multiplication is not supported in general.

DS6202 Digital I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2690 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS2690 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6121 Multi-I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6201 Digital I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6201 Digital I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet](#) of the DS6202 Digital I/O Board ([SCALEXIO Hardware Installation and Configuration](#)).

MicroAutoBox III Hardware Dependencies (PWM/PFM In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital In 4	Digital In 5	Digital In/Out 6	Digital In/Out 8	Digital In/Out 10
Hardware		<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module 		DS1554 Engine Control I/O Module	DS1521 Bus Board
Frequency	Maximum	1 MHz	1 MHz	10 MHz	10 MHz	1 MHz
	Time resolution	12.5 ns	12.5 ns		12.5 ns	8 ns
Event generation		✓	✓		✓	–
Input voltage range		0 ... +40 V	0 ... +40 V	0 ... +15 V	0 ... +15 V	0 ... +60 V
Signal type		Voltage	Voltage		Voltage	Voltage
Threshold	Input threshold voltage	+2 V (typ.)	+2 V (typ.)	+1 V ... +7.5 V	+1 V ... +7.5 V	+2 V (typ.)
	Input hysteresis voltage	1 V (typ.)	1 V (typ.)	0.7 V (typ.)	0.7 V (typ.)	0.6 V (typ.)
Pulse filter	Minimum pulse duration	0 ... 12.75 µs	0 ... 12.75 µs		0 ... 12.75 µs	0 ... 0.01 s
Supported interface type		Ground-based	Ground-based		Ground-based	Ground-based
Circuit diagram		Digital In 4 on page 1572	Digital In 5 on page 1573	Digital In/Out 6 on page 1588	Digital In/Out 8 on page 1589	Digital In/Out 10 on page 1591
Required channels		1	1		1	1

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to [DS1552 Multi-I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1554 Engine Control I/O Module For more module-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more module-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Digital Pulse In

Where to go from here

Information in this section

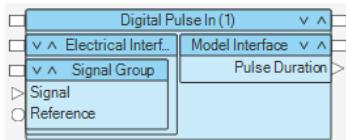
Introduction (Digital Pulse In).....	338
Overviews (Digital Pulse In).....	339
Configuring the Function Block (Digital Pulse In).....	341
Hardware Dependencies (Digital Pulse In)	344

Introduction (Digital Pulse In)

Introduction to the Function Block (Digital Pulse In)

Function block purpose The Digital Pulse In function block type lets you measure the pulse duration of digital voltage signals coming from an external device.

Default display The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features These are the main features:

- Transforming analog signals to digital signals.
- Measuring the pulse duration of high or low pulses of the digital signals.
- Generating and providing I/O events at start and/or end of measured pulses.

Supported channel types The Digital Pulse In function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	–	Digital In 4
Hardware	–	<ul style="list-style-type: none">▪ DS1511▪ DS1511B1▪ DS1513

Overviews (Digital Pulse In)

Where to go from here

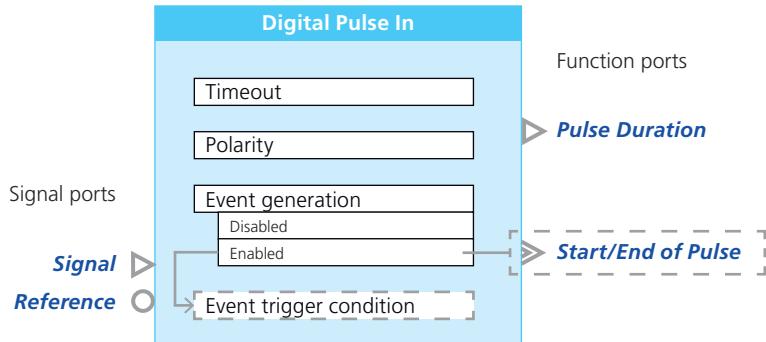
Information in this section

- | | |
|--------------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Digital Pulse In)..... | 339 |
| Overview of Tunable Properties (Digital Pulse In)..... | 340 |

Overview of Ports and Basic Properties (Digital Pulse In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Pulse Duration

This function output port writes the measured pulse duration to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 s ... Timeout ▪ -1.0 s: Measurement aborted due to timeout
Dependencies	–

Start/End of Pulse

This event port provides an I/O event at the start and/or the end of each pulse, depending on the specified event trigger condition.

For the event trigger condition, you have to specify if pulse start, pulse end, or both are used for triggering (via the Event trigger condition property).

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse In) on page 344.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse In) on page 344.
Dependencies	–

Overview of Tunable Properties (Digital Pulse In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Timeout	✓	–
Event trigger condition	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Threshold	✓	–
Polarity	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Digital Pulse In)

Where to go from here

Information in this section

- | | |
|-------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Digital Pulse In)..... | 341 |
| Configuring Standard Features (Digital Pulse In)..... | 343 |

Configuring the Basic Functionality (Digital Pulse In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

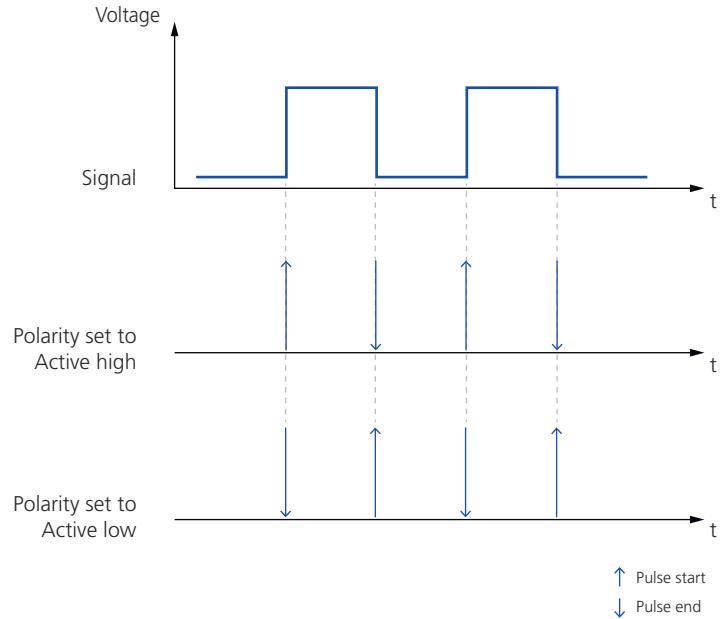
- Specifying the part of the digital signal to be measured (high or low).
- Specifying a timeout value to abort pulse duration measurement.
- Providing I/O events.

Specifying the part of the digital signal to be measured

The Digital In 4 channel type provides a fixed threshold value to transform the input signal to binary values. The hysteresis is also fixed. You can handle digital signals of different polarity without using inverters on the hardware side by setting the Polarity property as required. The setting determines which part of the signal you measure:

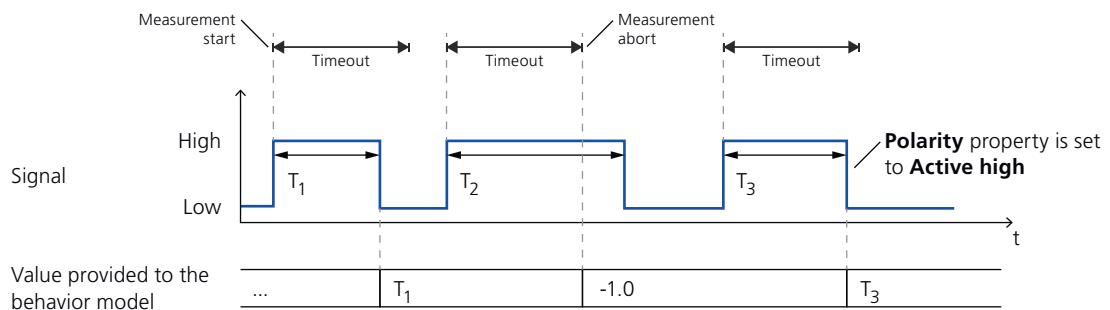
- Active high
The high pulse of the digital signal is measured and used for event generation.
- Active low
The low pulse of the digital signal is measured and used for event generation.

The setting of the **Polarity** property also determines whether signal edges are evaluated as start or end of a pulse. Refer to the following illustration:



Specifying a timeout value

You have to specify a timeout value. The possible range depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Digital Pulse In\)](#) on page 344. After this time span, the measurement of the pulse duration is aborted. To indicate the aborted measurement, the value -1.0 is written to the behavior model via the Pulse Duration function port. Refer to the following illustration:



Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Digital Pulse In function block can generate I/O events each time a pulse is measured. The **Event trigger condition** property lets you specify whether the I/O event is triggered at pulse start, pulse end, or at both conditions.

Note

When the Event trigger condition property is set to Pulse end or Pulse start and end, I/O events are also generated at the end of a pulse if the measurement was aborted due to timeout.

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Pulse property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide](#) .

Configuring Standard Features (Digital Pulse In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The electrical interface of the Digital Pulse In function block does not provide configurable standard features.
-----------------------------	------------------------------------------------------------------------------------------------------------------

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital In 4](#) on page 1572

Model interface	The interface to the behavior model of the function block provides the following standard features.
------------------------	-----------------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Digital Pulse In)

MicroAutoBox III Hardware Dependencies (Digital Pulse In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital In 4	
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	
Event generation	<input checked="" type="checkbox"/>	
Input voltage range		0 ... +40 V
Threshold	Input threshold voltage	+2 V (typ.)
	Input hysteresis voltage	1 V (typ.)
Input frequency	Maximum	1 MHz
	Time resolution	12.5 ns
Timeout		1 µs ... 1 s
Circuit diagram		Digital In 4 on page 1572
Required channels	1	

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Digital Pulse Capture

Where to go from here

Information in this section

Introduction (Digital Pulse Capture).....	346
Overviews (Digital Pulse Capture).....	347
Configuring the Function Block (Digital Pulse Capture).....	352
Hardware Dependencies (Digital Pulse Capture).....	358

Introduction (Digital Pulse Capture)

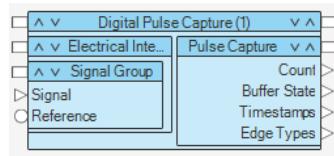
Introduction to the Function Block (Digital Pulse Capture)

Function block purpose

The Digital Pulse Capture function block converts signals coming from an external device (e.g., ECU) into corresponding time stamps, angle positions and edge polarities.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Capturing edges and the associated timestamps and angle positions.
- Specifying the trigger conditions for capturing edges.
- Generating I/O events and providing them to the behavior model each time an edge is detected in the input signal.

Supported channel types

The Digital Pulse Capture function block supports the following channel types:

	SCALEXIO						MicroAutoBox III
Channel type	Digital In 1	Flexible In 2	Flexible In 1	Digital In 3	Flexible In 3	Digital In/Out 5	Digital In/Out 10
Hardware	DS2680		DS2601	DS6101		DS6202	DS1521

Overviews (Digital Pulse Capture)

Where to go from here

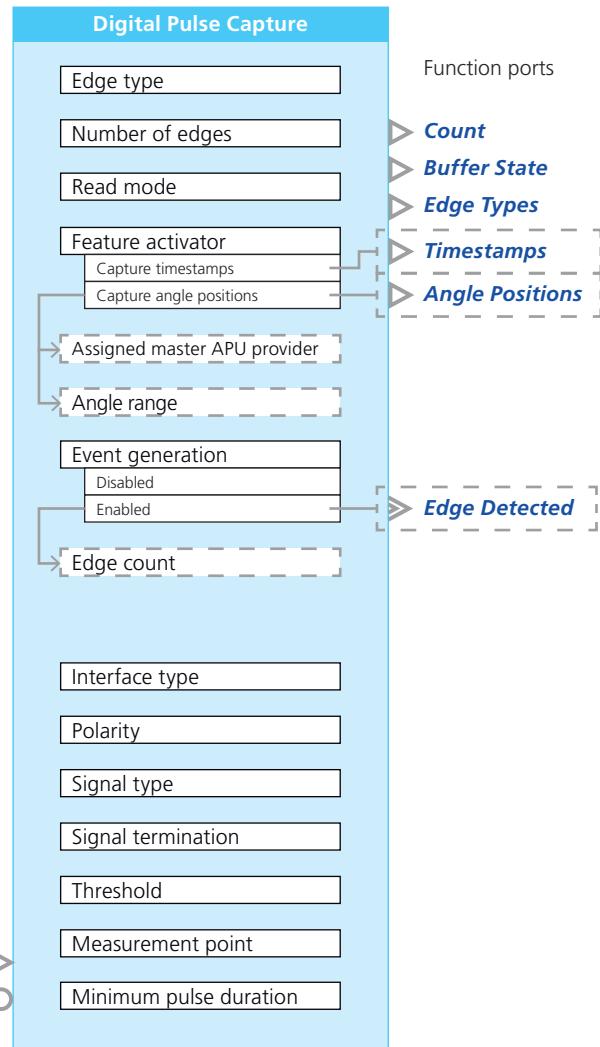
Information in this section

Overview of Ports and Basic Properties (Digital Pulse Capture).....	347
Overview of Tunable Properties (Digital Pulse Capture).....	351

Overview of Ports and Basic Properties (Digital Pulse Capture)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Count

This function outport writes the number of captured signal edges to the behavior model. It indicates the number of valid values in the Timestamps, Angle Positions and Edges vector.

Value range	<ul style="list-style-type: none"> 1 ... n Number of edges The maximum value depends on the value specified at the Number of edges property.
Dependencies	–

Buffer State

This function outport writes status information on the buffer to the behavior model. You can use this status information within your behavior model to check whether an error occurred during the measurement.

Value range	<ul style="list-style-type: none"> ▪ 0: OK The internal buffer is OK. No error occurred. ▪ 1: Empty The internal buffer is OK, but empty. No edges were detected. ▪ 2: Overflow An overflow of the internal buffer occurred. Too many edges were detected. Data is lost.
Dependencies	–

Timestamps

This function outport writes the time vector to the behavior model. This vector contains the timestamps of the captured signal edges.

Value range	<ul style="list-style-type: none"> ▪ 0 s ... 612,489,500 s for each timestamp in the time vector ▪ The maximum number of timestamps in the time vector depends on the value specified at the Number of edges property.
Dependencies	Available only if the Capture timestamps feature is enabled via the Feature activator property.

Edge Types

This function outport writes the edge vector to the behavior model. It contains the edge types of the captured signal edges.

Value range	<ul style="list-style-type: none"> ▪ For each entry of the edge vector: ▪ 0: Falling edge ▪ 1: Rising edge ▪ The maximum number of entries in the edge vector depends on the value specified at the Number of edges property.
Dependencies	–

Edge Detected

This event port provides an I/O event each time an edge is detected during signal capturing and this edge matches the specified trigger condition.

For the trigger condition, you have to specify the number of detected edges after which an event is generated via the **Edge count** property.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled .

Angle Positions

This function outport writes the angle vector to the behavior model. This vector contains the angle positions of the captured signal edges.

Value range	▪ 0° ... 360° or 0° ... 720° for each data in the angle vector
-------------	----------------------------------------------------------------

	<ul style="list-style-type: none"> The range depends on the Angle range property setting of the function block. The maximum number of data in the angle position vector depends on the value specified at the Number of edges property.
Dependencies	<ul style="list-style-type: none"> SCALEXIO hardware: Available only if the Capture angle position feature is enabled via the Feature activator property. MicroAutoBox III hardware: Not supported for the Digital In/Out 10 channel type.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse Capture) on page 358.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Pulse Capture) on page 358.
Dependencies	–

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> SCALEXIO hardware: <ul style="list-style-type: none"> Available only if the Connected load property is set to Use external load. Not supported for Digital In 3, Flexible In 3 and Digital In/Out 5 channel types. MicroAutoBox III hardware: <ul style="list-style-type: none"> Not supported.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Supported only for the Flexible In 1 channel type. ▪ Available only if the Connected load property is set to Use external load. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported.

Overview of Tunable Properties (Digital Pulse Capture)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Parameter	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Edge count	✓	-
Edge type	✓	-
Read mode	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Signal termination	✓	-
Threshold	✓	-
Measurement point	✓	-
Minimum pulse duration	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Digital Pulse Capture)

Where to go from here

Information in this section

- | | |
|----------------------------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Digital Pulse Capture)..... | 352 |
| Configuring Standard Features (Digital Pulse Capture)..... | 355 |

Configuring the Basic Functionality (Digital Pulse Capture)

Overview

The main configuration features of the function block are:

- Specifying the capturing of edges (contiguous or overlapping, number of edges).
- Enabling or disabling the capturing of timestamps and angle position data to save computation time.
- Conditioning the input signal: specifying signal type (voltage or current), trigger threshold, pulse filter and activating signal termination.
- Specifying generation of I/O events.

Specifying the capturing of edges

You can choose between two modes to capture edges.

Depending on how do you want to process the incoming signal within your behavior model, you can capture the edges of the incoming signal as a complete image (**Contiguous**) or you can get the latest measured edges with every measuring step (**Overlapped**). You can specify this behavior via the **Read Mode** property.

▪ Contiguous:

The captured edge data is returned in ascending order related to the **Timestamps**, **Angle Positions** and **Edge Types** function ports.

For example, the timestamps increase with increasing index. The first element of the **Timestamps** vector contains the timestamp of the first event since the last read operation.

▪ Overlapped:

The captured edge data is returned in a reverse order related to the **Timestamps**, **Angle Positions** and **Edge Types** function ports.

For example, the timestamps decrease with increasing index. The first element of the Timestamps vector contains the timestamp of the most recent event. Each read operation always starts with the last edge captured. Unlike the contiguous mode, segments of edge data being read may overlap. If the number of edges captured between two read operations is less than the number of edges read (Number of edges), an overlap occurs and some old edge data may be read several times.

Avoiding buffer overflows The Number of edges property lets you specify the maximum number of signal edges to be read with each sample step. If more signal edges are measured, they are stored in an internal buffer.

Take care to adapt the setting of the Number of edges property to the expected number of signal pulses to avoid a buffer overflow.

Notes on triggered subsystems or enabled subsystems Do not use triggered subsystems or enabled subsystems in your application together with the Digital Pulse Capture function block.

If triggered subsystems or enabled subsystems are essential in your application, note the following hints:

- It is recommended to read the measured signal edges not later than the tenth sample step if the Number of edges property is ≤ 100 .
- If you set the Number of edges property to more than 100, the measured signal edges must be read with every third sample step.

Enabling or disabling the capturing of timestamps and angle positions

Depending on your requests, you can capture timestamps and angle positions and write them to the behavior model via specific function ports. Timestamp and angle position values are captured synchronously to the captured edges. To save computation time, you are recommended to disable capturing for data that is not required.

As a precondition for capturing angle positions you have to assign a master APUs provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Limitation Capturing angle positions is not supported by the following channel types:

- MicroAutoBox III hardware: Digital In/Out 10.

Specifying the signal type

You must select the Signal type according to the connected external device that delivers the input signal:

- Voltage

The input signal is delivered as a voltage signal, for example, a square-wave signal in the range 0 ... 5 V.

- Current

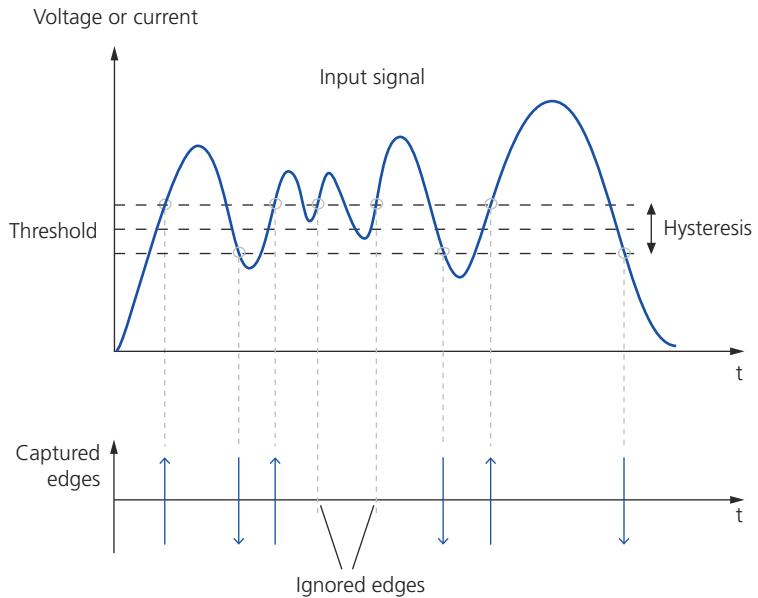
The input signal is delivered as a current signal, for example, a square-wave signal in the range 0 ... 500 mA.

Limitation The Current signal type is only supported by the following channel types:

- SCALEXIO hardware: Flexible In 1, Flexible In 2

Specifying the trigger threshold

The adjustable threshold value lets you compensate for any voltage/current offsets on the input signal. You should set the threshold value so as to capture all relevant edges of the waveform. The hysteresis value is fixed.



Edges of the input signal can be ignored for three reasons:

- The signal does not pass through the threshold
- The signal does not reach the threshold plus half the hysteresis
- The signal does not reach the threshold minus half the hysteresis

The value range of the threshold and the hysteresis constant depend on the assigned hardware resource and the selected signal type. For specific values, refer to [Hardware Dependencies \(Digital Pulse Capture\) on page 358](#).

Limitation The threshold value is fixed and cannot be changed for the following channel types:

- MicroAutoBox III hardware: Digital In/Out 10

Filtering the input signal

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the pulse duration depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Digital Pulse Capture\) on page 358](#).

You can also change the value of the **Minimum pulse duration** property via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Terminating the input signal

You can activate signal termination for differential input signals to prevent signal reflections at the line end.

Note

You cannot activate a signal termination for ground-based input signals.

Therefore, signal termination is supported only by the Digital In/Out 5 channel type. A $120\ \Omega$ resistor is connected in series to a 5 nF capacitor between the input lines. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Digital Pulse Capture function block can generate an I/O event each time an edge of the input signal is detected. However, with the Edge Count property, you can specify the number of detected edges after which an I/O event is generated. The maximum number depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Digital Pulse Capture\)](#) on page 358.

To use the I/O events in the behavior model, you have to assign them to a task via the Edge Detected property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Digital Pulse Capture)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO							
Configuration Feature	Digital In 1	Digital In 3	Digital In/Out 5	Flexible In 1	Flexible In 2	Flexible In 3	Further Information
Measurement point	Load side	–	–	▪ Load side ▪ ECU side ¹⁾	Load side	–	Specifying the Measurement Point for Input Signals on page 119
Interface type ²⁾	Ground-based	Ground-based	▪ Ground-based ▪ Differential	▪ Ground-based ▪ Galvanically isolated	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	✓	–	–	✓	✓	–	Specifying Current and Voltage Values for Channel Multiplication on page 95

Signal Ports							
Configuration Feature	Digital In 1	Digital In 3	Digital In/Out 5	Flexible In 1	Flexible In 2	Flexible In 3	Further Information
Trigger level of electronic fuse	–	–	–	✓	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	✓	✓	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	–	–	✓	✓	–	Specifying Load Settings on page 121

¹⁾ Supported only if you measure voltages (Signal type property is set to Voltage).

²⁾ Configurable only if you measure voltages (Signal type property is set to Voltage).

MicroAutoBox III		
Configuration Feature	Digital In/Out 10	Further Information
Measurement point	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Measurement range	–	–

Signal Ports		
Configuration Feature	Digital In 1	Further Information
Trigger level of electronic fuse	–	Specifying Fuse Settings on page 122
Failure simulation support	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MABX III
▪ Digital In 1 on page 1569	▪ Digital In/Out 10 on page 1591

SCALEXIO	MABX III
<ul style="list-style-type: none">▪ Digital In 3 on page 1572▪ Digital In/Out 5 on page 1586▪ Flexible In 1 on page 1593▪ Flexible In 2 on page 1595▪ Flexible In 3 on page 1603	

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Digital Pulse Capture)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Digital Pulse Capture).....	358
MicroAutoBox III Hardware Dependencies (Digital Pulse Capture).....	360

SCALEXIO Hardware Dependencies (Digital Pulse Capture)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital In 1	Flexible In 2	Flexible In 1
Hardware	DS2680 I/O Unit		DS2601 Signal Measurement Board
Maximum number of signal edges to be captured	500, each model step		500, each model step
Angle position capturing	✓		✓
Event generation	✓		✓
Maximum number of edges after which an event is generated	65,535		65,535
Input current range	<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 144 A_{RMS}¹⁾ for 30 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 86.4 A_{RMS}¹⁾ for 18 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +10 A_{RMS} for 1 channel ▪ 80 A_{RMS} for 10 channels (channel multiplication)
Input voltage range	0 ... +60 V		-60 V ... +60 V
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs	0.264 µs ... 131 µs
Threshold	Input threshold voltage	0 ... +24 V	<ul style="list-style-type: none"> ▪ Voltage signal type: 0 ... +24 V ▪ Current signal type: -18 A ... +18 A
	Input hysteresis voltage	200 mV (typ.)	<ul style="list-style-type: none"> ▪ Voltage signal type: 210 mV (typ.) ▪ Current signal type: 120 mA (typ.)
	DAC resolution	8 bit	14 bit

Channel Type		Digital In 1	Flexible In 2	Flexible In 1
Angular processing unit	Number of slaves	6	1	
	Maximum speed	168,000 °/s		
Measurement point		Load side		Voltage signal type: <ul style="list-style-type: none">▪ Load side▪ ECU side Current signal type: <ul style="list-style-type: none">▪ Load side
Configurable fuse		–		<ul style="list-style-type: none">▪ Load side▪ 0.5 A_{RMS} ... 10 A_{RMS}
Supported interface type		Ground-based	Ground-based	<ul style="list-style-type: none">▪ Ground-based▪ Galvanically isolated
Circuit diagram		Digital In 1 on page 1569	Flexible In 2 on page 1595	Flexible In 1 on page 1593
Required channels		1 (additional channels are required for current enhancements.)		

¹⁾ This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

Channel Type		Digital In 3	Flexible In 3	Digital In/Out 5
Hardware		DS6101 Multi-I/O Board		DS6202 Digital I/O Board
Maximum number of signal edges to be captured		500, each model step		500, each model step
Angle position capturing		✓		✓
Event generation		✓		✓
Maximum number of edges after which an event is generated		65,535		256
Input current range		–		–
Input voltage range		0 ... +60 V		0 ... +30 V
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs		8 ns ... 10 ms
Threshold	Input threshold voltage	0 ... +24 V	-10 V ... +10 V	0 ... +12 V
	Input hysteresis voltage	200 mV (typ.)		600 mV (typ.)
DAC resolution		8 bit		8 bit
Angular processing unit	Number of slaves	6		6
	Maximum speed	168,000 °/s		1,200,000 °/s
Measurement point		–		–
Configurable fuse		–		–
Supported interface type		Ground-based	Ground-based	<ul style="list-style-type: none">▪ Ground-based▪ Differential
Circuit diagram		Digital In 3 on page 1572	Flexible In 3 on page 1603	Digital In/Out 5 on page 1586
Required channels		1		1

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi I/O Board The following limitations apply to the DS6101 Multi I/O Board:

- Channel multiplication is not supported in general.
- The DS6101 allows the generation of max. 8 events on the board.

DS6202 Digital I/O Board The following limitations apply to the DS6202 Digital I/O Board:

- Channel multiplication is not supported in general.
- Only 16 Digital Pulse Capture function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (Digital Pulse Capture)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital In/Out 10	
Hardware	DS1521 Bus Board	
Maximum number of signal edges to be captured	500, each model step	
Angle position capturing	–	
Event generation	✓	
Maximum number of edges after which an event is generated	256	
Input voltage range	0 ... +60 V	
Signal type	Voltage	
Pulse filter	Minimum pulse duration	
Threshold	Input threshold voltage	
	Input hysteresis voltage	
Supported interface type	Ground-based	

Channel Type	Digital In/Out 10
Circuit diagram	Digital In/Out 10 on page 1591
Required channels	1

More hardware data

DS1521 Bus Board For more module-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration](#) .

Engine Simulation

Where to go from here

Information in this section

Introduction to Engine Simulation.....	364
Angular Clock Setup.....	369
Engine Simulation Setup.....	373
Injection/Ignition (Voltage In, Current In).....	377
Crank/Cam (Voltage Out, Current Sink, Digital Out).....	451
Knock Signal Out.....	498
Lambda Probe Simulation.....	509

Introduction to Engine Simulation

Where to go from here

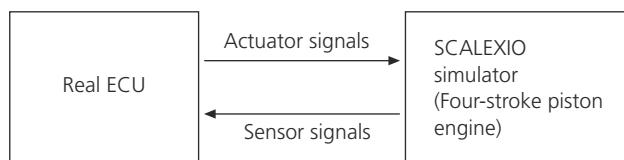
Information in this section

Basics on Engine Simulation with SCALEXIO.....	364
Operation Principle of Four-Stroke Piston Engines.....	365

Basics on Engine Simulation with SCALEXIO

Virtual engine connected to real ECU

Working with a SCALEXIO simulator means that you can simulate the behavior of a four-stroke piston engine. The virtual engine sends realistic sensor signals to and receives realistic actuator signals from a real ECU. Some engine components, for example, the throttle can be real and are part of the SCALEXIO simulator.



Available function blocks for engine simulation

ConfigurationDesk provides the following function block types for simulating the behavior of virtual piston engines with a SCALEXIO simulator:

Angular Clock Setup The Angular Clock Setup function block specifies the angle range and controls the rotational speed of the virtual piston engine. Furthermore, it provides the angular position of the virtual engine to the behavior model.

Engine Simulation Setup The Engine Simulation Setup function block specifies the characteristics of the virtual piston engine such as number of cylinders and top dead centers. Furthermore, it provides the angular position of the virtual cylinders to the behavior model.

Injection/Ignition In Injection/Ignition Voltage In and Injection/Ignition Current In function blocks measure injection or ignition pulses received from a real ECU.

Crank/Cam Out Crank/Cam Voltage Out, Crank/Cam Current Sink, and Crank/Cam Digital Out function blocks output wavetable-based signals for simulating crankshaft or camshaft sensors.

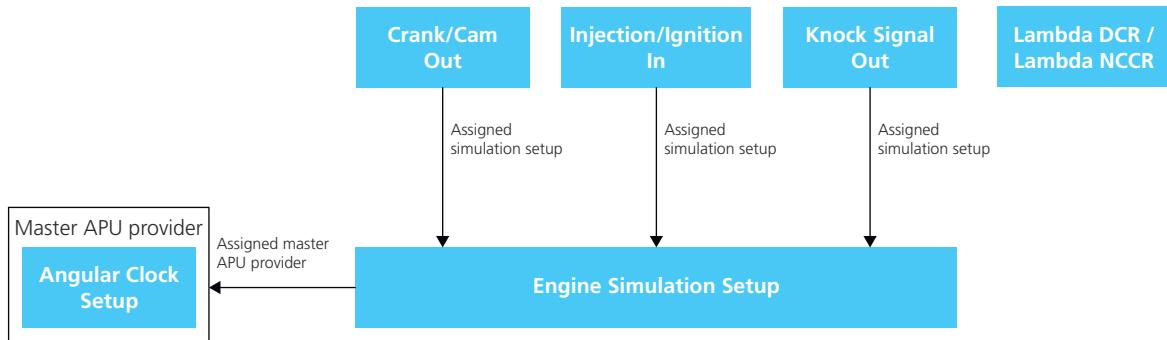
Knock Signal Out The Knock Signal Out function block simulates knock signals of specified cylinders of the virtual piston engine.

Lambda DCR The Lambda DCR function block simulates wide-band lambda probes by applying direct current regulation.

Lambda DCR The Lambda NCCR function block simulates wide-band lambda probes by applying Nernst-controlled current regulation.

Function block dependencies

The engine simulation function blocks Crank/Cam Out, Injection/Ignition In, Knock Signal Out, Lambda DCR, and Lambda NCCR can be used independently from each other. Except for Lambda DCR/Lambda NCCR function blocks, they all require an Engine Simulation Setup function block.



The **Engine Simulation Setup** block provides engine characteristics and makes use of a crankshaft angle counter via the **Angular Clock Setup** function block, which serves as master APU provider.

Available demo project

ConfigurationDesk provides the **EngineConfiguration** demo project. This project shows an implementation of the engine simulation functionality. You can use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to [EngineConfiguration Project: Simulating an Engine \(ConfigurationDesk Demo Projects\)](#).

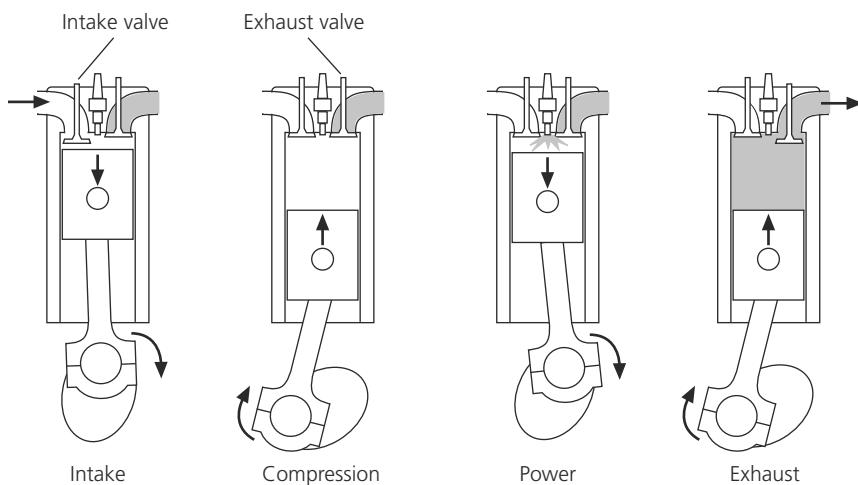
Operation Principle of Four-Stroke Piston Engines

Objective

To familiarize you with the basic functionality and terminology of four-stroke piston engines.

Four-stroke cycle

Today, internal combustion engines in cars, trucks, motorcycles, aircraft, construction machinery and many others, most commonly use a four-stroke cycle.



The four strokes make up one engine cycle and are as follows:

1. Intake:

The piston descends from the top to the bottom of the cylinder, reducing the pressure inside it. A mixture of fuel and air is forced into the cylinder. The intake valves then close.

2. Compression:

With both intake and exhaust valves closed, the piston returns to the top of the cylinder, compressing the fuel-air mixture.

3. Power:

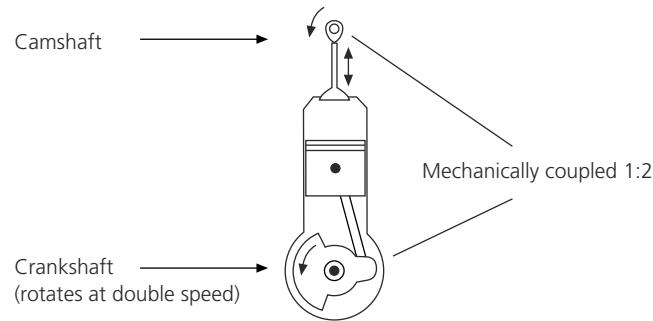
While the piston is at or close to top dead center, the compressed air-fuel mixture is ignited, usually by a spark plug (for a gasoline or spark ignition engine) or by the heat and pressure of compression (for a diesel cycle or compression ignition engine). The massive pressure resulting from the combustion of the compressed fuel-air mixture drives the piston back down toward bottom dead center with tremendous force.

4. Exhaust:

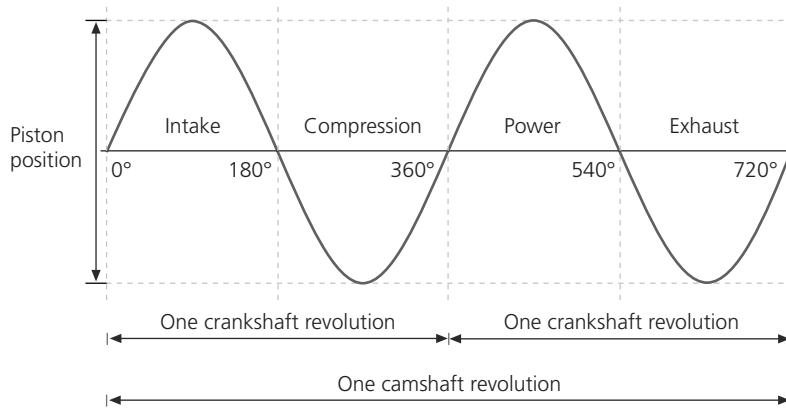
The piston once again returns to top dead center while the exhaust valve is open. This action evacuates the products of combustion from the cylinder by pushing the spent fuel-air mixture through the exhaust valves.

Crankshaft and camshaft

The reciprocating linear piston motion is translated into crankshaft rotation. The crankshaft transmits the engine power to the wheels via the gearbox. Commonly, one or even multiple camshafts control the intake and exhaust valves. They are mechanically coupled to the crankshaft. The four-stroke operation principle requires that the crankshaft rotates at double speed.

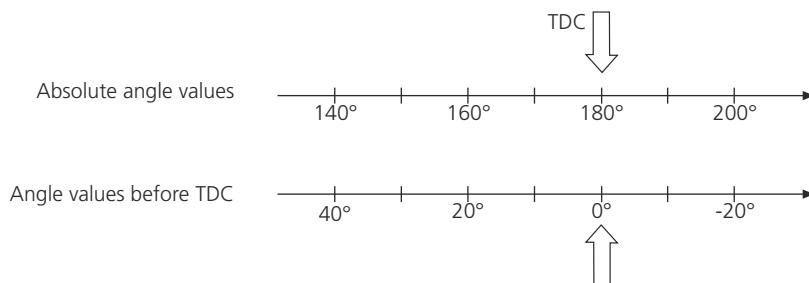


One engine cycle of a four-stroke piston engine corresponds to two crankshaft revolutions and one camshaft revolution. The angular position of the crankshaft corresponds to the position of the piston. Thus, if you know the position of the crankshaft, you can derive the position of the piston. However, you do not know the stroke, for example, whether it is compression or exhaust. To identify the stroke, the angular position of the camshaft must also be known. The following illustration shows the angular position of the crankshaft during one engine cycle.



Top dead center

Top dead center (TDC) is the upper reversal point of a piston after compression and exhaust, and is represented by an angle value in the range $0^\circ \dots 720^\circ$. TDC is a cylinder-specific attribute, and each cylinder has its own TDC. TDC is the datum point from which engine timing measurements are made. For example, injection and ignition timing is normally specified relative to the TDC as degrees before top dead center (BTDC). Positions before TDC have positive angle values, positions after TDC have negative ones.



Automatic engine stop/start

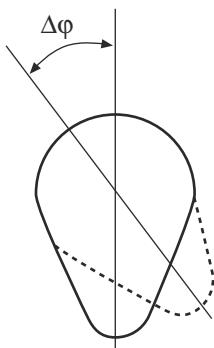
To save fuel, modern cars stop and restart their engines in a traffic jam or at a stop light automatically. The Smart Idle Stop System (SISS), for example, is an automatic stop/start method that does not use the electric starter to crank the engine back to life after shutdown. SISS stops the engine with all of the pistons near the midpoint of their strokes. To restart the engine, fuel is sprayed into the cylinder on its compression stroke and the corresponding spark plug is fired. Combustion drives the piston down, temporarily spinning the crankshaft backward. This reverse rotation compresses the fuel and air in an adjacent cylinder. Firing the spark plug in that cylinder initiates a second combustion event that halts the reverse rotation and restores normal engine operation.

Note

For the detection of reverse crankshaft rotation, a Reverse Crank sensor is needed (refer to [Intelligent crank sensor with direction detection](#) on page 453).

Camshaft phase shift

A camshaft phase shift is commonly a function of speed and load, and aims to save fuel and increase power. It means an angular displacement of a camshaft relative to the coupled crankshaft during operation, triggered by an external mechanism. Typically, a camshaft phase shift is in the range $0^\circ \dots 60^\circ$. The following illustration shows the angular displacement $\Delta\varphi$ of a cam when the camshaft is shifted.



Angular Clock Setup

Where to go from here

Information in this section

Introduction (Angular Clock Setup).....	369
Overviews (Angular Clock Setup).....	370
Configuring the Function Block (Angular Clock Setup).....	371

Introduction (Angular Clock Setup)

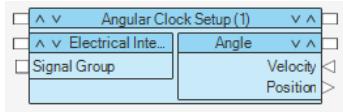
Introduction to the Function Block (Angular Clock Setup)

Function block purpose

The Angular Clock Setup function block configures and initializes the access to an angular processing unit (APU). The function block works as a provider: Other function blocks can use it to access an APU.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Controlling the rotational speed of a virtual engine as defined from the behavior model.
- Providing the access to angle units of SCALEXIO Processing Units or SCALEXIO processor boards.
- Providing the angular position of a virtual engine to the behavior model.

Supported channel types

The Angular Clock Setup function block supports the following channel types:

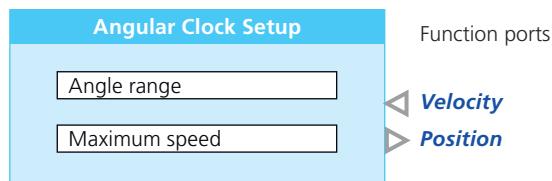
	SCALEXIO	MicroAutoBox III
Channel type	Angle Unit Set	-
Hardware	<ul style="list-style-type: none"> ▪ SCALEXIO Processing Unit ▪ DS6001 Processor Board 	-

Oversviews (Angular Clock Setup)

Overview of Ports and Basic Properties (Angular Clock Setup)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Velocity**

This function import lets you define the rotational speed and direction of the virtual engine from within the behavior model.

Value range	-1,200,000 °/s ... +1,200,000 °/s: <ul style="list-style-type: none"> ▪ > 0 °/s: Engine rotates forward ▪ 0 °/s: Engine stops ▪ < 0 °/s: Engine rotates backward
Dependencies	-

Position

This function outport writes the current angle position of the virtual engine to the behavior model.

Value range	Depends on the setting of the Angle range property: <ul style="list-style-type: none"> ▪ 0° ... 720° ▪ 0° ... 360°
Dependencies	-

Signal Ports The Angular Clock Setup function block does not provide signal ports.

Tunable properties The Angular Clock Setup function block type does not provide tunable properties.
Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (Angular Clock Setup)

Where to go from here	Information in this section
	Configuring the Basic Functionality (Angular Clock Setup)..... 371
	Configuring Standard Features (Angular Clock Setup)..... 372

Configuring the Basic Functionality (Angular Clock Setup)

Overview The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Selecting the angle range of the piston engine.
- Assigning an angle unit.
- Specifying the maximum expected speed.

Selecting the angle range of the piston engine You have to select the angle range that relates to one engine cycle. You can select either 360° or 720°. The engine cycle of a four-stroke piston engine, for example, covers 720°.

The angle range does not influence the step size of the angular processing unit (APU). The angle step size is always 0.011°, rounded to 3 decimal digits.

Assigning an angle unit You have to assign an angle unit of a SCALEXIO Processing Unit or processor board which is part of your SCALEXIO system. For more information on angle units and angular processing units, refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Specifying the maximum expected speed

The angular speed of the master APU might exceed the angular speed that can be processed by the slave APU of a hardware resource. Therefore, you can specify a maximum speed that fulfill the requirements of your application. ConfigurationDesk uses the specified maximum speed to optimize function blocks and to check if the hardware resources support the maximum speed.

Note

The Angular Clock Setup function block does not saturate the velocity of the virtual engine or the angular speed of the APU to the specified maximum speed.

Configuring Standard Features (Angular Clock Setup)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Engine Simulation Setup

Where to go from here

Information in this section

Introduction (Engine Simulation Setup).....	373
Overviews (Engine Simulation Setup).....	374
Configuring the Function Block (Engine Simulation Setup).....	375

Introduction (Engine Simulation Setup)

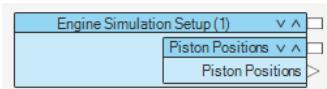
Introduction to the Function Block (Engine Simulation Setup)

Function block purpose

The Engine Simulation Setup function block type defines the basic design of virtual four-stroke piston engines, for example, number of cylinders.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Specifying the cylinders (number, status, and top dead centers) of the virtual piston engine.
- Providing the angular positions of the virtual engine's cylinders to the behavior model.

Oversviews (Engine Simulation Setup)

Where to go from here

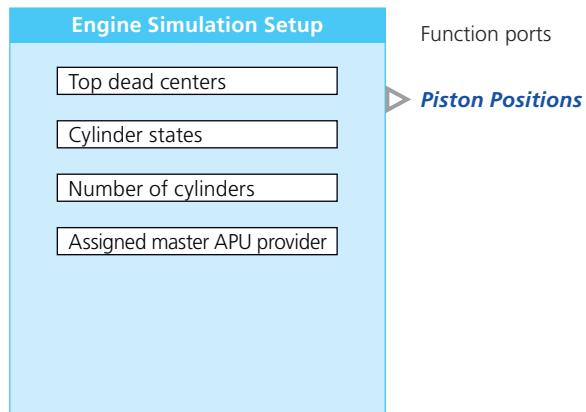
Information in this section

- [Overview of Ports and Basic Properties \(Engine Simulation Setup\)..... 374](#)
- [Overview of Tunable Properties \(Engine Simulation Setup\)..... 375](#)

Overview of Ports and Basic Properties (Engine Simulation Setup)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Piston Positions

This function outport writes the current angle positions of the virtual cylinders to the behavior model. The position values are provided relative to the TDC as degrees after top dead center (ATDC).

Value range	0° ... 720°
Dependencies	-

Signal Ports

The Engine Simulation Setup function block does not provide signal ports.

Overview of Tunable Properties (Engine Simulation Setup)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Cylinder states	✓	-
Top dead centers	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Engine Simulation Setup)

Where to go from here

Information in this section

- | | |
|--------------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Engine Simulation Setup)..... | 375 |
| Configuring Standard Features (Engine Simulation Setup)..... | 376 |

Configuring the Basic Functionality (Engine Simulation Setup)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Assigning a master APU provider
- Specifying the virtual piston engine.

Assigning a master APU provider

To receive the current piston positions of a virtual piston engine, you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Note

The Engine Simulation Setup function block supports only the 720° angle range. If you assign a master APU provider with a 360° angle range, it takes two APU cycles to run through the 720° angle range of the simulated piston engine. Therefore, the camshaft position of the engine simulation is not clearly related to the angle values of the master APU provider.

- Assign a master APU provider with a 720° angle range.

Specifying the virtual piston engine

The Engine Simulation Setup function block lets you configure the following characteristics of the virtual piston engine:

- Number of cylinders.
- Top Dead Center of each cylinder.
- Status of each cylinder (active/inactive).

Configuring Standard Features (Engine Simulation Setup)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Injection/Ignition (Voltage In, Current In)

Where to go from here

Information in this section

Introduction to Injection and Ignition Pulse Measurement.....	377
Injection/Ignition Voltage In.....	395
Injection/Ignition Current In.....	425

Introduction to Injection and Ignition Pulse Measurement

Where to go from here

Information in this section

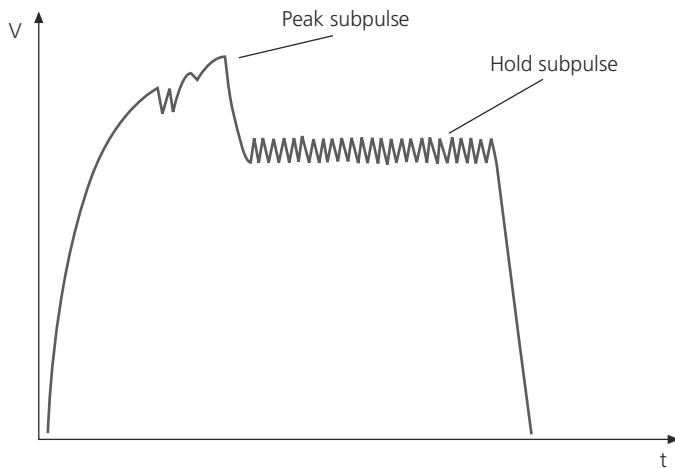
Basics on Measurement and Analysis of Injection and Ignition Pulses.....	377
Details on Window-Based Pulse Measurement Results.....	384
Details on Continuous Pulse Measurement Results.....	390
SCALEXIO Versus DS2211 Injection/Ignition Signal Evaluation.....	392

Basics on Measurement and Analysis of Injection and Ignition Pulses

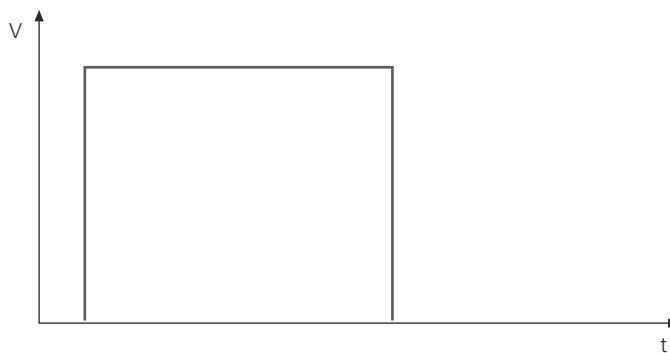
Typical injection and ignition signals

The SCALEXIO system receives injection and ignition signals from the real ECU. It counts and analyzes the injection and ignition pulses. Analyzing means capturing the start angle, end angle, and duration of the pulses.

Injection signal The following illustration shows a typical injection signal. The peak subpulse quickly opens the injection nozzle. The hold subpulse keeps the nozzle open.



Ignition signal The following illustration shows a typical ignition control signal. The pulse triggers the spark generation.



Pulse detection algorithm

The incoming signal is analyzed by a comparator. The comparator detects the beginning of a pulse and its end. The comparator transforms the input signal into a digital pulse pattern.

You can specify the following pulse detection properties:

- Threshold
- Hysteresis
- Edge type

Each property is a vector (width = 2). The first element refers to the pulse start, the second to the pulse end. The properties are provided from within the behavior model (function ports).

Note

Even small changes of the threshold, hysteresis, and/or edge type at run time, e.g., for calibration purposes, may lead to unpredictable system response. Thus, you must exactly know the incoming signal.

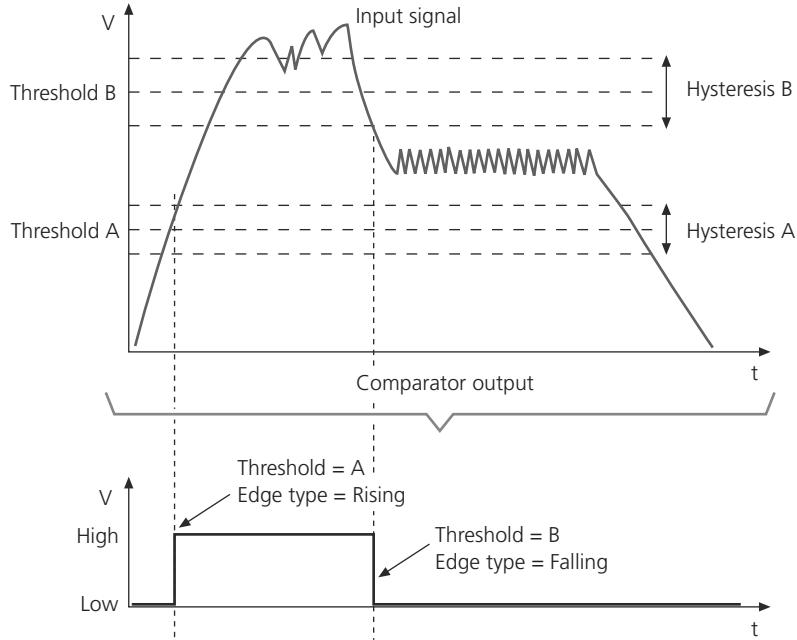
When a pulse has been detected, its pattern (start/end angles, start/end times, and/or duration) is captured. The pulse duration is derived from the time difference between the beginning of the pulse and its end.

Pulse configuration examples The properties for the pulse detection are provided from within the behavior model via the affected function ports.

The following configuration detects the peak subpulse of an injection signal:

- Threshold = [Threshold A, Threshold B]
- Hysteresis = [Hysteresis A, Hysteresis B]
- Edge type = [Rising (1), Falling (2)]

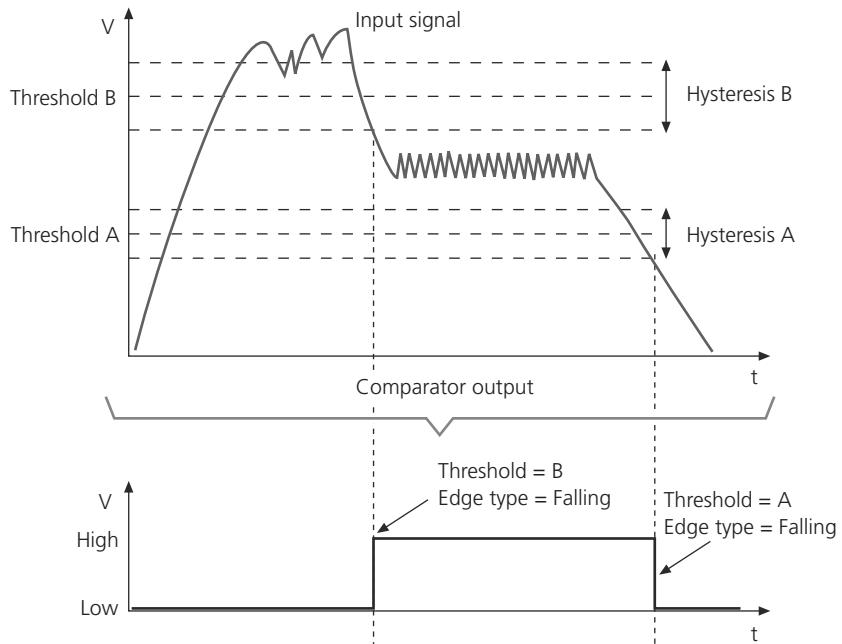
The following illustration shows how the properties affect pulse detection.



The next configuration detects the hold subpulse of the same injection signal:

- Threshold = [Threshold B, Threshold A]
- Hysteresis = [Hysteresis B, Hysteresis A]
- Edge type = [Falling (2), Falling (2)]

The following illustration shows how the properties affect pulse detection.



Preliminary calculation of the pulse duration A preliminary pulse duration is calculated when the sampling step of the behavior model starts while a pulse is active. This preliminary duration is provided until the end of the pulse is detected and the actual pulse duration can be calculated:

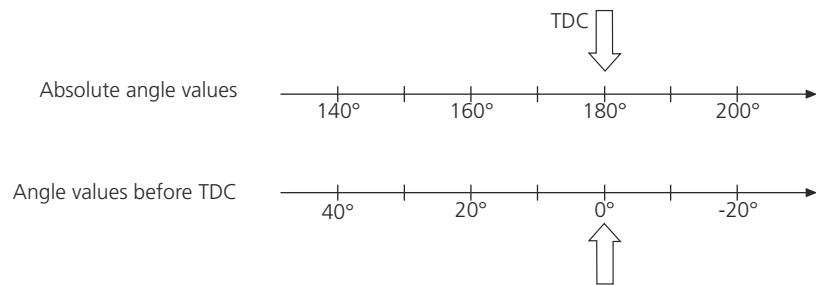
$$\text{Duration}_{\text{Preliminary}} = \text{Timestamp}_{\text{Sampling step}} - \text{Timestamp}_{\text{Pulse start}}$$

$\text{Duration}_{\text{Actual}} = \text{Timestamp}_{\text{Pulse end}} - \text{Timestamp}_{\text{Pulse start}}$

Note

Such an interim result is not calculated for end angles and end times. Here, the last pulse measurement result is returned which should be ignored until Pulse State has changed to 0 (Finished).

Angle values before TDC All the captured angle values which describe injection and ignition pulse patterns are before TDC and expressed as positive values in relation to TDC.

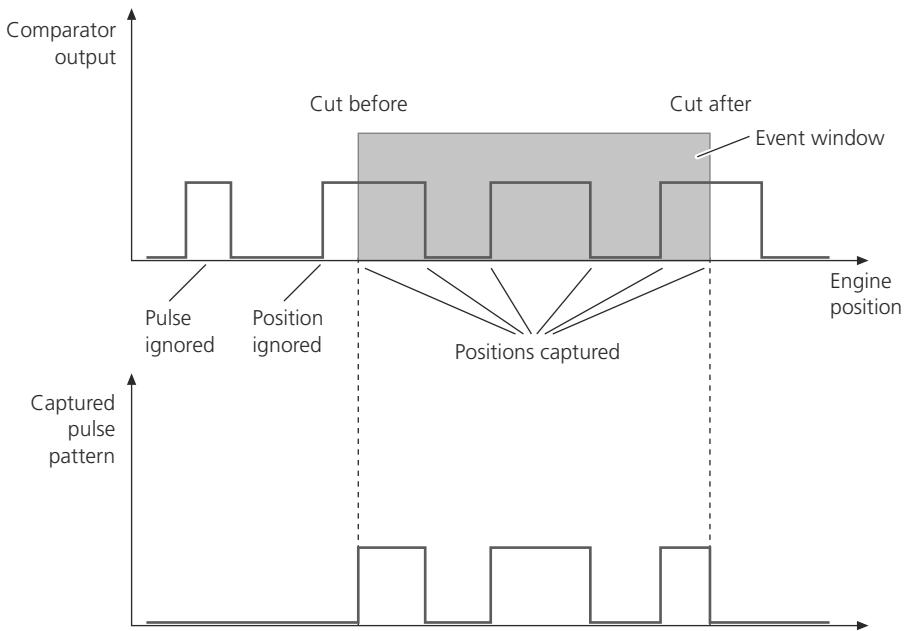


Accuracy of angle values The accuracy of angle values is 0.011° but decreases with higher speed. The loss of accuracy depends on the function (current versus voltage input) and the I/O board used.

Window-based pulse capture

You can define an angle range within the engine cycle as an event window in which injection and ignition pulses are detected. Pulses that occur outside the event window are ignored. You can define up to two event windows and use the second event window for simulation scenarios such as the aftertreatment of exhaust gases.

The following illustration shows the principle of window-based pulse capture.



If a pulse starts before (= cut before) or ends after (= cut after) an event window, the boundaries of the event window are captured instead of the outside pulse position.

Event window size The valid angle range of event windows is $[-1440^\circ \dots 1440^\circ]$ in relation to TDC, because injection can start in engine cycle n and end in engine cycle n+1. Injection then extends over two engine cycles.

Note

- An event window's start angle value is greater than its end angle value because angle values are expressed as positive values in relation to TDC (refer to [Angle values before TDC](#) on page 381).
- For one event window: The start angle and the end angle must not span an angle range greater than 720° .
For two event windows: The start angle of the first event window and the end angle of the second event window must not span an angle range greater than 720° .
- Event windows must not overlap. Note that the end angle of an event window always keeps a distance of at least 0.5° to a consecutive start angle. Events are not detected during this gap.

Continuous pulse capture

Continuous pulse capture has the following characteristics:

- Non-ending event window (actually, no event window is used and Number of event windows must be set to 0). The detected start and end angles fit the range $[TDC - 360^\circ \dots TDC + 360^\circ]$, with 0° representing the TDC.

- All pulses up to Number of expected pulses are captured (max: 100) and the data is written to a buffer. The size of the buffer equals Number of expected pulses.
- You cannot make use of trigger events.

Basic signal analysis

Basic signal analysis is always enabled and provides overall data and pulse-specific data.

Overall pulse capture data The following function outports provide the overall pulse capture data:

Function Outport	Description
Pulse Count	Returns the number of pulses that actually occurred within the event window.
Pulse State	Indicates whether the current pulse is finished. <ul style="list-style-type: none"> ▪ 0: Pulse is finished (no pulse) ▪ 1: Pulse is active
Pulse Cut State	Indicates whether pulses have been cut by the event window: <ul style="list-style-type: none"> ▪ 0: Event window completely wraps all pulses ▪ 1: Event window has cut the first pulse ▪ 2: Event window has cut the last pulse ▪ 3: Event window has cut the first pulse and the last pulse
Window State	Indicates whether the current value of the angle counter is within the event window: <ul style="list-style-type: none"> ▪ 0: Outside event window ▪ 1: Within event window
Buffer State	Indicates an overflow of the internal buffer: <ul style="list-style-type: none"> ▪ 0: Buffer OK ▪ 1: Buffer overflow

Pulse-specific data The following function outports provide the pulse-specific data:

Function Outport	Description
Pulse Start ¹⁾	Returns the start angles [°] of the pulses captured.
Pulse End ¹⁾	Returns the end angles [°] of the pulses captured.
Pulse Duration	Returns the durations [sec] of the pulses captured.
Pulse Start Timestamps ¹⁾	Returns the start times [sec] of the pulses captured.
Pulse End Timestamps ¹⁾	Returns the end times [sec] of the pulses captured.

¹⁾ By default, this function outport is disabled.

Extended signal analysis

Extended signal analysis is disabled by default, you can enable it via the Extended signal analysis property.

The following function outports provide the pulse-specific data:

Function Outport	Description
Pulse Sample Count	Returns the number of data points which the pulses captured consist of.
If the Signal type property is set to Voltage (default):	
Pulse Average Voltage ¹⁾	Returns the mean voltage value [V] of the pulses captured.
Pulse Minimum Voltage ¹⁾	Returns the minimum voltage value [V] of the pulses captured.
Pulse Maximum Voltage ¹⁾	Returns the maximum voltage value [V] of the pulses captured.
If the Signal type property is set to Current:	
Pulse Average Current ¹⁾	Returns the mean current value [A] of the pulses captured.
Pulse Minimum Current ¹⁾	Returns the minimum current value [A] of the pulses captured.
Pulse Maximum Current ¹⁾	Returns the maximum current value [A] of the pulses captured.

¹⁾ By default, this function outport is disabled.

Details on Window-Based Pulse Measurement Results

Data handling

The result of injection and ignition signal measurement is stored in *one or two data sets*. Each of the up to two event windows has its own data set. The data set is updated before each sampling step of the behavior model. The data sets and a buffer status flag (0: Buffer OK, 1: Buffer overflow) are stored in a common buffer. The following table shows the structure of the data set:

One data set per event window Each event window has its own data set for data transfer:

Data Set	
Overall data:	
Pulse Count	[0 ... 100] ¹⁾
Pulse State	0: Finished / 1: Active ¹⁾
Pulse Cut State	0: No cut / 1: Cut before / 2: Cut after / 3: Cut before and after ¹⁾

Data Set	
Window State	0: Closed / 1: Active ¹⁾
Pulse-specific data subsets [1 ... Number of expected pulses]:	
Pulse Start	[-1440° ... +1440°]
Pulse Stop	[-1440° ... +1440°]
Pulse Duration	[s]
Pulse Start Timestamps	[s]
Pulse End Timestamps	[s]
Pulse Sample Count	[0 ... 100] ¹⁾
Pulse Average Current/Voltage	I _{min} ... I _{max} / U _{min} ... U _{max}
Pulse Minimum Current/Voltage	I _{min} ... I _{max} / U _{min} ... U _{max}
Pulse Maximum Current/Voltage	I _{min} ... I _{max} / U _{min} ... U _{max}

¹⁾ Default: 0

The data of the first pulse captured is stored in the first data subset. Subsequently the data of the second pulse captured is stored in the second data subset, and so on (first-come-first-stored).

Note

The number of pulse-specific data subsets is fixed at Number of expected pulses.

Data reset The global event window data is reset to defaults at each start of the corresponding event window. The pulse-specific data subsets hold the existing values until they are overwritten by new values. The write position within each pulse-specific data subset, defining where the next pulse event is stored, is reset at each start of the corresponding event window as well.

Data transfer to the behavior model The behavior model reads the measurement results via the function outports from the buffer. The behavior model always reads out complete data sets.

The following table shows the availability of data at the function outports.

Function Outport	Vector size
Event window data: ▪ Pulse Count ▪ Pulse State ▪ Pulse Cut State ▪ Window State	Value specified at the Number of event windows property
Pulse data: ▪ Pulse Start ¹⁾ ▪ Pulse End ¹⁾ ▪ Pulse Duration ▪ Pulse Start Timestamps ¹⁾ ▪ Pulse End Timestamps ¹⁾ ▪ Pulse Sample Count ¹⁾	Total of the values specified at the Number of expected pulses property of the event windows ([E ₁ , ..., E _N , E ₁ , ..., E _M]): ▪ N = value specified at the Number of expected pulses

Function Outport	Vector size
<ul style="list-style-type: none"> Pulse Average Current/Voltage¹⁾ Pulse Minimum Current/Voltage¹⁾ Pulse Maximum Current/Voltage¹⁾ 	property of the 1st event window M = value specified at the Number of expected pulses property of the 2nd event window
Buffer State	1

¹⁾ By default, this function outport is disabled.

Example Suppose you have specified two event windows, Number of expected pulses is set to [2, 3], and the following pulse start angles have been captured:

- 1st window: (297°, 213°)
- 2nd window: (654°, 592°, 512°)

The Pulse Start function port hence outputs a 5-element vector signal, which reads as follows:

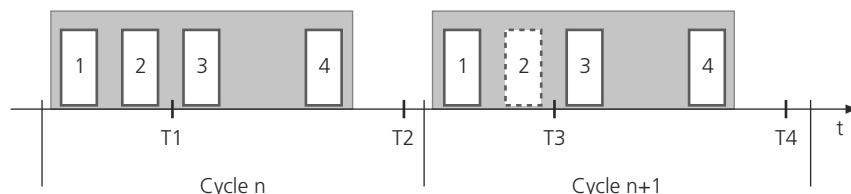
(297°, 213°, 654°, 592°, 512°)

Example of a window-based measurement result

Suppose pulse measurement is carried out as follows:

- Number of event windows = 1
- Number of expected pulses = 3
- Behavior model reads results at T1 ... T4
- Pulse #2 is missing during engine cycle n+1

Suppose Pulse Count and Pulse Duration are measured.



The following table shows the measurement results for different read accesses (T1, ..., T4) by the behavior model. If Pulse Count is smaller than Number of expected pulses (here: 3), the duration values of the pulses still to come do not relate to the current cycle but to the preceding cycle. If an expected pulse is missing, its duration value relates to the pulse that occurs next. At T4, for example, the 1st element of the pulse duration vector relates to pulse #1 and cycle n+1, the 2nd element to pulse #3 and cycle n+1, and the 3rd element to pulse #4 and cycle n+1.

Sample	Pulse Count Vector ¹⁾	Pulse Duration Vector ²⁾		
		1 st Element	2 nd Element	3 rd Element
T1	2	#1, n	#2, n	#3, n-1
T2	4	#1, n	#2, n	#3, n
T3	1	#1, n+1	#2, n	#3, n
T4	3	#1, n+1	#3, n+1	#4, n+1

¹⁾ Vector size = 1²⁾ Vector size = 3

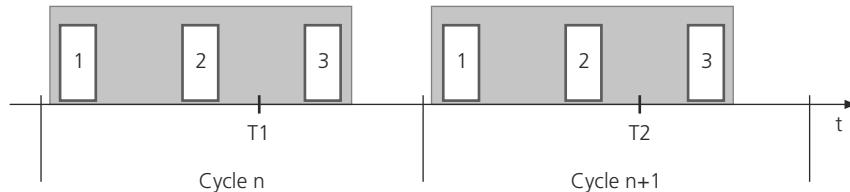
Periodic sampling

Suppose pulse measurement is carried out as follows:

- Number of expected pulses = 3
- Behavior model reads results once an engine cycle (periodically)

Suppose Pulse Count and Pulse Duration are measured.

If the behavior model reads the pulse duration vector from the buffer periodically, and the sample rate is too slow, elements of the pulse duration vector might not relate to the current engine cycle but to the prior one. In the following example, the 3rd element of the pulse duration vector always belongs to the foregoing engine cycle.



The following table shows the measurement results.

Sampling Time	Pulse Count	Pulse Duration Vector ¹⁾		
		1 st Element	2 nd Element	3 rd Element
T1	2	#1, n	#2, n	#3, n-1
T2	2	#1, n+1	#2, n+1	#3, n

¹⁾ Vector size = 3

Tip

Increase the sample rate of the behavior model so that the data set is read when Window State equals **Closed**.

Required sample rate As a rule of thumb the sample rate of the behavior model must be at least twice as fast as the engine cycle. An engine cycle at 20,000 rpm, for example, lasts 6 ms, hence the sample rate time of the behavior model must be 3 ms or faster.

Trigger events

You can use the following trigger events, which can be sent to the behavior model:

- First rising edge of a pulse in event window
- End of event window

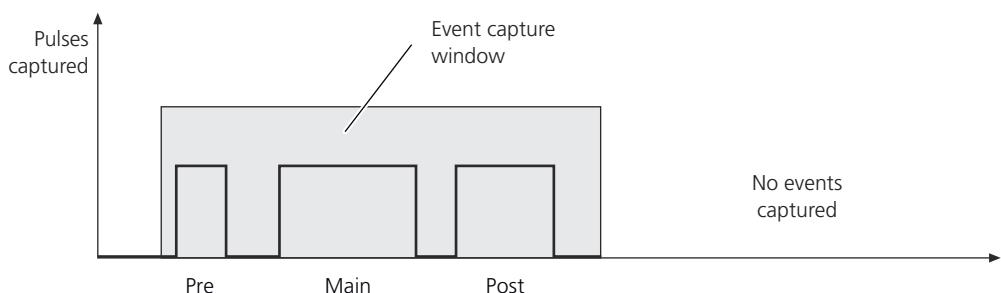
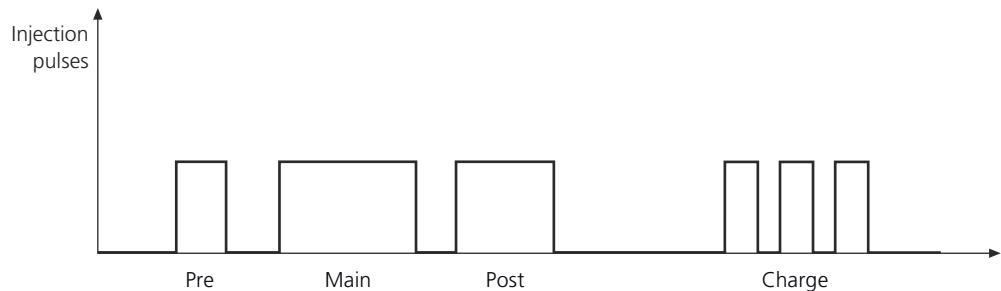
You can enable trigger events via the **Event generation** property, refer to [Configuring Standard Features \(Injection/Ignition Voltage In\)](#) on page 409.

Note

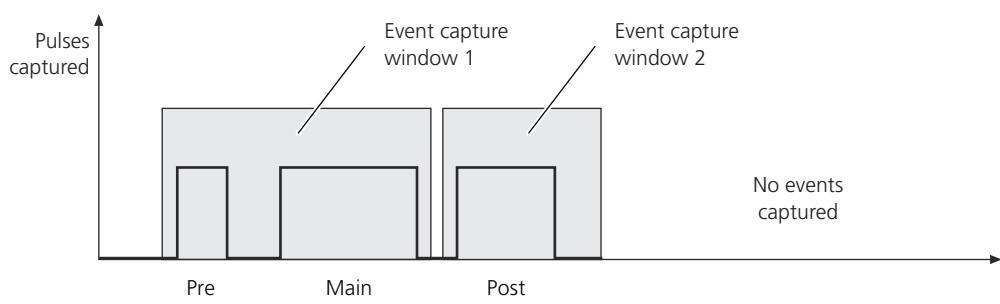
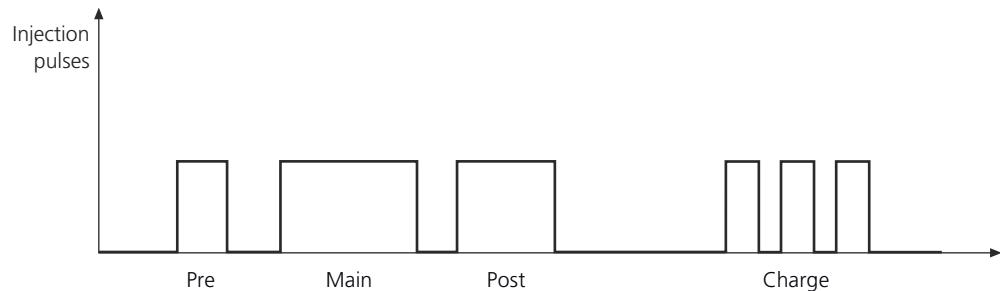
Event generation is not possible for continuous pulse measurement.

Event windows for special use cases

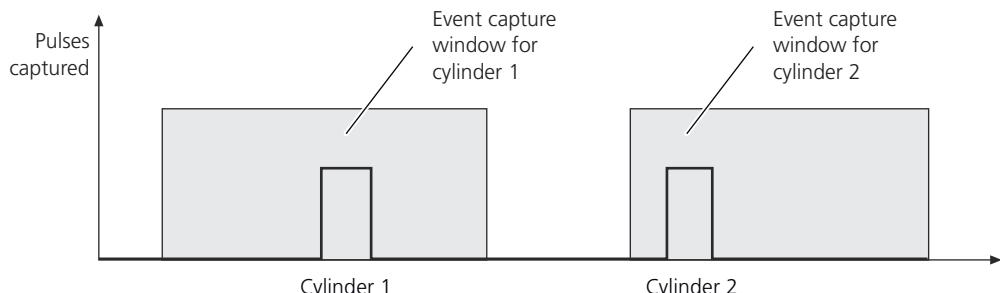
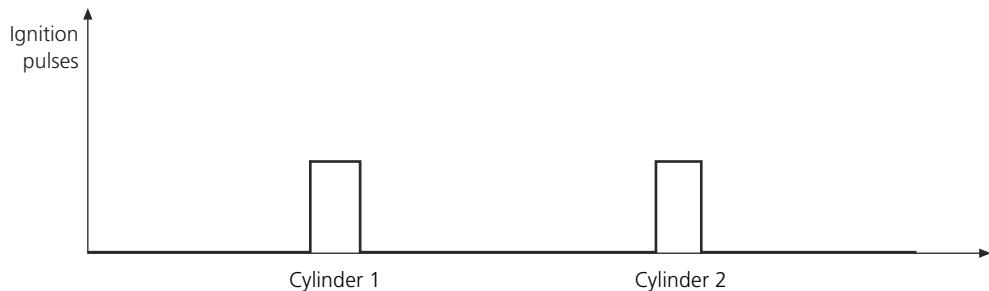
Common Rail Injection The start positions and durations of several injection pulses can be captured. Additional charge pulses can be ignored using an event window of an appropriate length.



You can use two event capture windows to distinguish between torque- and exhaust-related injection.



Double Spark Ignition There are two ignition pulses per coil but only one signal for the two cylinders. The cylinder is identified by its engine angle. You can use two event windows to specify possible ignition pulse positions.



Details on Continuous Pulse Measurement Results

Data handling

The result of injection and ignition signal measurement is stored in one data set. The data set is updated before each sampling step of the behavior model. The data sets and a buffer status flag (0: Buffer OK, 1: Buffer overflow) are stored in a common buffer. The following table shows the structure of the data set:

One single data set One single data set is used for data transfer:

Data Set	
Overall data:	
Pulse Count	[0 ... 100] ¹⁾
Pulse State	0: Finished / 1: Active ¹⁾
Pulse Cut State	0: No cut
Window State	1: Active
Pulse-specific data subsets [1 ... Number of expected pulses]:	
Pulse Start	[-1440° ... +1440°]
Pulse Stop	[-1440° ... +1440°]
Pulse Duration	[s]
Pulse Start Timestamps	[s]
Pulse End Timestamps	[s]
Pulse Sample Count	[0 ... 100] ¹⁾
Pulse Average Current/Voltage	I _{min} ... I _{max} / U _{min} ... U _{max}
Pulse Minimum Current/Voltage	I _{min} ... I _{max} / U _{min} ... U _{max}
Pulse Maximum Current/Voltage	I _{min} ... I _{max} / U _{min} ... U _{max}

¹⁾ Default: 0.

The data of the first pulse captured is stored in the first data subset. Subsequently the data of the second pulse captured is stored in the second data subset, and so on (first-come-first-stored).

Note

The number of pulse-specific data subsets is fixed at Number of expected pulses.

Data reset The global event data and the pulse-specific data subsets are reset before each sampling step of the simulation model. All pulses detected between previous and current sampling step are added to the data set. Additionally, if a pulse start was detected without corresponding pulse end, this pulse is added to the data set, by incrementing Pulse Count, setting Pulse State to Active and adding its angle to the Pulse Start data set.

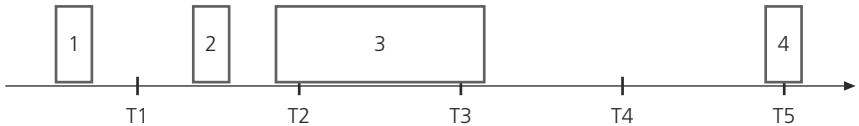
Note

A pulse start with no corresponding pulse end is added to the data set in the following sample steps, until its end pulse is detected. After storing both start and end angle once in the data set, they are removed in the next sampling step.

Example of a continuous measurement result

Suppose pulse measurement is carried out as follows:

- Number of event windows = 0
- Number of expected pulses = 3
- Behavior model reads reads at T1 ... T5



The following table shows the measurement results for different sample steps of the behavior model (T1 ... T5). If Pulse Count is smaller than Number of expected pulses (here: 3), the remaining entries are not valid

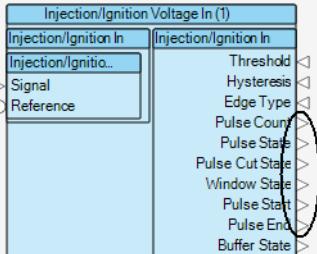
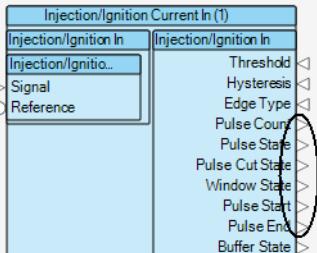
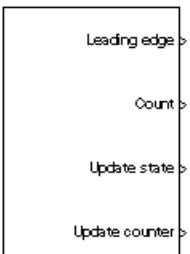
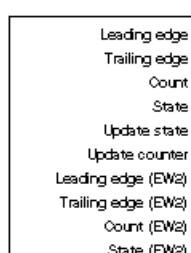
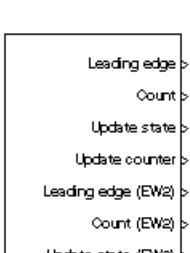
Sample	Pulse Count	Pulse State	Start Angles	End Angles
T1	1	0	#1	#1
T2	2	1	#2, #3	#2
T3	1	1	#3	—
T4	1	0	#3	#3
T5	1	1	#4	—

As a general rule, the value stored in Pulse Count describes the number of valid entries in the Pulse Start data subset , and Pulse Count minus Pulse State describes the number of valid entries in the Pulse End data subset.

SCALEXIO Versus DS2211 Injection/Ignition Signal Evaluation

Purpose

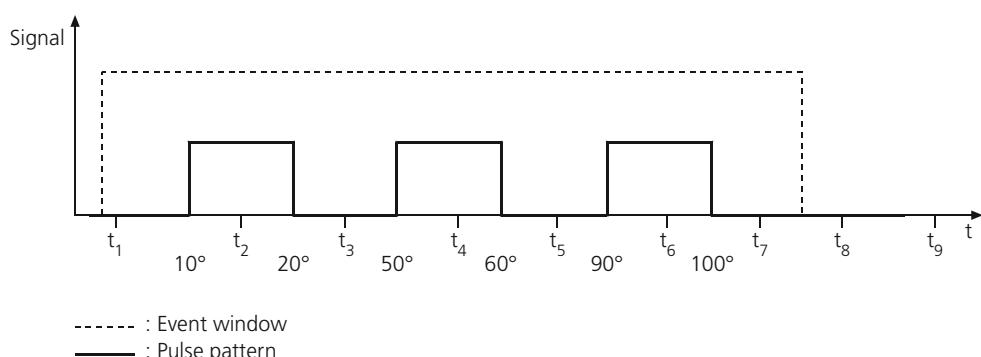
To compare the output values of these hardware-specific injection/ignition signal capture blocks (SCALEXIO versus DS2211):

SCALEXIO	DS2211
Injection/Ignition Voltage In	DS2211VARAPU_AUXCAP_Bx_Cy
	
	
	DS2211VARAPU_AUXCAP_B3_C1
	
	DS2211VARAPU_AUXCAP_B4_C1
	

Example of signal capture

This example demonstrates how the output values of a capturing block change when a signal is captured in window-based pulse capture mode.

The following illustration shows the captured signal. Three pulses are expected within one capture event window. The initial value of the pulses' start and end position is set to 999 degrees.



The following tables (SCALEXIO versus DS2211) show the values of the properties when the pulses are captured for the first time. Updated or changed data is italicized. t_n are the sample steps.

SCALEXIO

Output	Values									
Block execution	<i>t₁</i>	<i>t₂</i>	<i>t₃</i>	<i>t₄</i>	<i>t₅</i>	<i>t₆</i>	<i>t₇</i>	<i>t₈</i>	<i>t₉</i>	
Pulse data (angle values) ¹⁾ :										
Start1	999	<i>10</i>	10	10	10	10	10	10	10	10
Start2	999	999	999	<i>50</i>	50	50	50	50	50	50
Start3	999	999	999	999	999	<i>90</i>	90	90	90	90
End1	999	999	<i>20</i>	20	20	20	20	20	20	20
End2	999	999	999	999	<i>60</i>	60	60	60	60	60
End3	999	999	999	999	999	999	<i>100</i>	100	100	100
Event window data:										
Pulse Count	<i>0</i>	1	1	2	2	3	3	3	3	3
Pulse State	<i>0</i>	1	0	1	0	1	0	0	0	0
Window State	1	1	1	1	1	1	1	0	0	0
Pulse Cut State	<i>0</i>	0	0	0	0	0	0	0	0	0

¹⁾ Leading and trailing edges

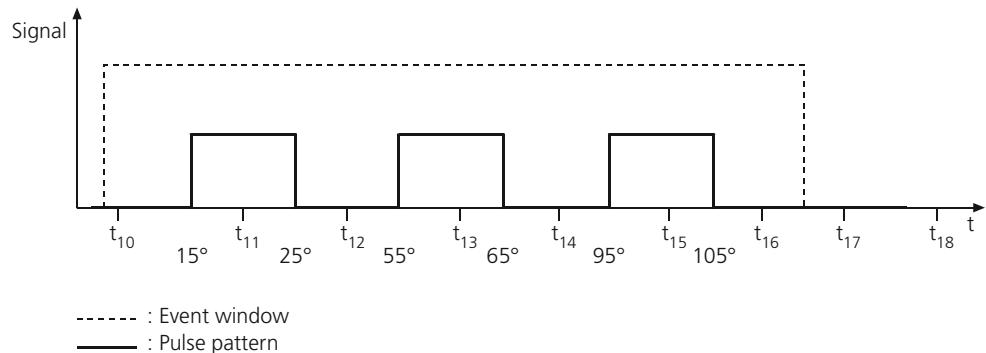
DS2211

Output	Values									
Block execution	<i>t₁</i>	<i>t₂</i>	<i>t₃</i>	<i>t₄</i>	<i>t₅</i>	<i>t₆</i>	<i>t₇</i>	<i>t₈</i>	<i>t₉</i>	
Pulse data (angle values) ¹⁾ :										
Start1	999	<i>10</i>	10	10	10	10	10	10	10	10
Start2	999	999	999	<i>50</i>	50	50	50	50	50	50
Start3	999	999	999	999	999	<i>90</i>	90	90	90	90
End1	999	999	<i>20</i>	20	20	20	20	20	20	20
End2	999	999	999	999	<i>60</i>	60	60	60	60	60
End3	999	999	999	999	999	999	<i>100</i>	100	100	100
Event window data: ²⁾										
Count	0	0	0	0	0	0	0	0	3	3
State	0	0	0	0	0	0	0	0	0	0
Update counter	0	-1	1	-2	2	-3	3	3	3	3
Update state	0	0	0	0	0	0	1	1	1	1

¹⁾ Leading and trailing edges

²⁾ The DS2211 event window data outputs are different to the SCALEXIO ones.

The following illustration shows the signal in a subsequent engine cycle. The pulses start 5 degrees later.



The following tables (SCALEXIO versus DS2211) show the values of the properties when the pulses are captured. Updated or changed data is italicized. t_n are the sample steps.

SCALEXIO

Output	Values								
Block execution	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
Pulse data:									
Start1	10	15	15	15	15	15	15	15	15
Start2	50	50	50	55	55	55	55	55	55
Start3	90	90	90	90	90	95	95	95	95
End1	20	20	25	25	25	25	25	25	25
End2	60	60	60	60	65	65	65	65	65
End3	100	100	100	100	100	100	105	105	105
Event window data:									
Pulse Count	0	1	1	2	2	3	3	3	3
Pulse State	0	1	0	1	0	1	0	0	0
Window State	1	1	1	1	1	1	1	0	0
Pulse Cut State	0	0	0	0	0	0	0	0	0

DS2211

Output	Values								
Block execution	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
Pulse data ¹⁾ :									
Start1	10	15	15	15	15	15	15	15	15
Start2	50	50	50	55	55	55	55	55	55
Start3	90	90	90	90	90	95	95	95	95
End1	20	20	25	25	25	25	25	25	25
End2	60	60	60	60	65	65	65	65	65

Output	Values									
End3	100	100	100	100	100	100	105	105	105	105
Event window data: ²⁾										
Count	3	3	3	3	3	3	3	3	3	3
State	0	0	0	0	0	0	0	0	0	0
Update counter	3	-1	1	-2	2	-3	3	3	3	3
Update state	1	0	0	0	0	0	1	1	1	1

¹⁾ Leading and trailing edges

²⁾ The DS2211 event window data outputs are different to the SCALEXIO ones.

Injection/Ignition Voltage In

Where to go from here

Information in this section

Introduction (Injection/Ignition Voltage In).....	395
Overviews (Injection/Ignition Voltage In).....	397
Configuring the Function Block (Injection/Ignition Voltage In).....	406
Hardware Dependencies (Injection/Ignition Voltage In).....	423

Introduction (Injection/Ignition Voltage In)

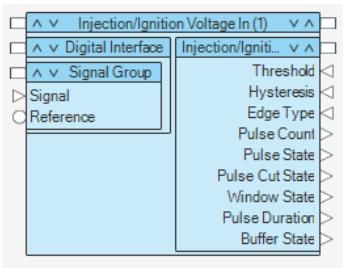
Introduction to the Function Block (Injection/Ignition Voltage In)

Function block purpose

The Injection/Ignition Voltage In function block type reads the positions and durations of the injection pulses that were generated by the ECU on the basis of a voltage input signal.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

**Main features**

These are the main features:

- Measuring injection or ignition pulses from an ECU within specified angular (event) windows or continuously.
- Buffering detected pulses.
- Supporting digital voltage measurement and analog voltage measurements.
- Providing the measurement results to the behavior model.
- Providing status information to the behavior model on pulses, event windows and the data buffer.
- Generating I/O events and providing them to the behavior model.

Supported channel types

Depending on the specified scope of signal analysis, the Injection/Ignition Voltage In function block supports a digital or a digital and an analog signal interface.

Digital Interface The Digital Interface supports the following channel types:

	SCALEXIO				MicroAutoBox III
Channel type	Flexible In 1	Flexible In 2	Flexible In 3		–
Hardware	DS2601	DS2680	DS6101		–

Analog Interface The Analog Interface is available only if Extended signal analysis is enabled. It supports the following channel types:

	SCALEXIO						MicroAutoBox III
Channel type	Flexible In 1	Flexible In 2 ¹⁾	Analog In 1 ²⁾	Flexible In 3 ²⁾	Analog In 4 ²⁾		–
Hardware	DS2601	DS2680		DS6101			–

¹⁾ Only for current measurement.

²⁾ Only for voltage measurement.

**Introduction to
Injection/Ignition pulse
measurement**

For more information, refer to [Introduction to Injection and Ignition Pulse Measurement](#) on page 377.

Oversviews (Injection/Ignition Voltage In)

Where to go from here

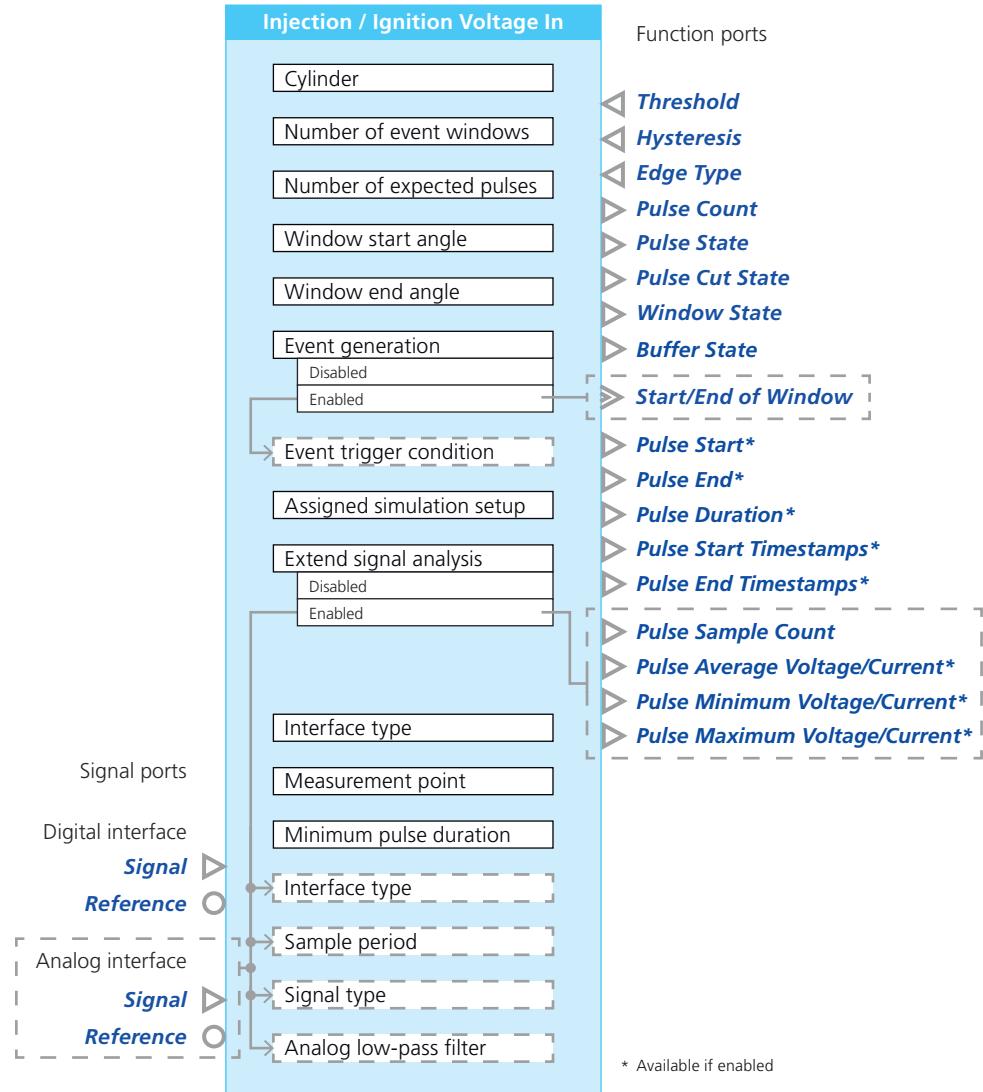
Information in this section

Overview of Ports and Basic Properties (Injection/Ignition Voltage In).....	397
Overview of Tunable Properties (Injection/Ignition Voltage In).....	405

Overview of Ports and Basic Properties (Injection/Ignition Voltage In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



If you enable Extended signal analysis, an analog electrical interface is added. This interface also provides identically labeled signal ports. However, in this case the assigned hardware resources affect the availability of all signal ports of the function block basically. For details on the mapping of the signal ports, refer to [Mapping Signal Ports When Using Extended Signal Analysis \(Injection/Ignition Voltage In\) on page 412](#).

Threshold

This function import reads a vector with the two threshold values for pulse detection from within the behavior model. The first threshold value is used for detecting the beginning of a pulse, the second for detecting its end.

Value range	▪ $U_{min} \dots U_{max}$ (in Volt) for each entry of the vector.
-------------	-------------------------------------------------------------------

- | | |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none"> ▪ The range depends on the following: ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection/Ignition Voltage In) on page 423. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ The vector size is two. The first entry refers to the pulse start, the second to the pulse end. |
| Dependencies | – |

Hysteresis

This function import reads a vector with the hysteresis values of the two thresholds from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0.25 V ... 6 V for each entry of the vector. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ The vector size is two. The first entry refers to the pulse start, the second to the pulse end.
Dependencies	–

Edge Type

This function import reads a vector with the edge types relevant for pulse detection from within the behavior model.

Value range	<p>For each entry of the vector:</p> <ul style="list-style-type: none"> ▪ 1: Rising ▪ 2: Falling <p>The vector size is two. The first entry refers to the pulse start, the second to the pulse end.</p>
Dependencies	–

Pulse Count

This function outport writes a vector with the number of pulses that actually occurred within the event window to the behavior model. The number of pulses is *independent* of the Number of expected pulses. The pulse counter increments at every rising pulse edge. Pulses that are only partially included by the event window, i.e., cut pulses, are also counted.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 100 for each entry of the vector. ▪ The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Pulse State	This function outport writes a vector with flags to the behavior model. The flags indicate whether the current pulse is finished or not. A pulse is finished when its <i>falling edge</i> was detected.
--------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Value range	<ul style="list-style-type: none"> ▪ For each entry of the vector: ▪ 0: Finished: No pulse is active. ▪ 1: Active: The last pulse is still active. ▪ The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Pulse Cut State	This function outport writes a vector that indicates whether pulses were cut by the event window to the behavior model. If a pulse starts before (= cut before) and/or ends after (= cut after) an event window, the boundaries of the event window are captured instead of the pulse edges outside the event window.
------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Value range	<ul style="list-style-type: none"> ▪ For each entry of the vector: ▪ 0: No cut Event window completely includes the captured pulses. ▪ 1: Cut before Event window cut the first pulse, i.e., only the ending edge of this pulse is within the event window. ▪ 2: Cut after Event window cut the last pulse, i.e., only the starting edge of this pulse is within the event window. ▪ 3: Cut before and after Event window cut the first pulse and the last pulse. ▪ The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Window State	This function outport writes a vector that indicates whether the current value of the angle counter is within the event window to the behavior model. The Window State function port switching to closed thereby indicates that the data set of an event window has completely been updated with the new pulse data.
---------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Value range	<ul style="list-style-type: none"> ▪ For each entry of the vector: ▪ 0: Closed ▪ 1: Active ▪ The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Pulse Start

This function outport writes a vector with the start angles of the captured pulses to the behavior model. If the first pulse is cut by the event window (cut before), the start angle of the event window is taken into account instead.

Value range	<ul style="list-style-type: none"> ▪ -1440° ... +1440° for each entry of the vector. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse start property is set to Enabled.

Pulse End

This function outport writes a vector with the end angles of the captured pulses to the behavior model. If the last pulse is cut by the event window (cut after), the end angle of the event window is taken into account instead.

Value range	<ul style="list-style-type: none"> ▪ -1440° ... +1440° for each entry of the vector. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse start property is set to Enabled.

Pulse Duration

This function outport writes a vector with the lengths of the captured pulses (in seconds) to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 3600 s for each entry of the vector. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse start property is set to Enabled.

Buffer State

This function outport writes state information on the internal buffer to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: OK The internal buffer is OK. No error occurred. ▪ 1: Overflow An overflow of the internal buffer occurred. Too many pulses were detected and data is lost. This can happen, if the
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>sample rate of the behavior model is too slow, and/or the number of pulses counted is too high.</p> <ul style="list-style-type: none"> ▪ 2: FIFO overflow An overflow of the transmission buffer between the computation node and I/O node occurred. Too much data was transmitted and data is lost. ▪ 3: Consistency lost Data consistency between digital and analog data is lost and a reset is in progress. ▪ 4: FIFO error An unrecoverable transmission error occurred and the connection is lost. An application restart is required.
Dependencies	–

Pulse Start Timestamps

This function outport writes a vector with the start time stamp values of the captured pulses to the behavior model. If the first pulse is cut by the event window (cut before), the start time stamp of the event window is used instead of the first pulse's start time stamp.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse end timestamps property is set to Enabled.

Pulse End Timestamps

This function outport writes a vector with the the end time stamp values of the captured pulses to the behavior model. If the last pulse is cut by the event window (cut after), the end time stamp of the event window is used instead of the last pulse's end time stamp.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse end timestamps property is set to Enabled.

Pulse Sample Count

This function outport writes a vector with the number of samples (data points) of the captured pulses to the behavior model. If the last pulse is cut by the event window (cut after), the end time stamp of the event window is used instead of the last pulse's end time stamp.

Note

The pulse sample counter returns unexpected results, if pulse duration is greater than 65,535 * sample periods.

Value range	<ul style="list-style-type: none"> 1 sample ... 100 samples for each entry of the vector. <p>The number of samples is influenced by the Sample period property of the block's analog signal interface.</p> <ul style="list-style-type: none"> The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property is set to Enabled.

Pulse Average Voltage/Current

This function outport writes a vector with the mean voltage/current values of the captured pulses to the behavior model. Whether it is for voltage (default) or current depends on the setting of the Signal type property.

Note

The pulse average voltage/current calculation returns unexpected results, if pulse duration is greater than 65,535 * sample periods.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt) or $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. For specific values, refer to Hardware Dependencies (Injection/Ignition Voltage In) on page 423. The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property and then the Capture pulse average values property are set to Enabled.

Pulse Minimum Voltage/Current

This function outport writes a vector with the minimum voltage/current values of the captured pulses to the behavior model. Whether it is for voltage (default) or current depends on the setting of the Signal type property.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt) or $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. For specific values, refer to Hardware Dependencies (Injection/Ignition Voltage In) on page 423. The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property and then the Capture pulse minimum values property are set to Enabled.

Pulse Maximum Voltage/Current

This function outport writes a vector with the maximum voltage/current values of the captured pulses to the behavior model. Whether it is for voltage (default) or current depends on the setting of the Signal type property.

Value range	<ul style="list-style-type: none"> ▪ $U_{\min} \dots U_{\max}$ (in Volt) or $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. ▪ For specific values, refer to Hardware Dependencies (Injection/Ignition Voltage In) on page 423. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property and then the Capture pulse maximum values property are set to Enabled.

Start/End of Window

This event port provides an I/O event each time the first rising edge in an event window is detected or when the end of an event window is reached, according to the setting of the Event trigger condition property.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection/Ignition Voltage In) on page 423.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection/Ignition Voltage In) on page 423.
Dependencies	–

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Supported only for Flexible In 1 and Flexible In 2 channel types.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Supported only for the Flexible In 1 channel type.

Overview of Tunable Properties (Injection/Ignition Voltage In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Event trigger condition	✓	–
Cylinder	✓	–
Window start angle	✓	–
Window end angle	✓	–

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Digital Interface		
Measurement point	✓	-
Minimum pulse duration	✓	-
Analog Interface ³⁾		
Sample period	✓	-
Analog low-pass filter	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ The analog interface unit is available if the Extended signal analysis property is set to Enabled.

Configuring the Function Block (Injection/Ignition Voltage In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Injection/Ignition Voltage In).....	406
Configuring Standard Features (Injection/Ignition Voltage In).....	409
Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Voltage In).....	412

Configuring the Basic Functionality (Injection/Ignition Voltage In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Defining engine properties
- Specifying minimum pulse duration
- Specifying event windows for pulse capturing

- Enabling function ports for measuring digital values of captured pulses
- Specifying extended signal analysis (measuring analog signal values of captured pulses)
- Providing I/O events

Defining engine properties

Defining virtual piston engine The Injection/Ignition Voltage In function block refers to a specific engine design by referencing an Engine Simulation Setup function block. The Engine Simulation Setup function block provides basic configuration data of the virtual engine, for example, the number of cylinders.

Selecting cylinder for measurement The Injection/Ignition Voltage In function block lets you specify the cylinder which the incoming injection or ignition signal is related to.

If you want to measure, for example, injection and ignition signals for the same cylinder, you would need two Injection/Ignition Voltage In function blocks both addressing this cylinder.

Filtering the input signal

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the pulse duration depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Injection/Ignition Voltage In\)](#) on page 423 .

You can also change the value of the **Minimum pulse duration** property via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying event windows for pulse capturing

You can specify an angle range within the engine cycle as an event window in which injection and ignition pulses are detected. Pulses occurring outside the event window are ignored.

- You can specify one or two event windows and their start and end angles (**Number of event windows** property is set to 1 or 2).
If you specify event windows with start and end angles, the values provided at the **Threshold**, **Hysteresis**, and **Edge Type** function ports take effect at the end of a window. This avoids data loss if the values at the function ports change during the measurement of a pulse.
- You can specify one event window that covers the whole engine cycle, i.e., for continuous pulse measurement (**Number of event windows** property is set to 0).
If you specify continuous pulse measurement, the values provided at the **Threshold**, **Hysteresis**, and **Edge Type** function ports take effect immediately. Note that a change of one of these values might result in data loss in the next model step.

Enabling function ports for measuring digital values of captured pulses

The Injection/Ignition Voltage In function block provides pulse measurement features, for example, for counting pulses, measuring pulse start/stop angles, and measuring pulse start/stop times. The measured values are provided to the behavior model via specific function ports.

You have to enable the function ports so you can use them.

Measuring analog signal values of captured pulses

Analog signal measurement With the Injection/Ignition Voltage In function block, you can additionally measure the analog signal values of the captured pulses (extended signal analysis). The measured values are provided to the behavior model via specific function ports.

If the function block detects a digital pulse, its analog measurement is started. The minimum, maximum, and average voltage or current values of the pulse are determined. These values can be provided to the behavior model.

You can specify the time between two samples with the **Sample period** property. The higher the sample rate, the more precise is the analog signal measurement.

Selecting the signal type You can select the physical signal type for analog signal measurement:

- **Voltage:** The analog signal measurement is based on the voltage of the captured pulse.
- **Current:** The analog signal measurement is based on the current of the captured pulse.

Limitations:

- The **Voltage** signal type is not supported for the **Flexible In 2** channel type.
- The **Current** signal type is not supported for the **Analog In 1**, **Analog In 4** and **Flexible In 3** channel types.

Using the analog low-pass filter You can use the analog low-pass filter, for example, to reduce signal noise. The filter characteristics depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Injection/Ignition Voltage In\)](#) on page 423.

For more basic information on using the filter, refer to [Analog and Digital Filtering with the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Limitation: The analog low-pass filter is supported only for the **Flexible In 1** channel type.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Injection/Ignition Voltage In function block can generate an I/O event each time a specific condition of an event window is detected. Via the **Event trigger condition** property you can specify when to trigger the I/O event, either when the first rising edge in an event window is detected or when the end of an event window is reached.

Note

Event generation is not possible for continuous pulse measurement.

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Window property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Injection/Ignition Voltage In)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Note

The channels of both interfaces (digital and analog) must be located on the same I/O board (or I/O unit).

Digital Interface	Flexible In 1	Flexible In 2	Flexible In 3	Further Information
Measurement point	<ul style="list-style-type: none"> ▪ Load side ▪ ECU side 	Load side	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Ground-based 	Ground-based	Ground-based	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	✓	✓	–	Specifying Current and Voltage Values for

Digital Interface	Flexible In 1	Flexible In 2		Flexible In 3		Further Information
						Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	✓	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	–	–	–	Specifying Load Settings on page 121
Analog Interface ¹⁾	Flexible In 1	Flexible In 2	Analog In 1	Flexible In 3	Analog In 4	
Interface type	<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Differential with ground sense 	Ground-based	Differential with ground sense	Differential with ground sense	Differential with ground sense	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	–	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	✓	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	✓	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	✓	–	–	Specifying Load Settings on page 121

¹⁾ The analog electrical interface unit is available if the Extended signal analysis property is set to Enabled.

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- For basic signal analysis:
 - [Flexible In 1 on page 1593](#)
 - [Flexible In 2 on page 1595](#)
 - [Flexible In 3 on page 1603](#)
- For extended signal analysis:
 - [Mapping Signal Ports When Using Extended Signal Analysis \(Injection/Ignition Voltage In\) on page 412.](#)

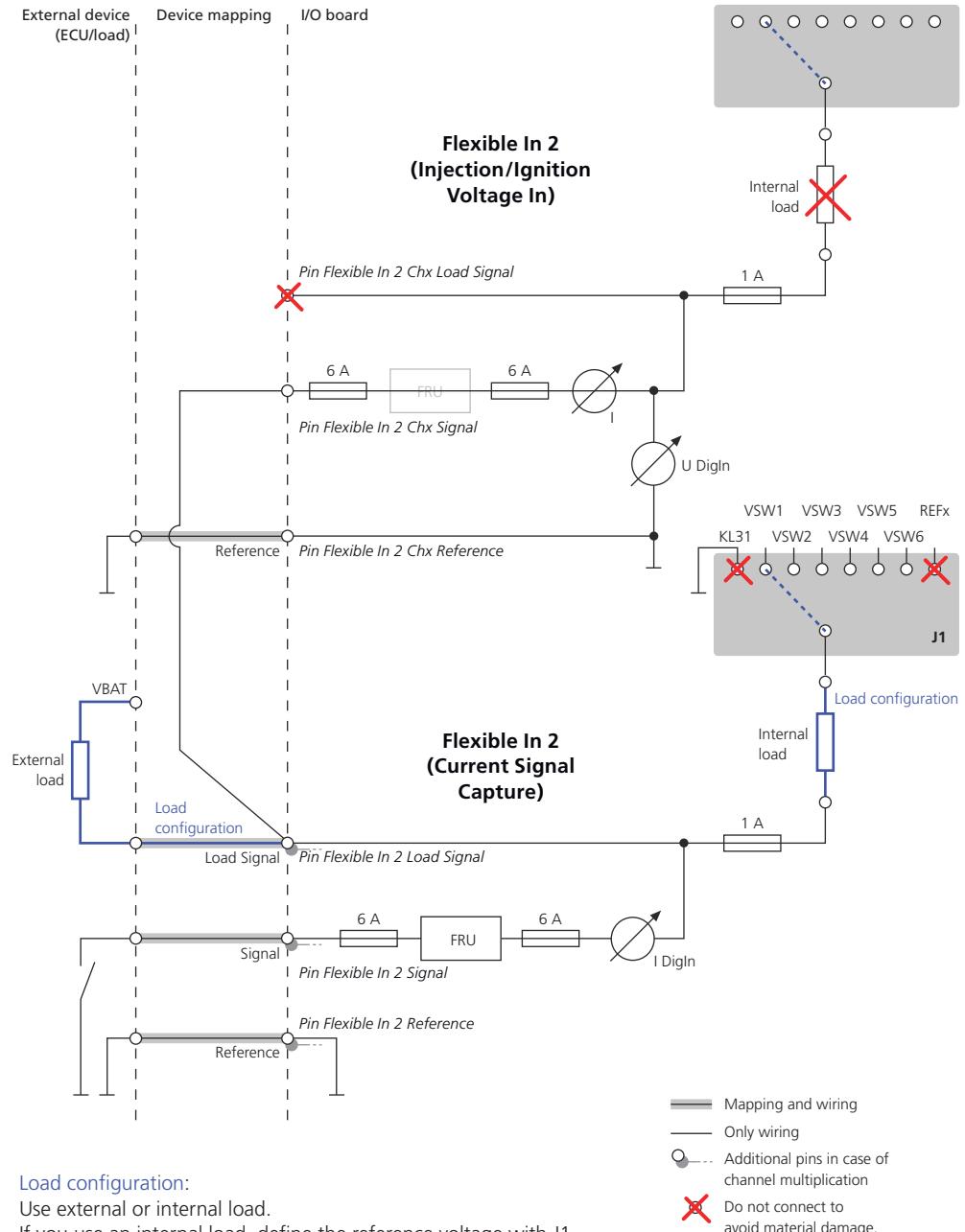
Model interface

The interface to the behavior model of the function block provides the following standard features.

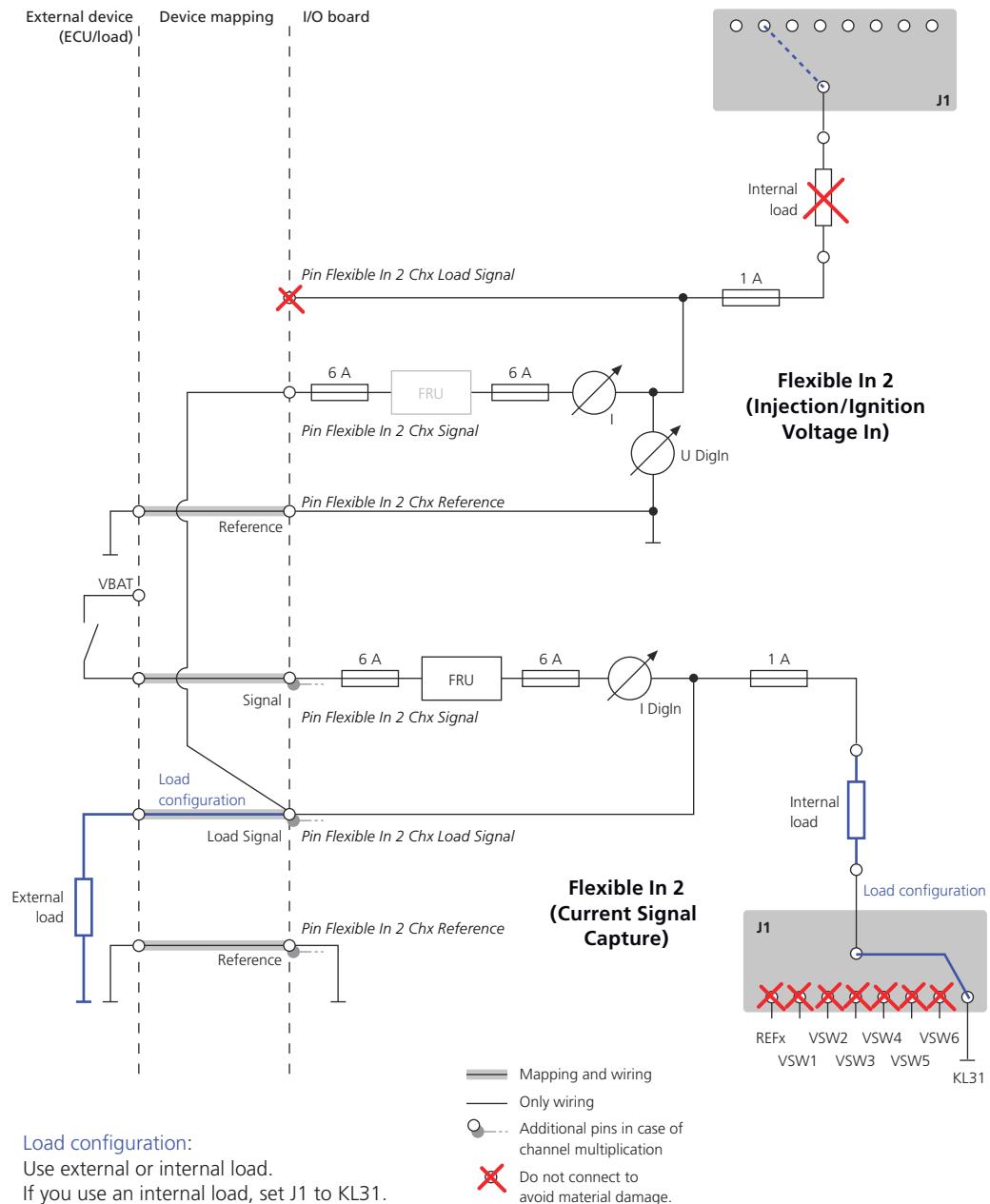
Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Voltage In)

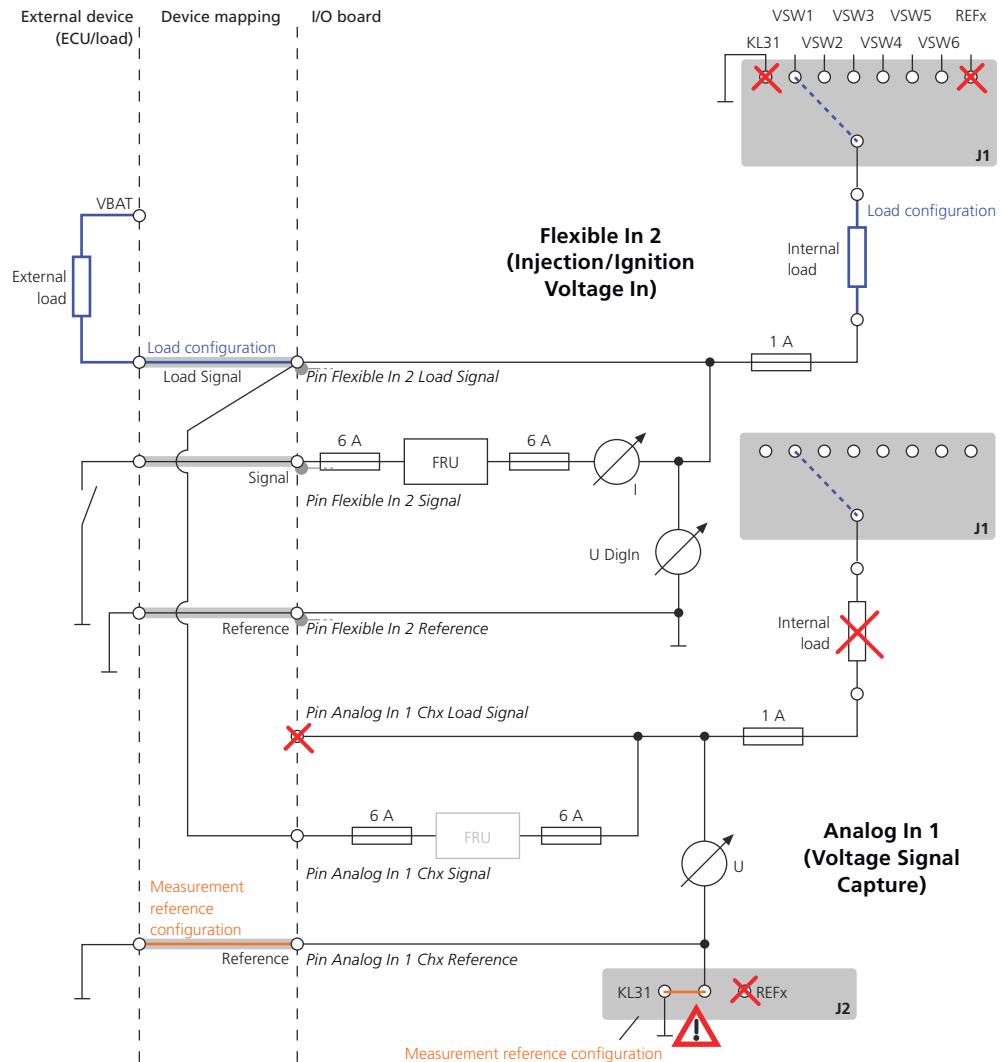
**Digital voltage measurement
with low-side controlled
analog current measurement
(on DS2680)**



**Digital voltage measurement
with high-side controlled
analog current measurement
(on DS2680)**

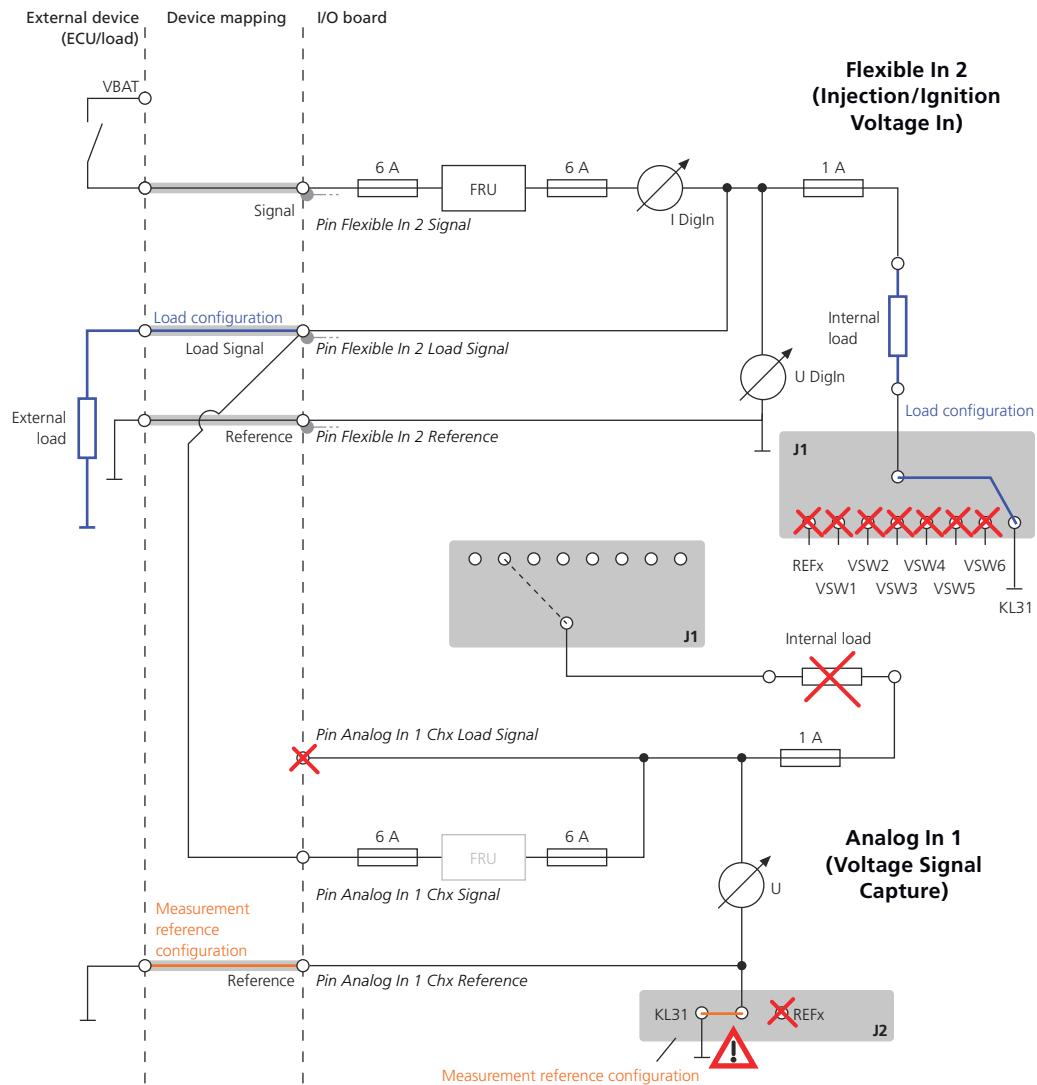


Low-side controlled digital voltage measurement with analog voltage measurement (on DS2680)

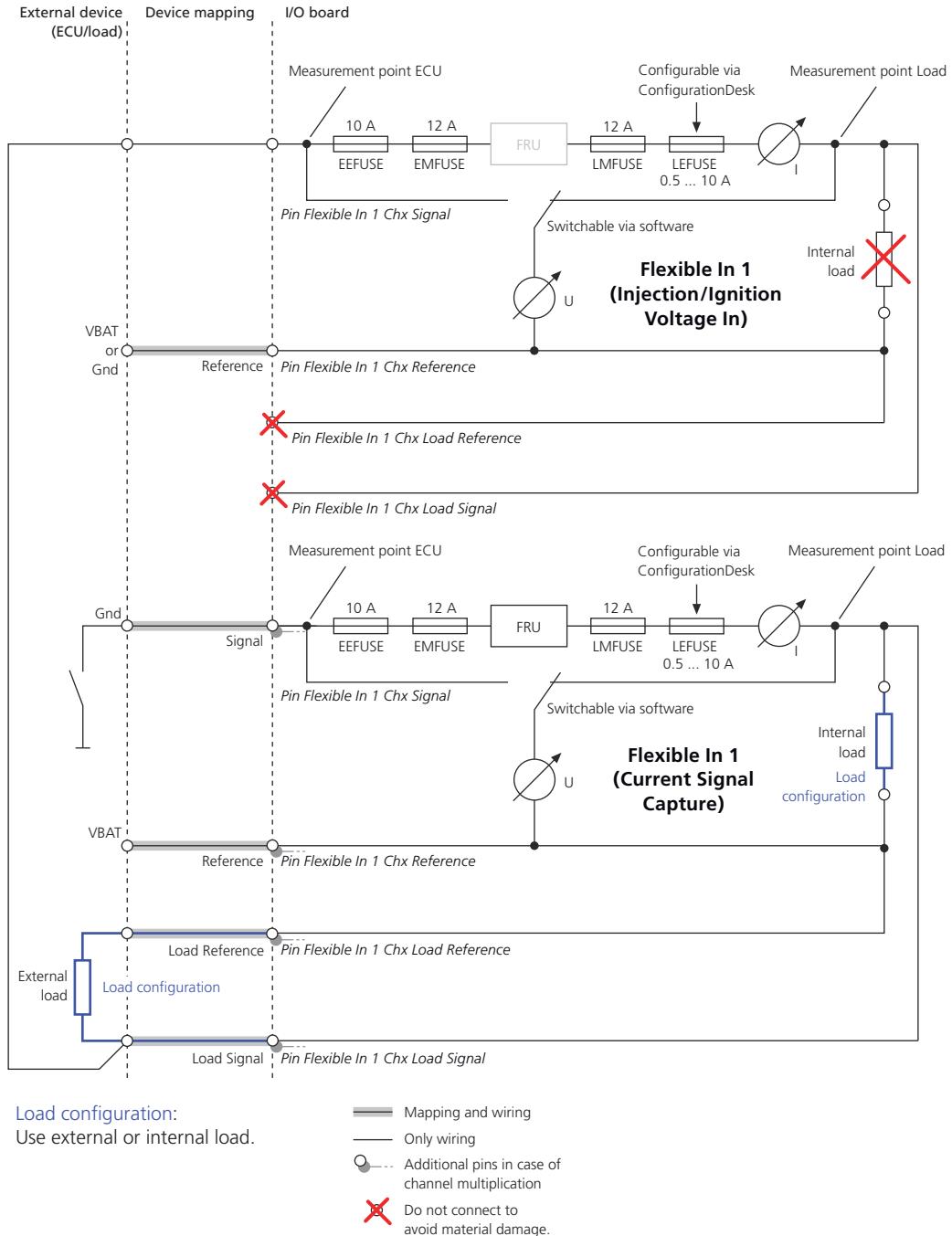


- Mapping and wiring
- Only wiring
- Additional pins in case of channel multiplication
- ✗ Do not connect to avoid material damage.

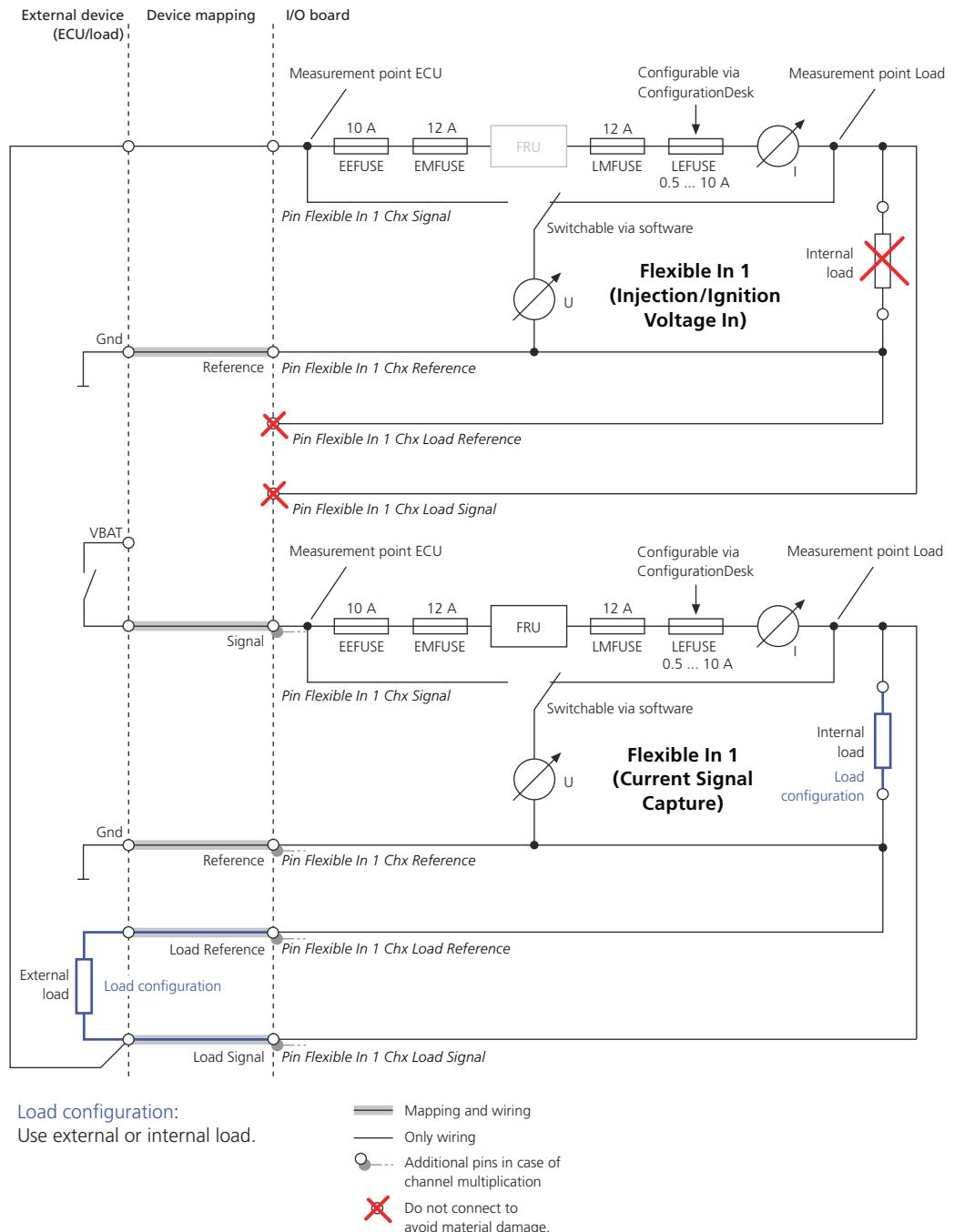
High-side controlled digital voltage measurement with analog voltage measurement (on DS2680)



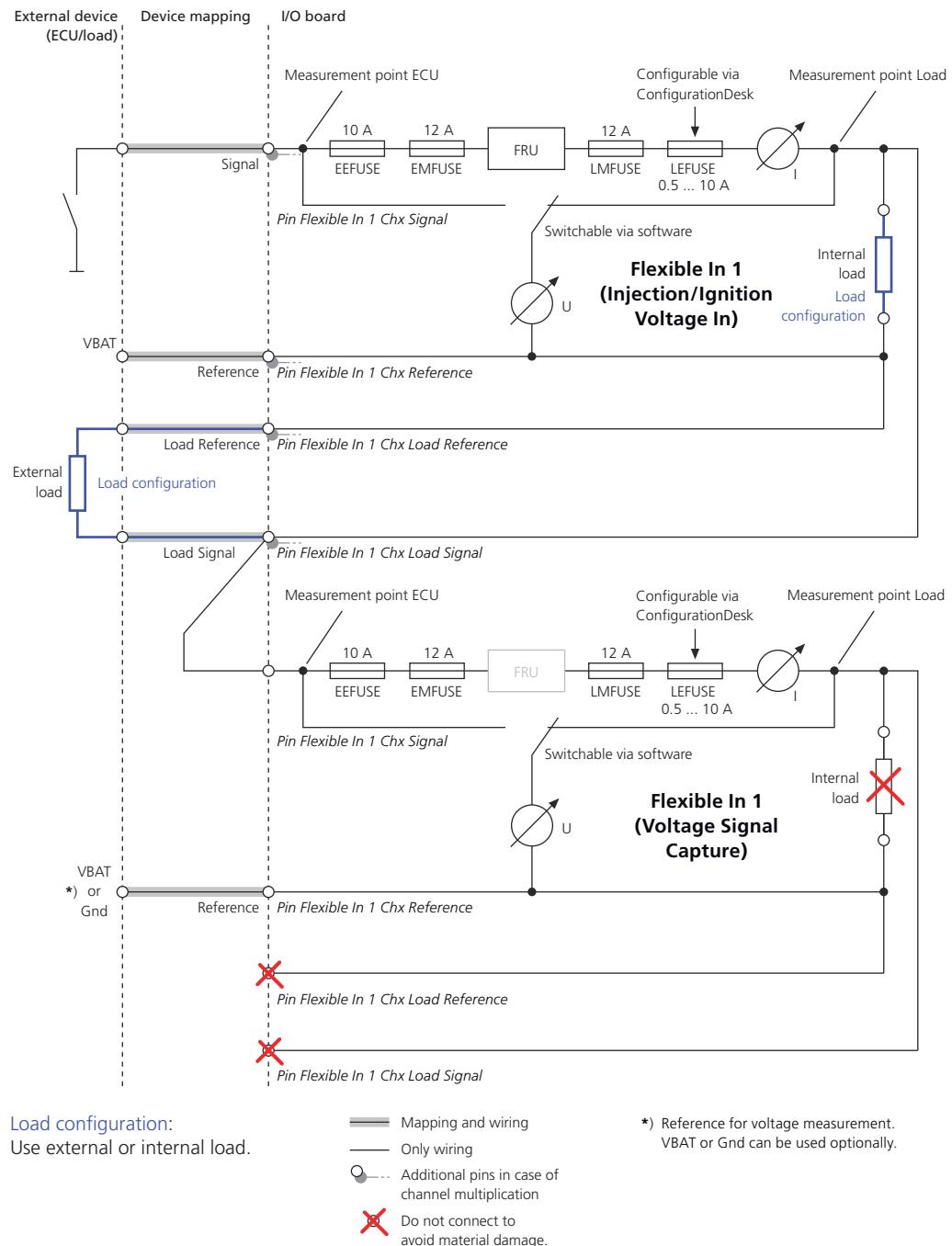
Low-side controlled digital voltage measurement with high-side controlled analog current measurement (on DS2601)



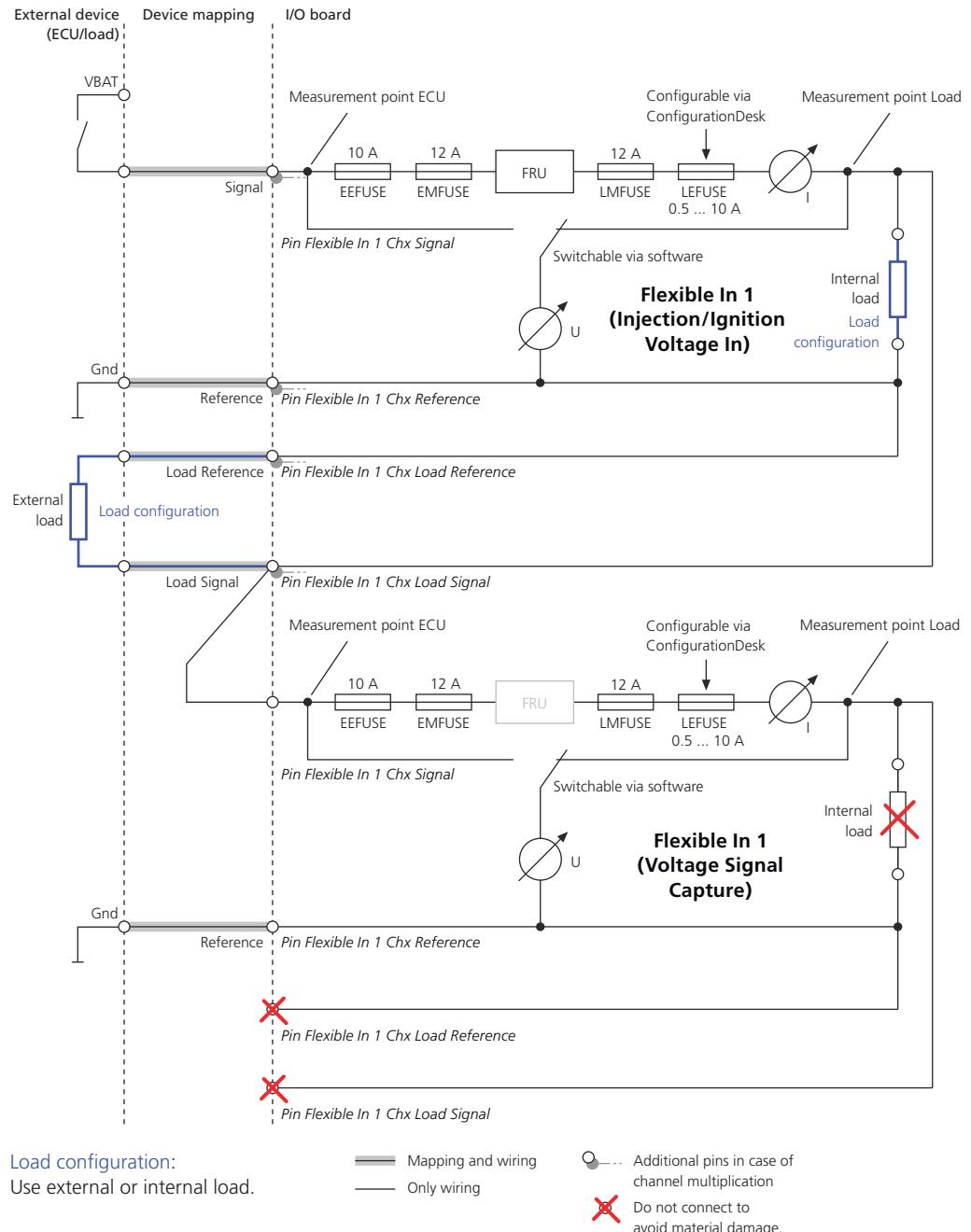
High-side controlled digital voltage measurement with high-side controlled analog current measurement (on DS2601)



Low-side controlled digital voltage measurement with analog voltage measurement (on DS2601)

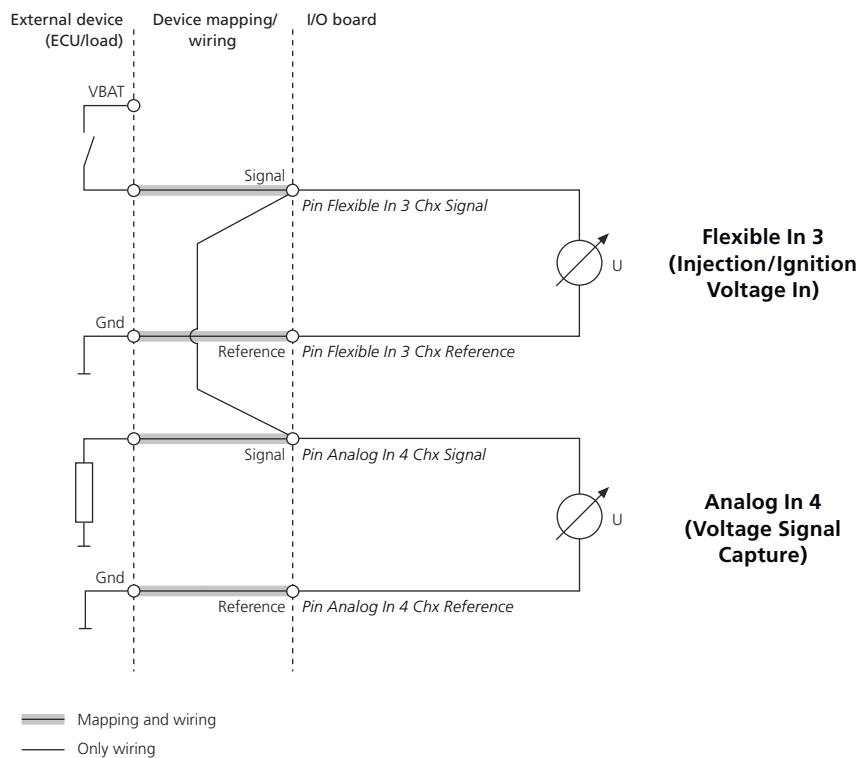


High-side controlled digital voltage measurement with analog voltage measurement (on DS2601)



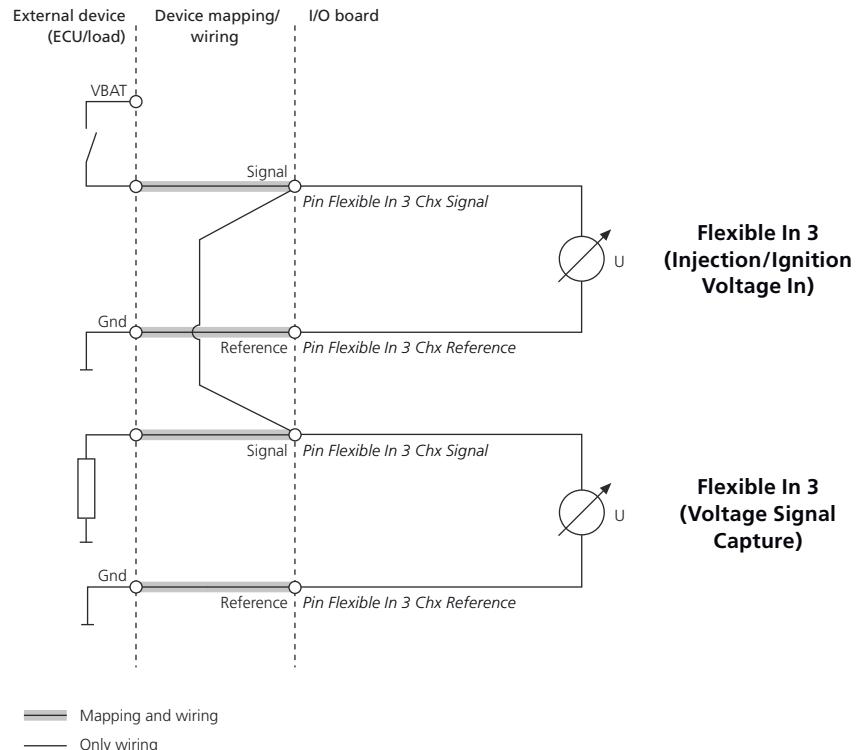
High-side controlled digital voltage measurement with analog voltage measurement (on DS6101)

Using Analog In 4 channel type for analog voltage measurement.



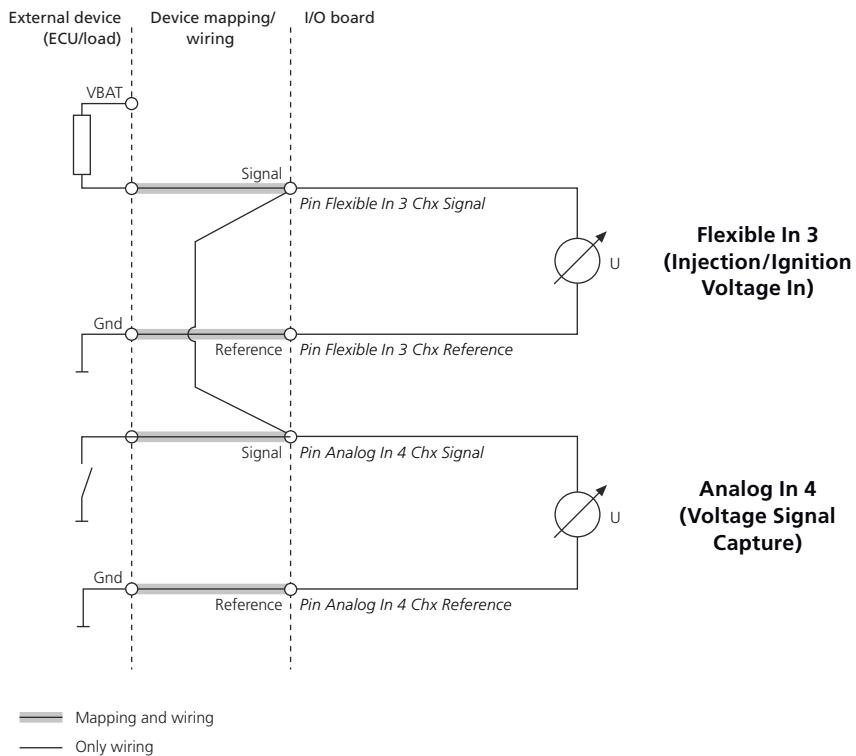
High-side controlled digital voltage measurement with analog voltage measurement (on DS6101)

Using Flexible In 3 channel type for analog voltage measurement.



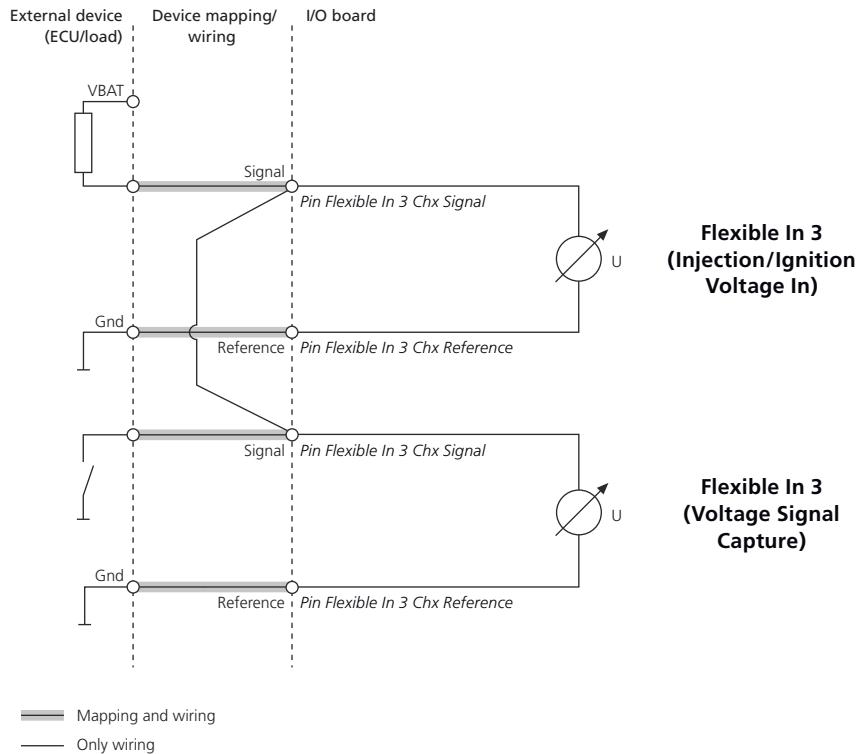
Low-side controlled digital voltage measurement with analog voltage measurement (on DS6101)

Using Analog In 4 channel type for analog voltage measurement.



Low-side controlled digital voltage measurement with analog voltage measurement (on DS6101)

Using Flexible In 3 channel type for analog voltage measurement.



Hardware Dependencies (Injection/Ignition Voltage In)

SCALEXIO Hardware Dependencies (Injection/Ignition Voltage In)

**Digital Interface:
Characteristics depending on
channel types**

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Flexible In 2	Flexible In 1	Flexible In 3
Hardware	DS2680 I/O Unit	DS2601 Signal Measurement Board	DS6101 Multi-I/O Board
Event generation	✓	✓	✓

Channel Type		Flexible In 2	Flexible In 1	Flexible In 3
Input current range		<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 86.4 A_{RMS}¹⁾ for 18 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +10 A_{RMS} for 1 channel ▪ 80 A_{RMS} for 10 channels (channel multiplication) 	–
Input voltage range		0 ... +60 V	-60 V ... +60 V	-60 V ... +60 V
Supported interface type		Ground-based	<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Ground-based 	Ground-based
Angular resolution		0.011°	0.011°	0.011°
Threshold	Input threshold range	0 ... +24 V	-60 V ... +60 V	-10 V ... +10 V
	Input hysteresis voltage	200 mV (typ.)	250 mV (typ.)	200 mV (typ.)
	DAC resolution	8 bit	14 bit	8 bit
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs	0.264 µs ... 131 µs	0.264 µs ... 131 µs
Angular processing unit	Number of slaves	6	1	6
	Maximum speed	168,000 °/s		
Measurement point		Load side	<ul style="list-style-type: none"> ▪ Load side ▪ ECU side 	–
Configurable fuse		–	<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	–
Circuit diagrams		Flexible In 2 on page 1595	Flexible In 1 on page 1593	Flexible In 3 on page 1603
Required channels		1 (additional channels are required for current enhancements.)		1

¹⁾ This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

Analog Interface: Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Flexible In 2	Analog In 1	Flexible In 1	Analog In 4	Flexible In 3
Hardware		DS2680 I/O Unit		DS2601 Signal Measurement Board	DS6101 Multi-I/O Board	
Input current range		0 ... +6 A _{RMS}		0 ... +10 A _{RMS}	–	
Input voltage range		0 ... +60 V		-60 V ... +60 V	0 ... + 60 V	-10 V ... +10 V
Supported interface type		Ground-based	Differential with ground sense	<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Differential with ground sense 	Differential with ground sense	
Measurement range per channel		±18 A		±30 A	–	
Sample period	Range	11.664 µs ... 0.1 s		4.012 µs ... 0.1 s	11.664 µs ... 0.1 s	
	Resolution	11.664 µs		68 ns	11.664 µs	

Channel Type	Flexible In 2	Analog In 1	Flexible In 1	Analog In 4	Flexible In 3
ADC resolution	16 bit		16 bit	16 bit	
Offset error	± 25 mA (typ.)	± 20 mV (typ.)	± 5 mA (typ.)	± 5 mV (typ.)	± 2 mV (typ.)
Gain error	$\pm 1.0\%$ (typ.)	$\pm 0.5\%$ (typ.)	$\pm 0.1\%$ (typ.)	$\pm 0.1\%$ (typ.)	$\pm 0.1\%$ (typ.)
Analog low-pass filter	Filter type	–		2 nd -order Bessel filter	–
	Edge frequency (-3dB cutoff frequency)			20 kHz	
Angular processing unit	Number of slaves	6	1	6	
	Maximum speed	168,000 °/s			
Configurable fuse	–		<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	–	
Circuit diagrams	Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Voltage In) on page 412				
Required channels	1				

General limitations

- Channel multiplication across several I/O boards is not supported.
- Basic signal analysis and extended signal analysis must use channels that are located on the same I/O board.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Injection/Ignition Current In

Where to go from here

Information in this section

Introduction (Injection/Ignition Current In).....	426
Overviews (Injection/Ignition Current In).....	427

Configuring the Function Block (Injection/Ignition Current In).....	436
Hardware Dependencies (Injection/Ignition Current In).....	449

Introduction (Injection/Ignition Current In)

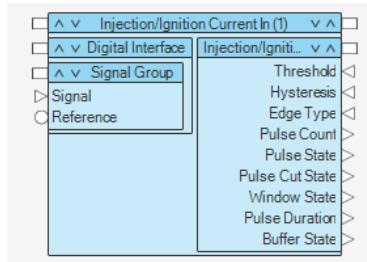
Introduction to the Function Block (Injection/Ignition Current In)

Function block purpose

The Injection/Ignition Current In function block type reads the positions and durations of the injection pulses that were generated by the ECU on the basis of a current input signal.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Measuring injection or ignition pulses from an ECU within specified angular (event) windows or continuously.
- Buffering detected pulses.
- Supporting digital current measurement and analog current measurements.
- Providing the measurement results to the behavior model.
- Providing status information to the behavior model on pulses, event windows and the data buffer.
- Generating I/O events and providing them to the behavior model.

Supported channel types

Depending on the specified scope of signal analysis, the Injection/Ignition Current In function block supports a digital or a digital and an analog signal interface.

Digital Interface The Digital Interface supports the following channel types:

	SCALEXIO		MicroAutoBox III
Channel type	Flexible In 1	Flexible In 2	–
Hardware	DS2601	DS2680	–

Analog Interface The Analog Interface is available only if Extended signal analysis is enabled. It supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Flexible In 1	Flexible In 2 ¹⁾	Analog In 1 ²⁾	–
Hardware	DS2601	DS2680	–	–

¹⁾ Only for current measurement.

²⁾ Only for voltage measurement.

**Introduction to
Injection/Ignition pulse
measurement**

For more information, refer to [Introduction to Injection and Ignition Pulse Measurement](#) on page 377.

Oversviews (Injection/Ignition Current In)

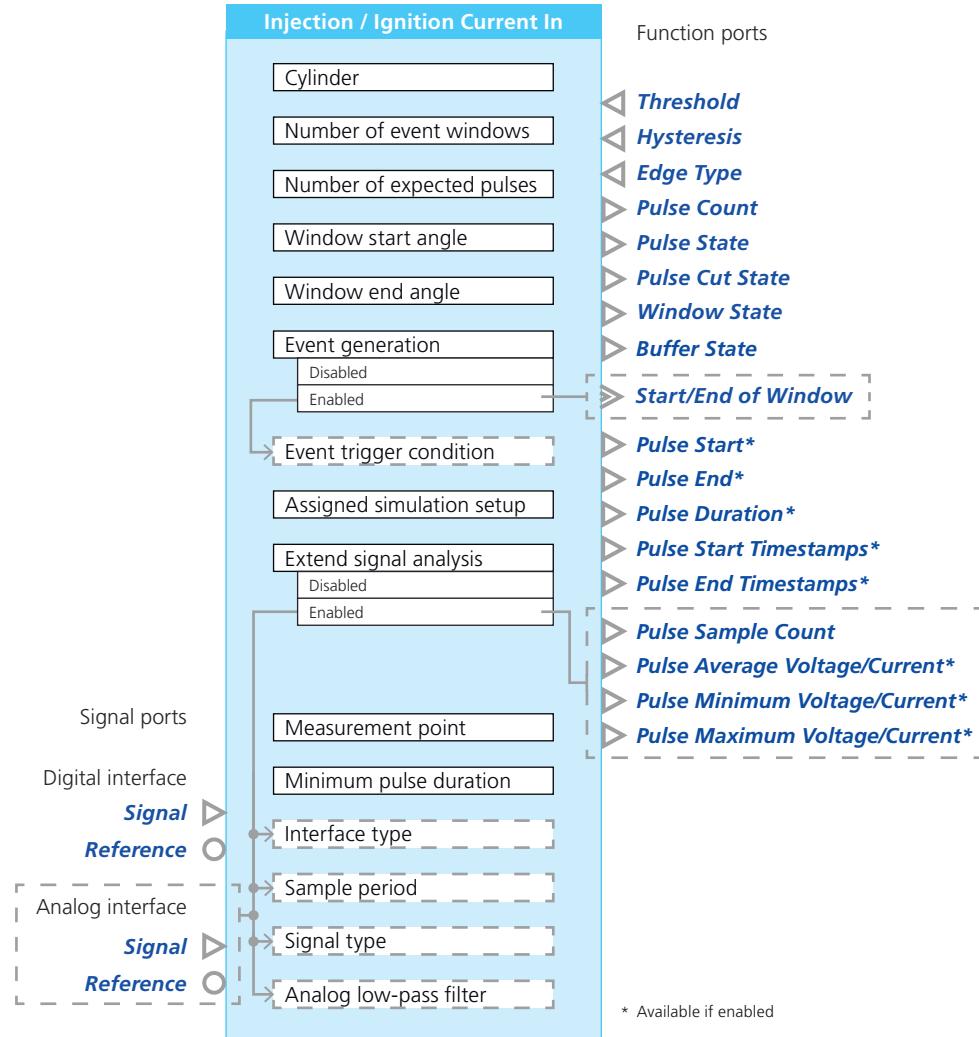
Where to go from here**Information in this section**

Overview of Ports and Basic Properties (Injection/Ignition Current In).....	427
Overview of Tunable Properties (Injection/Ignition Current In).....	435

Overview of Ports and Basic Properties (Injection/Ignition Current In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



If you enable Extended signal analysis, an analog electrical interface is added. This interface also provides identically labeled signal ports. However, in this case the assigned hardware resources affect the availability of all signal ports of the function block basically. For details on the mapping of the signal ports, refer to [Mapping Signal Ports When Using Extended Signal Analysis \(Injection/Ignition Current In\) on page 441](#).

Threshold

This function import reads a vector with the two threshold values for pulse detection from within the behavior model. The first threshold value is used for detecting the beginning of a pulse, the second for detecting its end.

Value range	<ul style="list-style-type: none"> $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. The range depends on the following: <ul style="list-style-type: none"> The assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection/Ignition Current In) on page 449.
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> If you use user saturation min/max values for saturation, the specified values can reduce the value range. The vector size is two. The first entry refers to the pulse start, the second to the pulse end.
Dependencies	–

Hysteresis

This function import reads a vector with the hysteresis values of the two thresholds from within the behavior model.

Value range	<ul style="list-style-type: none"> 0.125 A ... 3 A for each entry of the vector. If you use user saturation min/max values for saturation, the specified values can reduce the value range. The vector size is two. The first entry refers to the pulse start, the second to the pulse end.
Dependencies	–

Edge Type

This function import reads a vector with the edge types relevant for pulse detection from within the behavior model.

Value range	For each entry of the vector: <ul style="list-style-type: none"> 1: Rising 2: Falling The vector size is two. The first entry refers to the pulse start, the second to the pulse end.
Dependencies	–

Pulse Count

This function outport writes a vector with the number of pulses that actually occurred within the event window to the behavior model. The number of pulses is *independent* of the Number of expected pulses. The pulse counter increments at every rising pulse edge. Pulses that are only partially included by the event window, i.e., cut pulses, are also counted.

Value range	<ul style="list-style-type: none"> 0 ... 100 for each entry of the vector. The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Pulse State

This function outport writes a vector with flags to the behavior model. The flags indicate whether the current pulse is finished or not. A pulse is finished when its *falling* edge was detected.

Value range	<p>For each entry of the vector:</p> <ul style="list-style-type: none"> ▪ 0: Finished. No pulse is active. ▪ 1: Active. The last pulse is still active. ▪ The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Pulse Cut State

This function outport writes a vector that indicates whether pulses were cut by the event window to the behavior model. If a pulse starts before (= cut before) and/or ends after (= cut after) an event window, the boundaries of the event window are captured instead of the pulse edges outside the event window.

Value range	<ul style="list-style-type: none"> ▪ For each entry of the vector: <ul style="list-style-type: none"> ▪ 0: No cut Event window completely includes the captured pulses. ▪ 1: Cut before Event window cut the first pulse, i.e., only the ending edge of this pulse is within the event window. ▪ 2: Cut after Event window cut the last pulse, i.e., only the starting edge of this pulse is within the event window. ▪ 3: Cut before and after Event window cut the first pulse and the last pulse. ▪ The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Window State

This function outport writes a vector that indicates whether the current value of the angle counter is within the event window to the behavior model. The **Window State** function port switching to closed thereby indicates that the data set of an event window has completely been updated with the new pulse data.

Value range	<ul style="list-style-type: none"> ▪ For each entry of the vector: <ul style="list-style-type: none"> ▪ 0: Closed ▪ 1: Active ▪ The vector size depends on the value specified at the Number of event windows property. Its size is 1 if zero or one event window and 2 if two event windows are specified.
Dependencies	–

Pulse Start

This function outport writes a vector with the start angles of the captured pulses to the behavior model. If the first pulse is cut by the event window (cut before), the start angle of the event window is taken into account instead.

Value range	<ul style="list-style-type: none"> -1440° ... +1440° for each entry of the vector. The vector size is N + M, i.e., [E₁, ..., E_N, E₁, ..., E_M]. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse start property is set to Enabled.

Pulse End

This function outport writes a vector with the end angles of the captured pulses to the behavior model. If the last pulse is cut by the event window (cut after), the end angle of the event window is taken into account instead.

This function outport writes a vector with the end angles of the captured pulses to the behavior model. If the last pulse is cut by the event window (cut after), the end angle of the event window is taken into account instead.

Value range	<ul style="list-style-type: none"> -1440° ... +1440° for each entry of the vector. The vector size is N + M, i.e., [E₁, ..., E_N, E₁, ..., E_M]. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse start property is set to Enabled.

Pulse Duration

This function outport writes a vector with the lengths of the captured pulses (in seconds) to the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... 3600 s for each entry of the vector. The vector size is N + M, i.e., [E₁, ..., E_N, E₁, ..., E_M]. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse start property is set to Enabled.

Buffer State

This function outport writes state information on the internal buffer to the behavior model.

Value range	<ul style="list-style-type: none"> 0: OK The internal buffer is OK. No error occurred. 1: Overflow An overflow of the internal buffer occurred. Too many pulses were detected and data is lost. This can happen, if the
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>sample rate of the behavior model is too slow, and/or the number of pulses counted is too high.</p> <ul style="list-style-type: none"> ▪ 2: FIFO overflow An overflow of the transmission buffer between the computation node and I/O node occurred. Too much data was transmitted and data is lost. ▪ 3: Consistency lost Data consistency between digital and analog data is lost and a reset is in progress. ▪ 4: FIFO error An unrecoverable transmission error occurred and the connection is lost. An application restart is required.
Dependencies	–

Pulse Start Timestamps

This function outport writes a vector with the start time stamp values of the captured pulses to the behavior model. If the first pulse is cut by the event window (cut before), the start time stamp of the event window is used instead of the first pulse's start time stamp.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse end timestamps property is set to Enabled.

Pulse End Timestamps

This function outport writes a vector with the end time stamp values of the captured pulses to the behavior model. If the last pulse is cut by the event window (cut after), the end time stamp of the event window is used instead of the last pulse's end time stamp.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Capture pulse end timestamps property is set to Enabled.

Pulse Sample Count

This function outport writes a vector with the number of samples (data points) of the captured pulses to the behavior model. If the last pulse is cut by the event window (cut after), the end time stamp of the event window is used instead of the last pulse's end time stamp.

Note

The pulse sample counter returns unexpected results, if pulse duration is greater than 65,535*sample periods.

Value range	<ul style="list-style-type: none"> 1 sample ... 100 samples for each entry of the vector. <p>The number of samples is influenced by the Sample period property of the block's analog signal interface.</p> <ul style="list-style-type: none"> The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property is set to Enabled.

Pulse Average Voltage/Current

This function outport writes a vector with the mean voltage/current values of the captured pulses to the behavior model. Whether it is for voltage (default) or current depends on the setting of the Signal type property.

Note

The pulse average voltage/current calculation returns unexpected results, if pulse duration is greater than 65,535*sample periods.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt) or $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. For specific values, refer to Hardware Dependencies (Injection/Ignition Current In) on page 449. The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property and then the Capture pulse average values property are set to Enabled.

Pulse Minimum Voltage/Current

This function outport writes a vector with the minimum voltage/current values of the captured pulses to the behavior model. Whether it is for voltage (default) or current depends on the setting of the Signal type property.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt) or $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. For specific values, refer to Hardware Dependencies (Injection/Ignition Current In) on page 449. The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property and then the Capture pulse minimum values property are set to Enabled.

Pulse Maximum Voltage/Current

This function outport writes a vector with the maximum voltage/current values of the captured pulses to the behavior model. Whether it is for voltage (default) or current depends on the setting of the Signal type property.

Value range	<ul style="list-style-type: none"> ▪ $U_{\min} \dots U_{\max}$ (in Volt) or $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. ▪ For specific values, refer to Hardware Dependencies (Injection/Ignition Current In) on page 449. ▪ The vector size is $N + M$, i.e., $[E_1, \dots, E_N, E_1, \dots, E_M]$. N and M equal the number of expected pulses for the first and the optional second event window specified at the Number of expected pulses property.
Dependencies	Available only if the Extended signal analysis property and then the Capture pulse maximum values property are set to Enabled.

Start/End of Window

This event port provides an I/O event each time the first rising edge in an event window is detected or when the end of an event window is reached, according to the setting of the Event trigger condition property.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection/Ignition Current In) on page 449.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection/Ignition Current In) on page 449.
Dependencies	–

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Supported only for the Flexible In 1 channel type.

Overview of Tunable Properties (Injection/Ignition Current In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Event trigger condition	✓	–
Cylinder	✓	–
Window start angle	✓	–
Window end angle	✓	–

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Digital Interface		
Measurement point	✓	-
Minimum pulse duration	✓	-
Analog Interface ³⁾		
Sample period	✓	-
Analog low-pass filter	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ The analog interface unit is available if the Extended signal analysis property is set to Enabled.

Configuring the Function Block (Injection/Ignition Current In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Injection/Ignition Current In).....	436
Configuring Standard Features (Injection/Ignition Current In).....	439
Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Current In).....	441

Configuring the Basic Functionality (Injection/Ignition Current In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Defining engine properties
- Specifying minimum pulse duration
- Specifying event windows for pulse capturing

- Enabling function ports for measuring digital values of captured pulses
- Specifying extended signal analysis (measuring analog signal values of captured pulses)
- Providing I/O events

Defining engine properties

Defining virtual piston engine The Injection/Ignition Current In function block refers to a specific engine design by referencing an Engine Simulation Setup function block. The Engine Simulation Setup function block provides basic configuration data of the virtual engine, for example, the number of cylinders.

Selecting cylinder for measurement The Injection/Ignition Current In function block lets you specify the cylinder which the incoming injection or ignition signal is related to.

If you want to measure, for example, injection and ignition signals for the same cylinder, you would need two Injection/Ignition Current In function blocks both addressing this cylinder.

Filtering the input signal

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the pulse duration depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Injection/Ignition Current In\) on page 449](#).

You can also change the value of the **Minimum pulse duration** property via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying event windows for pulse capturing

You can specify an angle range within the engine cycle as an event window in which injection and ignition pulses are detected. Pulses occurring outside the event window are ignored.

- You can specify one or two event windows and their start and end angles (**Number of event windows** property is set to 1 or 2).
If you specify event windows with start and end angles, the values provided at the **Threshold**, **Hysteresis**, and **Edge Type** function ports take effect at the end of a window. This avoids data loss if the values at the function ports change during the measurement of a pulse.
- You can specify one event window that covers the whole engine cycle, i.e., for continuous pulse measurement (**Number of event windows** property is set to 0).
If you specify continuous pulse measurement, the values provided at the **Threshold**, **Hysteresis**, and **Edge Type** function ports take effect immediately. Note that a change of one of these values might result in data loss in the next model step.

Enabling function ports for measuring digital values of captured pulses

The Injection/Ignition Current In function block provides pulse measurement features, for example, for counting pulses, measuring pulse start/stop angles, and measuring pulse start/stop times. The measured values are provided to the behavior model via specific function ports.

You have to enable the function ports so you can use them.

Measuring analog signal values of captured pulses

Analog signal measurement With the Injection/Ignition Current In function block, you can additionally measure the analog signal values of the captured pulses (extended signal analysis). The measured values are provided to the behavior model via specific function ports.

If the function block detects a digital pulse, its analog measurement is started. The minimum, maximum, and average voltage or current values of the pulse are determined. These values can be provided to the behavior model.

You can specify the time between two samples with the **Sample period** property. The higher the sample rate, the more precise is the analog signal measurement.

Selecting the signal type You can select the physical signal type for analog signal measurement:

- **Voltage:** The analog signal measurement is based on the voltage of the captured pulse.
- **Current:** The analog signal measurement is based on the current of the captured pulse.

Limitations:

- The **Voltage** signal type is not supported for the Flexible In 2 channel type.
- The **Current** signal type is not supported for the Analog In 1, Analog In 4 and Flexible In 3 channel types.

Using the analog low-pass filter You can use the analog low-pass filter, for example, to reduce signal noise. The filter characteristics depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Injection/Ignition Current In\)](#) on page 449.

For more basic information on using the filter, refer to [Analog and Digital Filtering with the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Limitation: The analog low-pass filter is supported only for the Flexible In 1 channel type.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Injection/Ignition Current In function block can generate an I/O event each time a specific condition of an event window is detected. Via the **Event trigger condition** property you can specify when to trigger the I/O event, either when the first rising edge in an event window is detected or when the end of an event window is reached.

Note

Event generation is not possible for continuous pulse measurement.

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Window property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Injection/Ignition Current In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Note

The channels of both interfaces (digital and analog) must be located on the same I/O board (or I/O unit).

Digital Interface	Flexible In 1	Flexible In 2	Further Information	
Measurement point	Load side	Load side	Specifying the Measurement Point for Input Signals on page 119	
Channel multiplication	✓	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95	
Signal Ports				
Trigger level of electronic fuse	✓	–	Specifying Fuse Settings on page 122	
Failure simulation support	✓	✓	Specifying Settings for Failure Simulation on page 123	
Usage of loads	✓	✓	Specifying Load Settings on page 121	
Analog Interface ¹⁾	Flexible In 1	Flexible In 2	Analog In 1	
Interface type	▪ Galvanically isolated	Differential with ground sense	Differential with ground sense	Specifying the Circuit Type for Analog Output Signals on page 118

Digital Interface	Flexible In 1	Flexible In 2		Further Information
	<ul style="list-style-type: none"> ▪ Differential with ground sense 			
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	✓	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	✓	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	✓	Specifying Load Settings on page 121

¹⁾ The analog electrical interface unit is available if the Extended signal analysis property is set to Enabled.

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- For basic signal analysis:
 - [Flexible In 1 on page 1593](#)
 - [Flexible In 2 on page 1595](#)
- For extended signal analysis:
 - [Mapping Signal Ports When Using Extended Signal Analysis \(Injection/Ignition Current In\) on page 441](#).

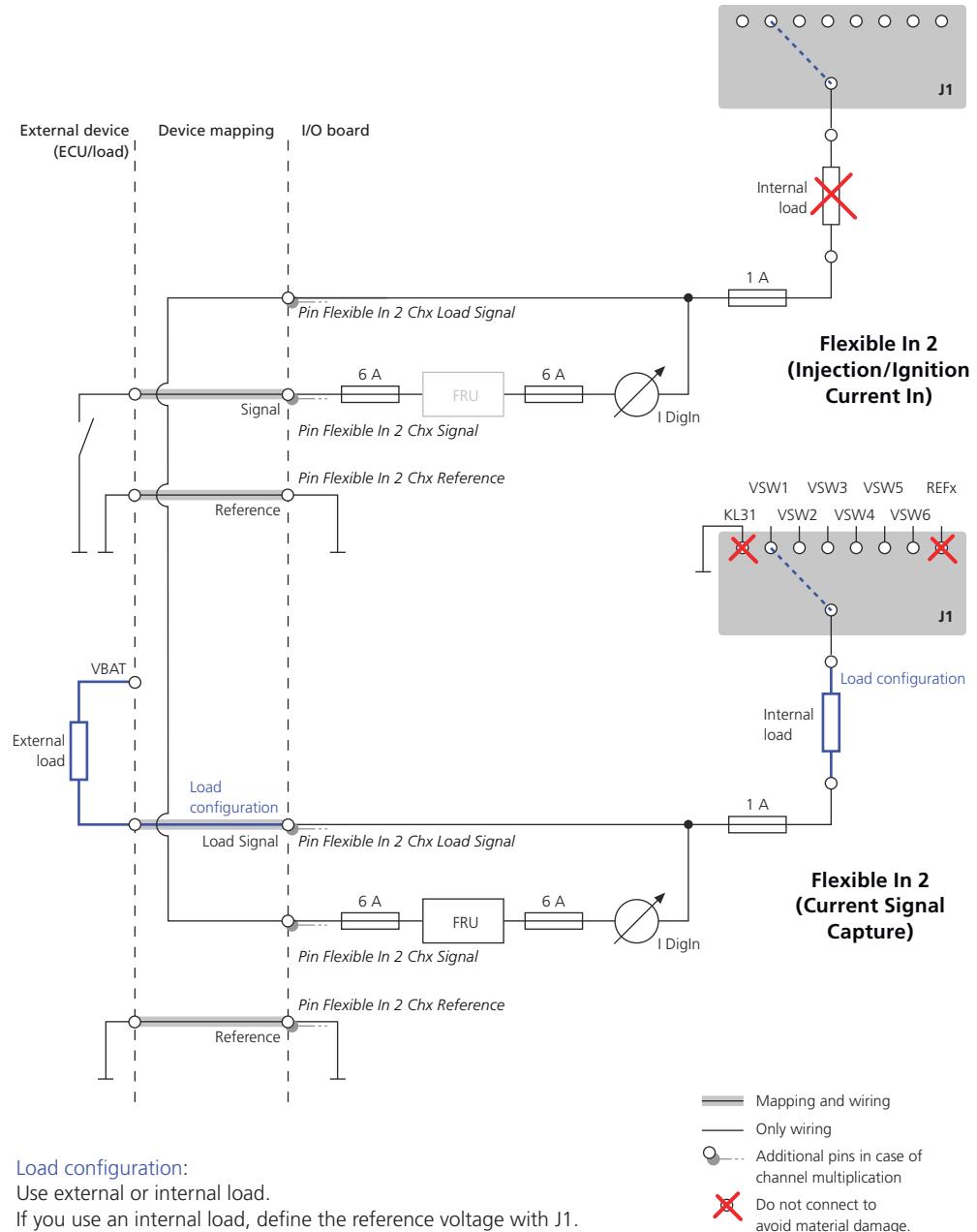
Model interface

The interface to the behavior model of the function block provides the following standard features.

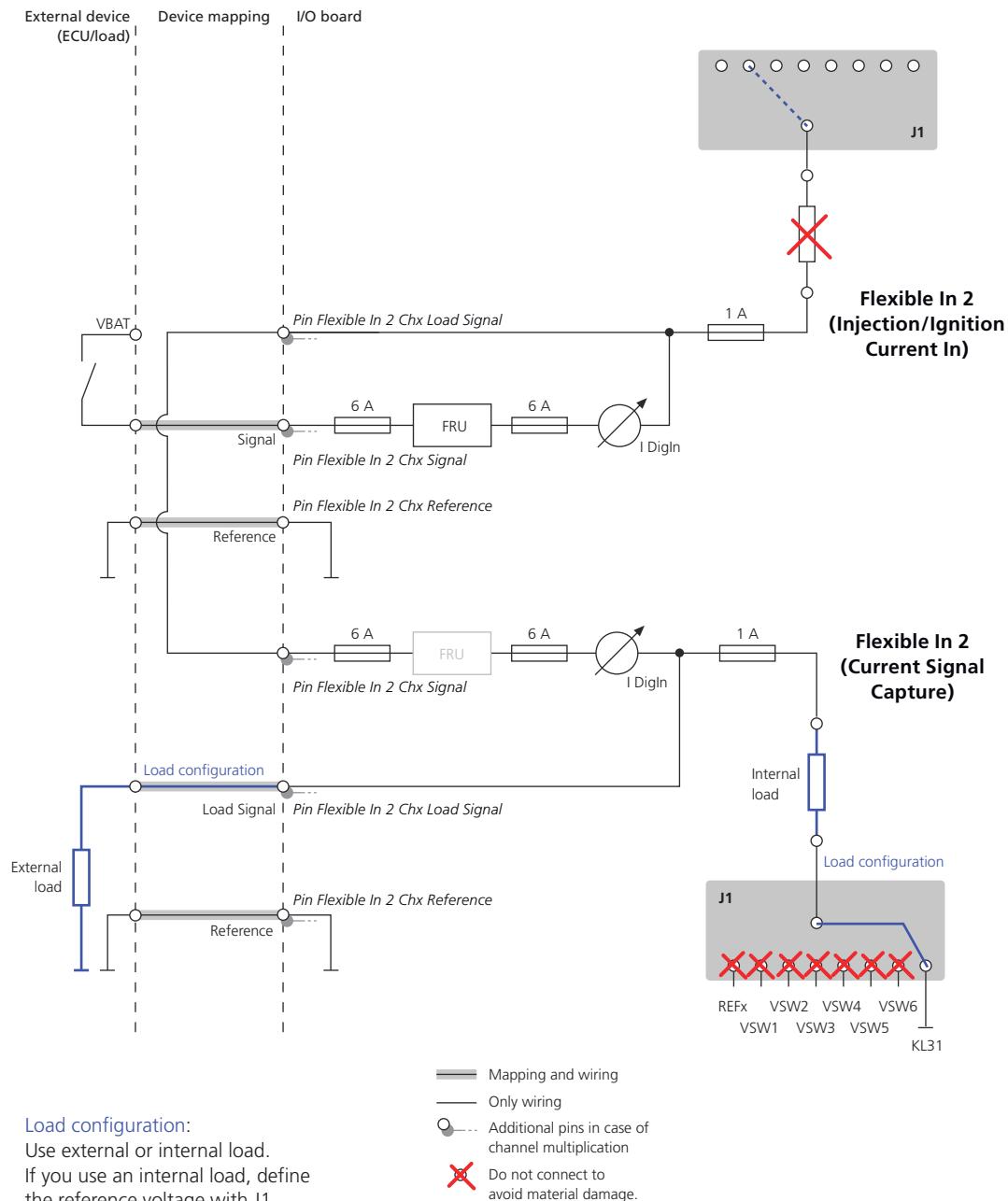
Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Current In)

**Digital current measurement
with low-side controlled
analog current measurement
(on DS2680)**



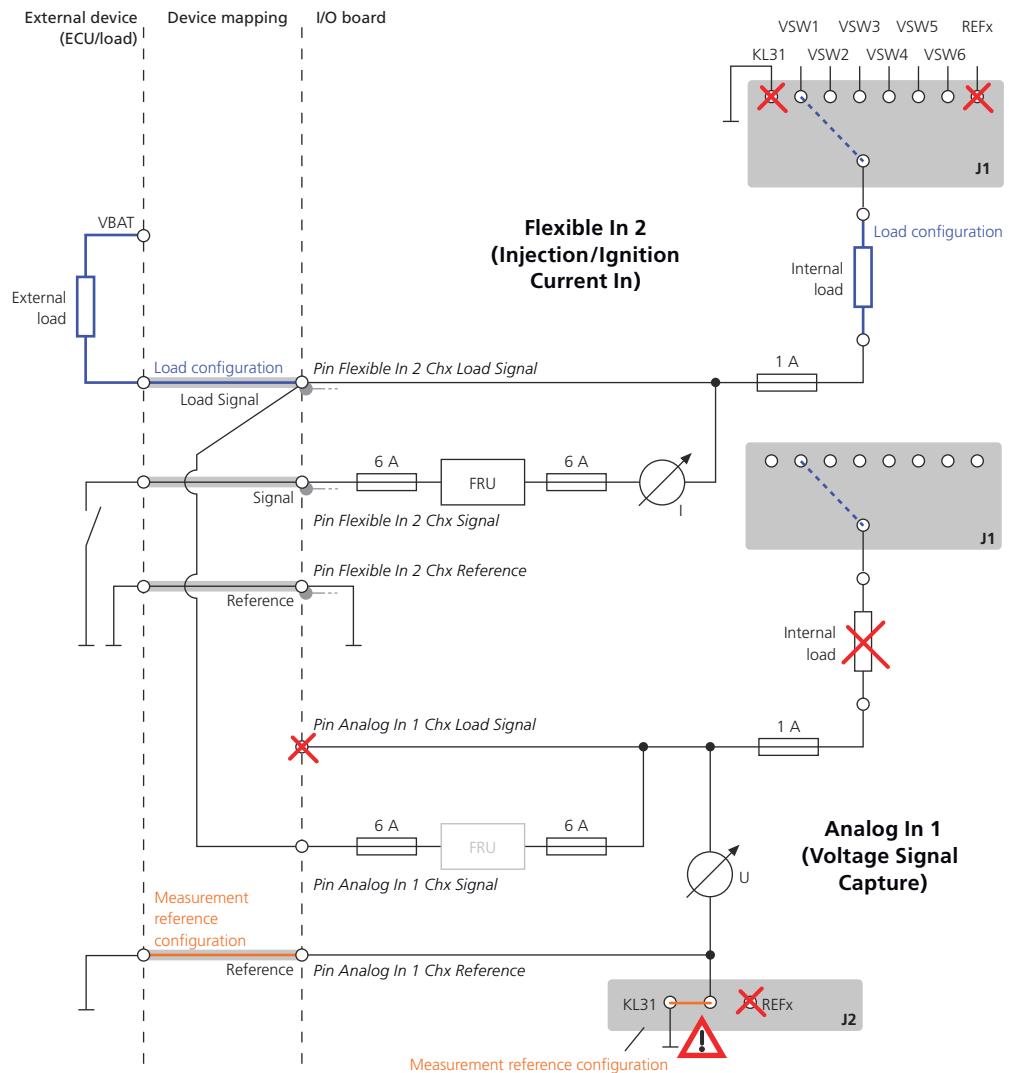
**Digital current measurement
with high-side controlled
analog current measurement
(on DS2680)**



Load configuration:

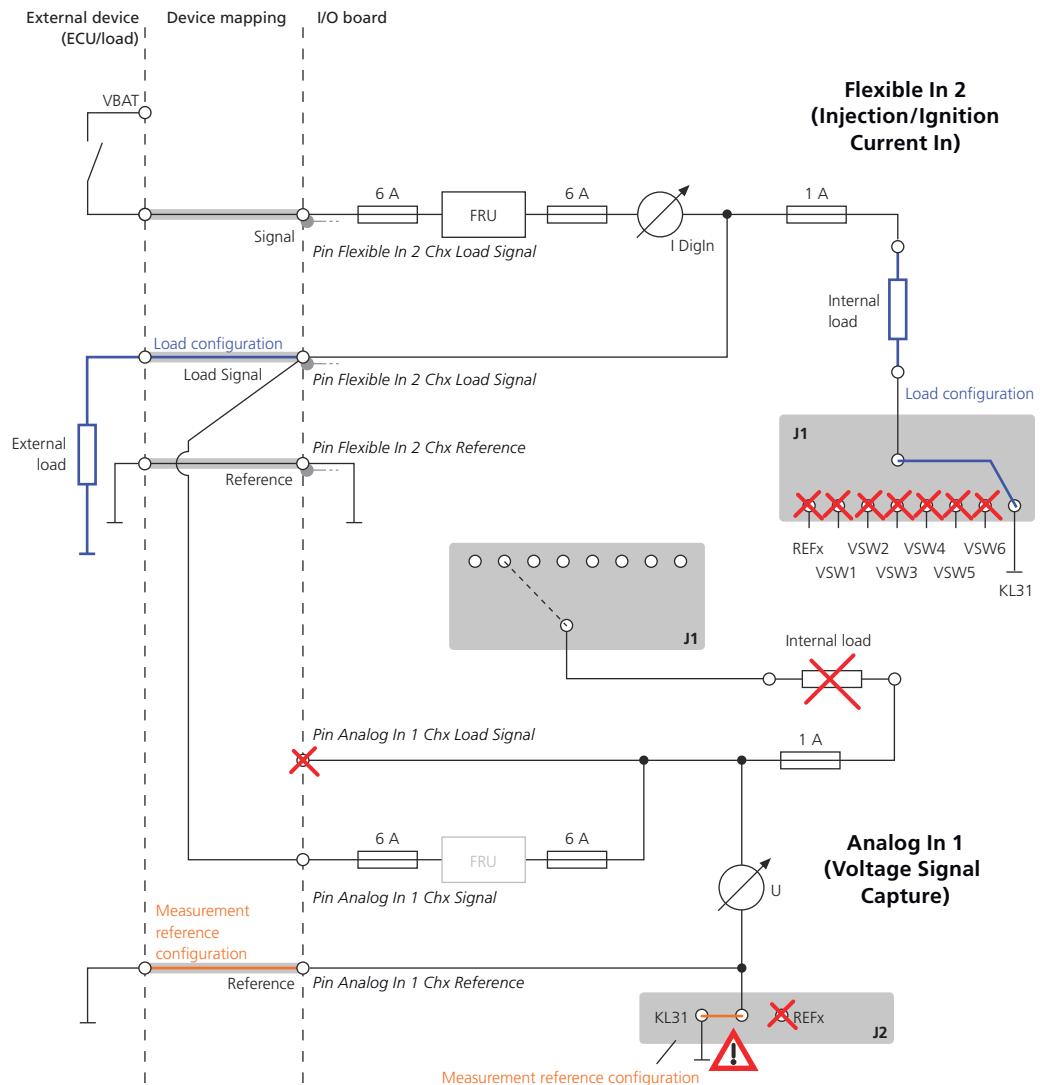
Use external or internal load.
If you use an internal load, define the reference voltage with J1.

Low-side controlled digital current measurement with analog voltage measurement (on DS2680)



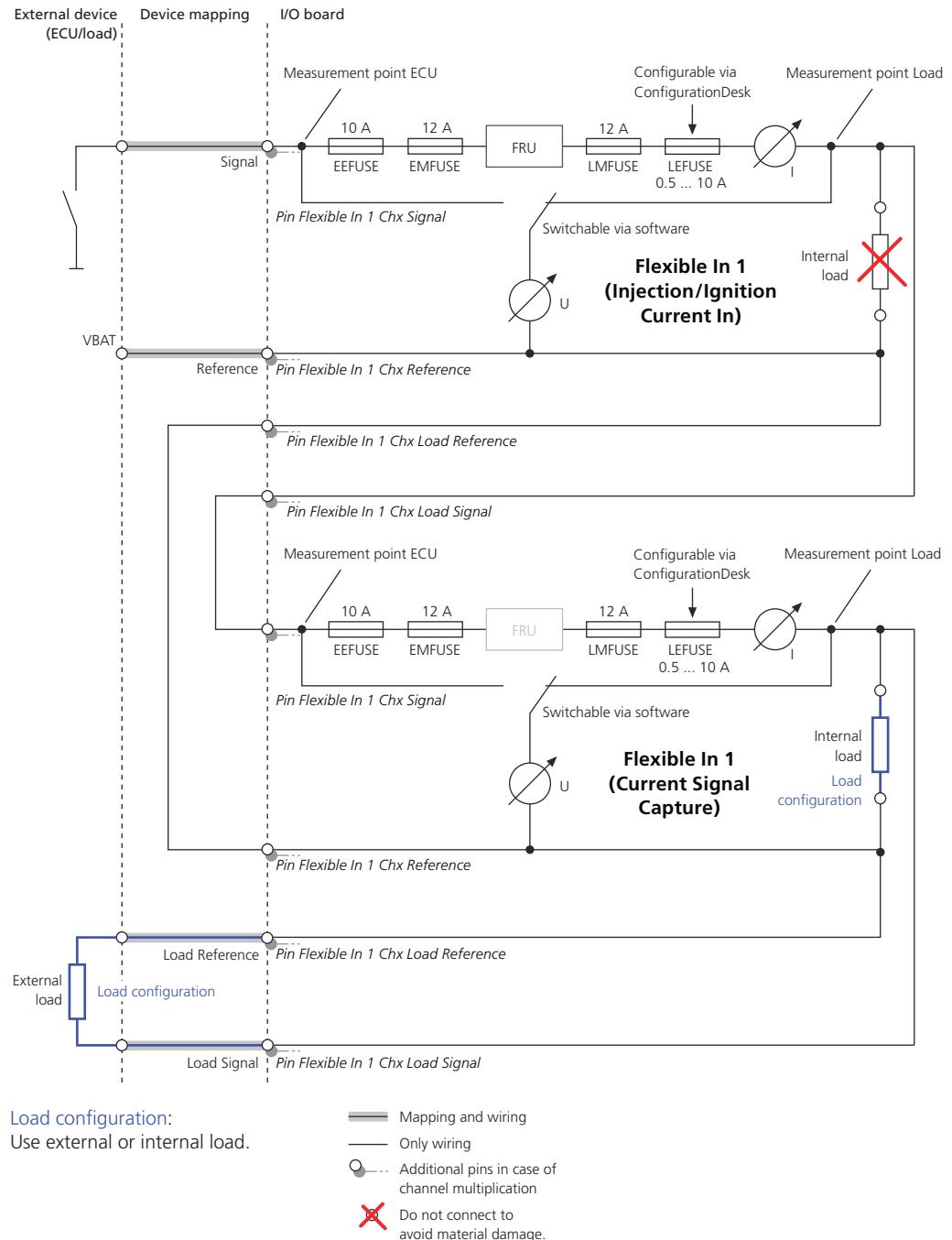
- Mapping and wiring
- Only wiring
- Additional pins in case of channel multiplication
- ✗ Do not connect to avoid material damage.

High-side controlled digital current measurement with analog voltage measurement (on DS2680)



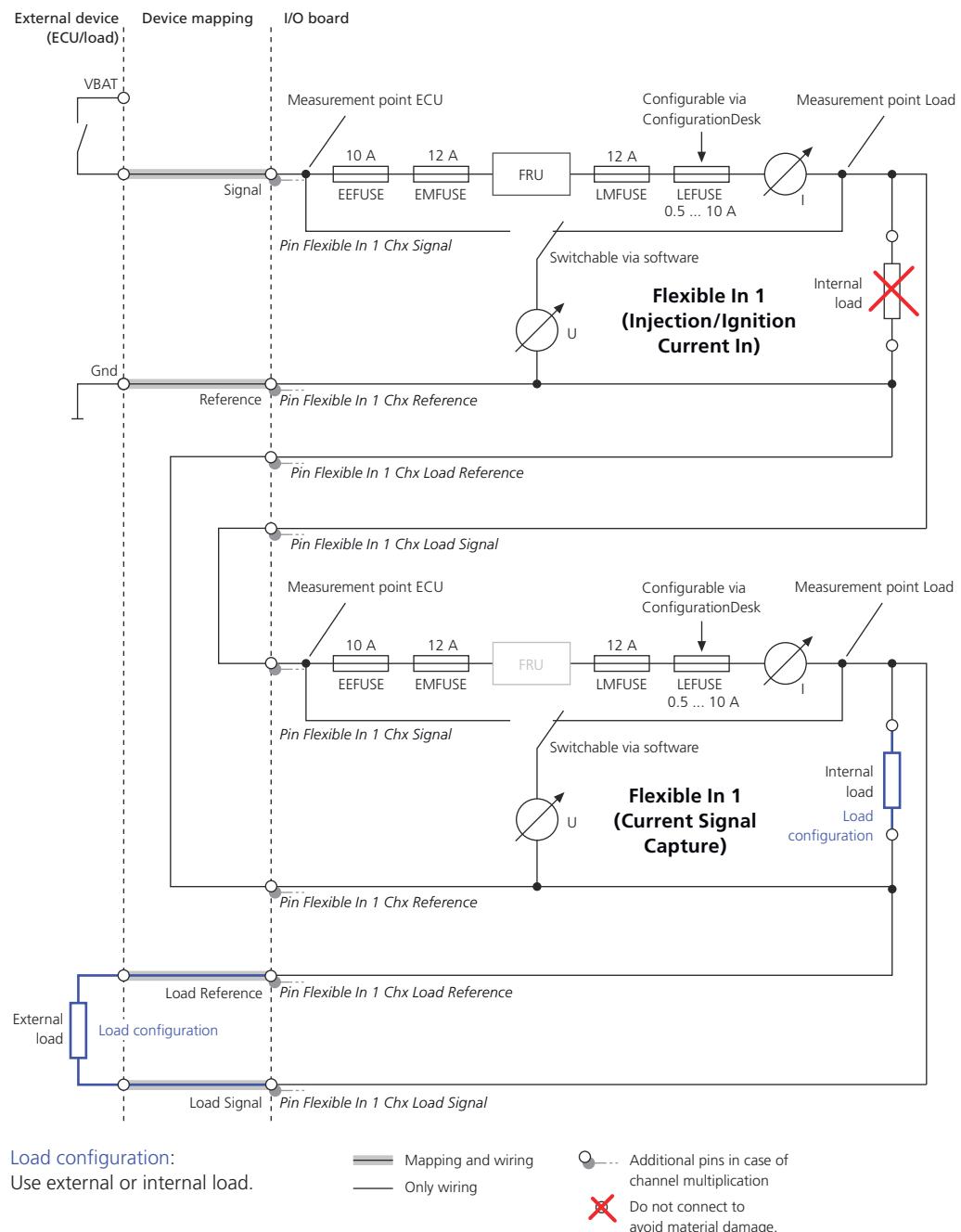
- Mapping and wiring
- Only wiring
- Additional pins in case of channel multiplication
- ✗ Do not connect to avoid material damage.

Low-side controlled digital current measurement with analog current measurement (on DS2601)

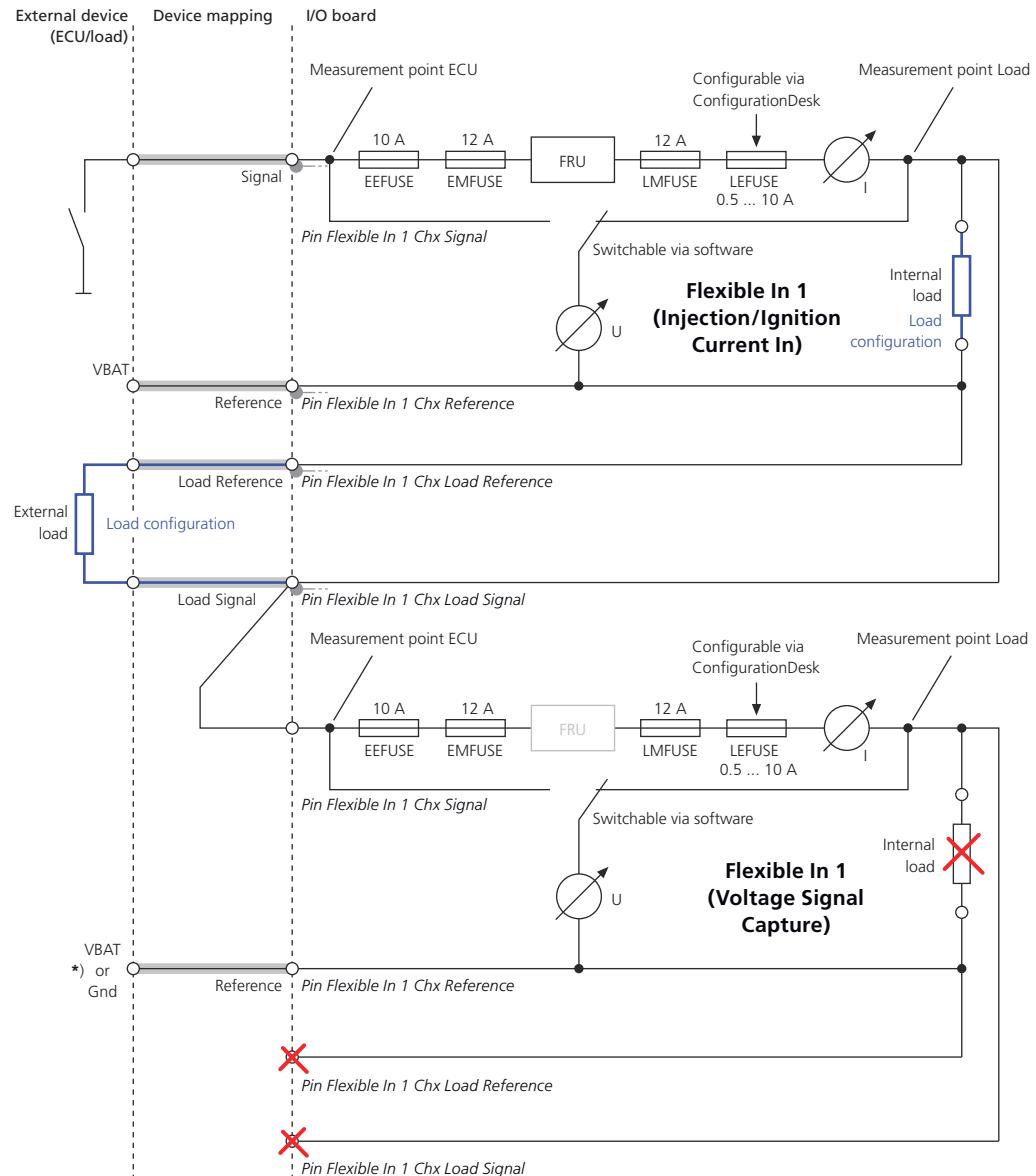


Load configuration:
Use external or internal load.

High-side controlled digital current measurement with analog current measurement (on DS2601)



Low-side controlled digital current measurement with analog voltage measurement (on DS2601)

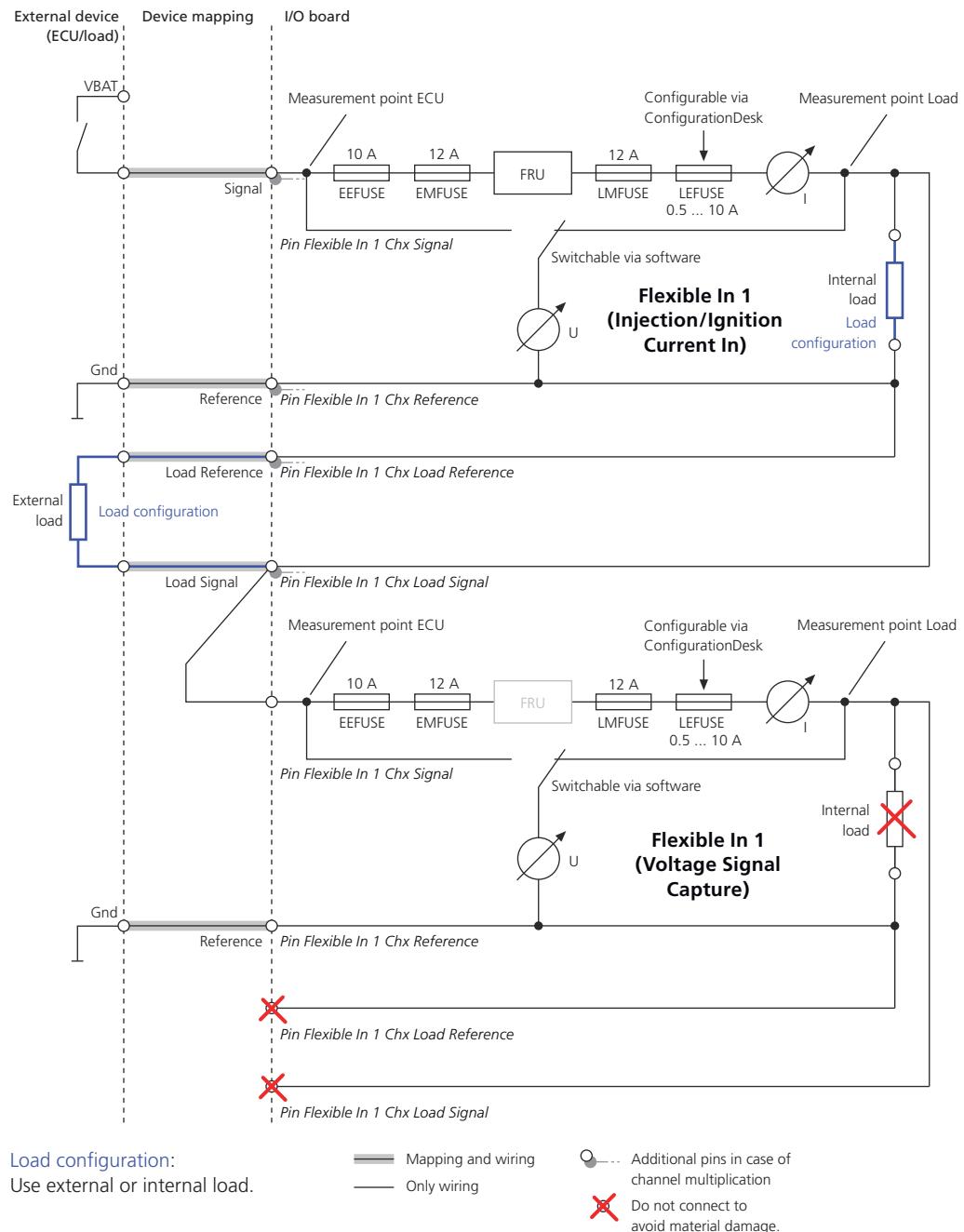


Load configuration:
Use external or internal load.

- Mapping and wiring
- Only wiring
- Additional pins in case of channel multiplication
- ✗ Do not connect to avoid material damage.

*) Reference for voltage measurement.
VBAT or Gnd can be used optionally.

**High-side controlled digital current measurement with analog voltage measurement
(on DS2601)**



Hardware Dependencies (Injection/Ignition Current In)

SCALEXIO Hardware Dependencies (Injection/Ignition Current In)

Digital Interface:
Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Flexible In 2	Flexible In 1
Hardware		DS2680 I/O Unit	DS2601 Signal Measurement Board
Event generation		✓	✓
Input current range		<ul style="list-style-type: none"> ▪ 0 ... +6 A_{RMS} for 1 channel ▪ 86.4 A_{RMS}¹⁾ for 18 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +10 A_{RMS} for 1 channel ▪ 80 A_{RMS} for 10 channels (channel multiplication)
Input voltage range		0 ... +60 V	-60 V ... +60 V
Threshold	Input threshold range	-18 A ... +18 A	-30 A ... +30 A
	Input hysteresis voltage	200 mA (typ.)	125 mA (typ.)
	DAC resolution	8 bit	14 bit
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs	0.264 µs ... 131 µs
Angular processing unit	Number of slaves	6	1
	Maximum speed	168,000 °/s	
Measurement point		Load side	
Configurable fuse		–	<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS}
Circuit diagrams		Flexible In 2 on page 1595	Flexible In 1 on page 1593
Required channels		1 (additional channels are required for current enhancements.)	

¹⁾ This value is the upper limit you can set in ConfigurationDesk and is used for channel multiplication (refer to [Specifying Current and Voltage Values for Channel Multiplication](#) on page 95). The DS2680 I/O Unit supports a maximum input current of 50 A.

Analog Interface:
Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog In 1	Flexible In 2	Flexible In 1
Hardware	DS2680 I/O Unit		DS2601 Signal Measurement Board
Input current range	0 ... +6 A _{RMS}		0 ... +10 A _{RMS}
Input voltage range	0 ... +60 V		-60 V ... +60 V

Channel Type		Analog In 1	Flexible In 2	Flexible In 1
Supported interface type		Differential with ground sense		<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Differential with ground sense
Measurement range per channel		$\pm 18\text{ A}$		$\pm 30\text{ A}$
Sample period	Range	$11.664\text{ }\mu\text{s} \dots 0.1\text{ s}$		$4.012\text{ }\mu\text{s} \dots 0.1\text{ s}$
	Resolution	$11.664\text{ }\mu\text{s}$		68 ns
ADC resolution		16 bit		16 bit
Offset error		$\pm 20\text{ mV}$ (typ.)	$\pm 25\text{ mA}$ (typ.)	$\pm 5\text{ mA}$ (typ.)
Gain error		$\pm 0.5\%$ (typ.)	$\pm 1.0\%$ (typ.)	$\pm 0.1\%$ (typ.)
Analog low-pass filter	Filter type	–		2 nd -order Bessel filter
	Edge frequency (-3dB cutoff frequency)	–		20 kHz
Angular processing unit	Number of slaves	6		1
	Maximum speed	$168,000\text{ }^\circ/\text{s}$		
Configurable fuse		–		<ul style="list-style-type: none"> ▪ Load side ▪ $0.5\text{ A}_{\text{RMS}} \dots 10\text{ A}_{\text{RMS}}$
Circuit diagrams		Mapping Signal Ports When Using Extended Signal Analysis (Injection/Ignition Current In) on page 441		
Required channels		1		

General limitations

- Channel multiplication across several I/O boards is not supported.
- Basic signal analysis and extended signal analysis must use channels that are located on the same I/O board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Crank/Cam (Voltage Out, Current Sink, Digital Out)

Where to go from here

Information in this section

Basics on Crankshaft and Camshaft Signals and Sensors.....	451
Creating Crankshaft/Camshaft Wavetables.....	458
Assigning Wavetables to the Real-Time Application.....	463
Upgrading DS2211 Wavetables.....	467
Crank/Cam Voltage Out.....	468
Crank/Cam Current Sink.....	478
Crank/Cam Digital Out.....	488

Basics on Crankshaft and Camshaft Signals and Sensors

Where to go from here

Information in this section

Operation Principle of Crankshaft and Camshaft Sensors.....	451
Basics on Crankshaft and Camshaft Signal Generation.....	454
Basics on Reverse Crankshaft Signal Generation.....	456

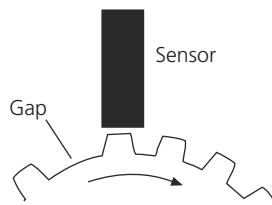
Operation Principle of Crankshaft and Camshaft Sensors

Objective

To familiarize you with the operation principle of crankshaft and camshaft sensors.

Crankshaft sensor (standard design)

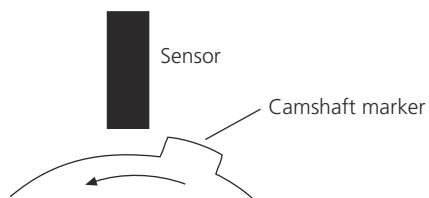
The crankshaft sensor is used to measure the angular position of the crankshaft. It is an optical or magnetic sensor which scans the radial surface of the rotating crankshaft wheel (also called the timing wheel). This is an even-toothed wheel flanged to the crankshaft. A typical number of teeth on a crankshaft wheel is "60 – 2", which means there are 60 teeth, and two of the teeth are missing to indicate a defined position, e.g., the 0° position.



The generated crankshaft signal is processed by a car's ECU and increments an angle counter inside the ECU. Injection and ignition pulse generation, for example, is based on the angle counter. The range of the angle counter is $0^\circ \dots 720^\circ$ ($720^\circ = 0^\circ$).

Camshaft sensor

The camshaft sensor is an optical or magnetic sensor which scans the radial surface of the rotating camshaft wheel. This is a smooth wheel flanged to the camshaft and equipped with one or more markers. The sensor detects the edges of the marker(s). A marker is specified by a start angle and an end angle, the angle values fit the range $0^\circ \dots 720^\circ$.

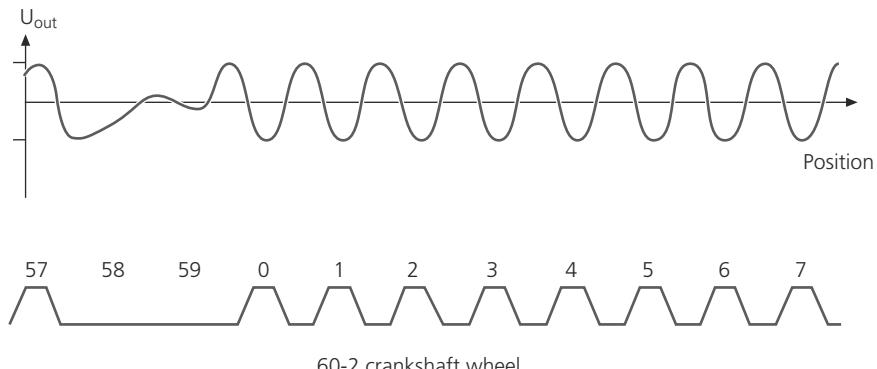


The generated camshaft signal is processed by the engine management system, for example, a car's ECU. The signal is used to synchronize the measurement of the crankshaft position. It helps identify the stroke, for example, as compression or exhaust. In addition, camshaft position measurement is used to measure a camshaft phase shift.

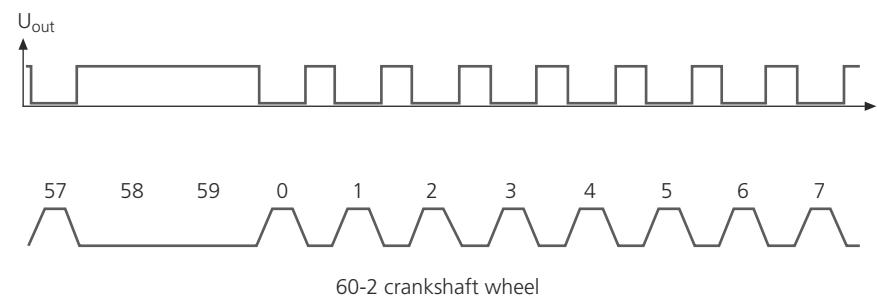
Sensor type

Crankshaft and camshaft sensors can be either passive or active.

Passive sensors Generate an analog signal. A rising tooth edge on the wheel corresponds to the signal crossing zero from positive to negative signal level. The following illustration shows a passive crankshaft signal.



Active sensors Generate a digital signal (=TTL signal). Usually, a rising tooth edge on the wheel corresponds to a falling signal edge, and vice versa. The following illustration shows an active crankshaft signal.

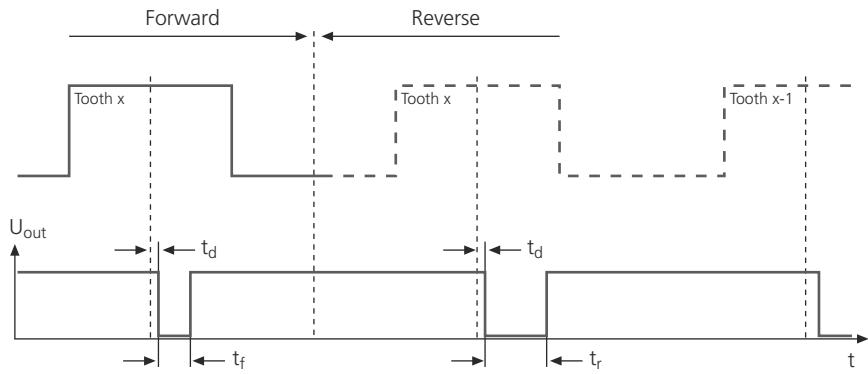


Intelligent crank sensor with direction detection

To detect reverse crankshaft rotation, a more sophisticated type of crankshaft sensor is needed. Reverse crankshaft rotation happens, for example, in the following situations:

- Existing pressure in cylinders due to compression is reduced after engine shutdown.
- The engine is restarted according to combustion, as is done by modern Stop-Start Automatics.

Unlike standard crankshaft sensors, so-called *reverse crankshaft sensors* can get exact information about the crankshaft position, even on reverse rotation of the crankshaft. Reverse crankshaft sensors generate pulse signals. The temporal pulse length of these signals depends on the current rotation direction (e.g., forward: $t_f = 45 \mu s$, reverse: $t_r = 90 \mu s$). The following illustration shows the signal generation of a reverse crankshaft sensor. The trigger point for pulse generation in the example is a tooth's center (td is a time delay, i.e., the sensor's reaction time between trigger and pulse).



Basics on Crankshaft and Camshaft Signal Generation

Simulation approach

Instead of having a real engine with real sensor equipment, the SCALEXIO system mimics real crankshaft and camshaft signals. For the SCALEXIO system, crankshaft and camshaft signals are encoded by numeric data stored in so-called *wavetables*.

Crankshaft and camshaft wavetables

Crankshaft as well as camshaft wavetables are 1-dimensional arrays comprising $2^{16} = 65536$ values of float type in the range [- 1.0 ... +1.0]. They are stored as CSV files. Wavetables always cover one engine cycle (0° ... 720°). Thus, the angular resolution of crankshaft and camshaft wavetables - rounded to 3 decimal digits - is 0.011°. Each wavetable value relates to a certain angle counter value. The following table illustrates a crankshaft wavetable (the angle values are not part of the wavetable)

Value_1	0.000°
Value_2	0.011°
Value_3	0.022°
...	...
Value_65535	719.978°
Value_65536	719.989°

Signal generation and conditioning

Looking up wavetable values During engine simulation the active wavetable is accessed like a look-up table. The current value of the angle counter acts as the index. The index determines which element of the wavetable is to be read out. As the angle counter continuously increments during engine simulation, the wavetable values are read out consecutively. The wavetables values are read out backwards, if reverse crankshaft and camshaft rotation is simulated.

Note

With a high engine speed, it can happen that wavetable values are skipped, hence, the angular resolution decreases.

Conditioning wavetable values The wavetable values are normalized values, hence they are not returned as the actual output signal.

Tip

Conditioning lets you generate different output signals that all refer to the same wavetable.

The processing of the wavetable values depends on the simulated sensor type (passive or active).

- If you simulate a passive crankshaft sensor, the wavetable values are multiplied by the amplitude (refer to the Amplitude function port).

$$\text{Output Signal} = \text{Wavetable Value} * \text{Amplitude}$$

The following table shows an example wavetable and the resulting output signal for an amplitude of 20 V.

Wavetable	Output Signal
0.0	0 V
0.5	10 V
-1.0	-20 V

- If you simulate an active crankshaft sensor, the wavetable values are interpreted either as 0 or 1.

$$\text{Output Signal} = 0, \text{ if Wavetable Value} \leq 0.0$$

$$\text{Output Signal} = 1, \text{ if Wavetable Value} > 0.0$$

The following table shows an example wavetable and the resulting output signal.

Wavetable	Output Signal
0.0	0
0.5	1
-1.0	0

Tool-based wavetable creation

The demo examples offer MATLAB scripts (M files) for creating crankshaft and camshaft tables automatically. For details on the tools, refer to [Creating Crankshaft/Camshaft Wavetables](#) on page 458.

On-board wavetable memory

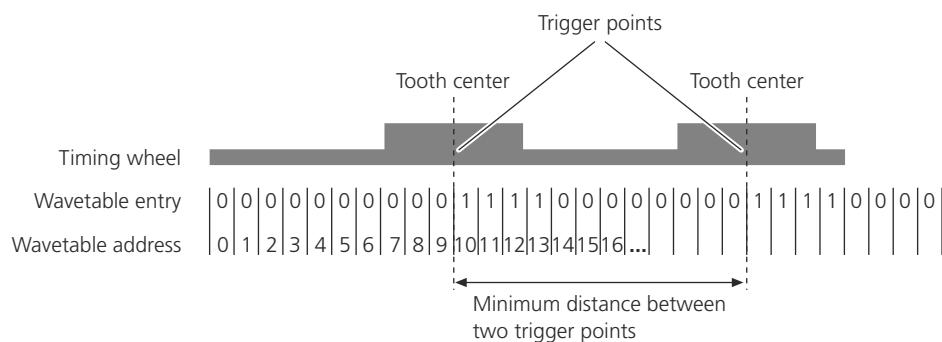
Wavetables are stored in the *wavetable memory* of an I/O board when the real-time application is downloaded to the target hardware. The size of the

wavetable memory depends on the I/O board. For the maximum number of wavetables that can be stored, refer to [Configuring the Basic Functionality \(Crank/Cam Voltage Out\)](#) on page 473.

Basics on Reverse Crankshaft Signal Generation

Reverse crankshaft wavetables

Basically, reverse crankshaft signal generation is identical to standard crankshaft signal generation, as a wavetable is looked up. However, wavetables used for reverse crankshaft signal generation must meet some additional requirements:



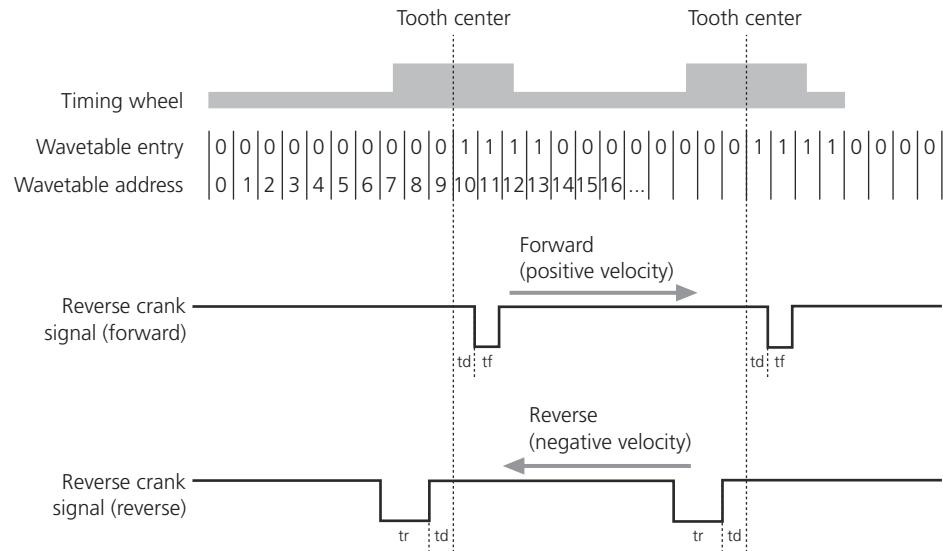
- The minimum distance between two trigger points must be 364 wavetable entries ($= 4.0^\circ$). If you use shorter distances, the behavior will not be defined. Note that for a 60-2 wheel, for example, around 546 wavetable entries correspond to one tooth.
 - A trigger point for low active pulse generation is defined by a 0-1 transition of the wavetable entries.

Note

Whether a trigger point refers to the rising edge of a tooth (as for standard crankshaft signals) or to the tooth center depends on the sensor.

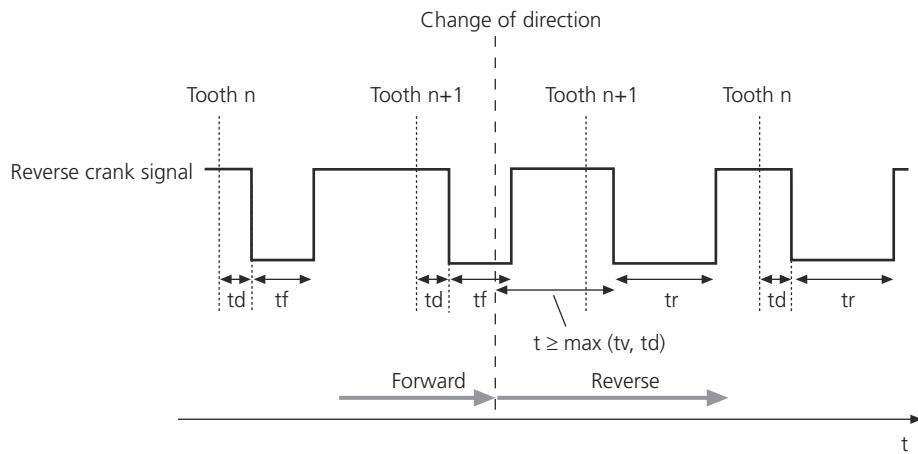
Output signal sent to ECU

The SCALEXIO system outputs a low active pulse signal. Whether forward or reverse rotation of the crankshaft is assumed depends only on the sign of the signal that is connected to the Velocity function port of the Angular Clock Setup function block in the ConfigurationDesk application. As long as the sign is positive, forward rotation is assumed and the angle counter increases. The wavetable entries are looked up in ascending order. For a negative sign, i.e., reverse rotation, the angle counter decreases. The wavetable entries are then looked up in descending order. The following illustration shows the relationship between wavetable access (forward or reverse) and PWM signal generation:



t_f is the pulse length that relates to forward rotation of the crankshaft. t_r is the pulse length that relates to reverse rotation of the crankshaft.

Change of direction The illustration shows a low active reverse crankshaft sensor signal. The rotation direction changes shortly after the trigger point of tooth $n+1$ has passed. At the *change of direction* mark, you can see that the active pulse t_f is followed by a non-active pulse $\max(t_v, t_d)$, i.e., the greater value of t_v and t_d . t_v is the user-defined inactive time between two pulses of different rotation directions. t_d is a time delay, i.e., the sensor's reaction time between trigger and pulse. After $\max(t_v, t_d)$ an active pulse t_r is generated. The pulse after $\max(t_v, t_d)$ is always a complete one. This ensures the correct transmission of the rotation direction.



Rotation speed limit The time between two trigger points must be $\max(t_f+t_d, t_f+t_v, t_r+t_d, t_r+t_v)$, at least, when the direction of rotation is the same. If the rotation speed becomes too fast, the active pulse is cut by the next trigger point.

Wavetable tool support	You can create crankshaft and camshaft tables by using a tool, refer to How to Create Reverse Crankshaft Wavetables on page 461.
-------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------

Creating Crankshaft/Camshaft Wavetables

Purpose	You can use MATLAB scripts (M files) to conveniently create crankshaft and camshaft wavetable files.
----------------	------------------------------------------------------------------------------------------------------

Where to go from here	Information in this section										
	<table><tr><td>Basics on Crankshaft/Camshaft Wavetable Files.....</td><td>458</td></tr><tr><td>How to Create Analog Crankshaft Wavetables.....</td><td>459</td></tr><tr><td>How to Create Digital Crankshaft Wavetables.....</td><td>459</td></tr><tr><td>How to Create Reverse Crankshaft Wavetables.....</td><td>461</td></tr><tr><td>How to Create Camshaft Wavetables.....</td><td>462</td></tr></table>	Basics on Crankshaft/Camshaft Wavetable Files.....	458	How to Create Analog Crankshaft Wavetables.....	459	How to Create Digital Crankshaft Wavetables.....	459	How to Create Reverse Crankshaft Wavetables.....	461	How to Create Camshaft Wavetables.....	462
Basics on Crankshaft/Camshaft Wavetable Files.....	458										
How to Create Analog Crankshaft Wavetables.....	459										
How to Create Digital Crankshaft Wavetables.....	459										
How to Create Reverse Crankshaft Wavetables.....	461										
How to Create Camshaft Wavetables.....	462										

Basics on Crankshaft/Camshaft Wavetable Files

CSV files	Crankshaft/camshaft wavetables are stored in CSV files. CSV means comma-separated values.
------------------	-------------------------------------------------------------------------------------------

CSV requirements CSV files must meet the following basic requirements:

- ASCII file
- Entries are separated by a comma (",")
- New line at the end of a file
- No additional data or text is included

Crankshaft/camshaft wavetable-specific requirements Crankshaft/camshaft wavetable files must also meet the following requirements:

- A crankshaft/camshaft wavetable file consists of 65,536 entries.
- A crankshaft/camshaft wavetable entry consists of one floating-point value (e.g., 1.0).
- The format of the decimal point is a dot (".").
- The values fit the range [-1.0 ... 1.0].

How to Create Analog Crankshaft Wavetables

Objective	If you want to simulate a passive (analog) crankshaft sensor you must provide an analog crankshaft wavetable.
------------------	---------------------------------------------------------------------------------------------------------------

Preconditions	MATLAB is open. The Current Folder contains the <code>crank_syn.m</code> script, which creates analog crankshaft wavetables.
----------------------	------------------------------------------------------------------------------------------------------------------------------



The `crank_syn.m` script originally resides in the `<DocumentsFolder>\EngineConfigurationDemo\EngineConfiguration\EngineExample\Wavetables` folder.

Restriction	The number of gaps is limited to one.
--------------------	---------------------------------------

Method	<p>To create analog crankshaft wavetables</p> <ol style="list-style-type: none"> The <code>crank_syn.m</code> script creates an analog wavetable that corresponds to a 60-2 crank wheel. However, you can change the setting by editing the following part of the script. <pre>% absolute number of teeth per 360 deg NUM_TEETH_ABS = 60; % real number of teeth per 360 deg (absolute - missing % teeth) NUM_TEETH_REAL = 58;</pre> <ol style="list-style-type: none"> In the Current Folder dialog, right-click <code>crank_syn.m</code> and select Run. The <code>crank_syn.m</code> script is being executed.
---------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Result	A Figure dialog opens in MATLAB, displaying the wavetable. The Current Folder contains the <code>crank_syn.csv</code> crankshaft wavetable file.
---------------	--------------------------------------------------------------------------------------------------------------------------------------------------

How to Create Digital Crankshaft Wavetables

Objective	If you want to simulate an active (digital) crankshaft sensor you must provide a digital crankshaft wavetable.
------------------	----------------------------------------------------------------------------------------------------------------

Preconditions

MATLAB is open. The Current Folder contains the `crank_digital_syn.m` script, which creates digital crankshaft wavetables.



The `crank_digital_syn.m` script originally resides in the `<InstallationFolder>\Demos\ConfigurationDesk\EngineConfigurationDemo\EngineConfiguration\EngineExample\Wavetables` folder.

Restriction

The number of gaps is limited to one.

Method**To create digital crankshaft wavetables**

- 1 The `crank_digital_syn.m` script creates a digital wavetable that corresponds to a 60-2 crank wheel. However, you can change the setting by editing the following part of the script.

```
% Crankshaft definitions
TEETH_NUMBER = 60; % Number of teeth overall
TEETH_MISSING = 2; % Number of Missing teeth
OFFSET_DEGREE = 0.00; % OFFSET_DEGREE [in degree,
% -360° ... + 360°]
```

The offset value is identical to the angular position of the first rising edge after the gap, for example:

Offset	First Rising Edge After The Gap
0°	0°
+3°	3°
-12°	348° (= -12° + 360°)

- 2 In the Current Folder dialog, right-click `crank_digital_syn.m` and select Run.

The `crank_digital_syn.m` script is being executed.

Result

A Figure dialog opens in MATLAB, displaying the wavetable. The Current Folder contains the `crank_digital_syn.csv` crankshaft wavetable file.

How to Create Reverse Crankshaft Wavetables

Objective

If you want to simulate a reverse crankshaft sensor you can use a digital crankshaft wavetable. Depending on the sensor it might be necessary to specify an angle offset.

Preconditions

MATLAB is open. The Current Folder contains the `crank_digital_syn.m` script, which creates digital crankshaft wavetables.



The `crank_digital_syn.m` script originally resides in the `<DocumentsFolder>\EngineConfigurationDemo\EngineConfiguration\EngineExample\Wavetables` folder.

Restriction

The number of gaps is limited to one.

Method

To create reverse crankshaft wavetables

- 1 The `crank_digital_syn.m` script creates a digital wavetable that corresponds to a 60-2 crank wheel. However, you can change the setting by editing the following part of the script.

```
% Crankshaft definitions
TEETH_NUMBER = 60; % Number of teeth overall
TEETH_MISSING = 2; % Number of Missing teeth
OFFSET_DEGREE = 0.00; % OFFSET_DEGREE [in degree,
% -360° ... + 360°]
```

If a trigger point does not refer to the rising edge of a tooth (as for standard crankshaft signals) but to the tooth center, for example, you have to specify an offset:

```
OFFSET_DEGREE = 1/4 * 360° / TEETH_NUMBER
```

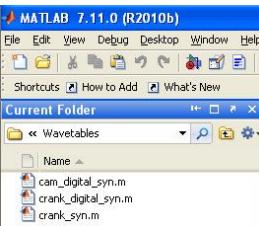
- 2 In the Current Folder dialog, right-click `crank_digital_syn.m` and select Run.

The `crank_digital_syn.m` script is being executed.

Result

A Figure dialog opens in MATLAB, displaying the wavetable. The Current Folder contains the `crank_digital_syn.csv` crankshaft wavetable file.

How to Create Camshaft Wavetables

Objective	Engine simulation requires one or more camshaft wavetables.
Preconditions	MATLAB is open. The Current Folder contains the <code>cam_digital_syn.m</code> script, which creates camshaft wavetables. 
	The <code>cam_digital_syn.m</code> script originally resides in the <code><DocumentsFolder>\EngineConfigurationDemo\EngineConfiguration\EngineExample\Wavetables</code> folder.
Signal configuration	You can configure the generation of the camshaft signal by editing the following part of the <code>cam_digital_syn.m</code> script. <pre>% edge positions EDGE1_POS = 339; EDGE2_POS = 419;</pre> In the example above, a cam marker with start angle = 339° and end angle = 419° is simulated.
	Limitation For the <code>cam_digital_syn.m</code> script, the number of cam markers is limited to one.
Method	To create camshaft wavetables 1 In the Current Folder dialog, right-click <code>cam_digital_syn.m</code> and select Run File. The <code>cam_digital_syn.m</code> script is executed.
Result	A Figure dialog opens in MATLAB, displaying the wavetable. The Current Folder contains the resulting <code>cam_digital_syn.csv</code> camshaft wavetable file.

Assigning Wavetables to the Real-Time Application

Purpose	You can select the wavetables that are used for crankshaft/camshaft signal generation.
----------------	----------------------------------------------------------------------------------------

Where to go from here	Information in this section
	How to Import Wavetables to a Crank/Cam Function Block463
	Switching the Active Wavetable at Run Time465
	Replacing Wavetable Files at Run Time466

How to Import Wavetables to a Crank/Cam Function Block

Objective	Crank/Cam function blocks require at least one wavetable for crankshaft/camshaft signal generation.
------------------	-----------------------------------------------------------------------------------------------------

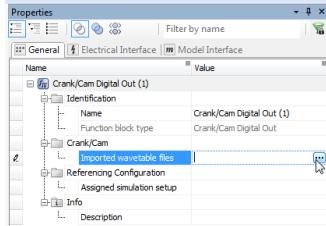
Maximum number of wavetables	The total number of wavetables that can be imported is limited as follows:
	Hardware
	DS2680 I/O Unit
	DS2621 Signal Generation Board
	DS6101 Multi-I/O Board

For more information, refer to one of the following chapters:

- [Configuring the Basic Functionality \(Crank/Cam Voltage Out\) on page 473](#)
- [Configuring the Basic Functionality \(Crank/Cam Current Sink\) on page 483](#)
- [Configuring the Basic Functionality \(Crank/Cam Digital Out\) on page 493](#)

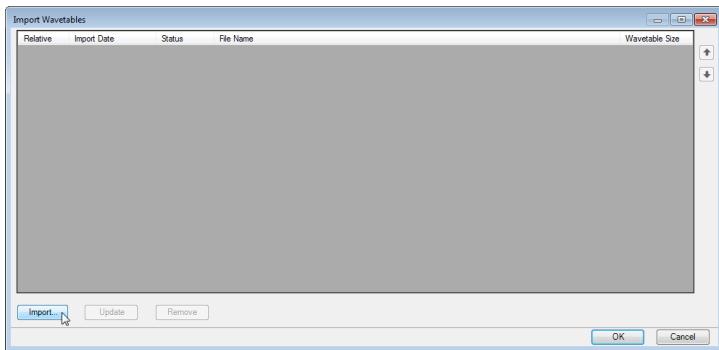
Method	To import wavetables to a Crank/Cam function block
	<ol style="list-style-type: none"> 1 Select the Crank/Cam function block. Its properties are displayed in the Properties Browser.

- 2 Click the Browse button in the Imported wavetable files field.



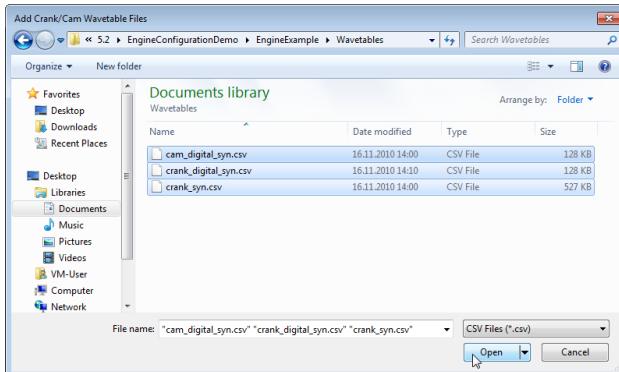
The Import Wavetables dialog opens.

- 3 Click Import.



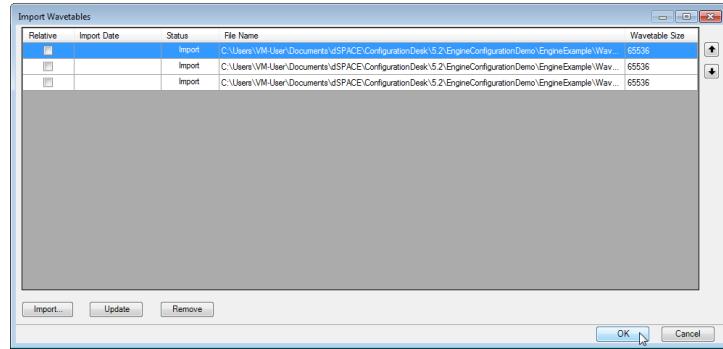
The Add Crank/Cam Wavetable Files dialog opens.

- 4 Select the wavetable file(s) and click Open. Multi-selection is possible.



The wavetable file is checked for validity, for example, appropriate number of values (65536) and value range [-1.0 ... +1.0].

- 5 When you have added all the required wavetables, click **OK** in the Import Wavetables dialog.



Result

You have imported wavetables to the Crank/Cam function block. The wavetable files are indexed [1 ... N]. 1 refers to the wavetable file at the beginning of the list (here: `Crank_Cam_Wave_1.csv`). The index is required when you specify, for example, the active wavetable (refer to [Switching the Active Wavetable at Run Time](#) on page 465).

Note

If a wavetable memory overflow is detected during the build process, all the Crank/Cam function blocks that are related to this wavetable memory have only the default wavetable assigned.

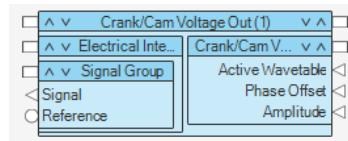
Switching the Active Wavetable at Run Time

Objective

To select the wavetable that is currently used for crankshaft or camshaft signal generation. Switching is possible at run time. Each Crank/Cam function block is related to a set of wavetables, one of which is the active wavetable.

Active wavetable selection at run time

The Active Wavetable function port of the Crank/Cam Out function is connected to the behavior model. The behavior model sends a signal to this function port. The signal must be an integer in the range [1 ... N]. N is the total number of wavetables linked to the function block. If the function port is not connected to the behavior model, the initial port value is used (default: 1).



Replacing Wavetable Files at Run Time

downwave.exe

The `downwave.exe` utility replaces one of the wavetables linked to function block when the real-time application is stopped by the experiment tool. The change takes effect when the application status changes from stopped to running. The utility is located in `<InstallationFolder>\SCALEXIO\Win32`.

Syntax

The `downwave.exe` utility is invoked by using the following syntax:

```
downwave.exe [general options] [action-specific options]  
[download options]
```

The following *general* options are available:

/b IP address of the real-time system or an alias name (refer to the `hosts` file in `%SYSTEM_ROOT%\system32\drivers\etc` which contains the mapping of host names to IP addresses).

/appl Name of the application

/id Identifier of the application

/t Identifier of the target board, e.g., DS2621 (default), DS2680, or DS6101.

/q Quiet mode (only error messages are issued)

/v Display verbose information

/? Display available options

The following *action-specific* options are available (select one):

/list List all the wavetables that relate to the application.

/rep Replace an existing wavetable file (default). This requires the *download* options (see below).

/new Download a new wavetable file. Replacing existing wavetables is disabled.

/f Force download of wavetables. Existing wavetables are replaced, missing wavetables are downloaded.

The following *download* options are available:

/block Name of the function block to which the wavetable to be replaced is linked, for example:

Use " " when the string includes blanks.

/idx Index of the wavetable that is to be replaced. To identify a wavetable's index, check the **Imported wavetable files** property of the function block in the **Properties Browser**. The wavetable at the beginning of the list is indexed as 1.

/src Path and name of the replacing wavetable file (CSV file), for example:

You must specify the path if the wavetable file is stored in a different folder than the **downwave** utility. Use " " when the string includes blanks.

Example #1

The command `downwave.exe /b 192.168.1.1 /list` lists all the wavetables that are linked to the application loaded on the real-time system whose IP address is 192.168.1.1. A message like the following is displayed:

```
downwave.exe - Wavetable download tool for SCALEXIO, Ver. 1.1 - 32,
(C) 2010 by dSPACE GmbH

List of wavetables for application 'Application_001.x86' (ID:
0x7891F7) on target (IP: 192.168.1.1).

Function block: ' Digital Out 1'

Index 1 (source file: _1)
Index 2 (source file: _2)
Index 3 (source file: _3)
Index 4 (source file: _4)

Function block: ' Digital Out 2'

Index 2 (source file: _0)
```

Example #2

The command

`downwave.exe /b 192.168.1.1 /src _11.csv /block
" Digital Out 1" /idx 1` performs the following replacement:

- Replaced wavetable:

This wavetable is stored on the real-time system whose IP address is 192.168.1.1. The wavetable is linked to the Digital Out 1 function block. It is the 1st entry in the Wavetable files list (`_1.csv`).

- Replacing wavetable:

This is the `_11.csv` wavetable file. It is stored in the same folder as the `downwave.exe` utility.

Upgrading DS2211 Wavetables

How to Convert a MAT Wavetable to CSV

Objective

To convert a DS2211 wavetable file to a SCALEXIO-compatible wavetable file.

MatConv.exe	The MatConv.exe conversion utility converts a DS2211 wavetable MAT file to a wavetable CSV file (comma separated values) which is compatible with the SCALEXIO system. The utility is located in <InstallationFolder>\Exe.
Preconditions	<ul style="list-style-type: none">▪ Crankshaft/camshaft wavetable files must be compatible with the SCALEXIO system. Therefore they must meet the requirements (for example, the number of entries) specified in Basics on Crankshaft/Camshaft Wavetable Files on page 458.▪ You must have installed RTLib1005 or RTLib1006.
Method	To convert a MAT wavetable to CSV 1 Invoke the MatConv.exe conversion utility by the following syntax: <code>matconv [MAT file] /csv</code> For example: <code>matconv wave1.mat /csv</code>
Result	The wavetable MAT file is converted to a wavetable CSV file.
Next step	After the wavetable is converted, you must link it to the Crank/Cam function block and build the real-time application in ConfigurationDesk, again.

Crank/Cam Voltage Out

Where to go from here

Information in this section

Introduction (Crank/Cam Voltage Out).....	469
Overviews (Crank/Cam Voltage Out).....	470
Configuring the Function Block (Crank/Cam Voltage Out).....	473
Hardware Dependencies (Crank/Cam Voltage Out).....	476

Introduction (Crank/Cam Voltage Out)

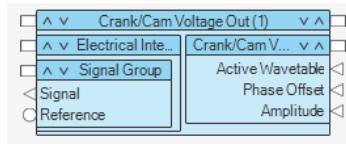
Introduction to the Function Block (Crank/Cam Voltage Out)

Function block purpose

The Crank/Cam Voltage Out function block type lets you output voltage signals to simulate inductive crankshaft and camshaft sensors.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Importing wavetable files and executing them for simulating crankshaft or camshaft sensor signals.
- Controlling the output signal (phase offset, amplitude, adding noise and voltage offset).
- Switching executed wavetables at runtime.

Supported channel types

The Crank/Cam Voltage Out function block supports the following channel types:

	SCALEXIO							MicroAutoBox III
Channel type	Analog Out 1	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 8	Analog Out 9	Flexible Out 1	-
Hardware	DS2680				DS6101			DS2621

Introduction to crankshaft/camshaft signal generation

For basic information, refer to [Basics on Crankshaft and Camshaft Signals and Sensors](#) on page 451.

Oversviews (Crank/Cam Voltage Out)

Where to go from here

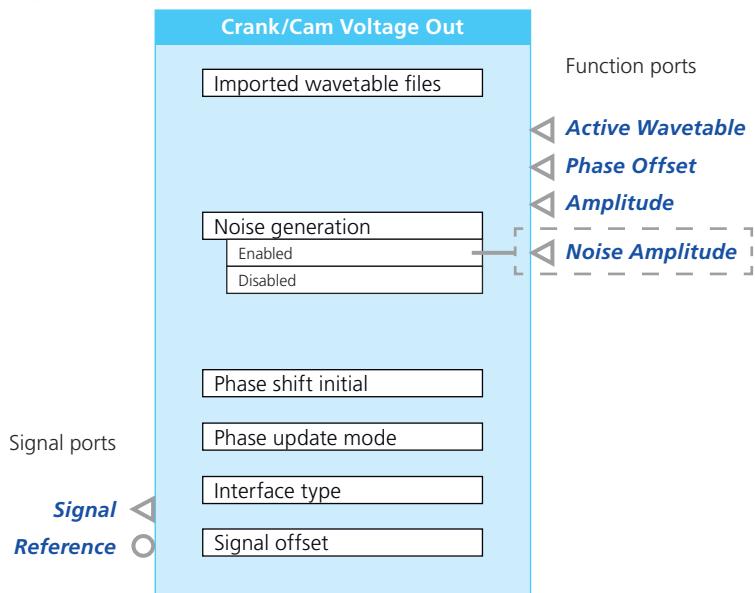
Information in this section

- [Overview of Ports and Basic Properties \(Crank/Cam Voltage Out\)..... 470](#)
- [Overview of Tunable Properties \(Crank/Cam Voltage Out\)..... 472](#)

Overview of Ports and Basic Properties (Crank/Cam Voltage Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... n ▪ n equals the number of imported wavetable files
Dependencies	-

Phase Offset

This function import lets you define the 0° position independently of the active wavetable from within the behavior model. Usually, a phase offset is applied to the camshaft according to the speed and load (camshaft phase shift).

Note

The phase offset is set relative to the initial phase shift offset defined by the Phase shift initial property.

Value range	<ul style="list-style-type: none"> ▪ $-360^\circ \dots +360^\circ$ ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Amplitude

This function import lets you define the amplitude of the output signal from within the behavior model. The amplitude of passive sensors (e.g., inductive sensors) typically depends on the angle velocity of the crankshaft or camshaft. The voltage output signal results from the current wavetable value $[-1.0 \dots +1.0]$ multiplied by the current value of the Amplitude function port.

Value range	<ul style="list-style-type: none"> ▪ $0 \text{ V} \dots U_{\max}$ (in Volt). ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Voltage Out) on page 476. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $0 \dots U_{\max \text{ Noise}}$ (in Volt). ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Voltage Out) on page 476. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ 0 V: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Noise generation property is set to Enabled. ▪ Supported only for Flexible Out 1, Analog Out 3 and Analog Out 8 channel types.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Voltage Out) on page 476.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Voltage Out) on page 476.
Dependencies	–

Overview of Tunable Properties (Crank/Cam Voltage Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Imported Wavetable Files ³⁾	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Phase update mode	✓	–
Phase shift initial	✓	–
Signal offset	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ Not possible by the experiment software but by the **downwave** assistance tool. Refer to [Replacing Wavetable Files at Run Time](#) on page 466.

Configuring the Function Block (Crank/Cam Voltage Out)

Where to go from here

Information in this section

- | | |
|----------------------------------------------------------------------------------|---------------------|
| Configuring the Basic Functionality (Crank/Cam Voltage Out)..... | 473 |
| Configuring Standard Features (Crank/Cam Voltage Out)..... | 474 |

Configuring the Basic Functionality (Crank/Cam Voltage Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Referencing virtual piston engine
- Referencing crankshaft/camshaft wavetables
- Setting a phase shift (typically for the camshaft)
- Adding noise to the output signal
- Specifying a signal offset

Referencing virtual piston engine

The Crank/Cam Out function block refers to a specific engine design by referencing an Engine Simulation Setup function block. The Engine Simulation Setup function block defines the number of cylinders (1 ... n) and provides further basic configuration data of the virtual engine.

Import wavetable files

Crankshaft and camshaft signals are encoded by numeric data stored in so-called wavetable files. A wavetable is related to a specific virtual sensor, for example, a 60-2 crankshaft wheel and a passive sensor.

The Import Wavetables dialog lets you import, update and remove wavetables files. For instructions, refer to [How to Import Wavetables to a Crank/Cam Function Block](#) on page 463.

Default wavetable ConfigurationDesk adds a default wavetable to the function block only if you do not import a wavetable file. The default wavetable ensures that at least one wavetable is provided for signal generation. All values of the default wavetable are 0.0.

Maximum number of wavetables Wavetables can be referenced by Crank/Cam and Angular Wavetable function blocks. The onboard memory of an I/O board limits the total number of wavetables that can be referenced by all function blocks of both types used in the application. Refer to the following table:

Hardware	Max. Number of Wavetables per Board
DS2680 I/O Unit	24
DS2621 Signal Generation Board	12
DS6101 Multi-I/O Board	12

A wavetable that is used n times by Crank/Cam or Angular Wavetable function blocks is stored n times in the wavetable memory. Potentially added default wavetables are also stored n times in the wavetable memory.

Initial phase shift

You can apply an angular offset (φ_{initial}) to the crankshaft and camshaft output signal. The actual phase shift φ_{actual} reads as follows:

$$\varphi_{\text{actual}} = \varphi_{\text{initial}} + \varphi_{\text{Phase offset}}$$

$\varphi_{\text{Phase offset}}$ is defined by the signal at the Phase offset function port, it can change at run time.

Adding noise to the output signal

The Crank/Cam Voltage Out function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Noise generation is supported only for the Flexible Out 1, Analog Out 3, and Analog Out 8 channel types.

Specifying a signal offset

You can apply a voltage offset to the output signal.

Limitation Applying a voltage offset to the output signal is *not* supported for the Analog Out 3 and Analog Out 8 channel types.

Related topics

HowTos

[How to Import Wavetables to a Crank/Cam Function Block.....](#) 463

Configuring Standard Features (Crank/Cam Voltage Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Analog Out 1	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 8	Analog Out 9	Flexible Out 1	Further Information
Interface type	Differential with ground sense	Galvanically isolated	Differential with ground sense	Differential with ground sense	Galvanically isolated	Differential with ground sense	Galvanically isolated	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	✓	–	–	✓	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports								
Trigger level of electronic fuse	–	–	–	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	✓	–	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 1](#) on page 1553
- [Analog Out 3](#) on page 1554
- [Analog Out 4](#) on page 1555
- [Analog Out 6](#) on page 1556
- [Analog Out 8](#) on page 1557

- [Analog Out 9](#) on page 1557
- [Flexible Out 1](#) on page 1604

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Crank/Cam Voltage Out)

SCALEXIO Hardware Dependencies (Crank/Cam Voltage Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 1	Analog Out 4	Analog Out 3	Flexible Out 1
Hardware	DS2680 I/O Unit			DS2621 Signal Generation Board
Noise generation	–		✓	✓
Noise amplitude range	–		<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication)
Output current range	0 ... +5 mA _{RMS}			0 ... +40 mA _{RMS}
Output voltage range	0 ... +10 V for 1 channel		<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication)
Reverse crankshaft signal generation	–			–
DC signal offset	✓		–	✓
Maximum number of wavetable files per board ¹⁾	24			12

Channel Type		Analog Out 1	Analog Out 4	Analog Out 3	Flexible Out 1
Angular processing unit	Number of slaves	6		1	
	Maximum speed	168,000 °/s			
Configurable fuse		–		–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Circuit diagrams		Analog Out 1 on page 1553	Analog Out 4 on page 1555	Analog Out 3 on page 1554	Flexible Out 1 on page 1604
Required channels		1		1 (additional channels are required for voltage enhancements.)	1 (additional channels are required for voltage enhancements.)

¹⁾ Total number of wavetable files that can be stored on the I/O board. For more information, refer to [Configuring the Basic Functionality \(Crank/Cam Voltage Out\)](#) on page 473.

Channel Type		Analog Out 6	Analog Out 8	Analog Out 9
Hardware		DS6101 Multi-I/O Board		
Noise generation		–	✓	–
Noise amplitude range		–	<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication) 	–
Output current range		0 ... +5 mA _{RMS}		
Output voltage range		0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	0 ... +10 V
Reverse crankshaft signal generation		–		
DC signal offset		✓	–	✓
Maximum number of wavetable files per board ¹⁾		12		
Angular processing unit	Number of slaves	6		
	Maximum speed	168,000 °/s		
Configurable fuse		–		
Circuit diagrams		Analog Out 6 on page 1556	Analog Out 8 on page 1557	Analog Out 9 on page 1557
Required channels		1	1 (additional channels are required for voltage enhancements.)	1

¹⁾ Total number of wavetable files that can be stored on the I/O board. For more information, refer to [Configuring the Basic Functionality \(Crank/Cam Voltage Out\)](#) on page 473.

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101](#) on page 1539.

General limitations

Channel multiplication across several I/O boards is not supported.

More hardware data	DS2680 I/O Unit For more board-specific data, refer to Data Sheet of the DS2680 I/O Unit (SCALEXIO Hardware Installation and Configuration) .
	DS2621 Signal Generation Board For more board-specific data, refer to Data Sheet of the DS2621 Signal Generation Board (SCALEXIO Hardware Installation and Configuration) .
	DS6101 Multi-I/O Board For more board-specific data, refer to Data Sheet of the DS6101 Multi-I/O Board (SCALEXIO Hardware Installation and Configuration) .

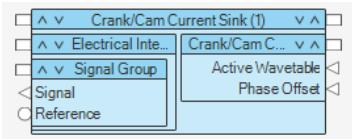
Crank/Cam Current Sink

Where to go from here	Information in this section								
	<table><tr><td>Introduction (Crank/Cam Current Sink).....</td><td>478</td></tr><tr><td>Overviews (Crank/Cam Current Sink).....</td><td>479</td></tr><tr><td>Configuring the Function Block (Crank/Cam Current Sink).....</td><td>482</td></tr><tr><td>Hardware Dependencies (Crank/Cam Current Sink).....</td><td>487</td></tr></table>	Introduction (Crank/Cam Current Sink).....	478	Overviews (Crank/Cam Current Sink).....	479	Configuring the Function Block (Crank/Cam Current Sink).....	482	Hardware Dependencies (Crank/Cam Current Sink).....	487
Introduction (Crank/Cam Current Sink).....	478								
Overviews (Crank/Cam Current Sink).....	479								
Configuring the Function Block (Crank/Cam Current Sink).....	482								
Hardware Dependencies (Crank/Cam Current Sink).....	487								

Introduction (Crank/Cam Current Sink)

Introduction to the Function Block (Crank/Cam Current Sink)

Function block purpose	The Crank/Cam Current Sink function block type lets you output current sink signals to simulate digital crankshaft, direction sensitive (reverse) crankshaft, and camshaft sensors.
Default display	The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

**Main features**

These are the main features:

- Importing wavetable files and executing them for simulating crankshaft or camshaft sensor signals.
- Controlling the output signal (phase offset, amplitude, adding noise).
- Switching executed wavetables at runtime.

Supported channel types

The Crank/Cam Current Sink function block supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Analog Out 4	Analog Out 9	Flexible Out 1	-
Hardware	DS2680	DS6101	DS2621	-

Introduction to crankshaft/camshaft signal generation

For basic information, refer to [Basics on Crankshaft and Camshaft Signals and Sensors](#) on page 451.

Oversviews (Crank/Cam Current Sink)

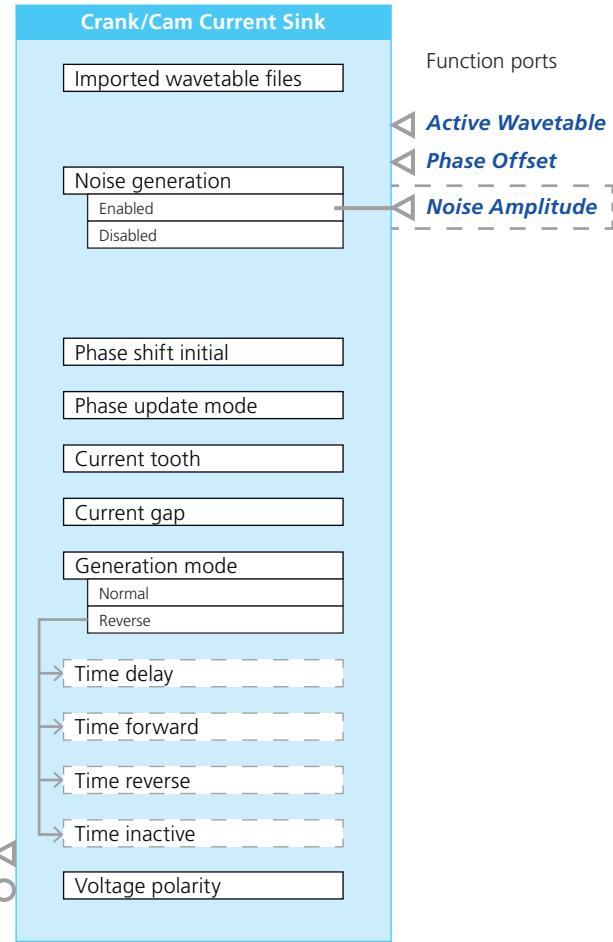
Where to go from here**Information in this section**

- | | |
|--------------------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Crank/Cam Current Sink)..... | 479 |
| Overview of Tunable Properties (Crank/Cam Current Sink)..... | 482 |

Overview of Ports and Basic Properties (Crank/Cam Current Sink)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... n n equals the number of imported wavetable files
Dependencies	-

Phase Offset

This function import lets you define the 0° position independently of the active wavetable from within the behavior model. Usually, a phase offset is applied to the camshaft according to the speed and load (camshaft phase shift).

Note

The phase offset is set relative to the initial phase shift offset defined by the Initial phase shift property.

Value range	<ul style="list-style-type: none"> ▪ -360° ... +360° ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... I_{max} (in Ampere). ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Current Sink) on page 487. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ 0 A: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Noise generation property is set to Enabled. ▪ Supported only for Flexible Out 1 channel type.

Signal

This signal port sinks current. It represents the electrical connection point of the logical signal of the function block.

Note

The hardware is passive and adjusts its internal resistance so that the defined current can flow at the signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Current Sink) on page 487 .
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the **Signal** signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Current Sink) on page 487 .
Dependencies	–

Overview of Tunable Properties (Crank/Cam Current Sink)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Imported Wavetable Files ³⁾	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Current gap	✓	-
Current tooth	✓	-
Phase update mode	✓	-
Phase shift initial	✓	-
Time delay ⁴⁾	✓	-
Time forward ⁴⁾	✓	-
Time reverse ⁴⁾	✓	-
Time inactive ⁴⁾	✓	-
Voltage polarity	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ Not possible by the experiment software but by the **downwave** assistance tool. Refer to [Replacing Wavetable Files at Run Time](#) on page 466.

⁴⁾ Available only if Generation mode is set to Reverse.

Configuring the Function Block (Crank/Cam Current Sink)

Where to go from here

Information in this section

- [Configuring the Basic Functionality \(Crank/Cam Current Sink\).....](#) 483
- [Configuring Standard Features \(Crank/Cam Current Sink\).....](#) 486

Configuring the Basic Functionality (Crank/Cam Current Sink)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Referencing virtual piston engine
- Referencing crankshaft/camshaft wavetables
- Setting a phase shift (typically for the camshaft)
- Specifying generation of reverse crankshaft signals
- Adding noise to the output signal
- Specifying current values for tooth and gap detection
- Specifying the signal polarity

Referencing virtual piston engine

The Crank/Cam Out function block refers to a specific engine design by referencing an Engine Simulation Setup function block. The Engine Simulation Setup function block defines the number of cylinders (1 ... n) and provides further basic configuration data of the virtual engine.

Import wavetable files

Crankshaft and camshaft signals are encoded by numeric data stored in so-called wavetable files. A wavetable is related to a specific virtual sensor, for example, a 60-2 crankshaft wheel and a passive sensor.

The Import Wavetables dialog lets you import, update and remove wavetables files. For instructions, refer to [How to Import Wavetables to a Crank/Cam Function Block](#) on page 463.

Default wavetable ConfigurationDesk adds a default wavetable to the function block only if you do not import a wavetable file. The default wavetable ensures that at least one wavetable is provided for signal generation. All values of the default wavetable are 0.0.

Maximum number of wavetables Wavetables can be referenced by Crank/Cam and Angular Wavetable function blocks. The onboard memory of an I/O board limits the total number of wavetables that can be referenced by all function blocks of both types used in the application. Refer to the following table:

Hardware	Max. Number of Wavetables per Board
DS2680 I/O Unit	24
DS2621 Signal Generation Board	12
DS6101 Multi-I/O Board	12

A wavetable that is used n times by Crank/Cam or Angular Wavetable function blocks is stored n times in the wavetable memory. Potentially added default wavetables are also stored n times in the wavetable memory.

Initial phase shift You can apply an angular offset (φ_{initial}) to the crankshaft and camshaft output signal. The actual phase shift φ_{actual} reads as follows:

$$\varphi_{\text{actual}} = \varphi_{\text{initial}} + \varphi_{\text{Phase offset}}$$

$\varphi_{\text{Phase offset}}$ is defined by the signal at the Phase offset function port, it can change at run time.

Specifying generation of reverse crankshaft signals

You can simulate reverse crankshaft rotation.

To generate reverse crankshaft signals, the Generation mode property has to be set to Reverse. Then the following properties are available to specify the characteristics of the generated reverse crankshaft signal:

- Time delay
- Time forward
- Time reverse
- Time inactive

For more information on the usage and the effects of the listed properties, refer to [Basics on Reverse Crankshaft Signal Generation](#) on page 456.

Limitation Reverse crankshaft signal generation is only supported for the Flexible Out 1 channel type.

Adding noise to the output signal

The Crank/Cam Current Sink function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Limitation Noise generation is supported only for the Flexible Out 1 channel type.

Specifying current values for tooth and gap detection

You can specify the currents that are generated when a crankshaft tooth or a crankshaft gap is detected, or camshaft marker versus no marker.

Current sink characteristics are related to the following configuration properties:

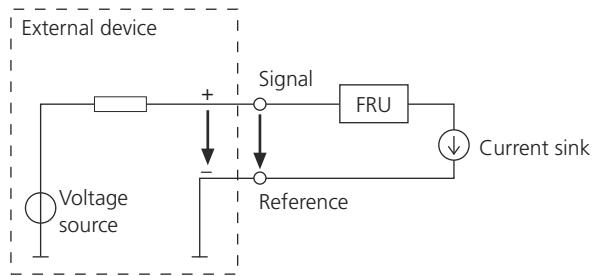
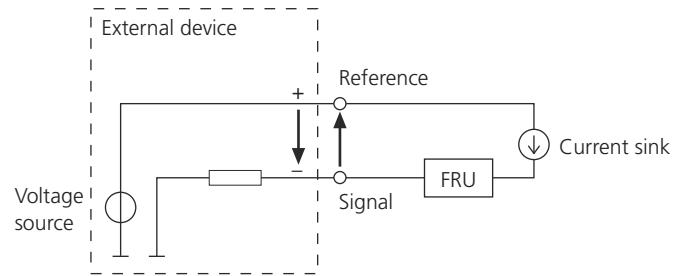
- Current tooth:
Current that is generated when a crankshaft tooth or a camshaft marker is detected (wavetable value > 0.0).
- Current gap:
Current that is generated when no crankshaft tooth or camshaft marker is detected (wavetable value ≤ 0.0).

Specifying the signal polarity

The Crank/Cam Current Sink function block is a current sink and must be fed by an external voltage source. Depending on the selected channel type, the voltage polarity affects the functionality of the function block.

Flexible Out 1 channel type You must specify the voltage polarity of the signal ports with the **Voltage polarity** property to ensure the correct behavior of the function block.

Note that changing the **Voltage polarity** property also changes the signal path that the failure routing unit (FRU) is used on. This means that you can use the FRU on either the one or the other signal path by switching the polarity of the voltage. The following illustration shows a connection example with an FRU on the signal path.

Positive polarity**Negative polarity****Note**

The Crank/Cam Current Sink function block adjusts only the current. The feed voltage must come from an external source as shown in the illustrations above.

The **Voltage polarity** property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

For details on the FRU, refer to [Electrical Error Simulation Concept \(SCALEXIO Hardware Installation and Configuration\)](#).

Analog Out 4, Analog Out 9 channel types The voltage polarity does not affect the functionality of the function block. The behavior is correct if the voltage polarity of the signal is negative as well as positive. Therefore the **Voltage polarity** property is not configurable for these channel types.

Specifying a signal offset	You can apply a voltage offset to the output signal.
Limitation	Applying a voltage offset to the output signal is <i>not</i> supported for the Analog Out 3 and Analog Out 8 channel types.

Configuring Standard Features (Crank/Cam Current Sink)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Analog Out 4	Analog Out 9	Flexible Out 1	Further Information
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 4](#) on page 1555
- [Analog Out 9](#) on page 1557
- [Flexible Out 1](#) on page 1604

Model interface	The interface to the behavior model of the function block provides the following standard features.												
<table border="1"> <thead> <tr> <th>Configuration Feature</th> <th>Further Information</th> </tr> </thead> <tbody> <tr> <td colspan="2">Function Ports</td></tr> <tr> <td>Initialization behavior</td> <td>Specifying Initialization and Stop Behavior on page 88</td></tr> <tr> <td>Stop behavior</td> <td></td></tr> <tr> <td>User saturation</td> <td>Specifying User Saturation on page 90</td></tr> <tr> <td>Test automation support</td> <td>Configuring Test Automation Support on page 92</td></tr> </tbody> </table>		Configuration Feature	Further Information	Function Ports		Initialization behavior	Specifying Initialization and Stop Behavior on page 88	Stop behavior		User saturation	Specifying User Saturation on page 90	Test automation support	Configuring Test Automation Support on page 92
Configuration Feature	Further Information												
Function Ports													
Initialization behavior	Specifying Initialization and Stop Behavior on page 88												
Stop behavior													
User saturation	Specifying User Saturation on page 90												
Test automation support	Configuring Test Automation Support on page 92												

Hardware Dependencies (Crank/Cam Current Sink)

SCALEXIO Hardware Dependencies (Crank/Cam Current Sink)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 4	Flexible Out 1	Analog Out 9
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board
Noise generation	–	✓	–
Noise amplitude range	–	0 ... 10 mA	–
Input current range	+0.1 mA ... +30 mA	+20 µA ... +40 mA	0.1 mA ... 30 mA
Input voltage range	The following conditions must be met: <ul style="list-style-type: none">▪ Range for the lower potential of signal and reference: -2 V ... +18 V▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V	-60 V ... +60 V	The following conditions must be met: <ul style="list-style-type: none">▪ Range for the lower potential of signal and reference: -2 V ... +18 V▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V
Reverse crankshaft signal generation	–	✓	–
Maximum number of wavetable files per board ¹⁾	24	12	12
Angular processing unit	Number of slaves Maximum speed	6 168,000 °/s	1 6
Configurable fuse	–	▪ Load side ▪ 5 mA _{RMS} ... 40 mA _{RMS}	–
Circuit diagrams	Analog Out 4 on page 1555	Flexible Out 1 on page 1604	Analog Out 9 on page 1557
Required channels	1		

¹⁾ Total number of wavetable files that can be stored on the I/O board. For more information, refer to [Configuring the Basic Functionality \(Crank/Cam Current Sink\)](#) on page 483.

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101](#) on page 1539.

General limitations	The Crank/Cam Current Sink function block does not support channel multiplication.
More hardware data	DS2680 I/O Unit For more board-specific data, refer to Data Sheet of the DS2680 I/O Unit (SCALEXIO Hardware Installation and Configuration) .
	DS2621 Signal Generation Board For more board-specific data, refer to Data Sheet of the DS2621 Signal Generation Board (SCALEXIO Hardware Installation and Configuration) .
	DS6101 Multi-I/O Board For more board-specific data, refer to Data Sheet of the DS6101 Multi-I/O Board (SCALEXIO Hardware Installation and Configuration) .

Crank/Cam Digital Out

Where to go from here	Information in this section
	Introduction (Crank/Cam Digital Out)..... 488
	Overviews (Crank/Cam Digital Out)..... 489
	Configuring the Function Block (Crank/Cam Digital Out)..... 493
	Hardware Dependencies (Crank/Cam Digital Out)..... 496

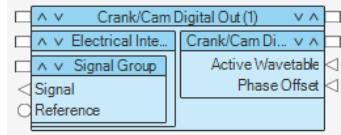
Introduction (Crank/Cam Digital Out)

Introduction to the Function Block (Crank/Cam Digital Out)

Function block purpose	The Crank/Cam Digital Out function block type lets you output digital signals to simulate digital crankshaft, direction-sensitive crankshaft, and camshaft sensors.
-------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

**Main features**

These are the main features:

- Importing wavetable files and executing them for simulating crankshaft or camshaft sensor signals.
- Controlling the phase offset of the output signal.
- Switching executed wavetables at runtime.

Supported channel types

The Crank/Cam Digital Out function block supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Digital Out 1	Digital Out 3	Flexible Out 1	-
Hardware	DS2680	DS6101	DS2621	-

Introduction to crankshaft/camshaft signal generation

For basic information, refer to [Basics on Crankshaft and Camshaft Signals and Sensors](#) on page 451.

Oversviews (Crank/Cam Digital Out)

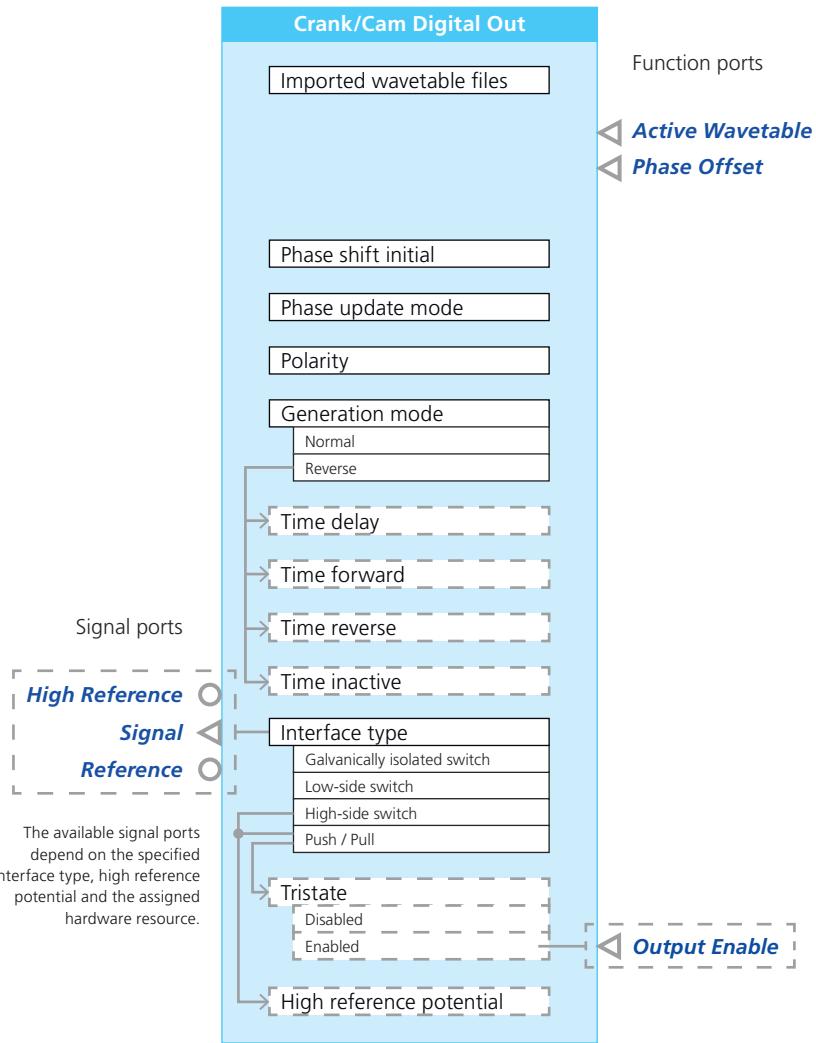
Where to go from here**Information in this section**

Overview of Ports and Basic Properties (Crank/Cam Digital Out)	490
Overview of Tunable Properties (Crank/Cam Digital Out)	492

Overview of Ports and Basic Properties (Crank/Cam Digital Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... n n equals the number of imported wavetable files
Dependencies	-

Phase Offset

This function import lets you define the 0° position independently of the active wavetable from within the behavior model. Usually, a phase offset is applied to the camshaft according to the speed and load (camshaft phase shift).

Note

The phase offset is set relative to the initial phase shift offset defined by the Phase shift initial property.

Value range	<ul style="list-style-type: none"> ▪ -360° ... +360° ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Digital Out) on page 496 .
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Digital Out) on page 496 .
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank/Cam Digital Out) on page 496.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Crank/Cam Digital Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Imported Wavetable Files ³⁾	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Current gap	✓	-
Current tooth	✓	-
Phase update mode	✓	-
Phase shift initial	✓	-
Time delay ⁴⁾	✓	-
Time forward ⁴⁾	✓	-
Time reverse ⁴⁾	✓	-
Time inactive ⁴⁾	✓	-
Polarity	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ Not possible by the experiment software but by the **downwave** assistance tool. Refer to [Replacing Wavetable Files at Run Time](#) on page 466.

⁴⁾ Available only if Generation mode is set to Reverse.

Configuring the Function Block (Crank/Cam Digital Out)

Where to go from here

Information in this section

- | | |
|----------------------------------------------------------------------------------|---------------------|
| Configuring the Basic Functionality (Crank/Cam Digital Out)..... | 493 |
| Configuring Standard Features (Crank/Cam Digital Out)..... | 494 |

Configuring the Basic Functionality (Crank/Cam Digital Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Referencing virtual piston engine
- Referencing crankshaft/camshaft wavetables
- Setting a phase shift (typically for the camshaft)
- Specifying generation of reverse crankshaft signals

Referencing virtual piston engine

The Crank/Cam Out function block refers to a specific engine design by referencing an Engine Simulation Setup function block. The Engine Simulation Setup function block defines the number of cylinders (1 ... n) and provides further basic configuration data of the virtual engine.

Import wavetable files

Crankshaft and camshaft signals are encoded by numeric data stored in so-called wavetable files. A wavetable is related to a specific virtual sensor, for example, a 60-2 crankshaft wheel and a passive sensor.

The Import Wavetables dialog lets you import, update and remove wavetables files. For instructions, refer to [How to Import Wavetables to a Crank/Cam Function Block](#) on page 463.

Default wavetable ConfigurationDesk adds a default wavetable to the function block only if you do not import a wavetable file. The default wavetable ensures that at least one wavetable is provided for signal generation. All values of the default wavetable are 0.0.

Maximum number of wavetables Wavetables can be referenced by Crank/Cam and Angular Wavetable function blocks. The onboard memory of an I/O board limits the total number of wavetables that can be referenced by all function blocks of both types used in the application. Refer to the following table:

Hardware	Max. Number of Wavetables per Board
DS2680 I/O Unit	24
DS2621 Signal Generation Board	12
DS6101 Multi-I/O Board	12

A wavetable that is used n times by Crank/Cam or Angular Wavetable function blocks is stored n times in the wavetable memory. Potentially added default wavetables are also stored n times in the wavetable memory.

Initial phase shift

You can apply an angular offset (φ_{initial}) to the crankshaft and camshaft output signal. The actual phase shift φ_{actual} reads as follows:

$$\varphi_{\text{actual}} = \varphi_{\text{initial}} + \varphi_{\text{Phase offset}}$$

$\varphi_{\text{Phase offset}}$ is defined by the signal at the Phase offset function port, it can change at run time.

Specifying generation of reverse crankshaft signals

You can simulate reverse crankshaft rotation.

To generate reverse crankshaft signals, the Generation mode property has to be set to Reverse. Then the following properties are available to specify the characteristics of the generated reverse crankshaft signal:

- Time delay
- Time forward
- Time reverse
- Time inactive

For more information on the usage and the effects of the listed properties, refer to [Basics on Reverse Crankshaft Signal Generation](#) on page 456.

Adding noise to the output signal

The an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Configuring Standard Features (Crank/Cam Digital Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and

important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 1	Digital Out 3	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	Depending on the channel type, refer to: <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Flexible Out 1) on page 98 ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109
Polarity of output signals	✓	✓	✓	–
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.			–
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 1 on page 1575](#)
- [Digital Out 3 on page 1577](#)
- [Flexible Out 1 on page 1604](#)

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Crank/Cam Digital Out)

SCALEXIO Hardware Dependencies (Crank/Cam Digital Out)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 3				
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board				
Output current range	0 ... +80 mA _{RMS}	0 ... +40 mA _{RMS}	<ul style="list-style-type: none"> ▪ 0 ... +140 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 				
Output voltage range	+5 V ... +60 V	-60 V ... +60 V	+5 V ... +60 V				
Reverse crankshaft signal generation	✓	✓	✓				
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 				
	In push-pull configuration the outputs can be set to high impedance (tristate) at run time.						
Configurable fuse	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–				
Maximum number of wavetable files per board ¹⁾	24	12	12				
Angular processing unit	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Number of slaves</td> <td style="padding: 2px;">6</td> </tr> <tr> <td style="padding: 2px;">Maximum speed</td> <td style="padding: 2px;">168,000 °/s</td> </tr> </table>	Number of slaves	6	Maximum speed	168,000 °/s	1	6
Number of slaves	6						
Maximum speed	168,000 °/s						
Circuit diagrams	Digital Out 1 on page 1575	Flexible Out 1 on page 1604	Digital Out 3 on page 1577				
Required channels	1						

¹⁾ Total number of wavetable files that can be stored on the I/O board. For more information, refer to [Configuring the Basic Functionality \(Crank/Cam Digital Out\)](#) on page 493.

General limitations

The Crank/Cam Digital Out function block does not support channel multiplication.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration](#) .

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration](#) .

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration](#) .

Knock Signal Out

Where to go from here

Information in this section

Introduction (Knock Signal Out).....	498
Overviews (Knock Signal Out).....	502
Configuring the Function Block (Knock Signal Out).....	505
Hardware Dependencies (Knock Signal Out).....	507

Introduction (Knock Signal Out)

Where to go from here

Information in this section

Introduction to the Function Block (Knock Signal Out).....	498
Basics on Knock Signal Generation.....	499

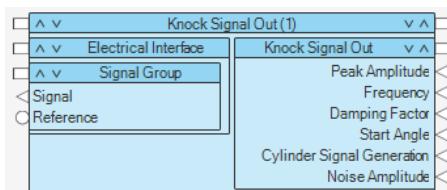
Introduction to the Function Block (Knock Signal Out)

Function block purpose

The Knock Signal Out function block type outputs voltage signals to simulate automotive knock sensor signals.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating up to 12 knock signals as defined from the behavior model.
- Mapping generated knock signals to specific cylinders.
- Controlling the output signal (amplitude, frequency, adding noise and/or offset).

Supported channel types

The Knock Signal Out function block supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Analog Out 3	Analog Out 8	Flexible Out 1	-
Hardware	DS2680	DS6101	DS2621	-

Basics on Knock Signal Generation

Definition of knocking

Knocking is a phenomenon connected with piston engines. It occurs when combustion of the air/fuel mixture in the cylinder starts off correctly in response to ignition by the spark plug, but one or more pockets of air/fuel mixture explode outside the envelope of the normal combustion front. The peak of the combustion process no longer occurs at the optimum moment for the four-stroke cycle. The shock wave creates the characteristic metallic knocking sound, and cylinder pressure increases dramatically. The effects of engine knocking range from inconsequential to completely destructive.

Do not confuse knocking and pre-ignition.

Knock signal curve

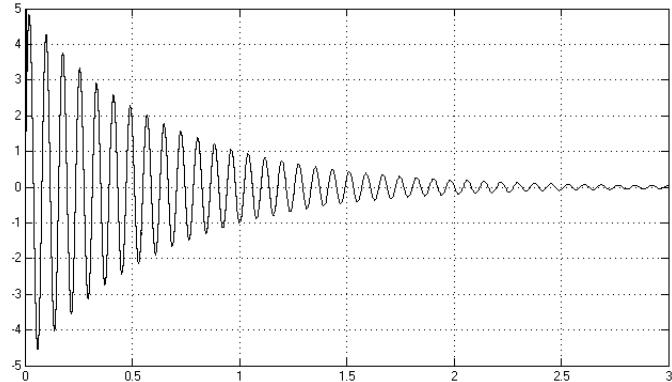
The simulated knock signals over time can be described by the following equation:

$$u(t) = A * e^{-2\pi d f t} * \sin(2\pi f t) + U_{\text{Offset}} + A_{\text{Noise}} * u_{\text{Noise}}(t)$$

where

$u(t)$	Knock signal over time
A	Peak amplitude
d	Damping factor
f	Frequency
U_{Offset}	Signal offset
$u_{\text{Noise}}(t)$	Random noise
A_{Noise}	Noise amplitude

The following illustration shows an example of a knock signal curve.



Note

The beginning of knock signals is not defined by time but by the piston position. Refer to the Start angle function port.

Generation of a knock signal using multiple input signals

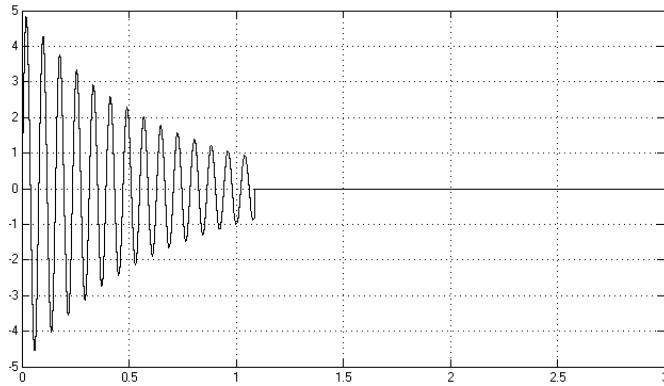
A Knock Signal Out function block lets you define a set of up to 12 knock signals which are provided from the behavior model. All signals are generated on the same channel. For each signal, you can specify a cylinder which the signal is mapped to. The available cylinders are defined in the dialog of the Engine Simulation Setup block.

Preemptive concatenation of knock signals The channel can generate one knock signal at a time. For knock signals that belong to the same function block the following applies: When a signal is going to start, a signal that is being output is stopped. The start angle of the new signal defines the termination point of the current signal.

Composition of curve segments

You can design a knock signal by concatenating curve segments.

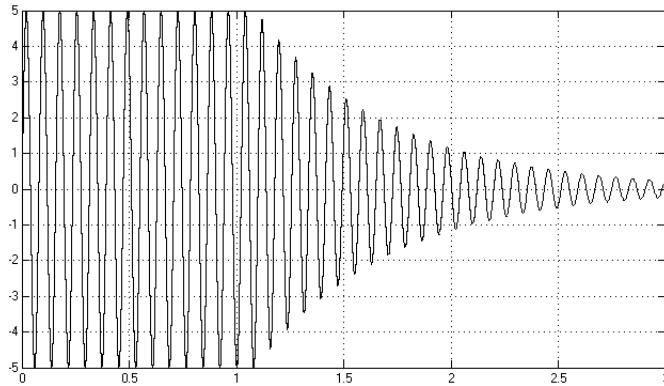
Switching off a knock signal To simulate a knock signal that stops at a specific engine position, you can use two knock signals, both mapped to the same cylinder. The first signal is the actual knock signal. The second knock signal is just used to switch off the first. It must have its amplitude set to 0, and its start angle defines the termination time of the first knock signal.



Delayed damping of a knock signal To simulate a knock signal that is not damped, first, but fades away at a specific engine position, you can use two knock signals, both mapped to the same cylinder. The first signal must have its Damping Factor set to 0, and the second ≠0. The start angle of the second signal defines the time when damping starts.

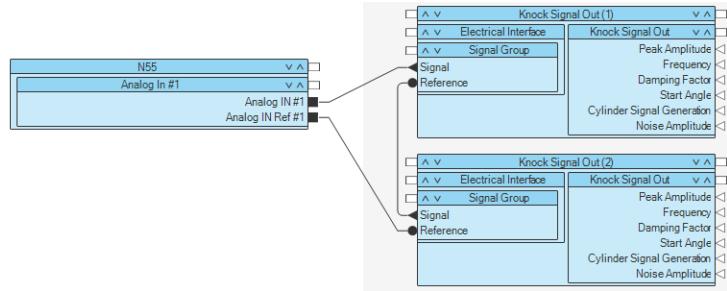
Note

A signal always starts with a function value of 0. Thus, the transition from the first to the second signal might be discontinuous.



Superposition of knock signals

You can superpose multiple knock signals if the channels that generate these signals are connected in series. Each of the superposed signals requires its own Knock Signal Out function block. The signal ports of the blocks must be connected as follows:

**Note**

For channels connected in series, only one channel may generate noise. For all the other channels, Noise Amplitude must be set to 0.

WARNING**High output voltage due to insufficient phase-shifting of signal peaks.**

Risk of serious personal injury and damage of connected devices.

- Ensure that the resulting voltage output never exceeds 60 V.

Overviews (Knock Signal Out)

Where to go from here

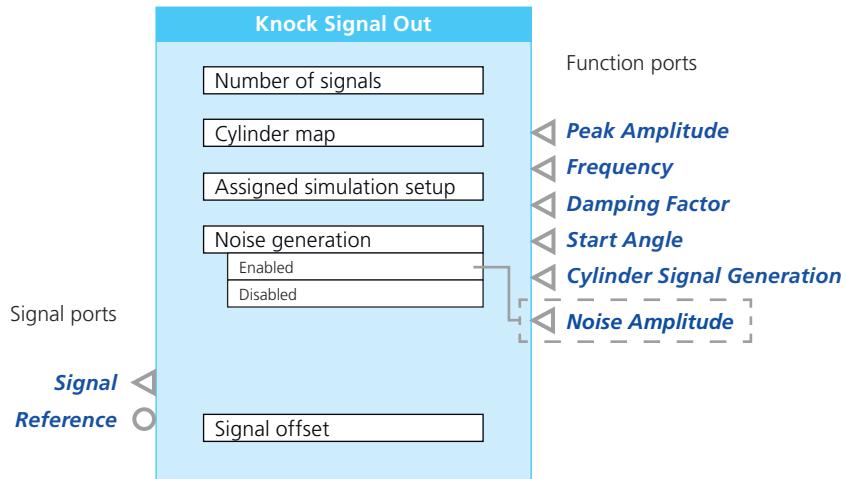
Information in this section

Overview of Ports and Basic Properties (Knock Signal Out).....	502
Overview of Tunable Properties (Knock Signal Out).....	505

Overview of Ports and Basic Properties (Knock Signal Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Peak Amplitude

This function import provides the amplitude of the generated knock signal. It can be provided for each knock signal separately.

Value range	<ul style="list-style-type: none"> ▪ 0 ... U_{\max} (in Volt) ▪ The maximum value depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Knock Signal Out) on page 507. ▪ The vector size depends on the value specified at the Number of signals property.
Dependencies	–

Frequency

This function import provides the frequency (in Hz) for the generated knock signal. It can be provided for each knock signal separately.

Value range	<ul style="list-style-type: none"> ▪ 250 Hz ... 60 kHz ▪ The vector size depends on the value specified at the Number of signals property.
Dependencies	–

Damping Factor

This function import provides the damping of the knock signal. The value is updated at the beginning of a knock sequence and is constant over the entire knock sequence. The value can be provided for each knock signal separately.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 1 ▪ The vector size depends on the value specified at the Number of signals property.
Dependencies	–

Start Angle

This function import provides the start angle [°] of the generated knock signal. The start angle can be in the range [-360° ... +360°] as "Before TDC" of the assigned cylinder. The value can be provided for each knock signal separately.

Note

If multiple signals are defined, the absolute angle values of the signals's start angles must not match.

Value range	<ul style="list-style-type: none"> ▪ -360° ... +360° ▪ The vector size depends on the value specified at the Number of signals property.
Dependencies	–

Cylinder Signal Generation

This function import enables/disables knock signal generation from the behavior model for each knock signal separately.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled The function block does not generate knock signals. ▪ 1: Enabled The function block generates knock signals. ▪ The vector size depends on the value specified at the Number of signals property.
Dependencies	–

Noise Amplitude

This function import provides the amplitude of the noise signal. A noise signal is output independent on whether a knock signal is being output.

Value range	<ul style="list-style-type: none"> ▪ 0 ... U_{\max} (in Volt) ▪ The maximum value depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Knock Signal Out) on page 507.
Dependencies	Available only if the Noise generation property is set to Enabled .

Signal

This signal port provides the generated knock signal.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Knock Signal Out) on page 507.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Knock Signal Out) on page 507.
Dependencies	–

Overview of Tunable Properties (Knock Signal Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Cylinder map	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Signal offset	✓ ³⁾	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ Only supported for Flexible Out 1 channel type

Configuring the Function Block (Knock Signal Out)

Where to go from here**Information in this section**

Configuring the Basic Functionality (Knock Signal Out).....	506
Configuring Standard Features (Knock Signal Out).....	506

Configuring the Basic Functionality (Knock Signal Out)

Overview	The function block type provides the following configuration features that influence the behavior of the basic functionality: <ul style="list-style-type: none">▪ Assigning an Engine Simulation Setup function block▪ Specifying the number of knock signals provided from the behavior model and their mapping to different cylinders▪ Adding noise and/or a signal offset to the output signal
Assigning an Engine Simulation Setup function block	You have to assign an Engine Simulation Setup function block that provides data about the virtual piston engine (number of cylinders, TDCs, etc.) and references a master APU provider.
Specifying number of knock signals	You can specify a set of up to 12 knock signals, which are provided from the behavior model. The signals can be assigned to different cylinders via the Cylinder map property. All signals are generated on the same channel.
Adding noise to the output signal	The Knock Signal Out function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.
Signal offset	You can apply a voltage offset to the generated knock signal. Limitation Applying a voltage offset to the knock signal is not supported for the Analog Out 3 channel type.

Configuring Standard Features (Knock Signal Out)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer.

You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Analog Out 3	Analog Out 8	Flexible Out 1	Further Information
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Flexible Out 1 on page 1604](#)
- [Analog Out 3 on page 1554](#)
- [Analog Out 8 on page 1557](#)

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Knock Signal Out)

SCALEXIO Hardware Dependencies (Knock Signal Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 3	Flexible Out 1	Analog Out 8
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board
Output current range	0 ... +5 mA _{RMS}	0 ... +40 mA _{RMS}	0 ... +5 mA _{RMS}

Channel Type	Analog Out 3	Flexible Out 1	Analog Out 8
Output voltage range	-20 V ... +20 V	-20 V ... +20 V	-20 V ... +20 V
Noise generation	✓	✓	✓
Noise amplitude range	0 ... 5 V	0 ... 5 V	0 ... 5 V
Resolution	14 bit	16 bit	14 bit
DC signal offset	—	✓	—
Configurable fuse	—	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	—
Angular processing unit	Number of slaves	6	1
	Maximum speed	168,000 °/s	
Circuit diagrams	Analog Out 3 on page 1554	Flexible Out 1 on page 1604	Analog Out 8 on page 1557
Required channels	1	1	1

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101 on page 1539](#).

General limitations

The Knock Signal Out function block does not support channel multiplication.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Lambda Probe Simulation

Purpose	With the Lambda NCCR and Lambda DCR function block types, you can simulate wide-band lambda probes. Using basic function block types, you can simulate two-step lambda probes.
----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Where to go from here	Information in this section
	Introduction to Lambda Probe Simulation 509 Lambda DCR 523 Lambda NCCR 534

Introduction to Lambda Probe Simulation

Where to go from here	Information in this section
	Definition of Lambda 509 SCALEXIO-Compatible Lambda Probe Types 510 Basics on the Simulation of Two-Step Lambda Probes 511 Basics on Simulation of Wide-Band Lambda Probes (DCR) 515 Basics on Simulation of Wide-Band Lambda Probes (NCCR) 518

Definition of Lambda

Oxygen content in exhaust gases	A lambda probe measures the remaining oxygen content in the exhaust gas of a combustion engine so that the engine ECU can determine the amount of fuel required to ensure optimal exhaust gas values. The remaining oxygen content is expressed by the variable λ (Greek: lambda):
----------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

$$\lambda = \frac{\text{Measured air-to-fuel ratio}}{\text{Stoichiometric air-to-fuel ratio}} = \frac{\text{Measured air-to-fuel ratio}}{14.7 : 1}$$

Combustion is best (causing minimum pollution) if the air/fuel ratio (AFR) is 14.7 : 1, thus $\lambda = 1$ (stoichiometric AFR).

Operation modes of piston engines

Modern automotive piston engines such as gasoline direct injection engines have three operation modes:

Operation Mode	Lambda	Engine Characteristic
Rich	$\lambda < 1.0$	Increases engine power
Stoichiometric	$\lambda = 1.0$	Minimum pollution
Lean	$\lambda > 1.0$	Decreases fuel consumption

Fixing of lambda probes

Typically, λ is measured before the exhaust gas reaches the catalyst and after it.

- Before the catalyst:
To monitor the combustion and optimize engine control a wide-band lambda probe measures how rich or lean the air/fuel mixture is.
- After the catalyst:
To monitor the efficiency of the catalyst a two-step lambda probe checks whether the air/fuel mixture is rich or lean.

SCALEXIO-Compatible Lambda Probe Types

Purpose

You can simulate various lambda probes with the SCALEXIO system.

Two-step lambda probes

You can simulate two-step lambda probes. No specific lambda function block types exists. You can use standard function block types.

For further details, refer to [Basics on the Simulation of Two-Step Lambda Probes](#) on page 511.

Wide-band lambda probes

You can simulate the following wide-band lambda probe types.

Manufacturer	Probe Type	Required Function Block
Bosch	LSU 4.2	Lambda NCCR
	LSU 4.9	
	LSU ADV	
NTK	L1H1	
	L1H2	
	LHA LZA-08-H6	
	6312-W1	
	ZFAS-U1	
	ZFAS-U2	
	ZFAS-U3	
Denso	PLUS2.1	Lambda DCR
	PLUS3.x	

If you want to simulate other wide-band lambda probe types, contact dSPACE. For Bosch LSU 5.1 probes a custom function block is available.

For further details, refer to [Basics on Simulation of Wide-Band Lambda Probes \(NCCR\)](#) on page 518 or [Basics on Simulation of Wide-Band Lambda Probes \(DCR\)](#) on page 515.

Basics on the Simulation of Two-Step Lambda Probes

Usage

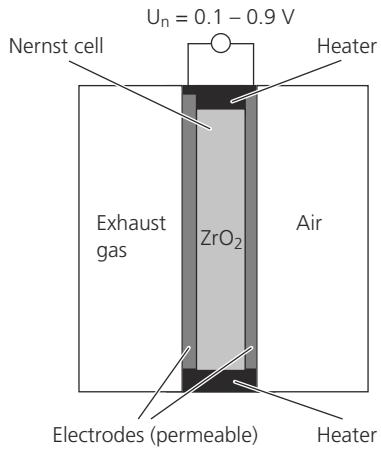
Two-step (or narrow-band or binary) lambda probes check whether the air/fuel mixture is rich ($\lambda < 1$) or lean ($\lambda > 1$).

Tip

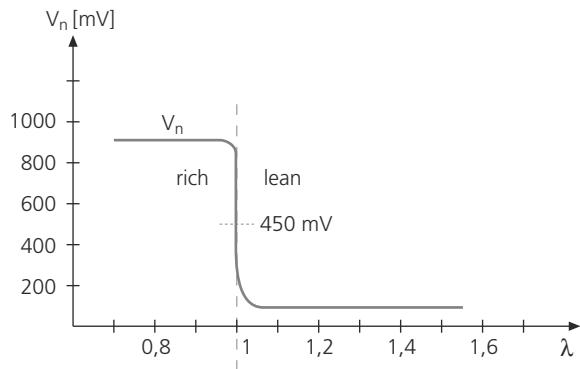
Simulation of two-step lambda probes is possible by using standard function blocks such as the Voltage Out and the Resistance Out function blocks.

Characteristics of two-step lambda probes

Sensor design Two-step lambda probes are based on the *Nernst cell*. This cell consists of a zirconia solid electrolyte (ZrO_2) which has two electrodes on each side. To reach the operating temperature quickly, it has an integrated heater.



Operating principle Exhaust gas passes through the electrode on one side of the cell, while air passes through the electrode on the other side. The Nernst cell generates a specific output voltage V_n according to the λ of the exhaust gas. When the air/fuel ratio is perfectly balanced ($\lambda = 1$), $V_n = 450$ mV is generated. When the fuel mixture goes rich ($\lambda < 1$), even just a little, V_n shoots up to its maximum output of about 900 mV. Conversely, when the fuel mixture goes lean ($\lambda > 1$), V_n drops to 100 mV. Resistance R_n of the Nernst cell is used to measure the temperature of the sensor. R_n decreases with higher sensor temperatures, and it increases with lower sensor temperatures. Typically, the heater is controlled by a PWM signal.



Every time V_n jumps or drops, the ECU responds by decreasing or increasing the amount of fuel that is delivered. This rapid flip-flopping allows the feedback fuel control system to maintain a more-or-less balanced average air-fuel mixture. However, two-step sensors only give a rich-lean indication, they cannot measure the exact air/fuel ratio. This proven approach is not accurate enough to meet the latest emissions requirements.

Tip

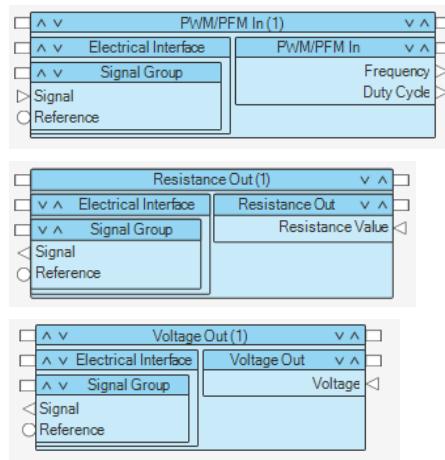
Two-step lambda probes are useful for tuning an engine for steady state cruising.

SCALEXIO simulation approach

The SCALEXIO simulator mimics the behavior of the lambda probe according to the following simulation approach:

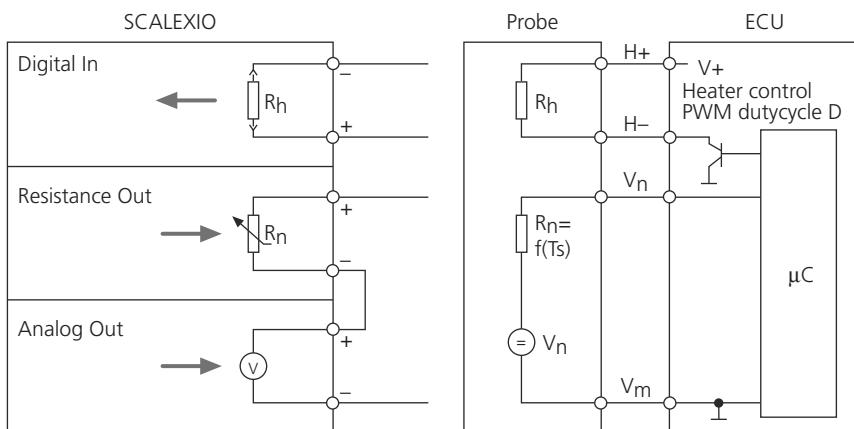
1. The behavior model (Simulink) simulates the engine and defines/changes the following properties as a function of time:
 - λ of the exhaust gas
 - Resistance R_n of the Nernst cell which correlates with the cell/sensor temperature T_s
 It then calculates V_n as a function of λ (refer to the $V_n-\lambda$ diagram in the previous illustration).
2. ConfigurationDesk outputs U_n and R_n to the ECU.
3. The process is repeated from step 1.

Required function blocks You can use the following standard function blocks for simulating two-step lambda probes:



SCALEXIO signal processing

The following illustration shows suitable channel types of the SCALEXIO hardware and their relation to the signals (pins) of a two-step lambda probe.



Signals to be connected The following signals must be connected from the ECU to the SCALEXIO hardware:

Signal	Description
H+	Battery supply (+ voltage)
H-	Low-side switch to battery supply (– voltage)
V _n	Nernst voltage and cell resistance measurement input
V _m	Signal ground of the ECU

Using loads To simulate lambda probes it is essential to use the following Lambda-specific loads.

Load	Description	Typical Value ¹⁾
R _h	Resistance of the heater	<ul style="list-style-type: none"> ▪ 3.3 Ω ▪ The load can be realized as internal or external loads. For instructions, refer to How to Connect Internal Loads (DS2601) (SCALEXIO Hardware Installation and Configuration)
R _n	Resistance of the Nernst cell	<ul style="list-style-type: none"> ▪ 47 Ω ... 940 Ω ▪ R_n is to be calculated in the behavior model and provided via the Resistance Out function block. Refer to Resistance Out (ConfigurationDesk Function Block Properties).

¹⁾ For more details, refer also to [Hardware Dependencies \(Lambda NCCR\)](#) on page 547

Suitable SCALEXIO hardware The following channel types are suitable and can be used depending on the SCALEXIO hardware:

Suitable Channel Types						Purpose	Function Block
DS2601	DS2621	DS2680	DS2690	DS6101	DS6201		
Flexible In 1	–	<ul style="list-style-type: none"> ▪ Digital In 1 ▪ Flexible In 2 	<ul style="list-style-type: none"> ▪ Digital In 2 ▪ Digital In/Out 1 	<ul style="list-style-type: none"> ▪ Digital In 3 ▪ Flexible In 3 	Digital In/Out 3	To measure the PWM heater signal of the ECU	PWM/PFM In
–	Flexible Out 1	Resistance Out 1	–	Resistance Out 2	–	To simulate the resistance of the sensor (sensor temperature)	Resistance Out
–	Flexible Out 1	<ul style="list-style-type: none"> ▪ Analog Out 1 ▪ Analog Out 4 	–	<ul style="list-style-type: none"> ▪ Analog Out 6 ▪ Analog Out 9 	–	To output the Nernst voltage U _n	Voltage Out

Basics on Simulation of Wide-Band Lambda Probes (DCR)

Usage

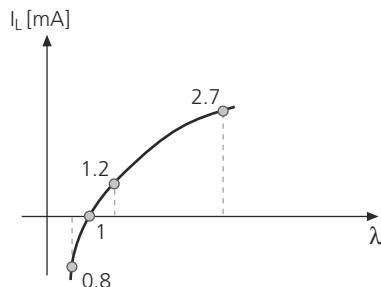
Wide-band lambda probes measure how rich or lean the air/fuel mixture is in the approximate range $\lambda = 0.7 \dots 1.7$. They eliminate the lean-rich cycling inherent in two-step sensors, allowing the control unit to adjust the fuel delivery and ignition timing of the engine much more rapidly.

Tip

Wide-band lambda probes are useful for tuning an engine for quick changes of loads and/or revolutions.

Characteristics of DCR lambda sensors

Operating principle A specific external voltage, i.e., the cell voltage U_0 , must be applied to the sensor by the ECU. The sensor outputs the cell current I_L as a function of λ . I_L is fed back to the ECU. The following illustration shows the $I_L-\lambda$ curve, which depends on the lambda probe type. Resistance R_n of the Nernst cell is used to measure the temperature T_s of the sensor. R_n decreases with higher sensor temperatures, and it increases with lower sensor temperatures. Typically, the heater is controlled by a PWM signal.



SCALEXIO simulation approach

The SCALEXIO simulator mimics the behavior of the lambda probe according to the following simulation approach:

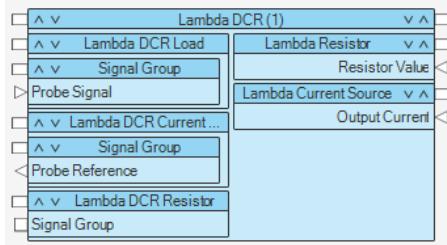
1. The behavior model simulates the engine and defines/changes the following properties as a function of time:

- λ of the exhaust gas
- Resistance R_n of the Nernst cell which correlates with the cell/sensor temperature T_s

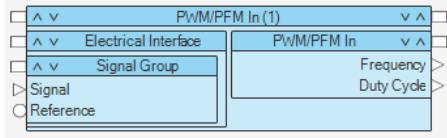
It then calculates the cell current I_L as a function of λ (refer to the $I_L-\lambda$ curve in the previous illustration).

2. ConfigurationDesk outputs I_L to the ECU.
3. The process is repeated from step 1.

Required function blocks You can use the Lambda DCR function block for simulating wide-band lambda probes (DCR):

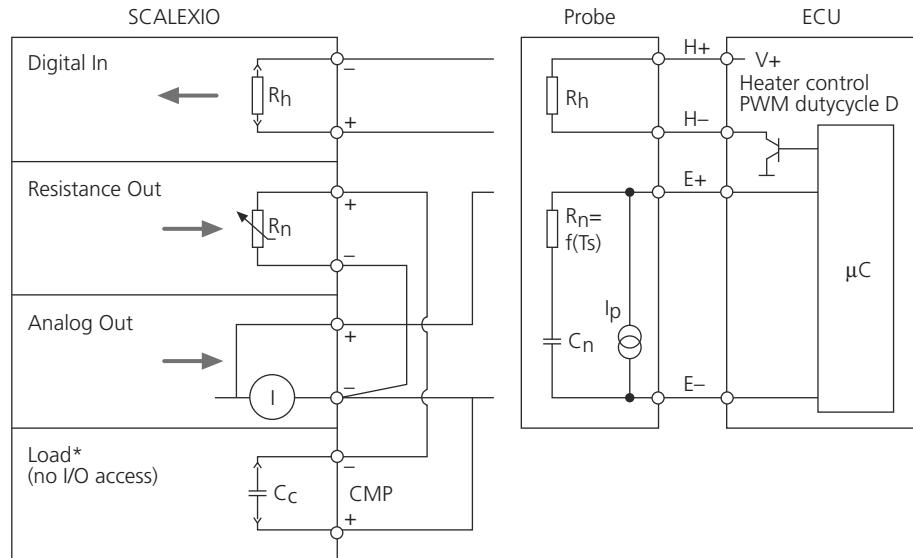


An additional function block is needed for processing the ECU's heater control signal, for example, the PWM/PFM In function block:



SCALEXIO signal processing

The following illustration shows suitable channel types of the SCALEXIO hardware and their relation to the signals (pins) of a wide-band lambda probe DCR.



* Using the DS6101, this channel type is not available. The capacitor must be connected externally.

The heater control could also be realized as a high-side switch.

Signals to be connected The following signals must be connected from the ECU to the SCALEXIO hardware:

Pin	Description
H+	Battery supply (+ voltage)
H-	Low-side switch to battery supply (– voltage)
E+	Measurement input for I_L
E-	Control output for U_0

Using loads To simulate lambda probes it is essential to use the following Lambda-specific loads.

Property	Description	Typical Value ¹⁾
R_h	Resistance of the heater	<ul style="list-style-type: none"> ▪ 3.3 Ω ▪ Using the DS2680, the DS2690 or the DS2601: The load can be realized as internal or external loads. For instructions, refer to SCALEXIO Hardware Installation and Configuration.
R_n	Resistance of the Nernst cell	<ul style="list-style-type: none"> ▪ 47 Ω ... 940 Ω ▪ R_n is to be calculated in the behavior model and provided via the Resistance Out function block. Refer to Resistance Out (ConfigurationDesk Function Block Properties).
C_c	Capacitance to galvanically decouple R_n	<ul style="list-style-type: none"> ▪ $\geq 47 \mu F$ ▪ Using the DS2680: Commonly, C_c is realized as internal load. For instructions, refer to How to Connect Internal Loads (DS2680-IL) (SCALEXIO Hardware Installation and Configuration). C_c might be overloaded during failure simulation. For instructions, refer to Configuring Standard Features (Lambda DCR) on page 529. ▪ Using the DS6101: C_c must be realized as external load.

¹⁾ For more details, refer also to [Hardware Dependencies \(Lambda DCR\)](#) on page 533

Suitable SCALEXIO hardware The following channel types are suitable and can be used depending on the SCALEXIO hardware:

Suitable Channel Types					Purpose	Function Block
DS2680	DS2690	DS6101	DS6201	DS6201		
<ul style="list-style-type: none"> ▪ Digital In 1 ▪ Flexible In 2 	<ul style="list-style-type: none"> ▪ Digital In 2 ▪ Digital In/Out 1 	<ul style="list-style-type: none"> ▪ Digital In 3 ▪ Flexible In 3 	Digital In/Out 3	Flexible In 1	To measure the PWM heater signal of the ECU	PWM/PFM In

Suitable Channel Types					Purpose	Function Block
DS2680	DS2690	DS6101	DS6201	DS2601		
Resistance Out 1	–	Resistance Out 2	–	–	To simulate the resistance of the sensor (sensor temperature)	Lambda DCR
Analog Out 2	–	Analog Out 7	–	–	To output the pump current I_p	
Load 1	–	–	–	–	Component channel for C_c	

For details on the I/O circuit using the Lambda DCR function block, refer to [Mapping Signal Ports \(Lambda DCR\)](#) on page 530.

Limitation For the DS2680 I/O Unit, only two wide-band lambda probes (NCCR/DCR) can be simulated at the same time.

Basics on Simulation of Wide-Band Lambda Probes (NCCR)

Usage

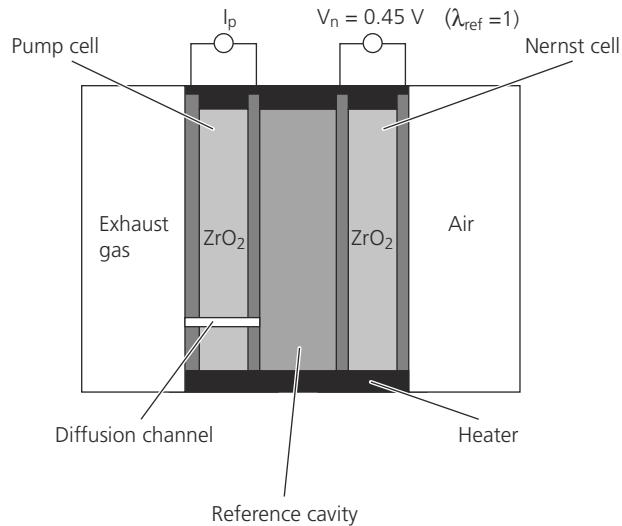
Wide-band lambda sensors measure how rich or lean the air/fuel mixture is in the approximate range $\lambda = 0.7 \dots 1.7$. They eliminate the lean-rich cycling inherent in two-step sensors, allowing the control unit to adjust the fuel delivery and ignition timing of the engine much more rapidly.

Tip

Wide-band lambda probes are useful for tuning an engine for quick changes of loads and/or revolutions.

Characteristics of NCCR lambda probes

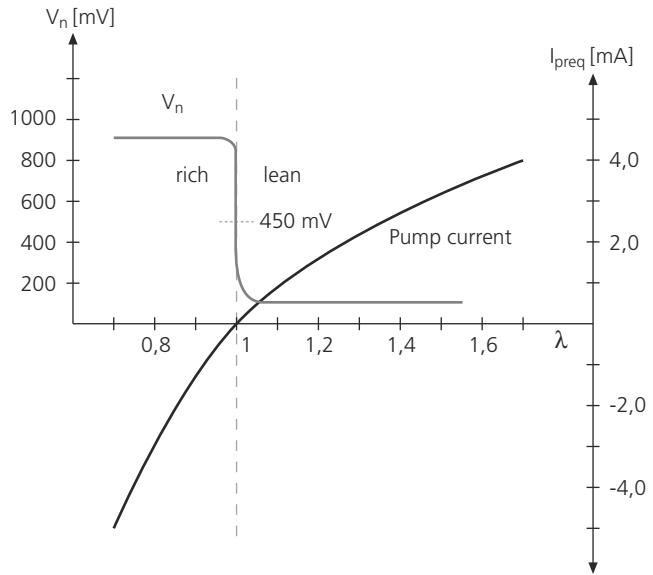
Sensor design This sensor type consists of a *Nernst cell* and a so-called *pump cell*. There is a reference cavity between the two cells. The exhaust gas passes a diffusion channel and can flow into the reference cavity. The two electrodes of the pump cell are connected to a current supply (ECU) so that a bipolar pump current I_p can be applied to the pump cell. The Nernst voltage U_n is fed to the ECU. To reach the operating temperature quickly, it has an integrated heater.



Operating principle A change in the oxygen content of the exhaust gas (λ) influences the oxygen concentration in the reference cavity (λ_{ref}):

Exhaust Gas	Reference Cavity	U_n
λ is constant	$\lambda_{\text{ref}} = 1$	450 mV
λ increases	$\lambda_{\text{ref}} = 1$ changes to $\lambda_{\text{ref}} > 1$ Oxygen ions leave the exhaust gas and flow into the reference cavity.	Decreases to 100 mV
λ decreases	$\lambda_{\text{ref}} = 1$ changes to $\lambda_{\text{ref}} < 1$ Oxygen ions leave the reference cavity and flow into the exhaust gas.	Increases to 900 mV

The pump current I_p calculated and generated by the ECU forces a contrary movement of oxygen ions through the ZrO₂ layer. This finally results in a steady state with $\lambda_{\text{ref}} = 1$ and $V_n = 450 \text{ mV}$. λ_{ref} does not change until λ of the exhaust gas changes again. The following illustration shows the relationship of λ (exhaust gas) and required pump current $I_{p, \text{req}}$. The $I_{p, \text{req}}-\lambda$ curve depends on the lambda probe type.



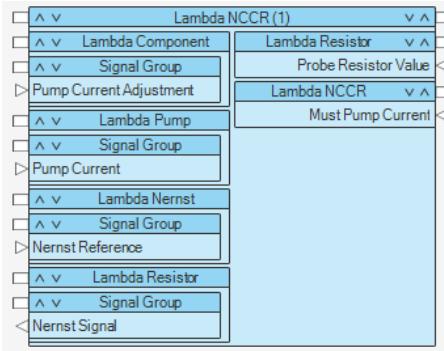
Resistance R_n of the Nernst cell is used to measure the temperature T_s of the sensor. R_n decreases with higher sensor temperatures, and it increases with lower sensor temperatures. Typically, the heater is controlled by a PWM signal.

SCALEXIO simulation approach

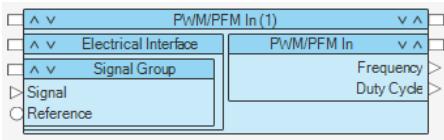
The SCALEXIO simulator mimics the behavior of the lambda probe according to the following simulation approach:

1. The behavior model simulates the engine and defines/changes the following properties as a function of time:
 - λ of the exhaust gas
 - Resistance R_n of the Nernst cell which correlates with the cell/sensor temperature T_s
 It then calculates the required pump current $I_{p,\text{req}}$ as a function of λ (refer to the $I_p-\lambda$ curve in the previous illustration).
2. The closed loop control for pump current I_p (for details, refer to [Configuring the Basic Functionality \(Lambda NCCR\)](#) on page 539):
 - ConfigurationDesk calculates a preliminary Nernst voltage V_n as a function of I_p and outputs it to the ECU.
 - The ECU calculates a preliminary pump current I_p as a function of V_n and outputs it to ConfigurationDesk.
 Finally the ECU provides $I_p = I_{p,\text{req}}$, hence, V_n is balanced at 450 mV.
3. The process is repeated from step 1.

Required function blocks You can use the Lambda NCCR function block for simulating wide-band lambda probes (NCCR):

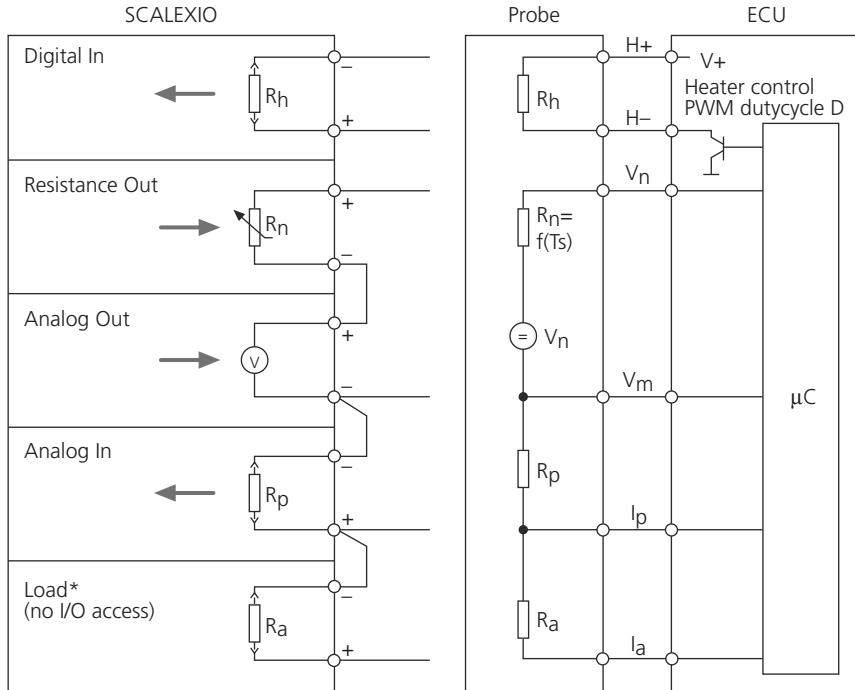


An additional function block is needed for processing the ECU's heater control signal, for example, the PWM/PFM In function block:



SCALEXIO signal processing

The following illustration shows suitable channel types of the SCALEXIO hardware and their relation to the signals (pins) of a wide-band lambda probe NCCR.



* Using the DS6101, this channel type is not available. R_a must be connected externally.

Signals to be connected The following signals must be connected from the ECU to the SCALEXIO hardware:

Pin	Description
H+	Battery supply (+ voltage)
H-	Low-side switch to battery supply (– voltage)
V _n	Nernst voltage and cell resistance measurement output
V _m	Constant voltage output as reference level
I _p	Control input for I _p
I _a	Gain adjustment of the I _p control circuit

Using loads To simulate lambda probes it is essential to use the following Lambda-specific loads.

Property	Description	Typical Value ¹⁾
R _h	Resistance of the heater	<ul style="list-style-type: none"> ▪ 3.3 Ω ▪ Using the DS2680, the DS2690 or the DS2601: The load can be realized as internal or external loads.
R _n	Resistance of the Nernst cell	<ul style="list-style-type: none"> ▪ 47 Ω ... 940 Ω ▪ R_n is to be calculated in the behavior model and provided via the Resistance Out function block. Refer to Resistance Out (ConfigurationDesk Function Block Properties).
R _p	Resistance of the pump cell	<ul style="list-style-type: none"> ▪ 200 Ω ▪ Using the DS2680: Commonly, R_p is realized as internal load. For instructions, refer to How to Connect Internal Loads (DS2680-IL) (SCALEXIO Hardware Installation and Configuration). R_p might be overloaded during failure simulation. For instructions, refer to Configuring Standard Features (Lambda NCCR) on page 543. ▪ Using the DS6101: R_p must be realized as external load.
R _a	Resistance for the I _p adjustment	<ul style="list-style-type: none"> ▪ 100 Ω ▪ Using the DS2680: Commonly, R_a is realized as internal load. For instructions, refer to How to Connect Internal Loads (DS2680-IL) (SCALEXIO Hardware Installation and Configuration). R_a might be overloaded during failure simulation. For instructions, refer to Configuring Standard Features (Lambda NCCR) on page 543. ▪ Using the DS6101: R_a must be realized as external load.

¹⁾ For more details, refer also to [Hardware Dependencies \(Lambda NCCR\)](#) on page 547

Suitable SCALEXIO hardware The following channel types are suitable and can be used depending on the SCALEXIO hardware:

Supported Channel Types					Purpose	Function Block
DS2680	DS2690	DS6101	DS6201	DS2601		
▪ Digital In 1 ▪ Flexible In 2	▪ Digital In 2 ▪ Digital In/Out 1	▪ Digital In 3 ▪ Flexible In 3	Digital In/Out 3	Flexible In 1	To measure the PWM heater signal of the ECU	PWM/PFM In
Resistance Out 1	–	Resistance Out 2	–	–	To simulate the resistance of the sensor (sensor temperature)	Lambda NCCR
Analog Out 2	–	Analog Out 7	–	–	To output the Nernst voltage U_N	
Analog In 2	–	Analog In 5	–	–	To measure the pump current I_P	
Load 1	–	–	–	–	Component channel for R_a	

Note

The Nernst voltage (Analog Out x) and the lambda pump current (Analog In x) must use the same I/O channel.

For details on the I/O circuit using the Lambda NCCR function block, refer to [Mapping Signal Ports \(Lambda NCCR\)](#) on page 544.

Limitation For the DS2680 I/O Unit, only two wide-band lambda probes (NCCR/DCR) can be simulated at the same time.

Lambda DCR

Where to go from here

Information in this section

Introduction (Lambda DCR)	524
Overviews (Lambda DCR)	525
Configuring the Function Block (Lambda DCR)	527
Hardware Dependencies (Lambda DCR)	533

Introduction (Lambda DCR)

Introduction to the Function Block (Lambda DCR)

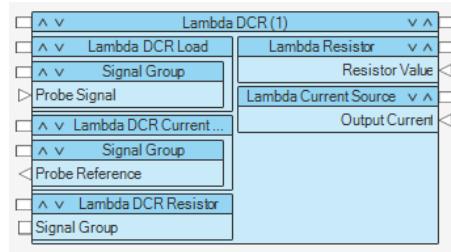
Function block purpose

With the Lambda DCR function block, you can simulate wide-band lambda probes. The functionality of the function block depends on the direct current regulation (DCR) method.

For an overview of the supported lambda probe types, refer to [SCALEXIO-Compatible Lambda Probe Types](#) on page 510.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Providing the Nernst-controlled pump current to the ECU as defined from the behavior model.
- Providing the resistance of the Nernst cell as defined from the behavior model.
- Providing the reference potential for the Nernst voltage.

Required channel types

The Lambda DCR function block requires the following hardware:

	SCALEXIO	MicroAutoBox III
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board
Channel types	Requires three channels based on the following	Requires two channels based on the following different channel types: <ul style="list-style-type: none"> ▪ Resistance Out 2 ▪ Analog Out 7

	SCALEXIO	MicroAutoBox III
	different channel types: <ul style="list-style-type: none">▪ Resistance Out 1▪ Analog Out 2▪ Load 1	

Oversviews (Lambda DCR)

Where to go from here

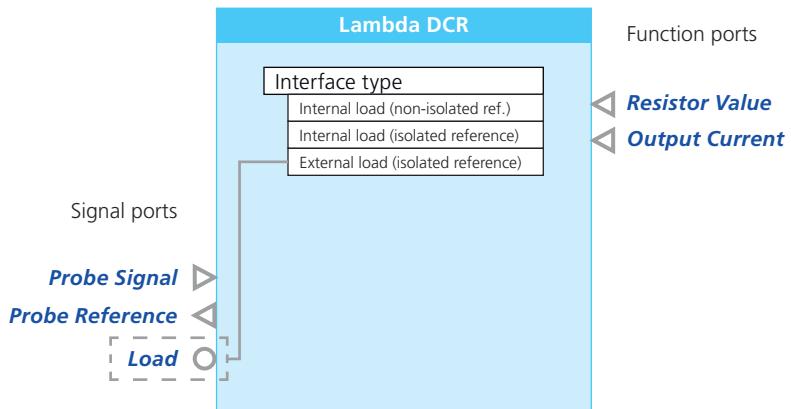
Information in this section

- Overview of Ports and Basic Properties (Lambda DCR)..... 525
 Overview of Tunable Properties (Lambda DCR)..... 526

Overview of Ports and Basic Properties (Lambda DCR)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Resistor Value

This function import provides the sensor resistance as a function of the simulated sensor temperature.

Value range	17 Ω ... 1 MΩ
Dependencies	-

Output Current

This function import provides the required cell current (in Ampere) as a function of the simulated λ .

Value range	-5 mA ... +5 mA
Dependencies	–

Probe Signal

This signal import reads the cell voltage U_0 generated by the ECU.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda DCR) on page 533.
Dependencies	–

Probe Reference

This signal output provides the cell current I_L to the ECU.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda DCR) on page 533.
Dependencies	–

Load

This signal port represents the electrical connection point to connect external loads.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda DCR) on page 533.
Dependencies	Available only if the Interface type property is set to External load (isolated reference).

Overview of Tunable Properties (Lambda DCR)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Lambda DCR)

Where to go from here

Information in this section

Configuring the Basic Functionality (Lambda DCR).....	527
Configuring Standard Features (Lambda DCR).....	529
Mapping Signal Ports (Lambda DCR).....	530

Configuring the Basic Functionality (Lambda DCR)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Selecting the electrical interface type

Selecting the electrical interface type

To adapt the electrical interface of the function block to the dSPACE real-time hardware that you want to assign to the function block, you have to select an electrical interface type.

Note

The availability of electrical interface units and their properties depends on the setting of this property. Therefore, you are strongly recommended to set this property first.

Select the setting according to the hardware that you want to assign:

- If you want to assign a channel set from the DS2680 I/O Unit, select the Internal load (non-isolated reference) or Internal load (isolated reference) setting as follows:
 - You are recommended to use the Internal load (isolated reference) setting. Modern ECUs require the sensor simulation to be similar to an ideal two-terminal circuit that generates identical current flows on both signal terminals (signal and reference).

The hardware behaves like an ideal two-terminal network with the same current flow on the signal and reference pins. The current flows into the circuit on one pin and out of the circuit on the other.

Note

Not every DS2680 I/O Unit provides a galvanically isolated reference. You can identify a DS2680 that supports this feature as follows: In the Platform Manager, navigate the hardware tree of the SCALEXIO system to the Analog Out 2 item (via DS2680 I/O Unit - DS2680 I/O Module). If the Galvanic isolation property has the Galvanically isolated or the Switchable setting, the DS2680 supports this feature.

- If you select Internal load (non-isolated reference), the current flows into the circuit via the reference pin and uses the internal GND of the dSPACE hardware as the return path.
- If your ConfigurationDesk application (containing a Lambda DCR function block) is created with ConfigurationDesk 5.1 or earlier, use the "non-isolated reference" setting to make the application compatible with the calculated external cable harness of the migrated previous application. If you use the "isolated reference" setting for the function block in the migrated application, you have to recalculate and build a new external cable harness.
- If you want to assign a channel set from the DS6101 Multi-I/O Board, select the External load (isolated reference) setting.

The DS6101 provides the connection of external loads and galvanic isolation of the reference signal. Therefore, the channels behave like an ideal two-terminal network with the same current flow on the signal and reference pins. The current flows into the circuit on one pin and out of the circuit on the other.

For circuit diagrams, refer to [Mapping Signal Ports \(Lambda DCR\)](#) on page 530.

Configuring Standard Features (Lambda DCR)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The electrical interface of the function block provides the following standard features.

Configuration Feature	Further Information
Signal Ports	
Failure simulation support ¹⁾	Specifying Settings for Failure Simulation on page 123
Usage of loads ¹⁾	Specifying Load Settings on page 121

¹⁾ Configurability depends on the selected electrical interface configuration and the electrical interface unit that the signal port belongs to.

Valid for the DS2680:

NOTICE

Capacitance C_c (to galvanically decouple R_n) might be overloaded during failure simulation.

Risk of material damage.

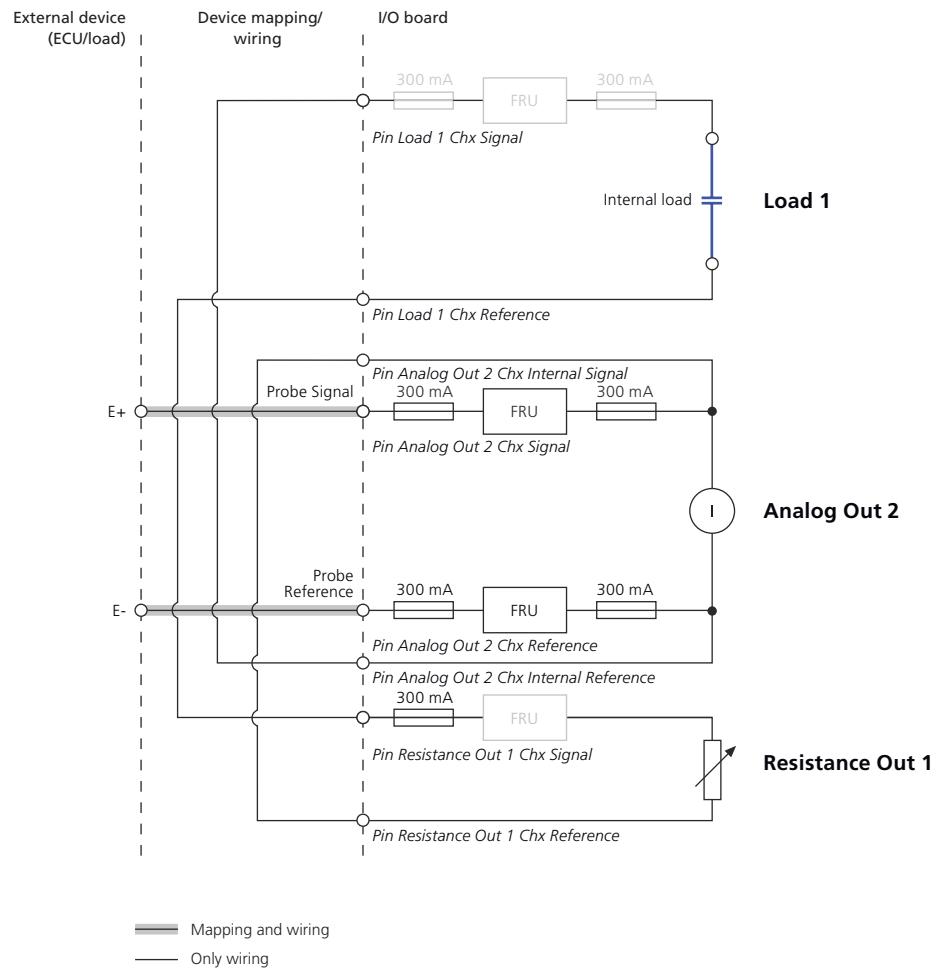
- Enforce load rejection for the Probe Signal signal port.
- For details on enforcing load rejection, refer to [Basics on Load Rejection \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(13ada334b1d3a4a8807415a42c9af955_img.jpg\)](#).

Model interface The interface to the behavior model of the function block provides the following standard features.

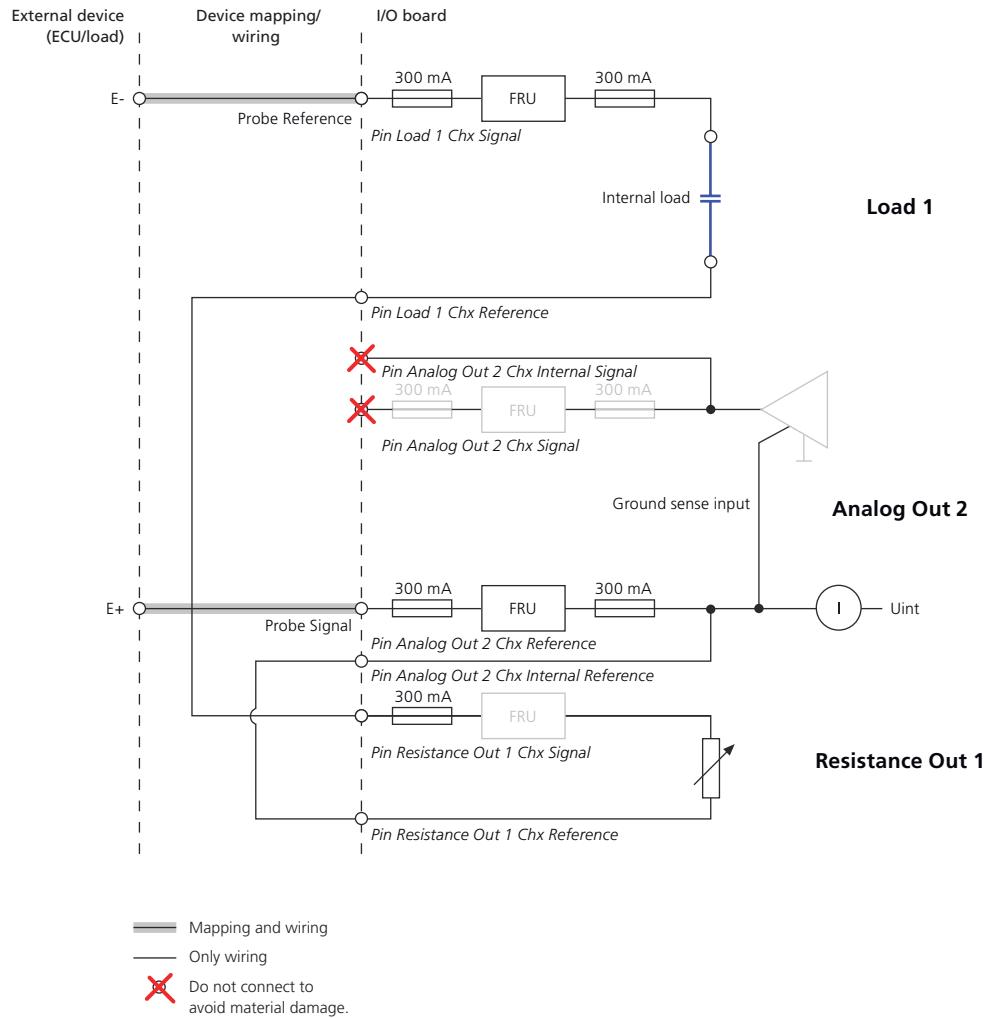
Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Mapping Signal Ports (Lambda DCR)

Purpose	To map and connect signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the used real-time hardware.
Circuit diagrams for the DS2680 I/O Unit	<p>The DS2680 provides internal load connection and isolation or non-isolation of the reference signal (selectable via Electrical interface selection property of the function block).</p> <p>Isolated reference In this configuration, the identical current flows into the circuit on one pin (Probe Reference port) and back out of the circuit on the other pin (Probe Signal port). Refer to the following illustration:</p>

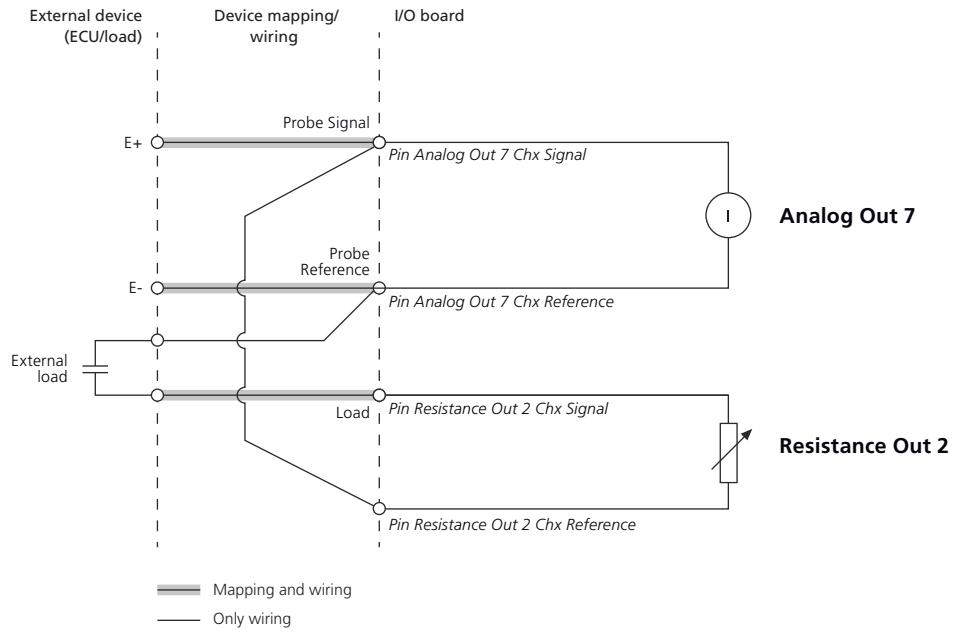


Non-isolated reference In this configuration, the current flows into the circuit via reference pin (Probe Reference port) and uses the internal GND of the dSPACE hardware as the return path. Refer to the following illustration:



Circuit diagram for the DS6101 Multi-I/O Board

The DS6101 provides external load connection and galvanic isolation of the reference signal.



Hardware Dependencies (Lambda DCR)

SCALEXIO Hardware Dependencies (Lambda DCR)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources that are supported by the function block type for the required channel types.

Hardware		DS2680 I/O Unit	DS6101 Multi-I/O Board
Required channel types		Resistance Out 1, Analog Out 2, Load 1	Resistance Out 2, Analog Out 7
Resistance Out	Channel type	Resistance Out 1	Resistance Out 2
	Current range	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW}/R)}$	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW}/R)}$
	Voltage range	-4 V ... +18 V	-4 V ... +18 V
	Resistance range	17 Ω ... 1 MΩ	17 Ω ... 1 MΩ
	Resolution of conductance	16 bit	16 bit
	Accuracy	±2% (typ.)	±2% (typ.)

Hardware		DS2680 I/O Unit	DS6101 Multi-I/O Board
Analog Out	Channel type	Analog Out 2	Analog Out 7
	FIU current range	0 ... +300 mA _{RMS}	–
	Output voltage range	-10 V ... +10 V	-10 V ... +10 V
	Output current range	-5 mA ... +5 mA	-5 mA ... +5 mA
	Resolution	14 bit	14 bit
Load	Channel type	Load 1	–
	Current range	0 ... +300 mA _{RMS}	(Connect load externally)
	Voltage range	-60 V ... +60 V	
	Internal load power dissipation	max. 2 W	
Circuit diagram		Mapping Signal Ports (Lambda DCR) on page 530	
Required channels		3 (one of each channel type)	2 (one of each channel type)

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration](#) .

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration](#) .

Lambda NCCR

Where to go from here**Information in this section**

Introduction (Lambda NCCR)	535
Overviews (Lambda NCCR)	536
Configuring the Function Block (Lambda NCCR)	539
Hardware Dependencies (Lambda NCCR)	547

Introduction (Lambda NCCR)

Introduction to the Function Block (Lambda NCCR)

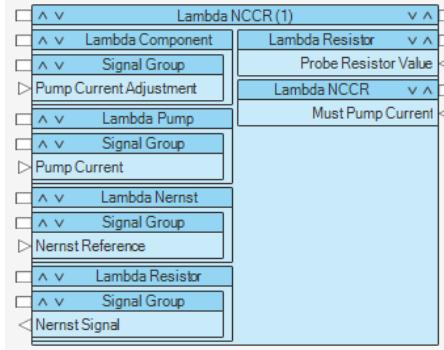
Function block purpose

With the Lambda NCCR function block, you can simulate wide-band lambda probes. The function block type depends on the current regulation method, i.e., Nernst-controlled current regulation (NCCR).

For an overview of the supported lambda probe types, refer to [SCALEXIO-Compatible Lambda Probe Types](#) on page 510.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Providing the Nernst-controlled pump current as defined from the behavior model.
- Providing the resistance of the Nernst cell as defined from the behavior model.
- Providing the reference potential for the Nernst voltage.

Required channel types

The Lambda NCCR function block requires the following hardware:

	SCALEXIO	MicroAutoBox III
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board
Channel types	Requires four channels based on the	Requires three channels based on the

	SCALEXIO	MicroAutoBox III
	following different channel types: <ul style="list-style-type: none">▪ Resistance Out 1▪ Analog Out 2▪ Load 1▪ Analog In 2	following different channel types: <ul style="list-style-type: none">▪ Resistance Out 2▪ Analog Out 7▪ Analog In 5

Oversviews (Lambda NCCR)

Where to go from here

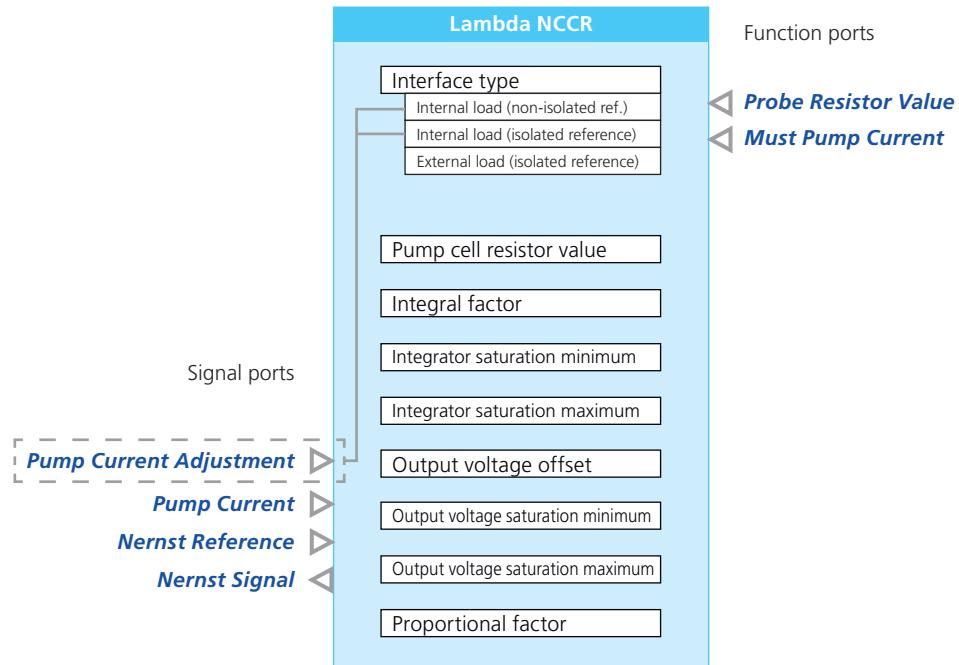
Information in this section

- Overview of Ports and Basic Properties (Lambda NCCR)..... 536
Overview of Tunable Properties (Lambda NCCR)..... 538

Overview of Ports and Basic Properties (Lambda NCCR)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Probe Resistor Value**

This function import provides the Nernst cell resistance as a function of the simulated Nernst cell temperature.

Value range	17 Ω ... 1 MΩ
Dependencies	–

Must Pump Current

This function import provides the required pump current (in Ampere) as a function of the simulated λ .

Value range	-10 mA ... +10 mA
Dependencies	–

Pump Current Adjustment

This signal import reads in the adjustment current I_a applied by the ECU and passes it across the gain adjustment resistor R_a of the pump current control circuit.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda NCCR) on page 547.
Dependencies	Available only if the Interface type property is set to Internal load (non-isolated reference) or Internal load (isolated reference).

Pump Current

This signal import reads in the pump current I_p applied by the ECU and passes it to the pump current control circuit.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda NCCR) on page 547.
Dependencies	–

Nernst Reference

This signal import represents the reference to the Nernst Signal signal port (V_m).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda NCCR) on page 547.
Dependencies	–

Nernst Signal

This signal output outputs the Nernst voltage V_n to the ECU.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda NCCR) on page 547.
Dependencies	–

Overview of Tunable Properties (Lambda NCCR)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Electrical Interface		
Pump cell resistor value	✓	-
Proportional factor	✓	-
Integral factor	✓	-
Output voltage offset	✓	-
Output voltage saturation min/max	✓	-
Integrator saturation min/max	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Lambda NCCR)

Where to go from here

Information in this section

Configuring the Basic Functionality (Lambda NCCR).....	539
Configuring Standard Features (Lambda NCCR).....	543
Mapping Signal Ports (Lambda NCCR).....	544

Configuring the Basic Functionality (Lambda NCCR)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Selecting the electrical interface type
- Specifying the pump current

Selecting the electrical interface type

To adapt the electrical interface of the function block to the dSPACE real-time hardware that you want to assign to the function block, you have to select an electrical interface type.

Note

The availability of electrical interface units and their properties depends on the setting of this property. Therefore, you are strongly recommended to set this property first.

Select the setting according to the hardware that you want to assign:

- If you want to assign a channel set from the DS2680 I/O Unit, select the Internal load (non-isolated reference) or Internal load (isolated reference) setting as follows:
 - You are recommended to use the Internal load (isolated reference) setting. Modern ECUs require the sensor simulation to be similar to an ideal two-terminal circuit that generates identical current flows on both signal terminals (signal and reference).

The hardware behaves like an ideal two-terminal network with the same current flow on the signal and reference pins. The current flows into the circuit on one pin and out of the circuit on the other.

Note

Not every DS2680 I/O Unit provides a galvanically isolated reference. You can identify a DS2680 that supports this feature as follows: In the Platform Manager, navigate the hardware tree of the SCALEXIO system to the Analog Out 2 item (via DS2680 I/O Unit - DS2680 I/O Module). If the Galvanic isolation property has the Galvanically isolated or the Switchable setting, the DS2680 supports this feature.

- If you select Internal load (non-isolated reference), the current flows into the circuit via the reference pin and uses the internal GND of the dSPACE hardware as the return path.

The reference pin is only used as a “sensing signal” for sensing an ECU GND potential as a reference for the Nernst voltage, and is therefore currentless in this configuration.

- If your ConfigurationDesk application (containing a Lambda NCCR function block) is created with ConfigurationDesk 5.1 or earlier, use the “non-isolated reference” setting to make the application compatible with the calculated external cable harness of the migrated previous application. If you use the “isolated reference” setting for the function block in the migrated application, you have to recalculate and build a new external cable harness.
- If you want to assign a channel set from the DS6101 Multi-I/O Board, select the External load (isolated reference) setting.

The DS6101 provides the connection of external loads and galvanic isolation of the reference signal. Therefore, the channels behave like an ideal two-terminal network with the same current flow on the signal and reference pins. The current flows into the circuit on one pin and out of the circuit on the other.

For circuit diagrams, refer to [Mapping Signal Ports \(Lambda NCCR\)](#) on page 544.

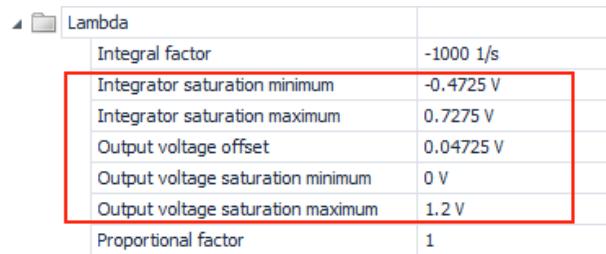
Specifying pump current

The pump current I_p is generated by the ECU and corresponds to λ of the exhaust gas. Finally the ECU provides $I_p = I_{p, \text{req}}$, hence U_n is balanced at 450 mV.

Pump current control properties The following pump current control properties (for a detailed property description, refer to [Lambda Component Electrical Interface Properties \(Lambda NCCR\) \(ConfigurationDesk Function Block Properties\)](#)) are available:

Property in ConfigurationDesk	Symbol	Description
Integral factor	K_i	Defines how quick the actual pump current approximates the required pump current
Proportional factor	K_p	Defines how I_p affects U_n . <ul style="list-style-type: none"> ▪ Bosch: 1 ▪ non-Bosch: 0
Pump cell resistor value	R_p	—
—	$I_{p, \text{req}}$	Required pump current
Integrator saturation minimum, Integrator saturation maximum	$U_{s\text{min}}, U_{s\text{max}}$	Integrator saturation limits which refer to GND
Output voltage saturation minimum, Output voltage saturation maximum	$U_{o\text{min}}, U_{o\text{max}}$	Output voltage saturation limits which refer to U_{ref}
Output voltage offset	U_{off}	—

You can specify the control properties in ConfigurationDesk (electrical interface of the Lambda NCCR function block), except for $I_{p, \text{req}}$, which is a run-time property (behavior model). The following illustration shows the pump current control properties and their default values for a Bosch LSU 4.2 probe.



Integral factor	-1000 1/s
Integrator saturation minimum	-0.4725 V
Integrator saturation maximum	0.7275 V
Output voltage offset	0.04725 V
Output voltage saturation minimum	0 V
Output voltage saturation maximum	1.2 V
Proportional factor	1

Note

The default values of the saturation limits and the voltage offset were identified by comprehensive testing. Do not change them. Otherwise, the pump current control might not work.

Control properties of different lambda probe types The following table shows appropriate property settings for different lambda probe types.

Manufacturer	Type	K _i	K _p	R _p	C _p	R _t
Bosch	LSU 4.2	-1000	1	200 Ω	—	0 Ω
	LSU 4.9	-100	1	100 Ω	approx. 10 μF	91 Ω
	LSU ADV	-1000	1	300 Ω ¹⁾	—	0 Ω
	LSU ADV (if operated with CJ135 Interface IC)	-1000	1	100 Ω	22 μF	0 Ω
NTK	L1H1	-1000	0	200 Ω	—	0 Ω
	L1H2	-1000	0	200 Ω	—	0 Ω
	LHA LZA-08-H6	-1000	0	200 Ω	—	0 Ω
	6312-W1	-1000	0	200 Ω	—	0 Ω
	ZFAS-U1	-1000	0	200 Ω	—	0 Ω
	ZFAS-U2	-100	0	110 Ω	approx. 40 μF	33 Ω
	ZFAS-U3	-100	0	680 Ω	approx. 22 μF	24 Ω

¹⁾ If λ cannot be adjusted, reduce R_p (Minimum: 100 Ω)

R_p, C_p and R_t must be connected to the hardware as internal and/or external load. Refer to [Mapping Signal Ports \(Lambda NCCR\)](#) on page 544. The value of R_t is highly temperature-dependent. The stated value is derived from a lambda probe working at operating temperature. If R_t is not required, you always have to short-circuit the pins (R_t = 0 Ω).

Note

Depending on the parametrization of the ECU, new generation lambda probe interface ICs, such as CJ135 or ATIC142, might have problems with the settings of the simulated lambda probe. In this case, contact dSPACE Support to evaluate whether an alternative pump control setting must be used to fix the problems.

Configuring Standard Features (Lambda NCCR)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The electrical interface of the function block provides the following standard features.
-----------------------------	------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Signal Ports	
Failure simulation support ¹⁾	Specifying Settings for Failure Simulation on page 123
Usage of loads ¹⁾	Specifying Load Settings on page 121

¹⁾ Configurability depends on the selected electrical interface configuration and the electrical interface unit that the signal port belongs to.

Valid for the DS2680:

NOTICE

The loads R_p (Resistance of the pump cell), and R_a (Resistance for the I_p adjustment) might be overloaded during failure simulation.

Risk of material damage.

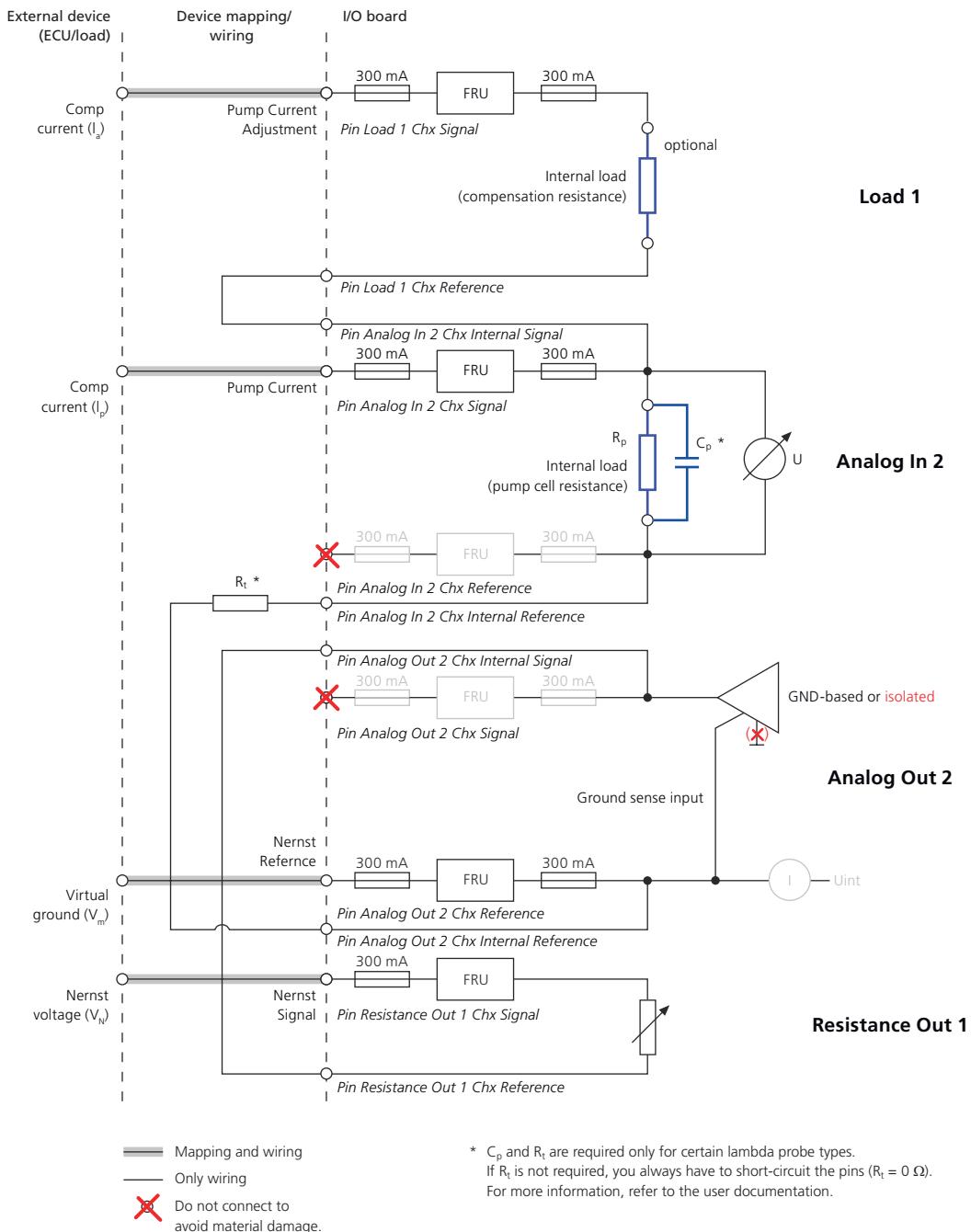
- Enforce load rejection for the Pump Current and Pump Current Adjustment signal ports.
- For details on enforcing load rejection, refer to [Basics on Load Rejection \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(e3d499ec04bd814692ea15042aa151b7_img.jpg\)](#).

Model interface	The interface to the behavior model of the function block provides the following standard features.
------------------------	-----------------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Mapping Signal Ports (Lambda NCCR)

Purpose	To map and connect signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the used real-time hardware.
Circuit diagrams for the DS2680 I/O Unit	The DS2680 provides internal load connection and isolation or non-isolation of the reference signal (selectable via Electrical interface selection property of the function block).

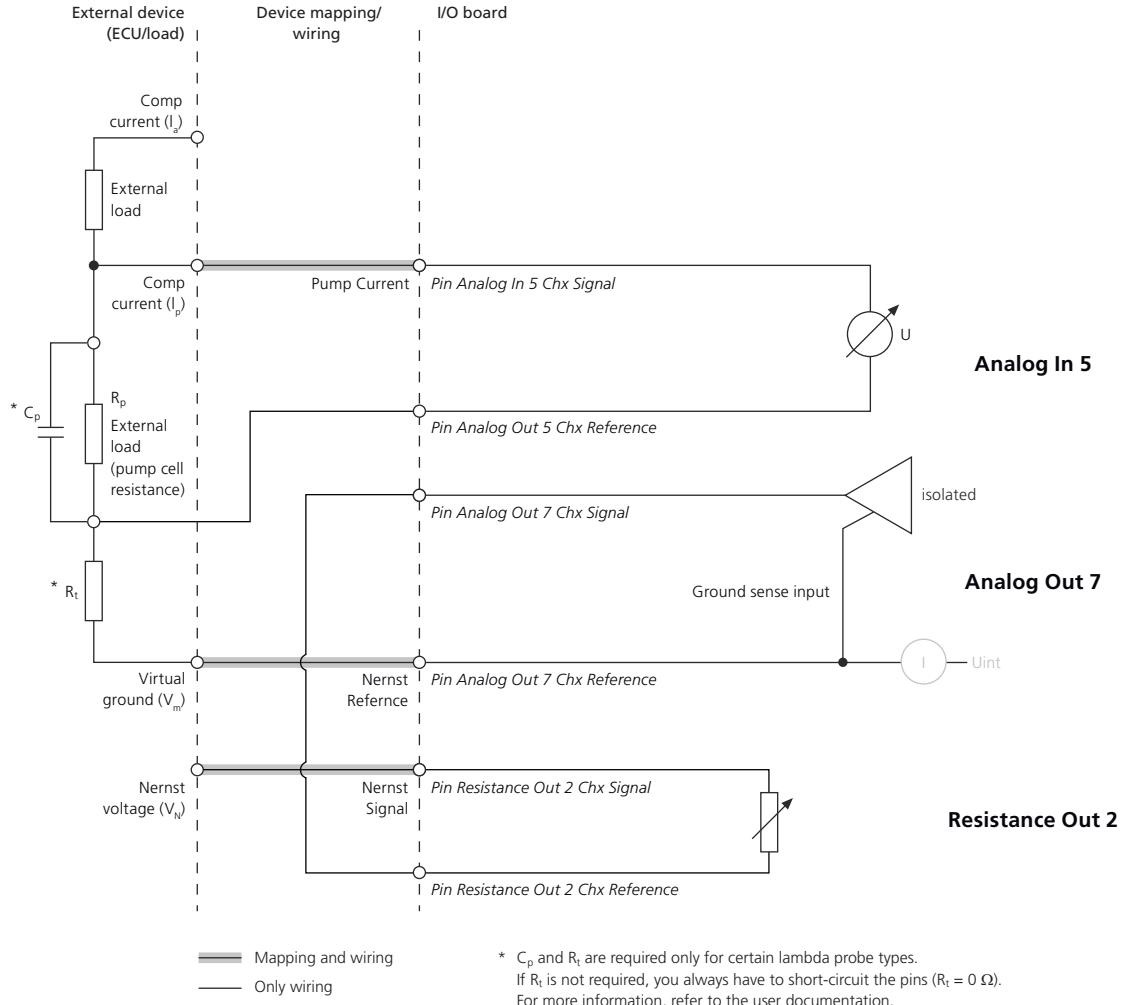


The values for R_p , C_p and R_t depend on the lambda probe type used. Refer to [Configuring the Basic Functionality \(Lambda NCCR\)](#) on page 539.

If R_t is not required, you always have to short-circuit the pins ($R_t = 0 \Omega$).

Circuit diagram for the DS6101 Multi-I/O Board

The DS6101 provides external load connection and galvanic isolation of the reference signal.



The values for R_p , C_p and R_t depend on the lambda probe type used. Refer to [Configuring the Basic Functionality \(Lambda NCCR\)](#) on page 539.

If R_t is not required, you always have to short-circuit the pins ($R_t = 0 \Omega$).

Hardware Dependencies (Lambda NCCR)

SCALEXIO Hardware Dependencies (Lambda NCCR)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources that are supported by the function block type for the required channel types.

Hardware		DS2680 I/O Unit	DS6101 Multi-I/O Board
Required channel types		Resistance Out 1, Load 1, Analog Out 2, Analog In 2	Resistance Out 2, Analog Out 7, Analog In 5
Resistance Out	Channel type	Resistance Out 1	Resistance Out 2
	Current range	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW/R})}$	Whichever occurs first: ▪ $I_{Max} = \pm 80 \text{ mA}$ or ▪ $I_{Max} = \sqrt{(250 \text{ mW/R})}$
	Voltage range	-4 V ... +18 V	-4 V ... +18 V
	Resistance range	17 Ω ... 1 MΩ	17 Ω ... 1 MΩ
	Resolution of conductance	16 bit	16 bit
	Accuracy	±2% (typ.)	±2% (typ.)
Load	Channel type	Load 1	– (Connect load externally)
	Current range	0 ... +300 mA _{RMS}	
	Voltage range	-60 V ... +60 V	
	Internal load power dissipation	max. 2 W	
Analog Out	Channel type	Analog Out 2	Analog Out 7
	FIU current range	0 ... +300 mA _{RMS}	–
	Output voltage range	-10 V ... +10 V	-10 V ... +10 V
	Output current range	-5 mA ... +5 mA	-5 mA ... +5 mA
	Resolution	14 bit	14 bit
Analog In	Channel type	Analog In 2	Analog In 5
	FIU current range	0 ... 300 mA _{RMS}	–
	Input voltage range	-60 V ... +60 V	-60 V ... +60 V
	Internal load power dissipation	max. 2 W	–
	Voltage measuring range	-2 V ... +2 V	-2 V ... +2 V
	Resolution	12 bit	12 bit
	Measurement point	Load side	–
Circuit diagram		Mapping Signal Ports (Lambda NCCR) on page 544	
Required channels		4 (one of each required channel type)	3 (one of each required channel type)

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Engine Control

Where to go from here

Information in this section

Introduction to Engine Control.....	550
Engine Control Setup.....	556
Crank In.....	563
Cam In.....	587
Injection Out.....	604
Ignition Out.....	617
Engine Angular Pulse Out.....	629
Knock In.....	637
Lambda Probe In.....	647

Introduction to Engine Control

Where to go from here

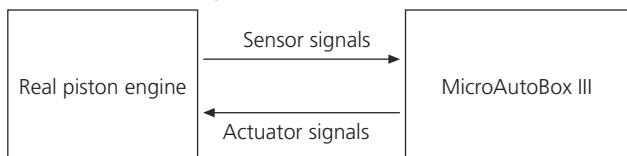
Information in this section

- | | |
|--------------------------------------------------------|-----|
| Basics on Engine Control with MicroAutoBox III..... | 550 |
| Operation Principle of Four-Stroke Piston Engines..... | 552 |

Basics on Engine Control with MicroAutoBox III

MicroAutoBox III connected to real piston engine

The MicroAutoBox III is used to develop controllers for real piston engines. The Micro AutoBox III sends actuator signals to and receives sensor signals from the controlled piston engine.



Available function blocks for engine control

ConfigurationDesk provides the following function block types for controlling a piston engine via the MicroAutoBox III:

Crank In The Crank In function block evaluates the crankshaft signal for synchronizing the crankshaft with the angular processing unit (APU) and for measuring engine position, speed, and rotational direction.

Cam In The Cam In function block evaluates the camshaft signal for supporting crankshaft/APU synchronizing and for measuring camshaft phase shift.

Engine Angular Pulse Out The Engine Angular Pulse Out function block generates a pulse pattern depending on the engine position.

Engine Control Setup The Engine Control Setup function block is used to specify the characteristics of the piston engine such as number of cylinders and top dead centers. Furthermore, it can generate I/O events depending on the engine position.

Ignition Out The Ignition Out function block generates ignition pulses for assigned cylinders.

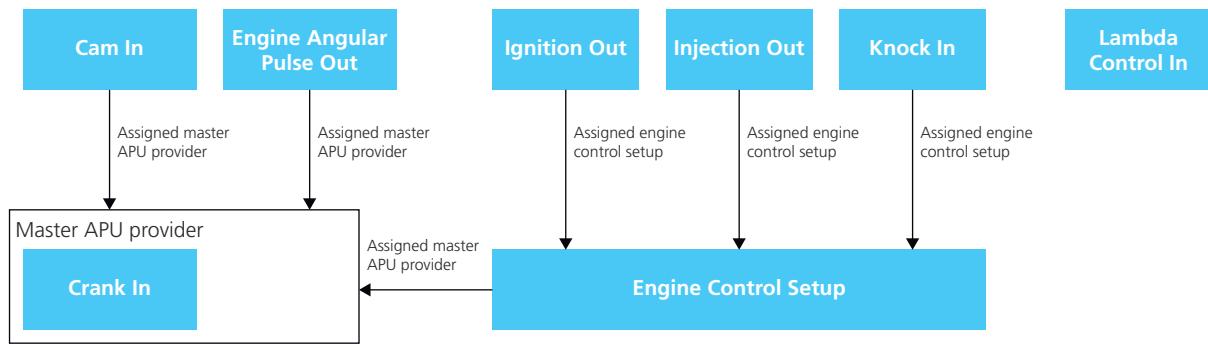
Injection Out The Injection Out function block generates injection pulses for assigned cylinders.

Knock In The Knock In function block analyzes knock signals from assigned cylinders.

Lambda Control In The Lambda Control In function block operates a wideband lambda probe to measure the oxygen content in the exhaust gas of the controlled piston engine.

Function block dependencies

Engine control function blocks are functionally related with other engine control function blocks and must therefore be used in specific combinations (except for the Crank In and the Lambda Control In function blocks). Refer to the following illustration:



The Crank In function block is the only master APU provider for engine control applications. Together with the Cam In function block, it synchronizes the master APU with the crankshaft of the piston engine. For more information on APUs, refer to [Using Angular Processing Units \(APUs\) on page 126](#).

The Engine Control Setup function block provides engine characteristics and the angle range of the master APU inherited from the Crank In function block. It can also be used to simultaneously enable/disable the pulse generation of all the Ignition Out and Injection Out function blocks in the application.

The Lambda Control In function block works independently from all the other engine control function blocks.

Function blocks required by common tasks

Your tasks determine the type of engine control function blocks that you have to use. The following table lists some common tasks and the function blocks provided by ConfigurationDesk for performing them.

Task	Required function blocks							
	Crank In	Cam In	Engine Angular Pulse Out	Engine Control Setup	Injection Out	Ignition Out	Knock In	Lambda Control In
Measurement of	▪ Engine speed ▪ Engine direction	✓	–	–	–	–	–	–
	▪ Engine position	✓	✓	–	–	–	–	–

Task		Required function blocks							
		Crank In	Cam In	Engine Angular Pulse Out	Engine Control Setup	Injection Out	Ignition Out	Knock In	Lambda Control In
	▪ Camshaft phase shift								
Generation of	Angular pulses	✓	✓	✓	–	–	–	–	–
	Angular events	✓	✓	–	✓	–	–	–	–
	Injection pulses	✓	✓	–	✓	✓	–	–	–
	Ignition pulses	✓	✓	–	✓	–	✓	–	–
Measurement of	Knock signals	✓	✓	–	✓	–	–	✓	–
	Oxygen content in exhaust gas (λ)	–	–	–	–	–	–	–	✓

Maximum number of usable instances of function block types

The following table shows how many function blocks of each type can be used in a ConfigurationDesk application.

	Crank In	Cam In	Engine Angular Pulse Out	Engine Control Setup	Injection Out	Ignition Out	Knock In	Lambda Control In
Maximum number of function blocks	1	4	1	1	6	6	4	2

Available demo project

ConfigurationDesk provides the EngineControlDemo demo project. This project shows an implementation of the engine control functionality. You can use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to [EngineControlDemo Project: Controlling an Engine \(ConfigurationDesk Demo Projects\)](#).

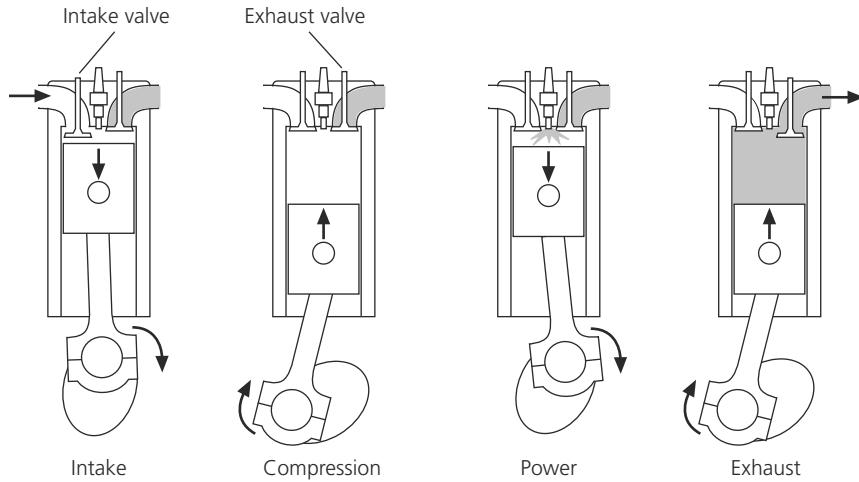
Operation Principle of Four-Stroke Piston Engines

Objective

Outlines the basic functionality and terminology of four-stroke piston engines.

Four-stroke cycle

Today, internal combustion engines in cars, trucks, motorcycles, construction machinery and many others, most commonly use a four-stroke cycle.



The four strokes make up one engine cycle and are as follows:

1. Intake:

The piston descends from the top to the bottom of the cylinder, reducing the pressure inside it. A mixture of fuel and air is forced into the cylinder. The intake valves then close.

2. Compression:

With both intake and exhaust valves closed, the piston returns to the top of the cylinder, compressing the fuel-air mixture.

3. Power:

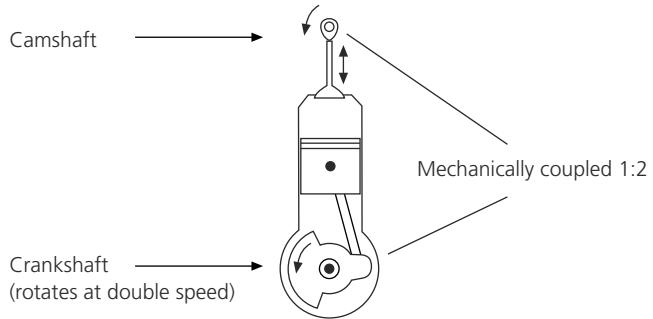
While the piston is at or close to top dead center, the compressed air-fuel mixture is ignited, generally by a spark plug (for a gasoline or spark ignition engine) or by the heat and pressure of compression (for a diesel cycle or compression ignition engine). The massive pressure resulting from the combustion of the compressed fuel-air mixture drives the piston back down toward bottom dead center with tremendous force.

4. Exhaust:

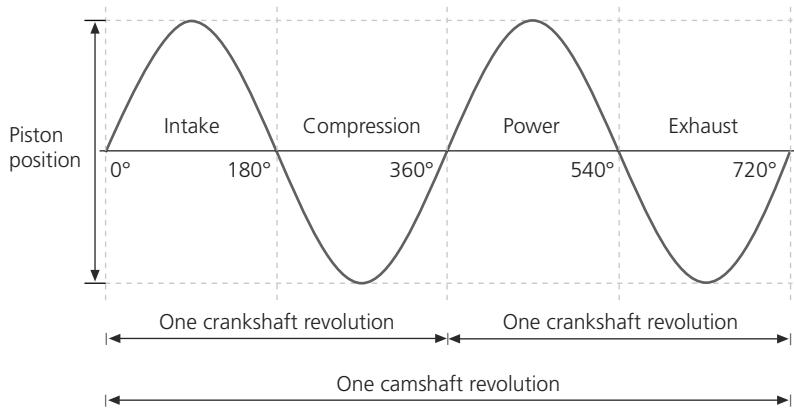
The piston once again returns to top dead center while the exhaust valve is open. This action evacuates the products of combustion from the cylinder by pushing the spent fuel-air mixture through the exhaust valves.

Crankshaft and camshaft

The linear up and down piston motion is translated into crankshaft rotation. The crankshaft transmits the engine power to the wheels via the gearbox. Commonly, one or even multiple camshafts control the intake and exhaust valves. They are mechanically coupled to the crankshaft. The four-stroke operation principle requires that the crankshaft rotates at double speed referred to the camshaft.

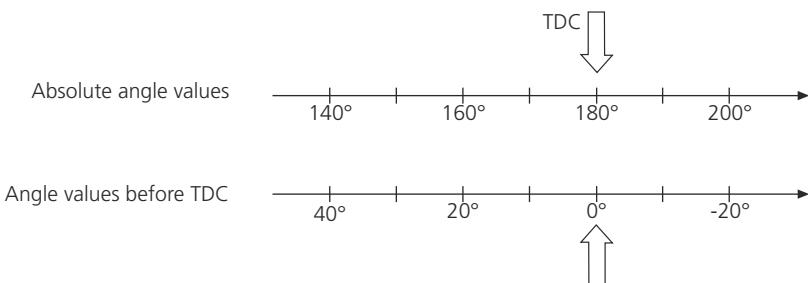


One engine cycle of a four-stroke piston engine equals two crankshaft revolutions and one camshaft revolution. The angular position of the crankshaft corresponds to the position of the piston. Therefore, if you know the position of the crankshaft, you can derive the position of the piston. However, you do not know which stroke it is (compression, exhaust, etc.) unless you also know the angular position of the camshaft.



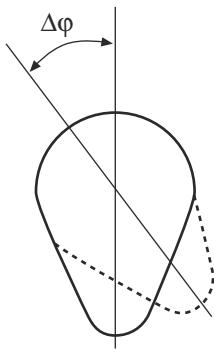
Top dead center

Top dead center (TDC) is the upper reversal point of a piston after compression and exhaust, and is represented by an angle value in the range $0^\circ \dots 720^\circ$. TDC is a cylinder-specific attribute, and each cylinder has its own TDC. TDC is the datum point from which engine timing measurements are made. For example, injection and ignition timing is normally specified relative to the TDC as degrees before top dead center (BTDC). Positions before TDC have positive angle values, positions after TDC have negative ones.



Camshaft phase shift

A camshaft phase shift means an angular displacement of a camshaft relative to the coupled crankshaft during operation, triggered by an external mechanism. It is commonly a function of speed and load, and aims to save fuel and increase power. The following illustration shows the angular displacement $\Delta\varphi$ of a cam when the camshaft is shifted.



Engine Control Setup

Where to go from here

Information in this section

Introduction (Engine Control Setup).....	556
Overviews (Engine Control Setup).....	557
Configuring the Function Block (Engine Control Setup).....	559

Introduction (Engine Control Setup)

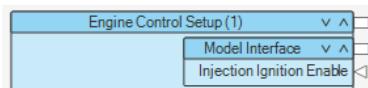
Introduction to the Function Block (Engine Control Setup)

Function block purpose

The Engine Control Setup function block specifies basic characteristics of an existing (real) piston engine that you want to control. The function block works as a provider: Other function blocks can use it to obtain information on the characteristics of the specified piston engine.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Specifying the number of cylinders and their top dead centers of the controlled piston engine.
- Generating single or periodic I/O events and providing them to the behavior model.
- Global enabling the generation of injection and ignition pulses.

Oversviews (Engine Control Setup)

Where to go from here

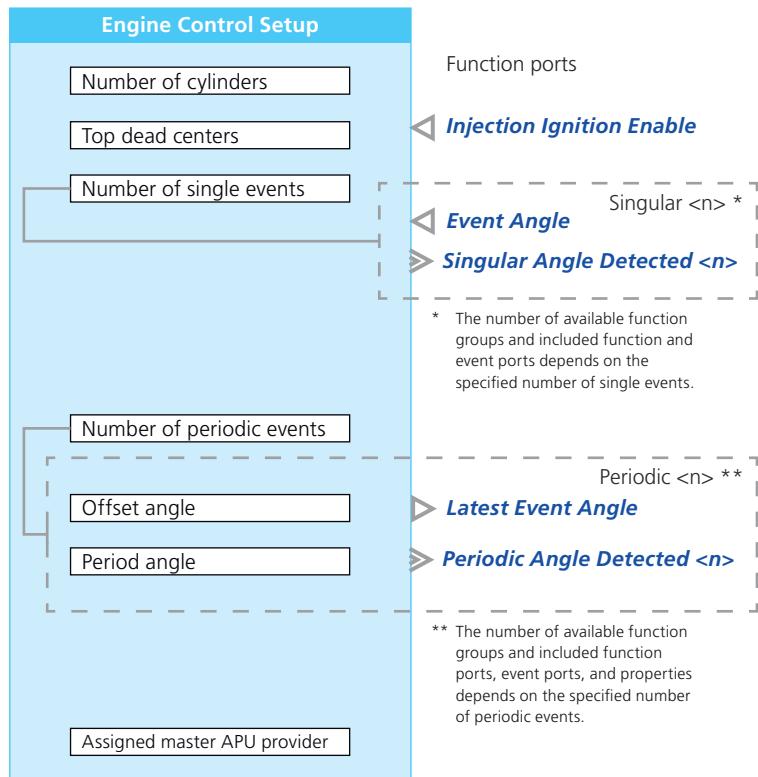
Information in this section

- | | |
|------------------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Engine Control Setup)..... | 557 |
| Overview of Tunable Properties (Engine Control Setup)..... | 559 |

Overview of Ports and Basic Properties (Engine Control Setup)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Injection Ignition Enable

From within the behavior model, this function import lets you globally enable the generation of injection/ignition pulses for all the Injection Out and Ignition Out function blocks that use the Engine Control Setup function block as setup

block. This might be required, for example, if all the cylinders of the piston engine have to be turned on or off simultaneously.

Each of the connected **Injection Out** and **Ignition Out** function blocks generates pulses only if its own **Injection Enable** or **Ignition Enable** function import is also set to **Enabled**.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	—

Event Angle

This function import lets you define the angle value for generating single I/O events from within the behavior model. Each time this value matches the engine cycle angle, the function block generates an I/O event.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° ▪ The effective range depends on the setting of the Angle range property of the assigned master APU provider: <ul style="list-style-type: none"> ▪ 360°: 0° ... 360° ▪ 720°: 0° ... 720° <p>If you enter values outside the effective range, the system converts them to positive values matching the effective range. Example: -400° is converted to +320°, +830° is converted to +110°.</p>
Dependencies	Available only if the value of the Number of single events property is greater than 0 (zero).

Latest Event Angle

This function outport provides the value of the engine cycle angle to the behavior model at which an I/O event of a sequence of periodic events is generated. This value is updated each time an I/O event is generated.

Value range	<ul style="list-style-type: none"> ▪ 0° ... 359.9° ▪ 0° ... 719.9° ▪ The range depends on the setting of the Angle range property of the assigned master APU provider.
Dependencies	Available only if the value of the Number of periodic events property is greater than 0 (zero).

Singular Angle Detected <n>

This event port provides single I/O events to the behavior model. They are generated each time the engine cycle angle matches the value at the **Event Angle** function import.

Value range	—
Dependencies	Available only if the value of the Number of single events property is greater than 0 (zero).

Periodic Angle Detected <n>

This event port provides a sequence of periodic I/O events to the behavior model. The sequence starts when the engine cycle angle matches the value of the Offset angle property. The following I/O events of the sequence are generated periodically each time the engine cycle angle increases by the value specified at the Period angle property.

Value range	–
Dependencies	Available only if the value of the Number of periodic events property is greater than 0 (zero).

Overview of Tunable Properties (Engine Control Setup)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Offset angle	✓	–
Period angle	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Engine Control Setup)

Where to go from here**Information in this section**

Configuring the Basic Functionality (Engine Control Setup).....	560
Configuring Standard Features (Engine Control Setup).....	561

Configuring the Basic Functionality (Engine Control Setup)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Assigning a master APU provider.
- Specifying the piston engine.
- Specifying the generation of I/O events.

Assigning a master APU provider

You have to assign a master APU provider that references an angular processing unit (APU). The APU is part of the MicroAutoBox III and provides the engine cycle position data of the piston engine to the function block. For more information on angular processing units, refer to [Using Angular Processing Units \(APUs\) on page 126](#).

Specifying the piston engine

The Engine Control Setup function block lets you configure the following characteristics of the piston engine:

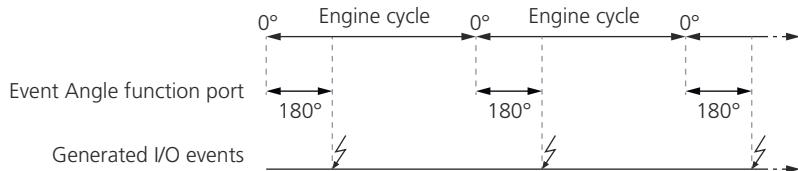
- Number of cylinders (max. 6).
- Top dead center of each cylinder.
- Receiving current engine cycle positions from a real piston engine.

Generating I/O events

The function block can generate single I/O events and sequences of periodic I/O events to use them as trigger sources for runnable functions in the behavior model.

Single I/O events You have to specify the number of independent single I/O events. Each of them is generated once per engine cycle when the engine cycle angle matches the value at the related Event Angle function import. The related event port provides the generated I/O events to the behavior model.

The following sample illustration shows single I/O events generated at the 180° angle position of each engine cycle.



To use the I/O events in the behavior model, you have to assign them to a task via the [Single Angle Detected <n>](#) property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

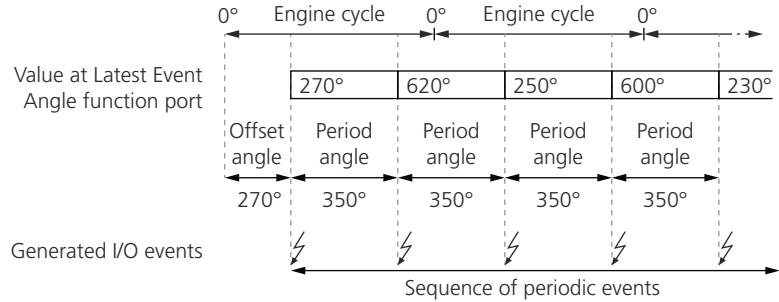
Periodic I/O events You have to specify the number of independent periodic I/O event sequences. For each sequence, the following information is required:

- You have to specify the angular offset of the start point of the sequence with respect to the 0° position of the engine cycle at the related **Offset angle** property.
- You have to specify the angular distance between generated events at the related **Period angle** property.

The first I/O event of a sequence is generated when the engine cycle angle matches the value of the related offset angle. The following I/O events are generated each time the engine cycle angle increases by the value of the related period angle.

The related event port provides the generated I/O events to the behavior model. The engine cycle angle of a generated I/O event is available at the related **Latest Event Angle** function output until it is overwritten when the next I/O event is generated. Event generation stops when the crank speed falls below 5 rpm or reverse crankshaft rotation is detected. Reverse crankshaft rotation is only evaluated when reverse crank mode is enabled.

The following sample illustration shows a sequence of periodic I/O events generated according to an offset angle of 270° and a period angle of 350° .



To use the I/O events in the behavior model, you have to assign them to a task via the **Periodic Angle Detected <n>** property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Engine Control Setup)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	This function block does not provide an electrical interface.
-----------------------------	---------------------------------------------------------------

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Crank In

Where to go from here

Information in this section

Introduction (Crank In).....	563
Overviews (Crank In).....	572
Configuring the Function Block (Crank In).....	578
Hardware Dependencies (Crank In)	586

Introduction (Crank In)

Where to go from here

Information in this section

Introduction to the Function Block (Crank In).....	563
Basics on Crankshaft Sensor Signals.....	564
Supported Crankshaft Wheels.....	565
Basics on Synchronizing the Crankshaft.....	566
Basics on Speed Measurement.....	571

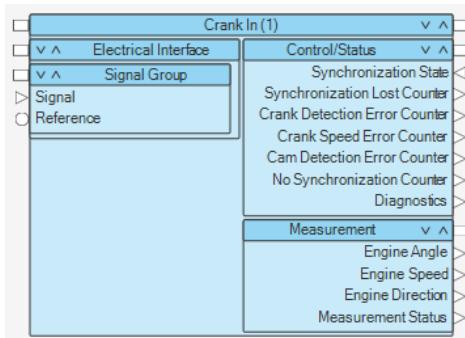
Introduction to the Function Block (Crank In)

Function block purpose

The Crank In function block type lets you measure the crankshaft of a piston engine and calculates the current position, speed and rotational direction of the engine. For performing these measurements, the Crank In function block must be used in combination with at least one Cam In function block.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

**Main features**

These are the main features:

- Synchronizing the crankshaft and the master angular processing unit (APU).
- Measuring and evaluating engine angle, speed, and rotational direction, and providing the results to the behavior model.
- Detecting measurement errors.
- Providing error counter readings and status information to the behavior model.
- Generating I/O events and providing them to the behavior model.
- Supporting active and passive crankshaft sensors.

Supported channel types

The Crank In function block supports the following channel types:

	SCALEXIO	MicroAutoBox III	
Channel type	–	Digital In 9	Digital In 10
Hardware	–	DS1554	

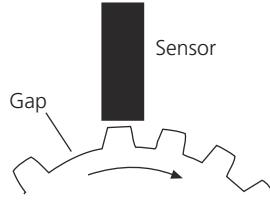
Basics on Crankshaft Sensor Signals

Introduction

The crankshaft sensor generates a signal that describes the motion of the crankshaft. The crankshaft signal is particularly used for speed measurement and, together with the camshaft signal, for angle measurement, which is required for the generation of injection and ignition pulses and angle-based pulse patterns.

Crankshaft sensor

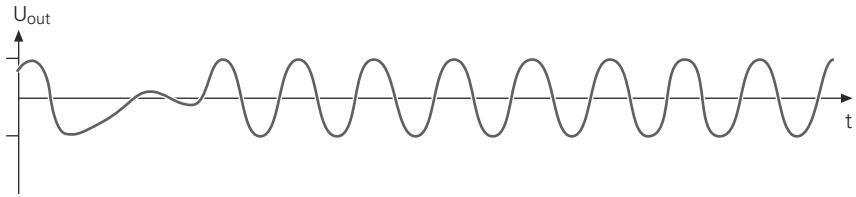
An optical or magnetic sensor scans the radial surface of the rotating crankshaft wheel. The crankshaft wheel is an even-toothed wheel flanged to the crankshaft. A gap is a place where teeth are missing.



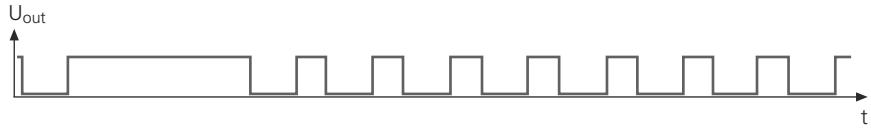
Signals of supported sensor types

Crankshaft signals are either passive or active, depending on the type of sensor you use. The amplitude of the signal generated also depends on the sensor.

Passive sensors Generate an analog signal, for example, as shown in the following illustration:



Active sensors Generate a digital signal, for example, a TTL signal as shown in the following illustration:



Intelligent crankshaft sensors with direction detection To detect reverse crankshaft rotation, a more sophisticated type of active crankshaft sensor is needed. Reverse crankshaft rotation happens, for example, in the following situations:

- Pressure in cylinders due to compression is reduced after engine shutdown.
- The engine is restarted according to combustion, as in modern stop-start automatics.

What is known as *reverse crankshaft sensors* evaluate forward and reverse rotation of the crankshaft. Reverse crankshaft sensors are active sensors and generate pulse signals. The pulse duration of these signals depends on the current rotational direction (e.g., $t_{\text{forward}} = 45 \mu\text{s}$, $t_{\text{reverse}} = 90 \mu\text{s}$).

Supported Crankshaft Wheels

Introduction

The engine control features of ConfigurationDesk support symmetric and asymmetric crankshaft wheels.

Symmetric crankshaft wheels

A crankshaft wheel is called symmetric if all the following conditions are met:

- There are the same number of teeth between all the gaps (segments).
- All the gaps comprise the same number of missing teeth.
- All the teeth have the same angular width.

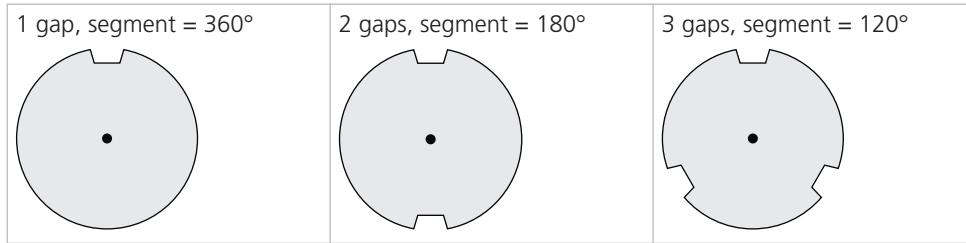
Note

Wheels that are named with respect to the crankshaft wheel parameters, for example, 60 – 2 wheel, are symmetric wheels.

Segments A segment comprises the following angle range:

$$\text{Segment} = 360^\circ / \text{number of gaps}$$

Example These are some examples of symmetric crankshaft wheels (schematic illustration):

**Asymmetric crankshaft wheels**

A crankshaft wheel is asymmetric if one or more of the following conditions are met:

- The number of teeth between the gaps differs.
- The number of missing teeth in the gaps differs.
- The angular width of the teeth differs.

Crankshaft wheel specifications

Crankshaft wheels are specified via wavetables files. For more information on specifying crankshaft wheels, refer to [Configuring the Basic Functionality \(Crank In\)](#) on page 579.

Basics on Synchronizing the Crankshaft

Definition of crankshaft synchronization

Crankshaft synchronization means synchronizing the crankshaft of the engine and the angular processing unit (APU) so that the actual engine position can be determined. This also requires a signal from a connected camshaft so that the APU can determine whether measured crankshaft edges belong to the first or the second crankshaft revolution (0 ... 359.9° or 360° ... 719.9°) within the

engine cycle. A camshaft signal is *not* required for the crankshaft synchronization of two-stroke engines, if the crankshaft wheel has only one segment or segments with different numbers of teeth.

Definition of the 0° position

By default, the 0° position of the angle measurement is internally defined to match the first wavetable data point. You can change this relationship by specifying a start angle $\neq 0^\circ$. All angle values relate to the 0° position. For more information on specifying start angles, refer to [Specifying the start angle for the engine cycle](#) on page 582.

Note

For the wavetables that are provided with the ConfigurationDesk EngineControlDemo demo project, the first data point relates to the first rising edge after a gap.

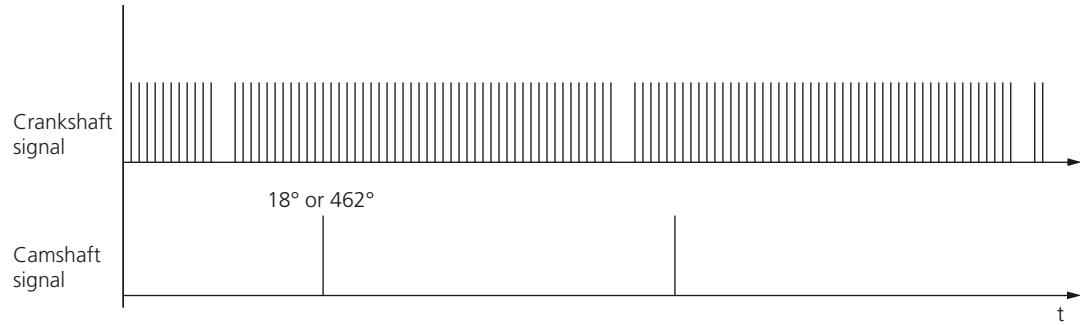
Synchronization of the crankshaft

The following example demonstrates the synchronization of the crankshaft for a sample piston engine.

The crankshaft wheel and the camshaft wheel of the sample piston engine are specified as follows:

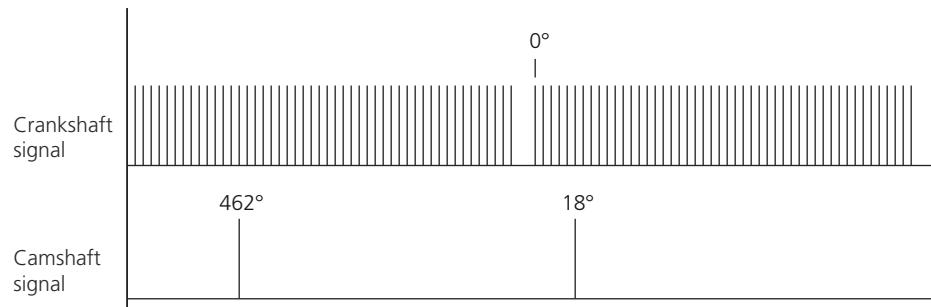
Wheel	Specification
Crankshaft	<ul style="list-style-type: none">▪ Number of teeth (incl. missing): 60▪ Number of gaps: 1▪ Number of missing teeth per gap: 2▪ Edge type: Rising▪ Start angle: 0° (i.e., the 0° position equals the first rising edge after the gap) <p>For more information on specifying crankshaft wheels, refer to Configuring the Basic Functionality (Crank In) on page 579.</p>
Camshaft	<ul style="list-style-type: none">▪ Number of camshaft pulses: 2▪ Edge type: Rising▪ Camshaft angles (rising edges): 18° and 462° <p>For more information on specifying camshaft wheels, refer to Configuring the Basic Functionality (Cam In) on page 597.</p>

The following signals (rising edges) from the crankshaft and camshaft wheels of the sample piston engine are measured:



The question now is: Where is the 0° position so that the measured crankshaft and camshaft signals match the crankshaft wheel and camshaft wheel specifications? Or in other words: Does the camshaft signal edge that was measured first correspond to 18° or 462° ?

There is only one solution for assigning the 0° position to a crankshaft signal edge so that the crankshaft and camshaft signals match the crankshaft/camshaft wheel specifications:



Note

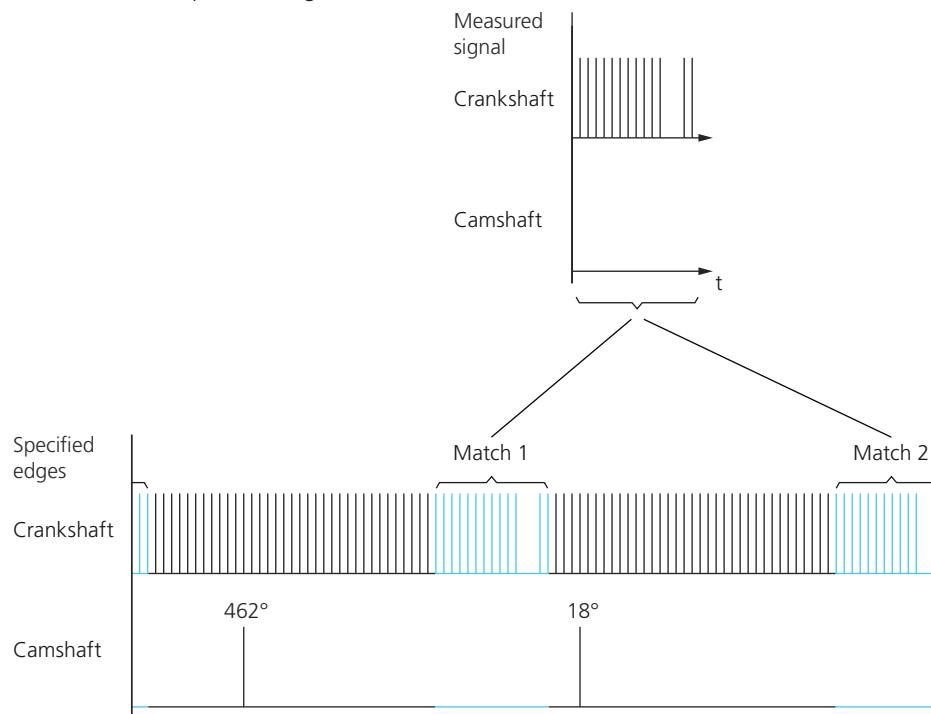
The pulses on the camshaft wheel pulses are placed unambiguously so that only one match exists.

Synchronization process

When the crankshaft and camshaft signals are measured, the system repeatedly tries (whenever a signal edge is detected) to match the signal edge pattern measured so far with the specified signal edge pattern. The longer the measured signal pattern is, the more likely is a definite match and therefore synchronization.

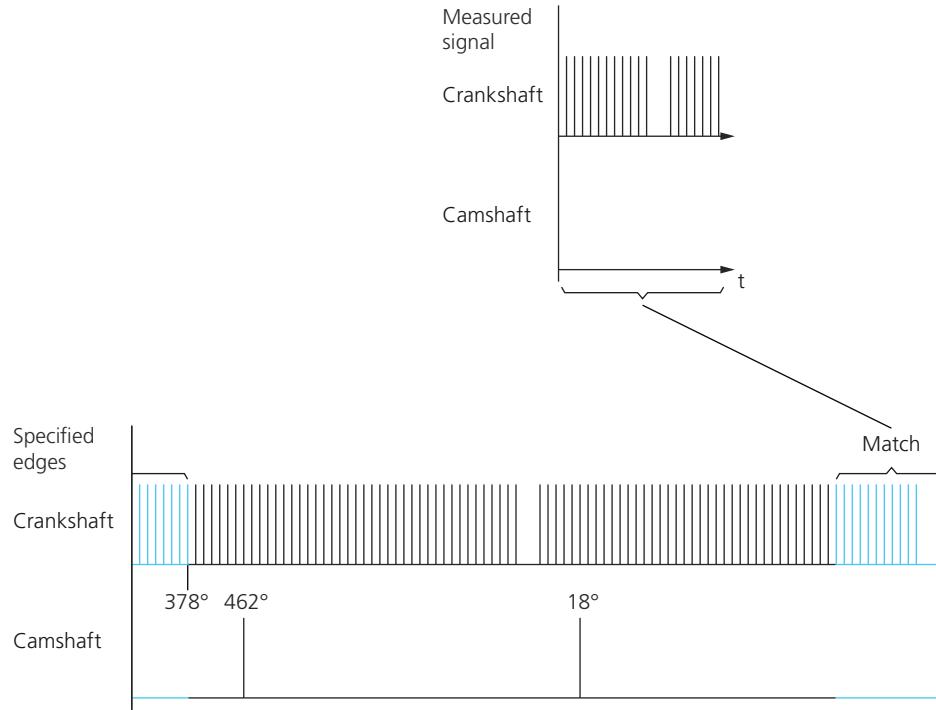
Example Refer to the following two signal match attempts:

1. The following signal pattern is measured in the first attempt and compared to the specified edges:



This leads to two matches with the specification. Synchronization is not yet achieved.

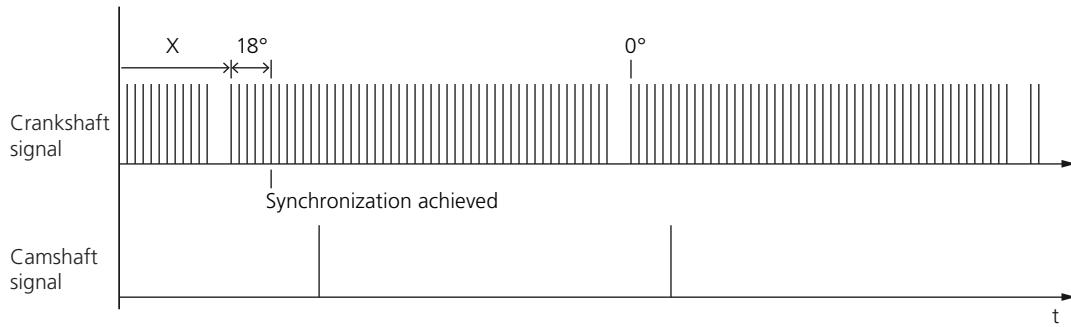
2. For the second attempt, the measured signal pattern has become longer:



The APU detects the *nonexistent* camshaft signal edge 18° after the gap. Because of this, the signal pattern measured until this point belongs to the edge that relates to the 462° camshaft pulse. The current engine position angle is computed to 378° and crankshaft synchronization is achieved.

Synchronization time For engine control, it is important to achieve synchronization as quickly as possible. For example, the generation of injection and ignition pulses cannot start until the crankshaft and camshaft signals are synchronized.

The actual synchronization time depends on the crankshaft and camshaft wheels used (or their specifications) and the positions of the crankshaft and the camshaft at start time (Time = 0). For the above example of measured crankshaft/camshaft signals, synchronization is achieved at $18^\circ + X$ after start time.



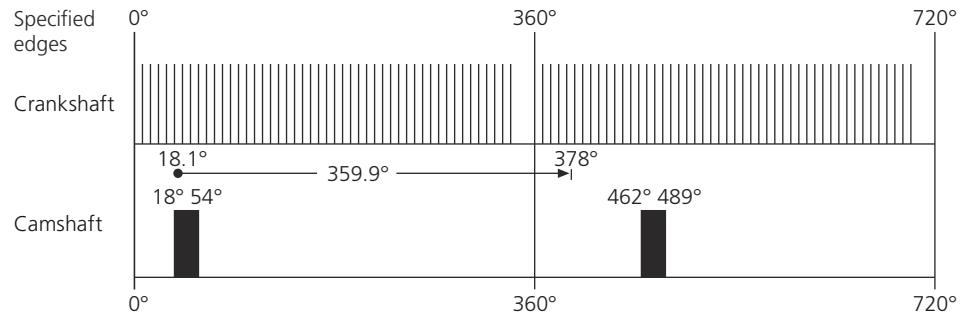
Worst-case synchronization time X depends on the engine position at start time plus two or three teeth (due to technical restrictions). A worst-case

synchronization time can be calculated for each crankshaft/camshaft wheel design according to the most unfavorable engine position at start time.

Refer to the following examples:

1. Edge type: Rising.

Synchronization is achieved after 359.9° at the latest ($359.9^\circ = 360^\circ - 18.1^\circ + 18^\circ$). The most unfavorable engine start position is at 18.1° .

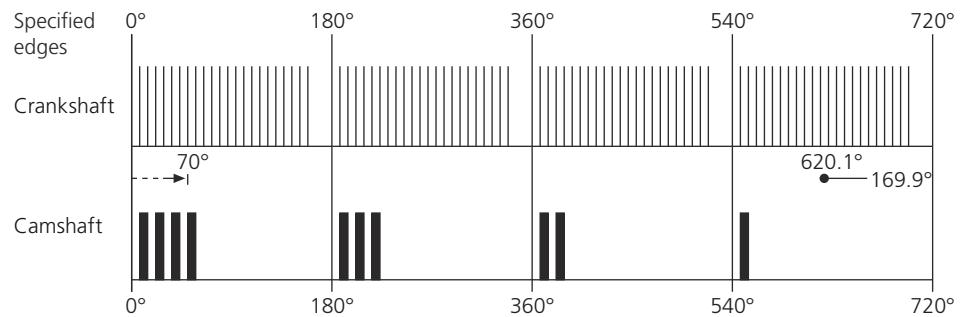


2. Edge type: Rising and falling.

Synchronization is achieved after 169.9° at the latest ($169.9^\circ = 720^\circ - 620.1^\circ + 70^\circ$). The most unfavorable engine start position is at 620.1° .

Specified camshaft angles (refer to the pulses in the following illustration from left to right):

- #1: $10^\circ \dots 20^\circ$, #2: $30^\circ \dots 40^\circ$, #3: $50^\circ \dots 60^\circ$, #4: $70^\circ \dots 80^\circ$
- #5: $190^\circ \dots 200^\circ$, #6: $210^\circ \dots 220^\circ$, #7: $230^\circ \dots 240^\circ$
- #8: $370^\circ \dots 380^\circ$, #9: $390^\circ \dots 400^\circ$
- #10: $550^\circ \dots 580^\circ$



Basics on Speed Measurement

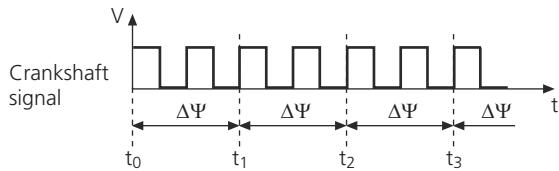
Introduction

Engine speed is evaluated by measuring the time that the crankshaft takes to rotate over a user-defined angle range $\Delta\psi$.

Speed calculation

Engine speed is calculated as follows:

$$\text{Speed}(t_i) = \Delta\psi / (t_i - t_{i-1})$$



- $\Delta\psi$ is the measurement angle based on the measurement of crankshaft signal edges and the crankshaft specification in the wavetable file.
- t_i is the time measured by the function block.

Speed measurement modes

Engine speed can be measured and calculated either in direct mode or in recursive mode. For more information on the measurement modes, refer to [Specifying the mode of engine speed measurement](#) on page 580.

Speed measurement range

Engine speed is measured in rpm (revolutions per minute) for the following range:

Rpm _{min}	Rpm _{max}
0	20,000

Reverse crankshaft rotation Engine speed is not measured when the crankshaft rotates in reverse direction. In this case, an engine speed of 0 is provided to the behavior model.

Oversviews (Crank In)

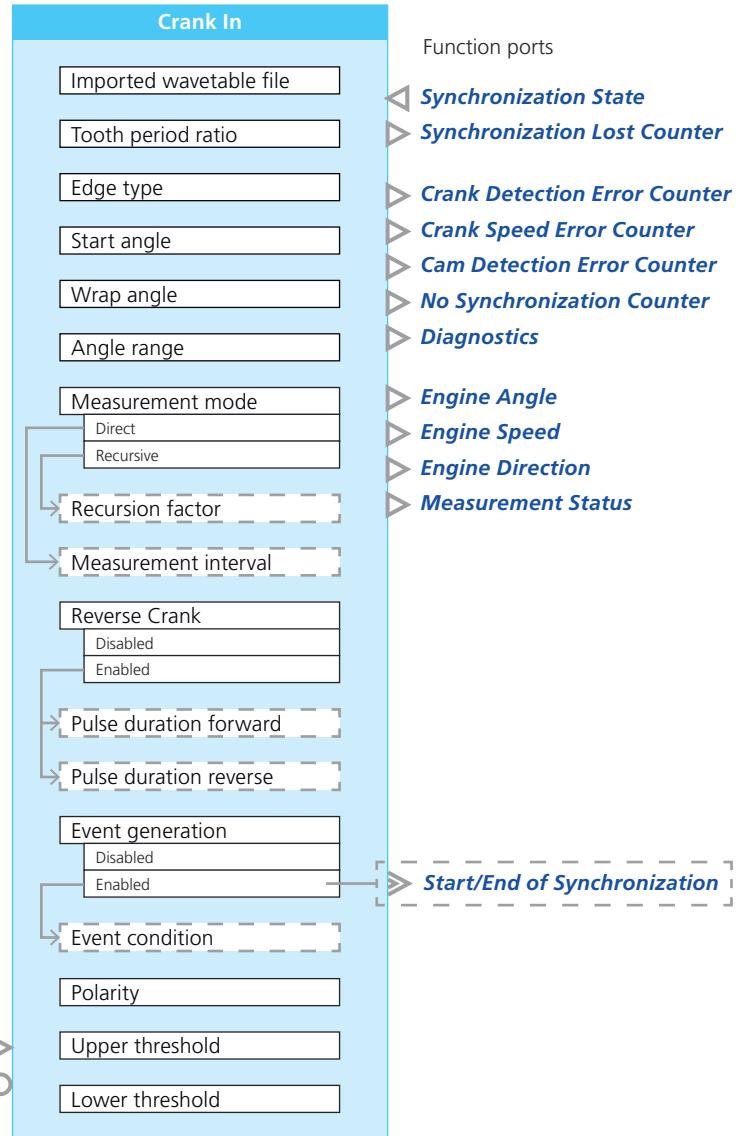
Where to go from here**Information in this section**

Overview of Ports and Basic Properties (Crank In)	573
Overview of Tunable Properties (Crank In)	578

Overview of Ports and Basic Properties (Crank In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Synchronization State

This function import enables/disables crankshaft synchronization from within the behavior model.

Value range

- 1: Enabled

Crankshaft synchronization can be started.

	<ul style="list-style-type: none"> ▪ 0: Disabled Crankshaft synchronization cannot be started. An established crankshaft synchronization is terminated.
Dependencies	—

Synchronization Lost Counter

This function outport writes the value of the *synchronization lost* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>synchronization lost</i> counter is incremented each time crankshaft synchronization is lost. ▪ The counter is reset each time the real-time application is stopped and then started again.
Dependencies	—

Crank Detection Error Counter

This function outport writes the value of the *crank detection error* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>crank detection error</i> counter is incremented each time the following condition is met: A detected edge in the crankshaft signal is outside the range determined via the Tooth period ratio property. ▪ The counter is reset each time the real-time application is stopped and then started again.
Dependencies	—

Crank Speed Error Counter

This function outport writes the value of the *crank speed error* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>crank speed error</i> counter is incremented each time the measured engine speed exceeds 20,000 rpm. ▪ The counter is reset each time the real-time application is stopped and then started again.
Dependencies	—

Cam Detection Error Counter

This function outport writes the value of the *cam detection error* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>cam detection error</i> counter is incremented each time following condition is met:
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>The angular distance of specified edges in a camshaft signal is smaller than twice the camshaft tolerance value of the related Cam In function block.</p> <ul style="list-style-type: none"> ▪ The counter is reset each time the real-time application is stopped and then started again.
Dependencies	—

No Synchronization Counter

This function outport writes the value of the *no synchronization* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>no synchronization</i> counter is incremented each time synchronization was not successful within one engine cycle. ▪ The counter is reset each time the real-time application is stopped and then started again.
Dependencies	—

Diagnostics

This function outport writes a value greater 0 (zero) to the behavior model, if one or more of the function block's error counters are incremented.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 31 ▪ 0: Indicates that no error counter was incremented since the function port was previously read. ▪ 1 ... 31: Indicate incremented error counters as follows: <ul style="list-style-type: none"> ▪ 1 (Bit 0 = 1): The <i>synchronization lost</i> counter is incremented. ▪ 2 (Bit 1 = 1): The <i>no synchronization</i> counter is incremented. ▪ 4 (Bit 2 = 1): The <i>crank speed error</i> counter is incremented. ▪ 8 (Bit 3 = 1): The <i>crank direction error</i> counter is incremented. ▪ 16 (Bit 4 = 1): The <i>cam detection error</i> counter is incremented. ▪ The error counters relate to specific bits as shown in the list. Combinations of bits are set when more than one error counter is incremented. To identify particular counters, you must evaluate the value of the <i>Diagnostics</i> function port in the behavior model. ▪ A bit is reset each time its corresponding error does no longer exist and the <i>Diagnostics</i> function port is read.
Dependencies	—

Engine Angle

This function outport provides the evaluated engine position angle to the behavior model.

Value range	<ul style="list-style-type: none"> The value range of this outport depends on the settings of the Wrap angle and Angle range properties as follows: [Wrap angle - Angle range ... Angle range] If the Angle range property is set to 360°, the wrap angle is internally saturated to 360°. The saturated wrap angle value is then used to calculate the lower limit of the value range.
Dependencies	—

Engine Speed

This function outport provides the evaluated engine speed to the behavior model.

Value range	0 ... 20,000 rpm
Dependencies	—

Engine Direction

This function outport provides the evaluated rotational direction of the engine to the behavior model.

Value range	<ul style="list-style-type: none"> 0: Undefined The engine's rotational direction is not evaluated, for example, because the engine speed is too low. 1: Forward The engine rotates in forward direction. -1: Reverse (Supported only if the Reverse Crank property is set to Enabled) The engine rotates in reverse direction.
Dependencies	—

Measurement Status

This function outport provides the current status of camshaft measurements and crankshaft measurements to the behavior model.

Value range	<ul style="list-style-type: none"> 0: Idle Measurement of engine speed, position, and rotational direction is initializing. Crankshaft and camshaft signals are evaluated. 1: Synchronization started Engine speed and rotational direction are available, crankshaft synchronization is started. 2: Synchronization completed (outputs disabled) Engine speed is below 5 rpm. Pulses of the Engine Angular Pulse Out function block and engine-angle-related I/O events cannot be generated, knock signals are not measured.
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 3: Synchronization completed (outputs enabled) Engine speed exceeds 10 rpm. Pulses and I/O events can be generated, knock signals can be measured. ▪ 4: No cam A Cam In function block is not used for the application and the Angle range property is set to 720°. Only engine speed is measured. ▪ -1: Synchronization error Crankshaft synchronization is stopped. Generation of ignition pulses, injection pulses, and engine angle related pulses as well as engine-angle-related I/O events is disabled, knock signals are not measured. Crankshaft synchronization can be initialized again by performing one of the following steps: <ul style="list-style-type: none"> ▪ Changing the state at the Synchronization State function port from Disabled to Enabled from in the behavior model. ▪ Stopping the real-time application and starting it again.
Dependencies	—

Start/End of Synchronization

This event port provides an I/O event each time the crankshaft synchronization (synchronized state) is reached and/or lost.

You have to specify, whether the start and/or end of the synchronized state are relevant for triggering I/O events (via the Event condition property).

Value range	—
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank In) on page 586.
Dependencies	—

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Crank In) on page 586.
Dependencies	—

Overview of Tunable Properties (Crank In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Pulse duration forward	✓	-
Pulse duration reverse	✓	-
Tooth period ratio	✓	-
Edge type	✓	-
Measurement mode	✓	-
Measurement interval	✓	-
Recursion factor	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Polarity	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Crank In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Crank In).....	579
Configuring Standard Features (Crank In).....	585

Configuring the Basic Functionality (Crank In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the crankshaft wheel characteristics.
- Specifying the evaluated (relevant) edge type.
- Specifying the mode of engine speed evaluation.
- Defining a time frame as a filter for valid edges.
- Specifying evaluation of rotational direction.
- Specifying the start angle for the engine cycle.
- Specifying the maximum engine angle provided to the behavior model.
- Specifying upper and lower trigger thresholds.
- Specifying signal polarity.
- Specifying the angle range of the master APU.
- Providing I/O events.

Specifying the crankshaft wheel characteristics

The specification of the crankshaft wheel (number of teeth and gaps, etc.) is based on wavetable files. You have to import a wavetable file that specifies the crankshaft wheel in use. The file format is CSV.

CSV files must meet the following basic requirements:

- ASCII file
- Entries are separated by a comma (",")
- New line at the end of a file

Wavetable files for the Crank in function block must also meet the following additional requirements:

- The first line has six numerical entries, commas are followed by a blank, and the line is terminated by a semicolon.

For example, the line *n, m, 60, 2, 1200, 2800;* means the following:

n	m	60	2	1200	2800
For internal use		Number of teeth (incl. missing) of the crank wheel.	Number of gaps of the crank wheel.	Lower limit of the value range of the Tooth period ratio property, specified in thousandths. The entry 1200 specifies a lower limit of 1.2.	Upper limit of the value range of the Tooth period ratio property, specified in thousandths. The entry 2800 specifies an upper limit of 2.8 .

- The second line has 7200 entries of 0 or 1 representing the teeth and gaps of the crank wheel at a resolution of 0.05°. The line has no blanks and is not terminated by a semicolon.

Wavetable files are supplied with the ConfigurationDesk EngineControlDemo demo project and located in the \Documents\dSPACE\ConfigurationDesk\<VersionNumber>\EngineControlDemo\EngineControl\Wavetables\ folder.

The supplied wavetable files are named as follows:

`CrankInWavetable_<Number of teeth (incl. missing)>_<Number of gaps>_<Number of missing teeth per gap>.csv.`

The folder with the wavetable files also includes a MATLAB script file (`CrankInWtCreate.m`) that lets you generate more wavetable files according to your requirements.

Specifying the relevant edge type

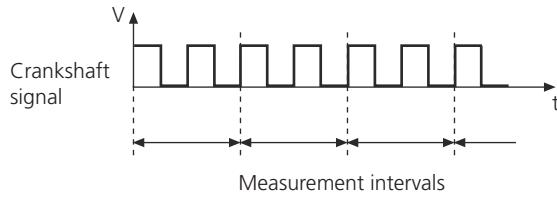
You have to specify whether the falling or the rising edges of the input signal are used to calculate engine speed, angle position, and rotational direction (relevant edges).

Limitation You cannot specify the relevant edge type for the Digital In 10 channel type. The Digital In 10 channel type supports only detection of falling edges.

Specifying the mode of engine speed measurement

You have to specify the mode for engine speed measurement and calculation:

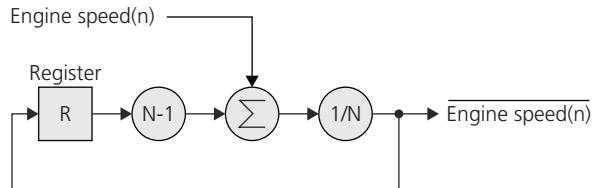
Direct mode To calculate the engine speed, the function block measures the time it takes the crankshaft to rotate over a user-defined angle range. You have to specify the measurement interval for which averaged speed measurements are consecutively performed. It is specified as the number of signal edges, which represent the angle range. Measurement intervals do not overlap. Refer to the following illustration:



For each measurement interval, the engine speed is calculated as revolutions per minute (rpm) and the result is provided to the behavior model via the Engine Speed function port.

Recursive mode Speed values are calculated from a *currently measured* speed value (measurement interval =1) and the *previously calculated* speed value. You have to specify the recursion weighting factor that determines how much the previous value contributes for the current calculation. The following equation is used to calculate the engine speed:

$$\text{Engine speed}(n) = \frac{\text{Engine speed}(n-1) * (N-1) + \text{Engine speed}(n)}{N}$$



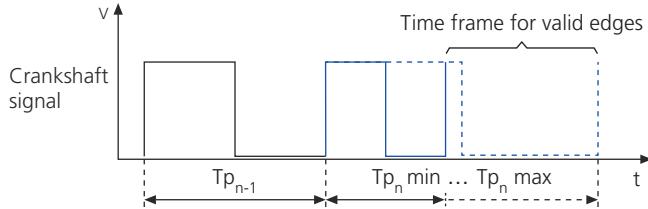
The expressions in the equation mean the following:

- $\bar{E}_{\text{ngine speed}}(n)$: Result of current engine speed calculation
- $\bar{E}_{\text{ngine speed}}(n-1)$: Result of previous engine speed calculation
- $E_{\text{ngine speed}}(n)$: Result of current engine speed measurement
- N : Specified recursion factor

For the recursive mode, an engine speed value is provided for each relevant edge.

Defining a time frame as a filter for valid edges

You have to specify the tooth period ratio to determine how much the duration of a tooth period (Tp_n) is allowed to increase/decrease during speed measurement with respect to the previous tooth period (Tp_{n-1}). A tooth period is the time interval between consecutive relevant edges in the crankshaft signal. The duration of tooth periods changes when engine speed changes. The following illustration shows the allowed minimum/maximum durations of Tp_n for a tooth period ratio of 1.5:



The interval given by $Tp_n \text{ min}$ and $Tp_n \text{ max}$ is the time frame in which the next edge must be detected to be valid. Edges detected outside this time frame are considered to be caused by defects such as a broken wire or signal distortion. The limiting values of the time frame are calculated as follows:

- $Tp_n \text{ min} = (\varphi_n / \varphi_{n-1}) \cdot Tp_{n-1} / \text{Tooth period ratio}$
- $Tp_n \text{ max} = (\varphi_n / \varphi_{n-1}) \cdot Tp_{n-1} \cdot \text{Tooth period ratio}$

φ_n and φ_{n-1} are the angular distances between adjacent relevant edges. They relate to Tp_n and Tp_{n-1} . Note that $(\varphi_n / \varphi_{n-1})$ also covers the conditions at crankshaft gaps and for crankshaft teeth of different width.

Note

The range of values that you can specify at the Tooth period ratio property depends on the crankshaft wheel of the piston engine. It is provided with the crankshaft wavetable file.

Effects of detecting invalid edges Detecting edges outside the specified time frame has the following effects:

- If crankshaft synchronization is not yet achieved:
 1. The speed measurement is invalid.
The *crank detection error* counter is incremented.
 2. The crankshaft synchronization is restarted and the value provided at the Measurement Status function port is temporarily set to 0.

If crankshaft synchronization is already achieved:

1. The speed measurement is valid.
The *crank detection error* counter is incremented.
2. Edges detected outside the specified time frame are continuously counted.
This is independent from the *crank detection error* counter and the result is not provided to the behavior model. Counting restarts from 0 each time no edge is detected outside the specified time frame for at least 720°.
If more than three invalid edges are consecutively counted, the crankshaft synchronization is restarted and the value provided at the **Measurement Status** function port is temporarily set to 0.

Specifying evaluation of rotational direction

You have to activate the reverse crank mode if your application requires information on the rotational direction of the crankshaft. Therefore, you must use an active crankshaft sensor that generates pulses with different durations for forward and reverse rotation. You have to specify the pulse durations for both rotational directions. Refer to the datasheet of the connected sensor.

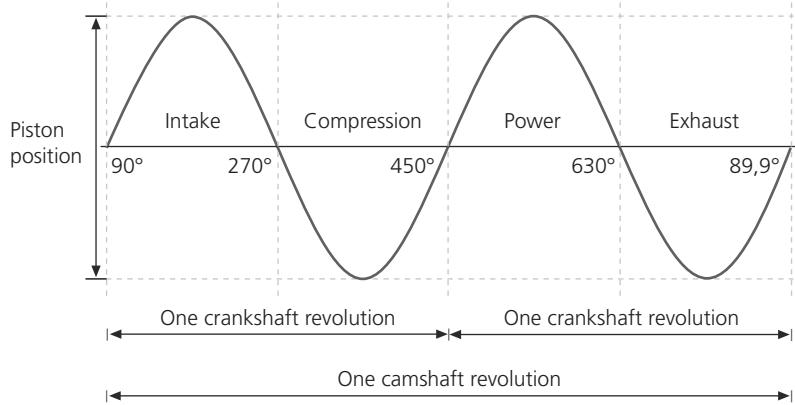
Reverse rotation has the following effects::

- The crankshaft remains synchronized if it was synchronized before rotational direction changed to reverse. Crankshaft synchronization is not started for reverse rotation.
- If the crankshaft is synchronized, engine angles are evaluated and provided to the behavior model for each detected edge. Angle values are not interpolated for reverse rotation.
- Engine speed is not evaluated and 0 (zero) is provided to the behavior model via the **Engine Speed** function port.

If information on rotational direction of the crankshaft is not required, you can deactivate the reverse crank mode. In this case, active or passive sensors can be used.

Specifying the start angle for the engine cycle

By default, the 0° position of engine position measurement is internally defined to match the first wavetable data point. All angle values relate to the 0° position. You can change this relationship by specifying a start angle other than 0°, for example, because the 0° position of the engine cycle does not match the first wavetable data point. As a result, the angular scale for the engine cycle is shifted by the specified start angle. This must be considered for the specification of all absolute angle values, for example, camshaft angles, TDCs, and angle-based I/O events. The following illustration shows a four-stroke engine cycle with a start angle of 90°.

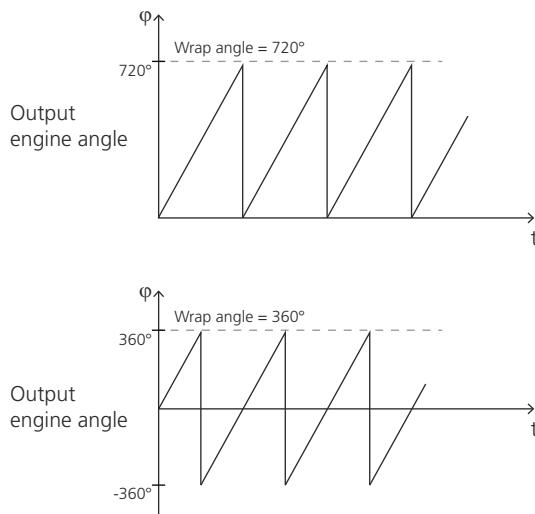
**Note**

For the wavetables that are provided with the ConfigurationDesk EngineControlDemo demo project, the first data point relates to the first rising edge after a gap.

Specifying the maximum engine angle value provided to the behavior model

The maximum engine angle value provided to the behavior model is specified at the **Wrap angle** property. You can change the default value of 720° to a value between 0° and 720°. The setting of the **Wrap angle** property affects exclusively the engine angle value provided to the behavior model.

When the measured engine angle is equal or greater than the specified wrap angle, the value specified at the **Angle range** property is subtracted from the measured engine angle. The result is provided to the behavior model. The following illustration shows the angle values of a four-stroke engine output to the behavior model for specified wrap angles of 720° and 360°:

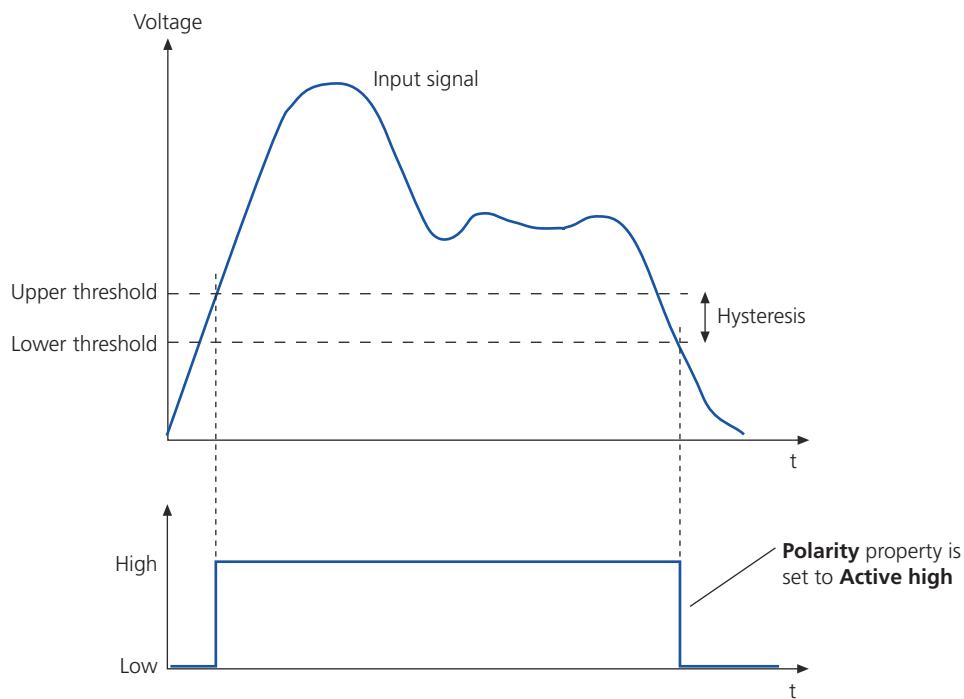


Note

If the Angle range property is set to 360°, the wrap angle is internally saturated to 360°. This prevents the angle value provided to the behavior model from exceeding the angle range of the engine.

Specifying upper and lower trigger thresholds

You have to specify lower and upper threshold values to transform the input signal into binary values. A rising edge of the input signal must exceed the upper threshold to be detected as high and a falling edge of the input signal must fall below the lower threshold to be detected as low. Note that you can adjust the hysteresis, which equals the difference of upper and lower threshold values.



The value range of the thresholds depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Crank In\)](#) on page 586.

Limitation The threshold value is fixed to 0 V without hysteresis and cannot be changed for the Digital In 10 channel type.

Specifying polarity of the signal

To adapt the real-time hardware to signals of different polarity without having to use inverters on the hardware, you can switch the **Polarity** of the input signal.

- Active high

The higher voltage value of the signal represents a binary 1 and the lower value represents a binary 0.

- Active low

The lower voltage value of the signal represents a binary 1 and the higher value represents a binary 0.

The setting is also relevant for the setting of the Edge type property.

Limitation The polarity setting is not supported for the Digital In 10 channel type.

Specifying the angle range of the master APU

The function block provides the master APU for all the engine control functionalities. You have to specify the angle range of the master APU according to the engine cycle of the controlled (real) piston engine. Select 720° for four-stroke engines or 360° for two-stroke engines.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

You have to specify at the Event condition property, if an event is generated when crankshaft synchronization is reached, crankshaft synchronization is lost, or when synchronization is reached and lost.

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Synchronization property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Crank In)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The electrical interface of the Crank In function block does not provide configurable standard features.

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital In 9](#) on page 1573
- [Digital In 10](#) on page 1574

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Crank In)

MicroAutoBox III Hardware Dependencies (Crank In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital In 9	Digital In 10
Hardware		DS1554 Engine Control I/O Module	
Event generation		✓	
Signal input	Edge frequency (-3 dB cutoff frequency)	320 kHz (typ.)	30 kHz (typ.)
	Input voltage range	-40 V ... +60 V	-60 V ... +60 V
	Input threshold voltage	<ul style="list-style-type: none"> ▪ -40 V ... +40 V ▪ Upper and lower threshold configurable separately 	<ul style="list-style-type: none"> ▪ 0 V ▪ Not configurable, no hysteresis
	Recognized edge type	<ul style="list-style-type: none"> ▪ Rising ▪ Falling 	Falling
Circuit diagram		Digital In 9 on page 1573	Digital In 10 on page 1574
Required channels		1	

More hardware data**DS1554 Engine Control I/O Module**

For more board-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration !\[\]\(1c17a0f6eba94cbe9da3d8b72a26af97_img.jpg\)](#)).

Cam In

Where to go from here

Information in this section

Introduction (Cam In).....	587
Overviews (Cam In).....	590
Configuring the Function Block (Cam In).....	596
Hardware Dependencies (Cam In)	603

Introduction (Cam In)

Where to go from here

Information in this section

Introduction to the Function Block (Cam In).....	587
Basics on Camshaft Sensor Signals.....	588
Basics on Camshaft Phase Shift Measurement.....	589

Introduction to the Function Block (Cam In)

Function block purpose

The Cam In function block type lets you measure the phase shift angle between a camshaft and the coupled crankshaft. Cam In must be used with the Crank In function block for synchronizing with the provided master APU and to evaluate the current engine position.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

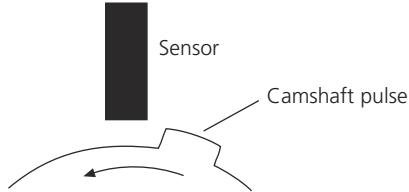


Main features	<p>These are the main features:</p> <ul style="list-style-type: none"> ▪ Supporting the synchronization of the angular processing unit with the crankshaft. ▪ Measuring camshaft phase shift values and providing them to the behavior model. ▪ Providing error and status information on phase measurements to the behavior model. ▪ Generating I/O events and providing them to the behavior model. ▪ Supporting active and passive camshaft sensors. 												
Supported channel types	<p>The Cam In function block supports the following channel types:</p> <table border="1"> <thead> <tr> <th></th> <th>SCALEXIO</th> <th colspan="2">MicroAutoBox III</th> </tr> </thead> <tbody> <tr> <td>Channel type</td> <td>–</td> <td>Digital In 9</td> <td>Digital In 10</td> </tr> <tr> <td>Hardware</td> <td>–</td> <td colspan="2">DS1554</td> </tr> </tbody> </table>		SCALEXIO	MicroAutoBox III		Channel type	–	Digital In 9	Digital In 10	Hardware	–	DS1554	
	SCALEXIO	MicroAutoBox III											
Channel type	–	Digital In 9	Digital In 10										
Hardware	–	DS1554											

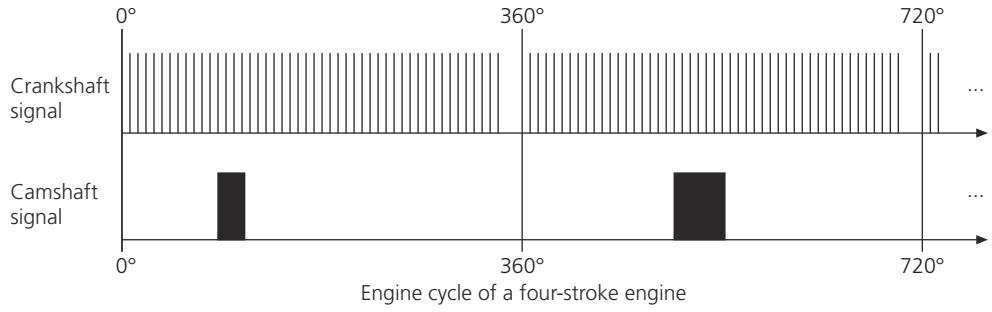
Basics on Camshaft Sensor Signals

Introduction	The camshaft signal describes the motion of the camshaft. It is used in combination with the crankshaft signal to identify the exact engine position and to measure camshaft phase shift.
---------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Camshaft sensor	An optical or magnetic sensor scans the radial surface of the rotating camshaft wheel that is flanged to the camshaft. The sensor detects the edges of the camshaft pulses located on the wheel.
------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



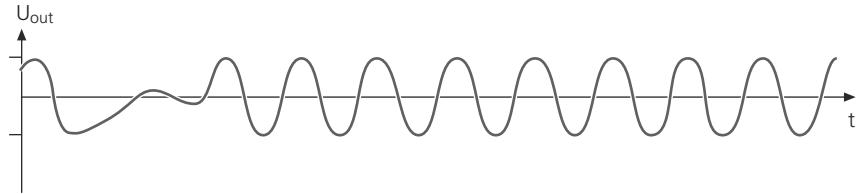
Camshaft pulses are specified by the angle values representing their edges. All angle values of the camshaft pulse specification relate to the crankshaft signal. The possible camshaft angle range for four-stroke engines is $0^\circ \dots 719.9^\circ$, as one camshaft revolution represents one engine cycle and corresponds to two crankshaft revolutions.



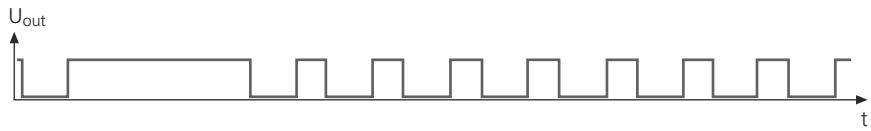
Signals of supported sensor types

Camshaft signals are either passive or active, depending on the type of sensor you use. The amplitude of the signal generated also depends on the sensor.

Passive sensors Generate an analog signal, for example, as shown in the following illustration:



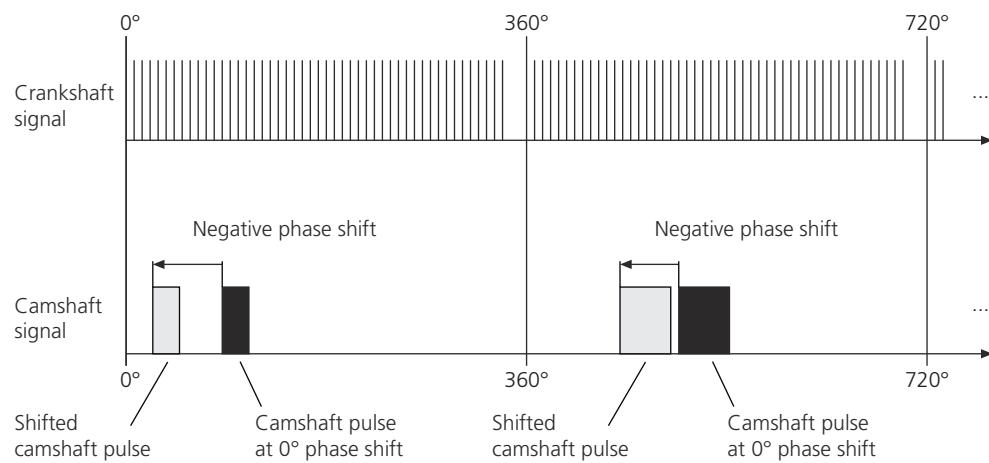
Active sensors Generate a digital signal, for example, a TTL signal as shown in the following illustration:



Basics on Camshaft Phase Shift Measurement

Definition of camshaft phase shift

Camshaft phase shift means an angular displacement of a camshaft relative to the coupled crankshaft, triggered by an external mechanism. The entire pulse pattern for the camshaft signal is shifted. A camshaft phase shift is commonly a function of speed and load. The following illustration shows a negative phase shift, i.e., the signal edges occur earlier than for the initial camshaft position.



Initialization of camshaft phase shift measurement

After phase shift measurement is initialized, the camshaft angle is measured and camshaft phase shift is evaluated for every detected edge of the camshaft signal. Initialization of the camshaft phase shift measurement cannot be performed stand-alone but depends also on synchronization of the coupled crankshaft. For more information on the synchronization mechanism, refer to [Basics on Synchronizing the Crankshaft](#) on page 566.

Oversviews (Cam In)

Where to go from here

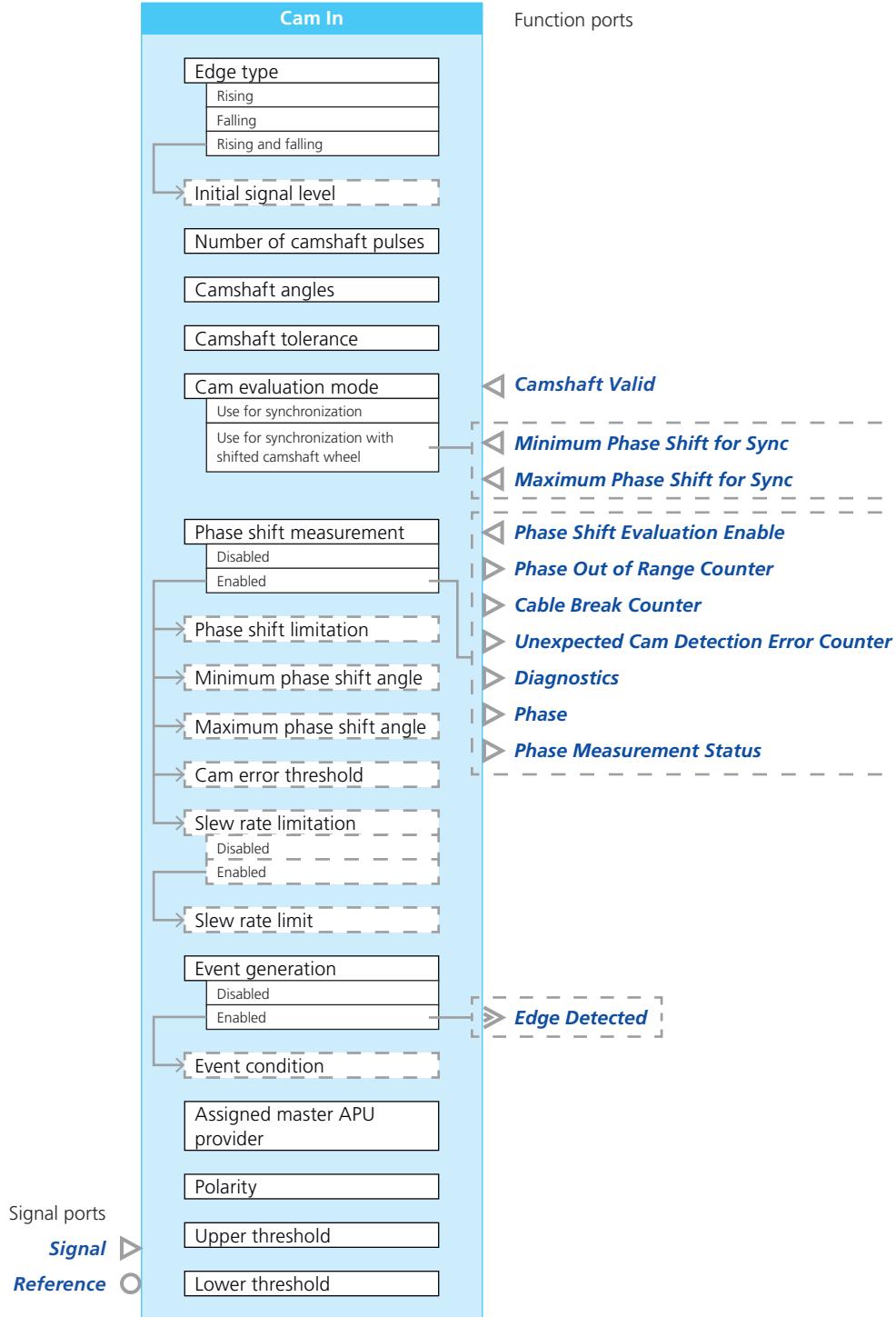
Information in this section

Overview of Ports and Basic Properties (Cam In).....	590
Overview of Tunable Properties (Cam In).....	596

Overview of Ports and Basic Properties (Cam In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Camshaft Valid**

This function import reads from within the behavior model, whether the camshaft signal is valid and can be used for synchronizing the coupled crankshaft.

Value range	<ul style="list-style-type: none"> ▪ 1: Valid The camshaft signal can be used for crankshaft synchronization. ▪ 0: Invalid The camshaft signal cannot be used for crankshaft synchronization. This value does not terminate an already established crankshaft synchronization.
Dependencies	—

Minimum Phase Shift for Sync

This function import reads the value of the minimum phase shift between camshaft and coupled crankshaft that is allowed during crankshaft synchronization from within the behavior model.

This function port is read only at the moment when the value at the Synchronization State function port of the assigned Crank In function block is changed from Disabled to Enabled. Therefore, the value at the Synchronization State function port must be changed each time an update of the Minimum Phase Shift for Sync function port is required.

Value range	<ul style="list-style-type: none"> ▪ -120° ... +120° ▪ The provided value is saturated to the value of the Minimum phase shift angle property.
Dependencies	Available only if the Cam evaluation mode property is set to Use for Synchronization with Shifted Camshaft Wheel.

Maximum Phase Shift for Sync

This function import reads the value of the maximum phase shift between camshaft and coupled crankshaft that is allowed during crankshaft synchronization from within the behavior model.

This function port is read only at the moment when the Synchronization State function port of the assigned Crank In function block is changed from Disabled to Enabled. Therefore, the value at the Synchronization State function port must be changed each time an update of the Maximum Phase Shift for Sync function is required.

Value range	<ul style="list-style-type: none"> ▪ -120° ... +120° ▪ The provided value is saturated to the value of the Maximum phase shift angle property.
Dependencies	Available only if the Cam evaluation mode property is set to Use for Synchronization with Shifted Camshaft Wheel.

Phase Shift Evaluation Enable

This function import enables/disables the evaluation of measured phase shift values from within the behavior model. The setting does not affect the crankshaft synchronization.

Value range	<ul style="list-style-type: none"> ▪ 1: Enabled Phase shift values are evaluated and output continuously to the behavior model. ▪ 0: Disabled The phase shift between the camshaft and the coupled crankshaft is not evaluated.
Dependencies	Available only if the Phase shift measurement property is set to Enabled.

Phase Out of Range Counter

This function outport writes the value of the *phase out of range* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>phase out of range</i> counter is incremented each time the measured phase shift value exceeds the range specified at the Minimum/Maximum phase angle properties by more than the value specified at the Camshaft tolerance property. ▪ The counter is reset each time the real-time application is stopped and then started again.
Dependencies	Available only if the Phase shift measurement property is set to Enabled.

Cable Break Counter

This function outport writes the value of the *cable break* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>cable break</i> counter is incremented each time the input signal from the camshaft did not change for three consecutive revolutions of the coupled crankshaft. ▪ The counter is reset each time the real-time application is stopped and then started again.
Dependencies	Available only if the Phase shift measurement property is set to Enabled.

Unexpected Cam Detection Error Counter

This function outport writes the value of the *unexpected cam detection error* counter to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 ▪ The <i>unexpected cam detection error</i> counter is incremented each time the following condition is met:
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>An edge is detected in the input signal that matches the specified edge type but is not specified at the Camshaft angles property. This might be caused, for example, by a noisy input signal.</p> <ul style="list-style-type: none"> The counter is reset each time the real-time application is stopped and then started again.
Dependencies	Available only if the Phase shift measurement property is set to Enabled.

Diagnostics

This function outport writes a value greater 0 (zero) to the behavior model if one or more of the function block's error counters are incremented.

Value range	<ul style="list-style-type: none"> 0 ... 7 0: Indicates that no error counter was incremented since the function port was previously read. 1 ... 7: Indicate incremented error counters as follows: <ul style="list-style-type: none"> 1 (Bit 0 = 1): The <i>unexpected cam detection</i> error counter is incremented. 2 (Bit 1 = 1): The <i>phase out of range</i> counter is incremented. 4 (Bit 2 = 1): The <i>cable break</i> counter is incremented. The error counters relate to specific bits as shown in the list. Combinations of bits are set when more than one error counter is incremented. To identify particular error counters, you must evaluate the value of the Diagnostics function port in the behavior model. A bit is reset when its corresponding error does no longer exist and the Diagnostics function port is read.
Dependencies	Available only if the Phase shift measurement property is set to Enabled.

Phase

This function outport provides the evaluated and possibly limited camshaft phase shift value to the behavior model.

Value range	<ul style="list-style-type: none"> -120° ... +120° The value range can be limited via phase shift limitation. Refer to Configuring the Basic Functionality (Cam In) on page 597.
Dependencies	Available only if the Phase shift measurement property is set to Enabled.

Phase Measurement Status

This function outport provides the status of phase measurements to the behavior model.

Value range	<ul style="list-style-type: none"> Idle (0): Phase shift measurement is initializing.
-------------	----------------------------------------------------------------------------------------------------------

	<p>The value 0° is written to the Phase function port.</p> <ul style="list-style-type: none"> ▪ Active (1): The camshaft signal is valid. The camshaft phase shift measurement is established. Current phase shift measurement results are written to the Phase function port. ▪ Error (-1): Evaluation of the camshaft signal is aborted because the phase shift measurement could not be initialized. The value 0° is written to the Phase function port. Initialization of the phase shift measurement restarts in the following cases: <ul style="list-style-type: none"> ▪ The state of the Phase Shift Evaluation Enable function port changes from Disabled to Enabled. ▪ The state of the Synchronization State function port of the related Crank In function block changes from Disabled to Enabled. This restarts all the related Cam In function blocks. ▪ The real-time application is stopped and then started again.
Dependencies	Available only if the Phase shift measurement property is set to Enabled.

Edge Detected

This event port provides an I/O event each time an edge is detected at the signal port and this edge matches the specified trigger condition.

You have to specify which type of edge of the input signal (rising and/or falling) is relevant for event generation (via the Event condition property).

Value range	—
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Cam In) on page 603.
Dependencies	—

Reference	This signal port is a reference port and provides the reference signal for the Signal signal port.
------------------	----------------------------------------------------------------------------------------------------

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Cam In) on page 603.
Dependencies	—

Overview of Tunable Properties (Cam In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Camshaft tolerance	✓	—
Maximum phase shift angle	✓	—
Minimum phase shift angle	✓	—
Function Ports		
Initial switch setting (Test Automation)	—	✓
Initial substitute value (Test Automation)	—	✓
Electrical Interface		
Polarity	✓	—

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Cam In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Cam In).....	597
Configuring Standard Features (Cam In).....	602

Configuring the Basic Functionality (Cam In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the characteristics of the connected camshaft wheel.
- Specifying the mode for crankshaft synchronization.
- Specifying phase shift measurement.
- Defining behavior for camshaft signal errors.
- Specifying upper and lower trigger thresholds.
- Specifying signal polarity.
- Assigning a master APU provider.
- Providing I/O events.

Specifying the characteristics of the connected camshaft wheel

You have to specify the characteristics of the connected camshaft wheel so that the camshaft can be synchronized and measured camshaft signals can be properly evaluated.

Number of pulses You have to specify the number of pulses of the connected camshaft wheel.

Edges for triggering measurements You have to specify whether the rising, falling, or rising *and* falling edges of the camshaft pulses are used for triggering measurements. For the Rising and falling setting you have to specify additionally, whether the initial signal level before the edge related to the first specified camshaft angle is expected to be low or high.

Limitation: You cannot specify the edge type for triggering measurements for the Digital In 10 channel type. The Digital In 10 channel type supports only falling edges.

Angle values of triggering edges You have to specify the angle values of the triggering edges for each pulse of the connected camshaft wheel.

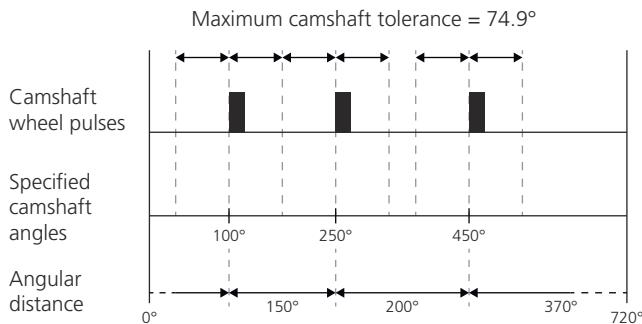
Camshaft wheel tolerances To consider, for example, production tolerances of camshaft wheels, you can specify an angular tolerance at the Camshaft tolerance property.

A specified tolerance value has the following effects:

- Crankshaft synchronization can be successfully performed only if the measured camshaft angles equal the specified angles plus/minus the specified camshaft tolerance. You have to take into account that larger camshaft tolerances might require larger angular intervals for the crankshaft synchronization process.
- During measurements, phase shift values provided to the behavior model are limited if both of the following conditions are met:
 - Phase shift limitation is enabled.
 - The measured values exceed the values specified at the Minimum/Maximum phase shift angle properties by more than the specified camshaft tolerance.

For more information on limiting phase shift values, refer to [Limiting provided phase shift values](#) on page 599.

The specified camshaft tolerance must be smaller than 50% of the shortest angular distance between all the specified camshaft angles. Otherwise, synchronizing the crankshaft might become impossible. For example, if the camshaft angles 100°, 250°, and 450° are specified, the shortest angular distance is 150° and the maximum camshaft tolerance is therefore 74.9°. Refer to the following illustration:



Specifying the crankshaft synchronization mode

You have to specify whether a phase shift of the camshaft is supported during crankshaft synchronization or not.

Synchronization without phase shift If crankshaft synchronization of your engine always starts with 0° camshaft phase shift, select **Use for synchronization** as the evaluation mode. For this setting, you have to ensure that the camshaft is not shifted during synchronization. Otherwise, the crankshaft synchronization might be impossible or faulty.

Synchronization with phase shift You can specify that a user-defined camshaft phase shift is considered during crankshaft synchronization by selecting **Use for synchronization with shifted camshaft wheel** as the evaluation mode. For this setting, the crankshaft synchronization can be performed when the phase shift of the camshaft wheel is within a user-defined range during synchronization. The allowed phase shift range is determined from within the behavior model via the **Maximum/Minimum Phase Shift for Sync** function ports.

Specifying phase shift measurement

You have to enable the phase shift measurement when your application requires current phase shift values, for example, to control the engine accordingly. If enabled, phase shift measurements are continuously performed and evaluated phase shift values, measurement status, and readings of error counters are written to the behavior model via the related function ports. If required, measured phase shift values can be first limited and then provided to the behavior model.

Valid phase shift range Unexpected edges in the camshaft signal, for example, due to noise, distort the measurement of the camshaft phase shift. To filter out invalid measurement values, you have to specify a range of valid phase

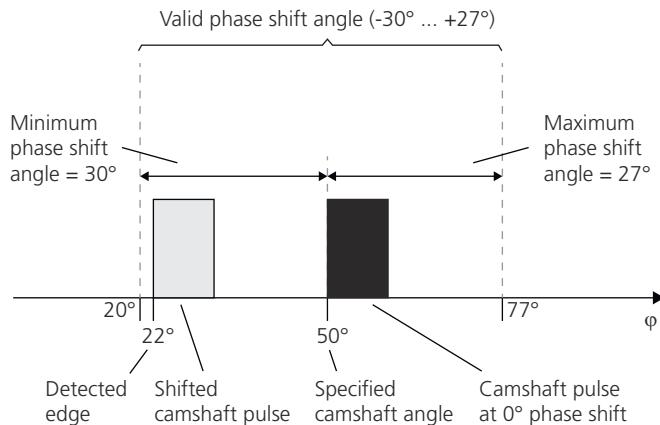
shift angles via the **Minimum phase shift angle** and the **Maximum phase shift angle** properties as follows:

- Both values can be positive or negative. The range of valid phase shift angles can therefore cover positive, negative, or positive and negative phase shift values.
- The specified minimum phase shift angle must be smaller than the specified maximum phase shift angle.
- Both values relate to the specified camshaft angles of the camshaft pulses.

Evaluating detected edges The system expects edges in the input signal according to the values specified at the Camshaft angles property. If an edge is detected, its measured angle value is compared to the next specified camshaft angle value. The edge is valid if it matches the following rules:

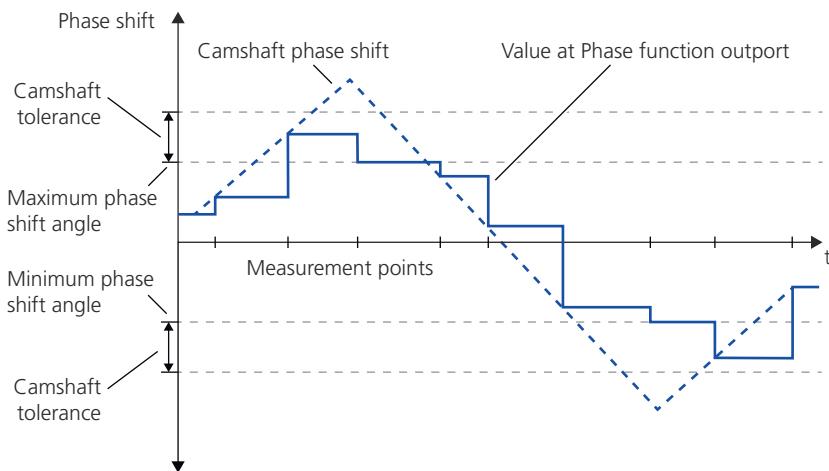
- *Measured angle \geq (specified camshaft angle + specified minimum phase shift angle)*
- *Measured angle \leq (specified camshaft angle + specified maximum phase shift angle)*

Example: The specified camshaft angle (rising edge) of a pulse is 50° and the specified minimum/maximum phase shift angles are -30° and $+27^\circ$. The rising edge for this pulse is expected between 20° and 77° of each engine cycle. The rising edge is detected at 22° and the phase shift angle is therefore $22^\circ - 50^\circ = -28^\circ$. The detected edge is valid.



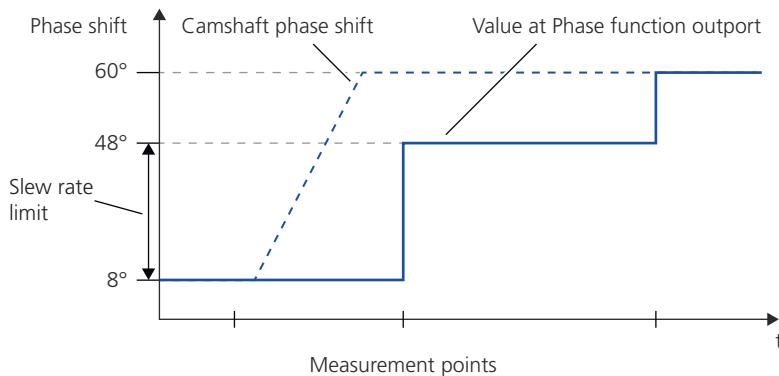
Each time an unexpected edge is detected in the input signal, the *unexpected cam detection error* counter is incremented.

Limiting provided phase shift values Measured phase shift values are invalid when they exceed the specified range of valid phase shift angles by more than the specified camshaft tolerance. If you want to filter invalid phase shift values to prevent them from being provided to the behavior model, you have to enable phase shift limitation. In this case, invalid values are limited to the specified minimum/maximum phase shift angles and then output to the behavior model. Refer to the following illustration:



Limiting the change of provided phase shift values For the provided camshaft phase shift value, you can limit the changes from measurement to measurement. Therefore, you have to enable slew rate limitation and specify the maximum allowed change at the Slew rate limit property.

For example, if Slew rate limit is set to 40° and camshaft phase shift rises from 8° to 60° between two measurements and then remains at 60°, the output values are 8°, 48°, 60°.



Defining behavior for camshaft signal errors

Errors in the camshaft signal that are detected during phase shift measurement (phase measurement status = 1) are continuously counted. The result is not provided to the behavior model. Counting restarts from 0 each time a number of consecutive valid edges is detected that is equal or greater than the number of specified camshaft angles.

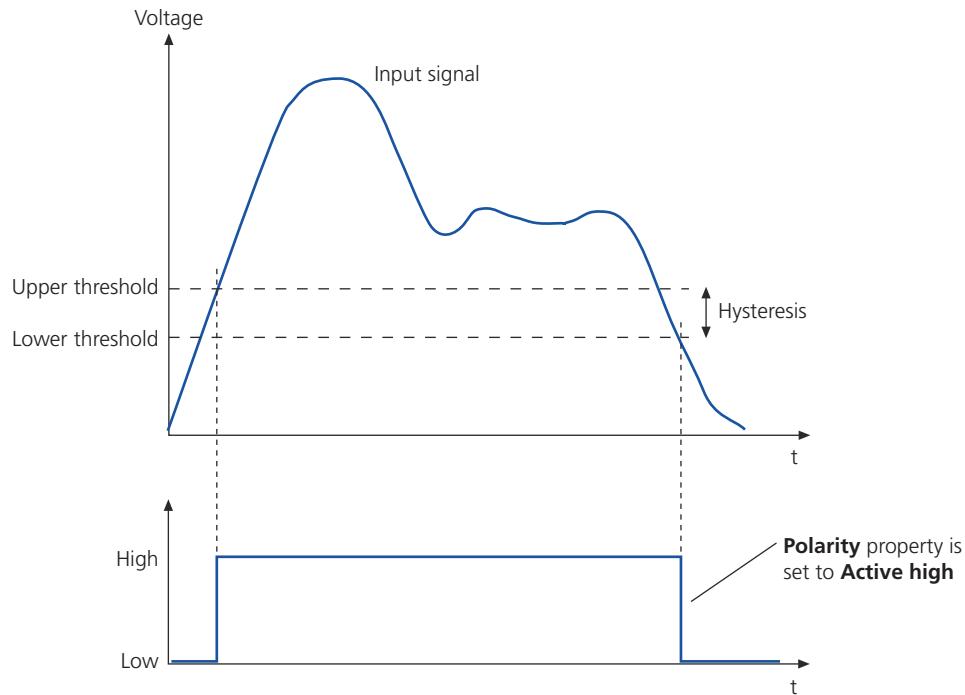
You can control the behavior for camshaft signal errors via the Cam error threshold property as follows:

- Specify the allowed number of camshaft signal errors at the Cam error threshold property. Counting more than the specified number of errors effects the following:
 - The phase shift measurement is stopped and initialized again.
 - The Phase Measurement Status function port is set to 0 until the initialization process is completed.

- Specify 0 (zero) if you want no phase shift measurement abort. In this case, errors in the camshaft signal are ignored.

Specifying upper and lower trigger thresholds

You have to specify lower and upper threshold values to transform the input signal into binary values. A rising edge of the input signal must exceed the upper threshold to be detected as high and a falling edge of the input signal must fall below the lower threshold to be detected as low. Note that you can adjust the hysteresis, which equals the difference of upper and lower threshold values.



The value range of the thresholds depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Cam In\)](#) on page 603.

Limitation The threshold value is fixed to 0 V without hysteresis and cannot be changed for the Digital In 10 channel type.

Specifying polarity of the signal

To adapt the real-time hardware to signals of different polarity without having to use inverters on the hardware, you can switch the Polarity of the input signal.

- Active high

The higher voltage value of the signal represents a binary 1 and the lower value represents a binary 0.

- Active low

The lower voltage value of the signal represents a binary 1 and the higher value represents a binary 0.

The setting is also relevant for the setting of the Event condition, Edge type, and Initial signal level properties.

Limitation The polarity setting is not supported for the Digital In 10 channel type.

Assigning a master APU provider For synchronizing the camshaft you have to reference a master angular processing unit (APU) that is provided by the Crank In function block.

Providing I/O events If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model. You have to specify at the Event condition property, whether the rising edges and/or the falling edges of the input signal are used to trigger event generation. Note that if you set the Polarity property to Active low, rising edges in the input signal are detected as falling edges and vice versa.

Note

- The Digital In 10 channel type supports only detection of *falling* edges.

To use the I/O events in the behavior model, you have to assign them to a task via the Edge Detected property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Cam In)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The electrical interface of the Cam In function block does not provide configurable standard features.

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital In 9](#) on page 1573
- [Digital In 10](#) on page 1574

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Cam In)

MicroAutoBox III Hardware Dependencies (Cam In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital In 9	Digital In 10
Hardware		DS1554 Engine Control I/O Module	
Event generation		✓	
Signal input	Edge frequency (-3 dB cutoff frequency)	320 kHz (typ.)	30 kHz (typ.)
	Input voltage range	-40 V ... +60 V	-60 V ... +60 V
	Input threshold voltage	<ul style="list-style-type: none"> ▪ -40 V ... +40 V ▪ Upper and lower threshold configurable separately 	<ul style="list-style-type: none"> ▪ 0 V ▪ Not configurable, no hysteresis
	Recognized edge type	<ul style="list-style-type: none"> ▪ Rising ▪ Falling 	Falling
Circuit diagram		Digital In 9 on page 1573	Digital In 10 on page 1574
Required channels		1	

More hardware data**DS1554 Engine Control I/O Module**

For more board-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Injection Out

Where to go from here

Information in this section

Introduction (Injection Out).....	604
Overviews (Injection Out).....	605
Configuring the Function Block (Injection Out).....	610
Hardware Dependencies (Injection Out)	615

Introduction (Injection Out)

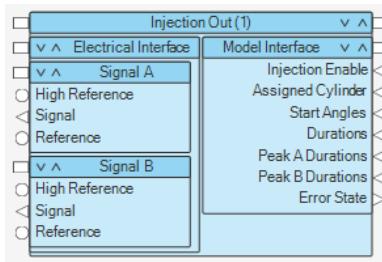
Introduction to the Function Block (Injection Out)

Function block purpose

The Injection Out function block type lets you generate injection pulses for a real piston engine.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating injection pulses according to values for pulse start angles and durations provided from within the behavior model.
- Generating injection pulses for direct injection or port injection.
- Limiting pulse durations to a specified maximum.
- Providing error status information to the behavior model.

Supported channel types

The Injection Out function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III	
Channel type	–	Digital Out 7	Digital In/Out 8
Hardware	–	DS1554	

Oversviews (Injection Out)

Where to go from here**Information in this section**

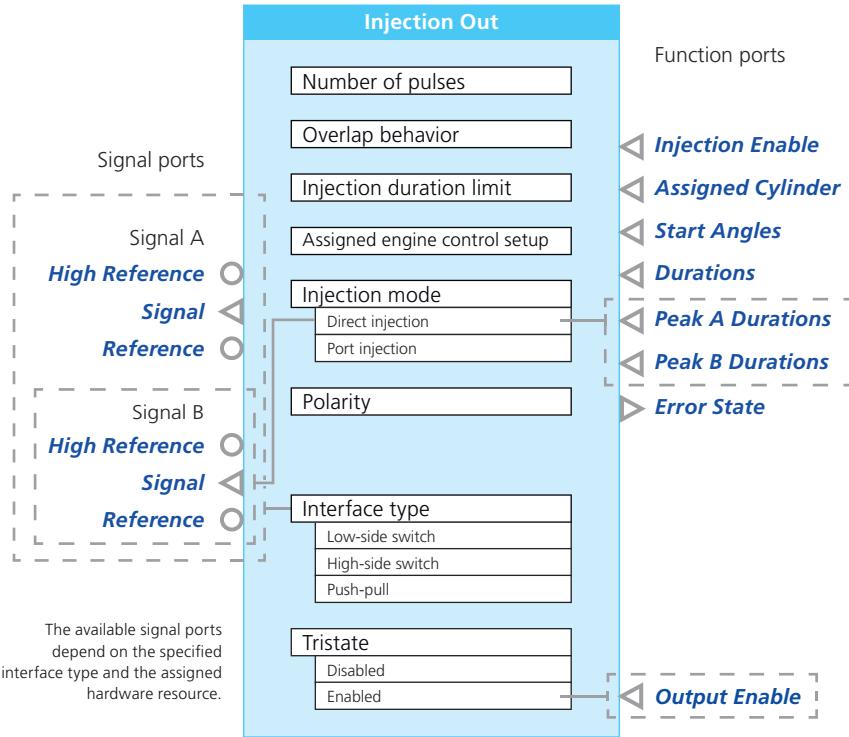
Overview of Ports and Basic Properties (Injection Out)..... 605

Overview of Tunable Properties (Injection Out)..... 609

Overview of Ports and Basic Properties (Injection Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Injection Enable

This function import enables the function block to generate injection pulses from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Setting this function port to Enabled takes effect only if the Injection Ignition Enable function port of the assigned Engine Control Setup function block is also set to Enabled.

Assigned Cylinder

This function import provides the number of the piston engine cylinder, for which the injection pulses are generated.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the setting of the Number of cylinders property of the assigned Engine Control Setup function block. If the provided value exceeds the value specified at the Number of cylinders property, the function block generates pulses for the cylinder with the highest number.
Dependencies	–

Start Angles

This function import reads a vector of angle values provided from within the behavior model. The values specify the start point of injection pulses related to the TDCs of assigned cylinders.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° for each entry in the vector. ▪ The effective range depends on the setting of the Angle range property of the assigned Engine Control Setup function block: <ul style="list-style-type: none"> ▪ 360°: -180° ... +180° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -250° is converted to +110°, +300° is converted to -60°. ▪ 720°: -360° ... +360° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -400° is converted to +320°, +830° is converted to +110°. ▪ The vector size depends on the number specified at the Number of pulses property. ▪ The vector entries must be provided in descending order, otherwise an error is indicated via the Error State function port.
Dependencies	-

Durations

This function import reads a vector of pulse duration values provided from within the behavior model. The values specify the length of injection pulses (in seconds).

Value range	<ul style="list-style-type: none"> ▪ 0 s ... 10 s for each entry in the vector. ▪ The vector size depends on the number specified at the Number of pulses property.
Dependencies	-

Peak A Durations

This function import reads a vector with peak A duration values provided from within the behavior model. The values specify the length of the peak A phase of injection pulses (in seconds).

Value range	<ul style="list-style-type: none"> ▪ 0 s ... 10 s for each entry in the vector. ▪ The vector size depends on the number specified at the Number of pulses property.
Dependencies	Available only if the Injection mode property is set to Direct injection.

Peak B Durations	<p>This function import reads a vector with peak B duration values provided from within the behavior model. The values specify the length of the peak B phase of injection pulses (in seconds).</p> <table border="1"> <tr> <td>Value range</td><td> <ul style="list-style-type: none"> ▪ 0 s ... 10 s for each entry in the vector. ▪ The vector size depends on the number specified at the Number of pulses property. </td></tr> <tr> <td>Dependencies</td><td>Available only if the Injection mode property is set to Direct injection.</td></tr> </table>	Value range	<ul style="list-style-type: none"> ▪ 0 s ... 10 s for each entry in the vector. ▪ The vector size depends on the number specified at the Number of pulses property. 	Dependencies	Available only if the Injection mode property is set to Direct injection.
Value range	<ul style="list-style-type: none"> ▪ 0 s ... 10 s for each entry in the vector. ▪ The vector size depends on the number specified at the Number of pulses property. 				
Dependencies	Available only if the Injection mode property is set to Direct injection.				
Error State	<p>This function outport writes status information on errors during pulse generation to the behavior model.</p> <table border="1"> <tr> <td>Value range</td><td> <ul style="list-style-type: none"> ▪ 0, 2, 4, 6 ▪ The value of the function port represents the following errors: <ul style="list-style-type: none"> ▪ 0: No errors occurred during pulse generation. ▪ 2: At least one of the following error conditions occurred: <ul style="list-style-type: none"> ▪ The angles for the injection pulses are not provided in a descending order. ▪ An injection pulse completely covers another one (e.g., first pulse 0° ... 100°, second pulse 20° ... 80°). ▪ The distance between two injection start or end angles is smaller than 0.1° engine cycle angles. ▪ More than two consecutive injection pulses overlap. ▪ 4: An injection pulse was limited to the value specified at the Injection duration limit property. ▪ 6: The errors indicated by values of 2 and 4 occurred at the same time. </td></tr> <tr> <td>Dependencies</td><td>–</td></tr> </table>	Value range	<ul style="list-style-type: none"> ▪ 0, 2, 4, 6 ▪ The value of the function port represents the following errors: <ul style="list-style-type: none"> ▪ 0: No errors occurred during pulse generation. ▪ 2: At least one of the following error conditions occurred: <ul style="list-style-type: none"> ▪ The angles for the injection pulses are not provided in a descending order. ▪ An injection pulse completely covers another one (e.g., first pulse 0° ... 100°, second pulse 20° ... 80°). ▪ The distance between two injection start or end angles is smaller than 0.1° engine cycle angles. ▪ More than two consecutive injection pulses overlap. ▪ 4: An injection pulse was limited to the value specified at the Injection duration limit property. ▪ 6: The errors indicated by values of 2 and 4 occurred at the same time. 	Dependencies	–
Value range	<ul style="list-style-type: none"> ▪ 0, 2, 4, 6 ▪ The value of the function port represents the following errors: <ul style="list-style-type: none"> ▪ 0: No errors occurred during pulse generation. ▪ 2: At least one of the following error conditions occurred: <ul style="list-style-type: none"> ▪ The angles for the injection pulses are not provided in a descending order. ▪ An injection pulse completely covers another one (e.g., first pulse 0° ... 100°, second pulse 20° ... 80°). ▪ The distance between two injection start or end angles is smaller than 0.1° engine cycle angles. ▪ More than two consecutive injection pulses overlap. ▪ 4: An injection pulse was limited to the value specified at the Injection duration limit property. ▪ 6: The errors indicated by values of 2 and 4 occurred at the same time. 				
Dependencies	–				
Output Enable	<p>This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.</p> <table border="1"> <tr> <td>Value range</td><td> <ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled </td></tr> <tr> <td>Dependencies</td><td>Available only if the Tristate property is set to Enabled.</td></tr> </table>	Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled 	Dependencies	Available only if the Tristate property is set to Enabled.
Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled 				
Dependencies	Available only if the Tristate property is set to Enabled.				
High Reference	<p>This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs operate as high-side switches or in push-pull configuration.</p>				

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection Out) on page 615.
Dependencies	Available only if the port is supported by the settings of the Interface type and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection Out) on page 615.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the **Signal** signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Injection Out) on page 615.
Dependencies	Available only if the port is supported by the settings of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Injection Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Overlap behavior	✓	–
Injection duration limit	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Electrical Interface Polarity	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Injection Out)

Where to go from here

Information in this section

- | | |
|----------------------------------------------------------|-----|
| Configuring the Basic Functionality (Injection Out)..... | 610 |
| Configuring Standard Features (Injection Out)..... | 614 |

Configuring the Basic Functionality (Injection Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the number of injection pulses to be generated.
- Specifying injection pulse characteristics.
- Specifying pulse overlap behavior.
- Specifying injection pulse limiting.
- Assigning an Engine Control Setup function block.

Specifying the number of injection pulses to be generated

You can specify a single injection pulse or a sequence of up to six injection pulses to be generated per engine cycle at the Number of pulses property.

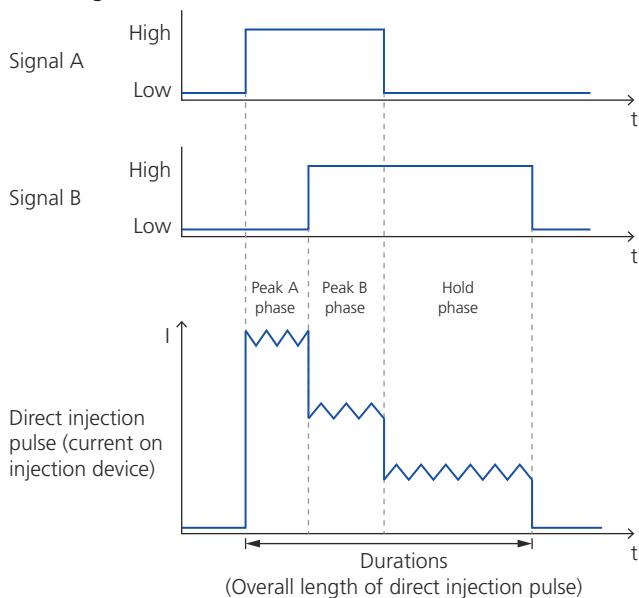
Specifying injection pulse characteristics

The function block can generate injection pulses for the following fuel injection modes:

- **Port injection:** Suitable for piston engines using injection devices commonly for more than one cylinder.
- **Direct injection:** Suitable for piston engines using individual injection devices for each of their cylinders.

You have to specify the injection mode according to the piston engine's construction so the function block generates appropriate injection pulses.

Pulses for direct injection Direct injection pulses are specified by values for start angle and durations provided via the related function ports from the behavior model. They consist of up to three phases (peak A, peak B, hold). The function block generates two pulses that encode the phases of the injection pulses and are output via separate channels (Signal A, Signal B). Refer to the following illustration:



The length of the peak A and peak B phases are specified by the values at the Peak A Durations and Peak B Durations function ports.

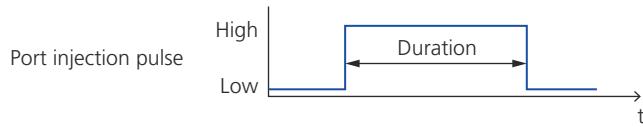
The value at the Durations function port specifies the overall length of an injection pulse and determines the length of the hold phase, peak B phase, and peak A phase as follows:

- For Durations greater than the sum of Peak A Durations and Peak B Durations, the length of the hold phase is equal to Durations minus the sum of Peak A Durations and Peak B Durations.
- For Durations equal to the sum of Peak A Durations and Peak B Durations, the length of the hold phase is 0 (zero).
- For Durations less than the sum of Peak A Durations and Peak B Durations, the peak B phase is shortened until the overall length of the injection pulse matches the Durations value. If shortening of the peak B phase is not sufficient, the peak A phase is shortened next.

Tip

The PS-DINJ 2/1 (DS1664) direct-injection driver module provides control inputs, to which Signal A and Signal B can be directly connected. For more information on the module, refer to [PS-DINJ 2/1 Module \(RapidPro System Hardware Reference\)](#).

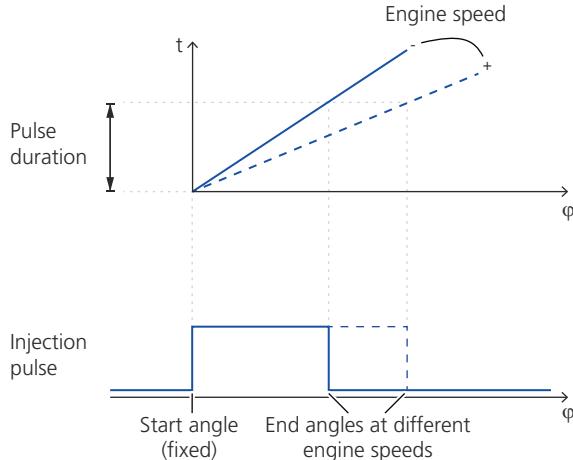
Pulses for port injection Port injection pulses are specified by values for start angle and duration provided via the related function ports from the behavior model. Port injection pulses consist of a single phase as shown in the following illustration:



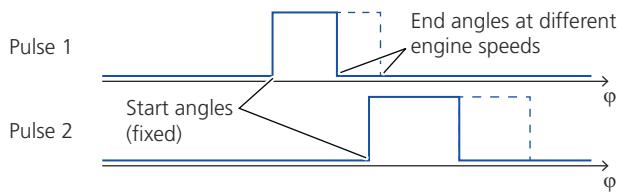
Port injection pulses are output via a single channel (Signal).

Specifying pulse overlap behavior

Overlap conditions Overlapping of injection pulses can be caused directly by the values for start angle and pulse duration provided from within the behavior model. Furthermore, pulses might overlap when engine speed increases. Refer to the following illustration:



With increasing engine speed, the end angle moves towards the start angle of the next pulse and both pulses might overlap. Refer to the following illustration:



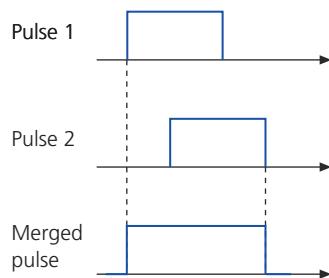
Overlap handling You have to specify how the function block handles overlapping injection pulses:

- **Merge:**

Two overlapping pulses are merged to a new pulse as follows:

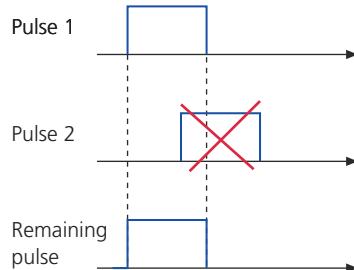
- The start angle of the first started pulse becomes the new start angle.
- The end angle of the last ending pulse becomes the new end angle.

Refer to the following illustration:



- **Remove:**

A pulse is removed if it starts while another one is active. Refer to the following illustration:



Note

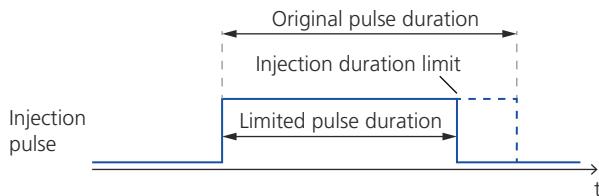
The following cases might lead to incorrect pulse generation:

- An injection pulse completely covers another one (e.g., first pulse 0° ... 100°, second pulse 20° ... 80°).
- More than two consecutive injection pulses overlap.

If any of these cases occurs, this is indicated as error and provided to the behavior model via the Error State function port.

- Specifying injection pulse limiting**
- Depending on your application, you might have to specify a maximum pulse duration. Without limiting, pulses might exceed maximum values for the following reasons:
- Duration values provided from within the behavior model are greater than the specified maximum.
 - Overlapping pulses are merged to a pulse with a duration that exceeds the specified maximum.

Pulses are cut off by the end of the interval specified at the **Injection duration limit** property. Refer to the following illustration:



Note

If a pulse is cut off, this is indicated as error and provided to the behavior model via the **Error State** function port.

- Assigning an Engine Control Setup function block**
- You have to assign an Engine Control Setup function block that provides data about the controlled piston engine (number of cylinders, TDCs, etc.) and references a master APU provider.

Configuring Standard Features (Injection Out)

- Scope**
- Each function block offers properties for configuring standard features of the electrical interface and the model interface.

- Electrical interface**
- The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the **Conflicts Viewer**. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 7	Digital In/Out 8	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–

Configuration Feature	Digital Out 7	Digital In/Out 8	Further Information
	<ul style="list-style-type: none"> ▪ Low-side switch ▪ Push-pull 		
Polarity of output signals	✓	✓	–
High impedance (tristate)	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.		–
Channel multiplication	–	–	–
Signal Ports			
Trigger level of electronic fuse	–	–	–
Failure simulation support	–	–	–

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 7](#) on page 1580
- [Digital In/Out 8](#) on page 1589

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Injection Out)

MicroAutoBox III Hardware Dependencies (Injection Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 7	Digital In/Out 8
Hardware	DS1554 Engine Control I/O Module	
Event generation	–	

Channel Type	Digital Out 7	Digital In/Out 8
Minimum output pulse width	<ul style="list-style-type: none"> ▪ High: 700 ns ▪ Low: 200 ns 	12.5 ns
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	The outputs can be set to high impedance (tristate) at run time.
Output current range	0 ... ±5 mA	0 ... ±10 mA
High-side reference voltage	+4.5 V ... +40 V	+5 V
Configurable fuse	–	
Circuit diagram	Digital Out 7 on page 1580	Digital In/Out 8 on page 1589
Required channels	1	

More hardware data

DS1554 Engine Control I/O Module For more board-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Ignition Out

Where to go from here

Information in this section

Introduction (Ignition Out).....	617
Overviews (Ignition Out).....	618
Configuring the Function Block (Ignition Out).....	623
Hardware Dependencies (Ignition Out)	627

Introduction (Ignition Out)

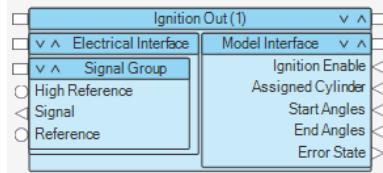
Introduction to the Function Block (Ignition Out)

Function block purpose

The Ignition Out function block type lets you generate ignition pulses for a real piston engine.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating ignition pulses according to values for pulse start and end angles provided from within the behavior model.
- Limiting pulse durations to a specified maximum.
- Providing error status information to the behavior model.

Supported channel types

The Ignition Out function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III	
Channel type	–	Digital Out 7	Digital In/Out 8
Hardware	–	DS1554	

Oversviews (Ignition Out)

Where to go from here**Information in this section**

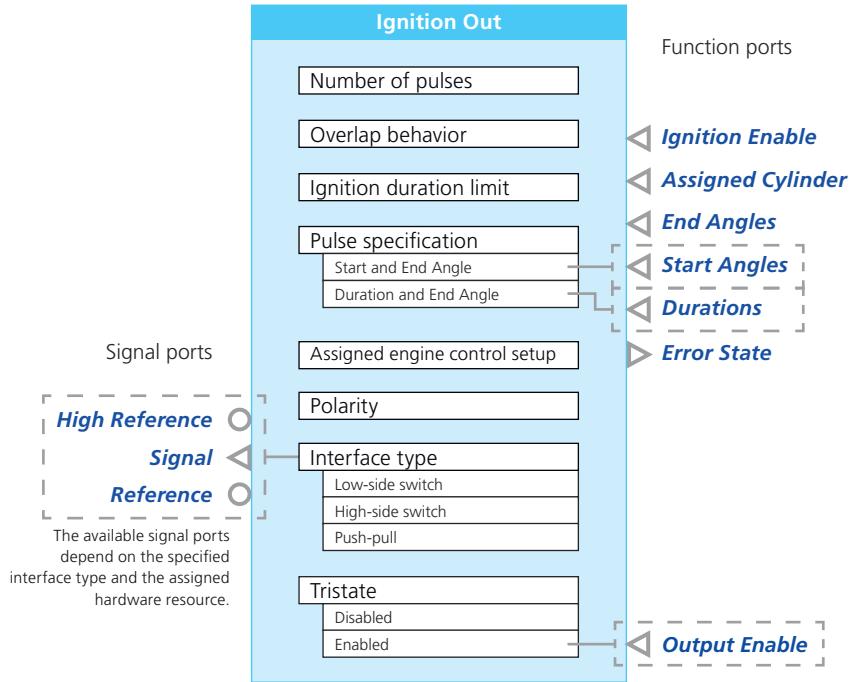
Overview of Ports and Basic Properties (Ignition Out)..... 618

Overview of Tunable Properties (Ignition Out)..... 622

Overview of Ports and Basic Properties (Ignition Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Ignition Enable

This function import enables the function block to generate ignition pulses from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Setting this function port to Enabled takes effect only if the Injection Ignition Enable function port of the assigned Engine Control Setup function block is also set to Enabled.

Assigned Cylinder

This function import provides the number of the piston engine cylinder, for which the ignition pulses are generated.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the setting of the Number of cylinders property of the assigned Engine Control Setup function block. If the provided value exceeds the value specified at the Number of cylinders property, the function block generates pulses for the cylinder with the highest number.
Dependencies	-

End Angles

This function import reads a vector of angle values provided from within the behavior model. The values specify the end of ignition pulses related to the TDCs of assigned cylinders. Ignition sparks are generated at this point.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° for each entry in the vector. ▪ The effective range depends on the setting of the Angle range property of the assigned Engine Control Setup function block: <ul style="list-style-type: none"> ▪ 360°: -180° ... +180° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -250° is converted to +110°, +300° is converted to -60°. ▪ 720°: -360° ... +360° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -400° is converted to +320°, +830° is converted to +110°. ▪ The vector size depends on the number specified at the Number of pulses property. ▪ The vector entries must be provided in descending order, otherwise an error is indicated via the Error State function port.
Dependencies	-

Start Angles

This function import reads a vector of angle values provided from within the behavior model. The values specify the start point of ignition pulses related to the TDCs of assigned cylinders.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° for each entry in the vector. ▪ The effective range depends on the setting of the Angle range property of the assigned Engine Control Setup function block: <ul style="list-style-type: none"> ▪ 360°: -180° ... +180° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -250° is converted to +110°, +300° is converted to -60°. ▪ 720°: -360° ... +360° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -400° is converted to +320°, +830° is converted to +110°. ▪ The vector size depends on the number specified at the Number of pulses property. ▪ The vector entries must be provided in descending order, otherwise an error is indicated via the Error State function port.
Dependencies	Available only if the Pulse specification property is set to Start and End Angle.

Durations

This function import reads a vector of pulse duration values provided from within the behavior model. The values specify the length of ignition pulses (in seconds).

Value range	<ul style="list-style-type: none"> ▪ 0 s ... 10 s for each entry in the vector. ▪ The vector size depends on the number specified at the Number of pulses property.
Dependencies	Available only if the Pulse specification property is set to Duration and End Angle .

Error State

This function outport writes status information on errors during pulse generation to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0, 2, 4, 6 ▪ The value of the function port represents the following errors: <ul style="list-style-type: none"> ▪ 0: No errors occurred during pulse generation. ▪ 2: At least one of the following error conditions occurred: <ul style="list-style-type: none"> ▪ The angles for the ignition pulses are not provided in a descending order. ▪ An ignition pulse completely covers another one (e.g., first pulse 0° ... 100°, second pulse 20° ... 80°). ▪ The distance between two ignition start or end angles is smaller than 0.1° engine cycle angles. ▪ More than two consecutive ignition pulses overlap. ▪ 4: An ignition pulse was limited to the value specified at the Ignition duration limit property. ▪ 6: The errors indicated by values of 2 and 4 occurred at the same time.
Dependencies	-

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Tristate property is set to Enabled .

High Reference

This signal port is a reference port and provides the high-side reference for the **Signal** signal port when the digital outputs operate as high-side switches or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Ignition Out) on page 627.
Dependencies	Available only if the port is supported by the settings of the Interface type and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Ignition Out) on page 627.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the **Signal** signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Ignition Out) on page 627.
Dependencies	Available only if the port is supported by the settings of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Ignition Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Overlap behavior	✓	–
Ignition duration limit	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Electrical Interface Polarity	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Ignition Out)

Where to go from here

Information in this section

- | | |
|---------------------------------------------------------|-----|
| Configuring the Basic Functionality (Ignition Out)..... | 623 |
| Configuring Standard Features (Ignition Out)..... | 626 |

Configuring the Basic Functionality (Ignition Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the number of ignition pulses to be generated.
- Specifying ignition pulse characteristics.
- Specifying pulse overlap behavior.
- Specifying ignition pulse limiting.
- Assigning an Engine Control Setup function block.

Specifying the number of ignition pulses to be generated

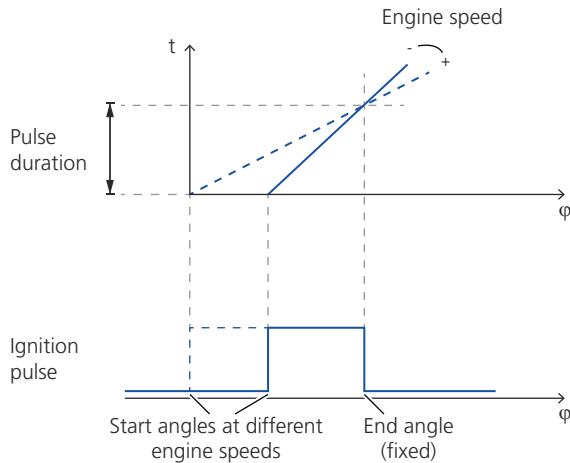
You can specify a single ignition pulse or a sequence of up to six ignition pulses to be generated per engine cycle at the Number of pulses property.

Specifying ignition pulse characteristics

The function block can generate ignition pulses specified by different pulse characteristics (pulse duration, start angles, and end angles). You have to select which set of characteristics is used.

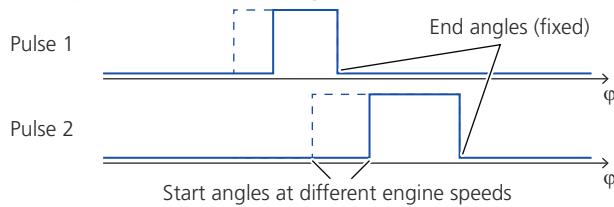
Pulses specified by start and end angles For this set, start and end angle values are provided from within the behavior model. Pulse duration depends on actual engine speed.

Pulses specified by duration and end angle For this set, duration and end angle values are provided from within the behavior model. Start angles are calculated for each pulse and depend on actual engine speed. Refer to the following illustration:



Specifying pulse overlap behavior

Overlap conditions Overlapping of ignition pulses can be caused directly by the values for duration and start/end angles provided from within the behavior model. Furthermore, pulses that are specified by duration and end angle might overlap when engine speed increases. With increasing engine speed, the start angle moves towards the end angle of the previous pulse and both pulses might overlap. Refer to the following illustration:



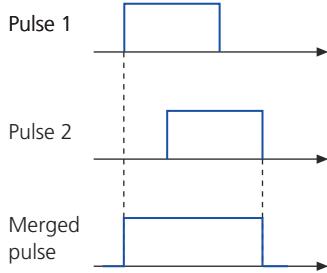
Overlap handling You have to specify how the function block handles overlapping ignition pulses:

- **Merge:**

Two overlapping pulses are merged to a new pulse as follows:

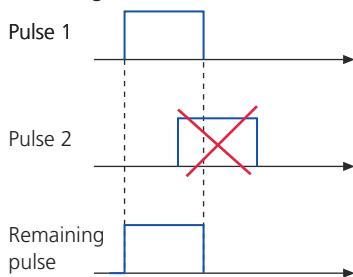
- The start angle of the first started pulse becomes the new start angle.
- The end angle of the last ending pulse becomes the new end angle.

Refer to the following illustration:



- Remove:

A pulse is removed if it starts while another one is active. Refer to the following illustration:



Note

The following cases might lead to incorrect pulse generation:

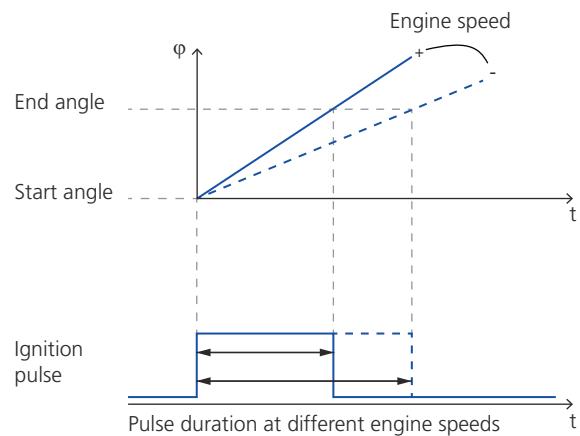
- An ignition pulse completely covers another one (e.g., first pulse 0° ... 100°, second pulse 20° ... 80°).
- More than two consecutive ignition pulses overlap.

If any of these cases occurs, this is indicated as error and provided to the behavior model via the Error State function port.

Specifying ignition pulse limiting

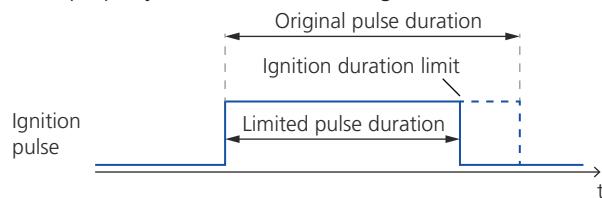
Depending on your application, you might have to specify a maximum pulse duration. Without limiting, pulses might exceed maximum values for the following reasons:

- Duration values provided from within the behavior model are greater than the specified maximum.
- Overlapping pulses are merged to a pulse with a duration that exceeds the specified maximum.
- Engine speed is so low that pulse duration exceeds the specified maximum. This might happen with pulses that are specified by start and end angle values. In this case, pulse duration depends on engine speed. Refer to the following illustration:



With decreasing engine speed, pulse duration increases.

Pulses are cut off by the end of the interval specified at the Ignition duration limit property. Refer to the following illustration:



Note

If a pulse is cut off, this is indicated as error and provided to the behavior model via the Error State function port.

Assigning an Engine Control Setup function block

You have to assign an Engine Control Setup function block that provides data about the controlled piston engine (number of cylinders, TDCs, etc.) and references a master APU provider.

Configuring Standard Features (Ignition Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer.

You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 7	Digital In/Out 8	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–
Polarity of output signals	✓	✓	–
High impedance (tristate)	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.		–
Channel multiplication	–	–	–
Signal Ports			
Trigger level of electronic fuse	–	–	–
Failure simulation support	–	–	–

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 7](#) on page 1580
- [Digital In/Out 8](#) on page 1589

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Ignition Out)

MicroAutoBox III Hardware Dependencies (Ignition Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 7	Digital In/Out 8
Hardware	DS1554 Engine Control I/O Module	
Event generation	—	
Minimum output pulse width	<ul style="list-style-type: none"> ▪ High: 700 ns ▪ Low: 200 ns 	12.5 ns
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull <p>The outputs can be set to high impedance (tristate) at run time.</p>	
Output current range	0 ... ±5 mA	0 ... ±10 mA
High-side reference voltage	+4.5 V ... +40 V	+5 V
Configurable fuse	—	
Circuit diagram	Digital Out 7 on page 1580	Digital In/Out 8 on page 1589
Required channels	1	

More hardware data

DS1554 Engine Control I/O Module For more board-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Engine Angular Pulse Out

Where to go from here

Information in this section

Introduction (Engine Angular Pulse Out).....	629
Overviews (Engine Angular Pulse Out).....	630
Configuring the Function Block (Engine Angular Pulse Out).....	632
Hardware Dependencies (Engine Angular Pulse Out)	636

Introduction (Engine Angular Pulse Out)

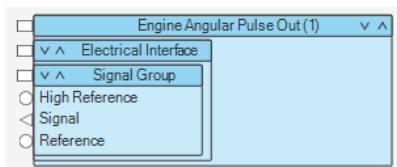
Introduction to the Function Block (Engine Angular Pulse Out)

Function block purpose

The Engine Angular Pulse Out function block generates a periodic pulse pattern based on engine position data retrieved from an assigned master APU provider. The generated pulse pattern can be used, for example, for starting A/D conversions.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

This is the main feature:

- Generating a configurable periodic pulse pattern.

Supported channel types	The Engine Angular Pulse Out function block type supports the following channel types:
Channel type	SCALEXIO
Hardware	MicroAutoBox III Digital Out 7 Digital In/Out 8 DS1554

Oversviews (Engine Angular Pulse Out)

Where to go from here

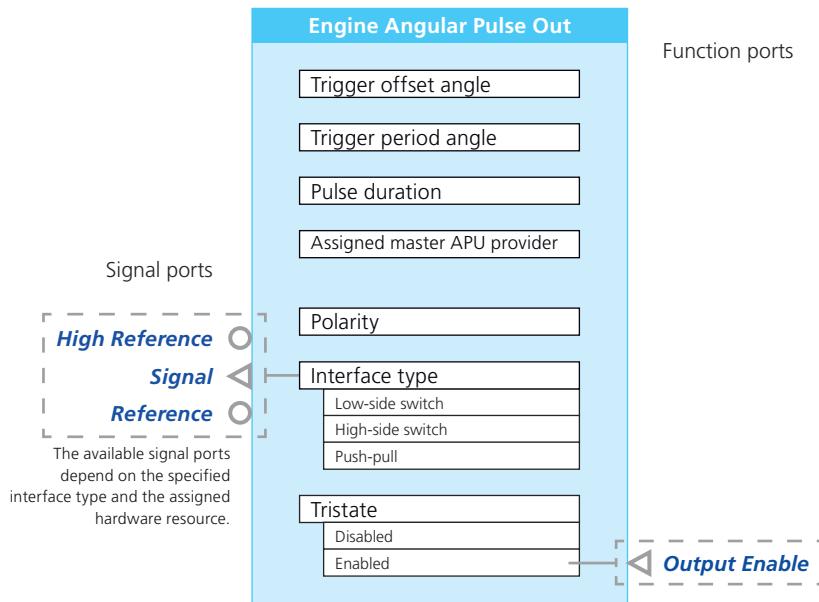
Information in this section

- [Overview of Ports and Basic Properties \(Engine Angular Pulse Out\)..... 630](#)
- [Overview of Tunable Properties \(Engine Angular Pulse Out\)..... 632](#)

Overview of Ports and Basic Properties (Engine Angular Pulse Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Tristate property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs operate as high-side switches or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Engine Angular Pulse Out) on page 636.
Dependencies	Available only if the port is supported by the settings of the Interface type and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Engine Angular Pulse Out) on page 636.
Dependencies	–

Reference

This signal port is a reference port and provides the low-side reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Engine Angular Pulse Out) on page 636.
Dependencies	Available only if the port is supported by the settings of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Engine Angular Pulse Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Trigger offset angle	✓	–
Trigger period angle	✓	–
Pulse duration	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Polarity	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Engine Angular Pulse Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (Engine Angular Pulse Out)..... 633

Configuring Standard Features (Engine Angular Pulse Out)..... 635

Configuring the Basic Functionality (Engine Angular Pulse Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

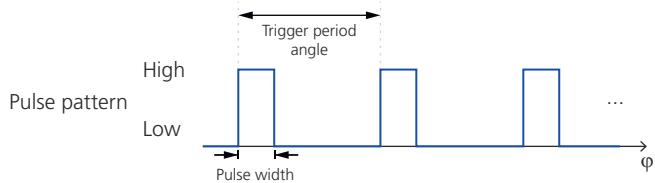
- Assigning a master APU provider.
- Specifying the characteristics of the generated pulse pattern.
- Specifying an offset for pulse pattern generation.

Assigning a master APU provider

You have to assign a master APU provider that references an angular processing unit (APU). The APU is part of the MicroAutoBox III and provides the engine cycle position data of the piston engine to the function block. For more information on angular processing units, refer to [Using Angular Processing Units \(APUs\) on page 126](#).

Specifying the characteristics of the generated pulse pattern

The function block generates a pulse pattern with pulses of equal width spaced at a specific angular distance. Refer to the following illustration:

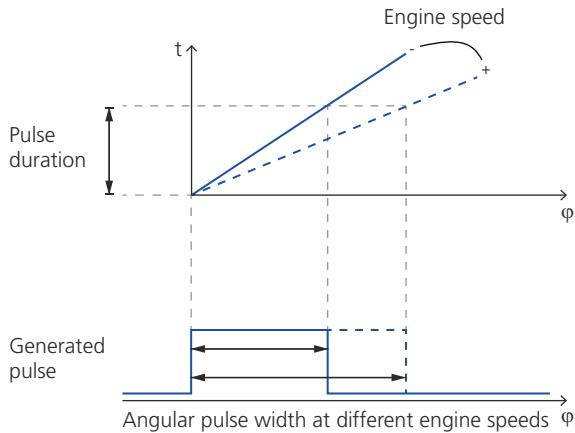


Trigger period angle You have to specify the trigger period angle. The trigger period angle determines the angular distance of the rising or falling edges of consecutive pulses. The specified value must match the angle range of the

assigned master APU provider. Otherwise, a conflict is generated and displayed in the Conflicts Viewer.

Pulse duration (pulse width) You have to specify the duration of the generated pulses (in seconds). The angular width of the generated pulses results from the specified pulse duration and the actual engine speed.

Effects of engine speed The angular width of the generated pulses changes when engine speed changes. Refer to the following illustration:



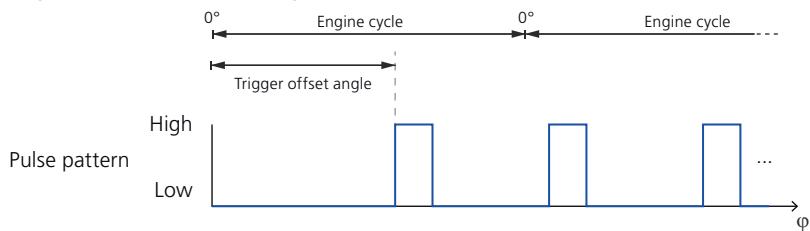
Depending on the specified pulse duration and trigger period angle, increasing engine speed might lead to overlapping pulses, because angular pulse width becomes equal to or greater than the specified trigger period angle.

Pulse overlap behavior Pulse generation might become erroneous when pulses overlap. To generate valid pulses, observe the maximum engine speed when specifying the pulse pattern. The following rule applies:

$$\text{Engine speed}_{\max} [\text{rpm}] < \text{trigger period angle } [{}^\circ] / \text{pulse duration } [\text{s}] / 6.$$

Specifying an offset for pulse pattern generation

By default, the function block starts generating the pulse pattern at the 0° position of an engine cycle. To postpone the start, you can specify a trigger offset angle. Refer to the following illustration:



The maximum trigger offset angle equals one engine cycle of 360° or 720° , depending on the angle range of the related master APU provider.

Configuring Standard Features (Engine Angular Pulse Out)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 7	Digital In/Out 8	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–
High impedance (tristate)	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.	–	–
Polarity of output signals	✓	✓	–
Channel multiplication	–	–	–
Signal Ports			
Trigger level of electronic fuse	–	–	–
Failure simulation support	–	–	–

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 7](#) on page 1580
- [Digital In/Out 8](#) on page 1589

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Engine Angular Pulse Out)

MicroAutoBox III Hardware Dependencies (Engine Angular Pulse Out)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 7	Digital In/Out 8
Hardware	DS1554 Engine Control I/O Module	
Event generation	–	
Minimum output pulse width	<ul style="list-style-type: none">▪ High: 700 ns▪ Low: 200 ns	12.5 ns
Interface type for digital outputs	<ul style="list-style-type: none">▪ Low-side switch▪ High-side switch▪ Push-pull <p>The outputs can be set to high impedance (tristate) at run time.</p>	
Output current range	0 ... ±5 mA	0 ... ±10 mA
High-side reference voltage	+4.5 V ... +40 V	+5 V
Configurable fuse	–	
Circuit diagram	Digital Out 7 on page 1580	Digital In/Out 8 on page 1589
Required channels	1	

More hardware data

DS1554 Engine Control I/O Module For more board-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Knock In

Where to go from here

Information in this section

Introduction (Knock In).....	637
Overviews (Knock In).....	640
Configuring the Function Block (Knock In).....	644
Hardware Dependencies (Knock In)	646

Introduction (Knock In)

Where to go from here

Information in this section

Introduction to the Function Block (Knock In).....	637
Basics on Knock Signal Measurement.....	638

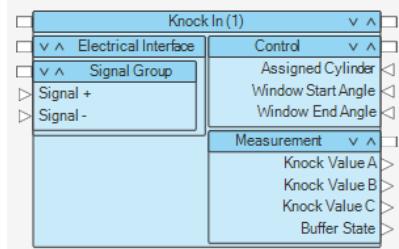
Introduction to the Function Block (Knock In)

Function block purpose

The Knock In function block lets you analyze the noise of a piston engine to avoid/minimize preignitions caused by improper ignition timing.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

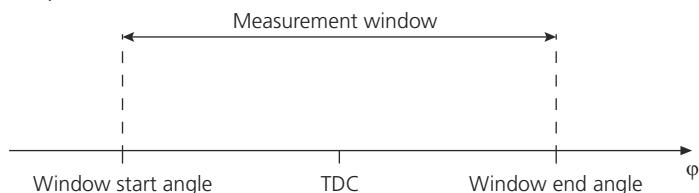


Main features	These are the main features: <ul style="list-style-type: none"> ▪ Analyzing measured knock signals and providing knock values for up to three different frequency bands to the behavior model. ▪ Supporting measurement windows that can be freely defined from within the behavior model. ▪ Generating I/O events and providing them to the behavior model.
Supported channel types	The Knock In function block supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	–	Analog In 15
Hardware	–	DS1554

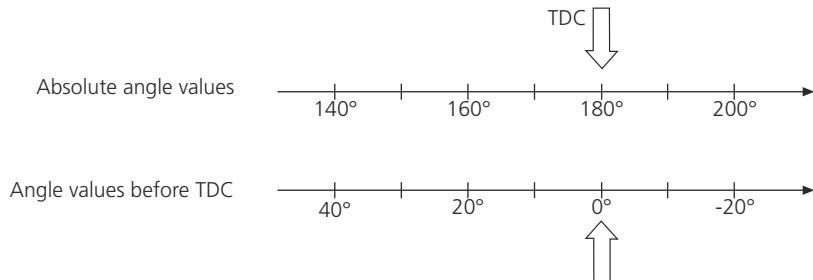
Basics on Knock Signal Measurement

Origin of knock signals	Ignition that occurs too early (preignition) is accompanied by a specific engine noise. Knock sensors convert this noise into a voltage signal (knock signal).
Knock sensor	Knock sensors are piezoelectric sensors mounted on the engine housing. They deliver an analog output voltage representing the engine noise. By using measurement windows around the top dead center (TDC) of particular cylinders, it is possible to measure the knock signal of multiple cylinders that are consecutively selected (assigned) with one single sensor.
Measurement window	Knock signals are measured within a specific angle range (measurement window) around the TDC of the cylinder whose knock signal is analyzed. The start and end angles of the measurement window and the number of the selected cylinder are provided from within the behavior model.



At the end of each measurement window, an I/O event can be generated.

TDC as reference All angle values that are provided to specify measurement windows are relative to *before TDC*. You get these values by subtracting the desired absolute angle value from the absolute TDC angle value.



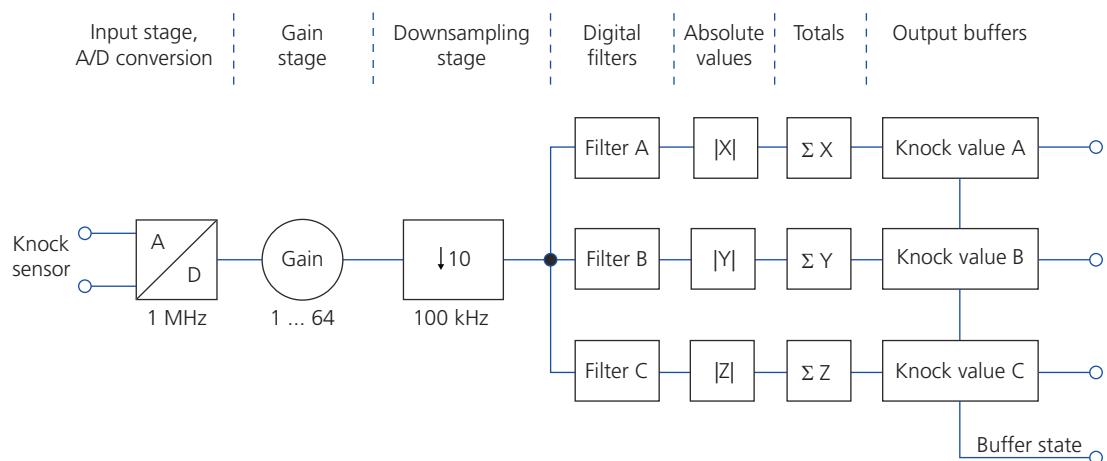
Example: Suppose a measurement window for a TDC of 180° must start at 140° and end at 230°. The required angle values are calculated as follows:

- Window start angle = 180° - 140° = +40°
- Window end angle = 180° - 230° = -50°

+40° and -50° must be provided as window start and end angles from within the behavior model.

Knock signal measurement and processing on MicroAutoBox III

The following illustration shows a block diagram of the signal measurement and processing components of a Knock In function block:



The MicroAutoBox III converts the analog knock signal into a digital signal at a sampling frequency of 1 MHz. The digitized signal is then multiplied by a user-definable gain factor and passes a 100 kHz downsampling stage. The downsampled knock signal is fed to three independent digital filters with finite impulse response (FIR filters) in parallel. The characteristics of each filter are determined by a set of loadable filter coefficients.

Starting with the beginning of each measurement window, the magnitudes of the filter outputs are integrated in totals. At the end of each measurement window, the totals of the three filters are transferred to the output buffers and provided to the behavior model via function outports. Another function outport provides the buffer state that indicates, whether the output buffers were read from the behavior model since their last update. Furthermore, an I/O event can be generated at the end of each measurement window.

Note that measuring knock values and generating the related I/O events starts when engine speed rises above 10 rpm. It stops when engine speed falls below 5 rpm or the crankshaft rotates in reverse direction. Reverse crankshaft rotation is only evaluated when reverse crank mode is enabled.

Oversviews (Knock In)

Where to go from here

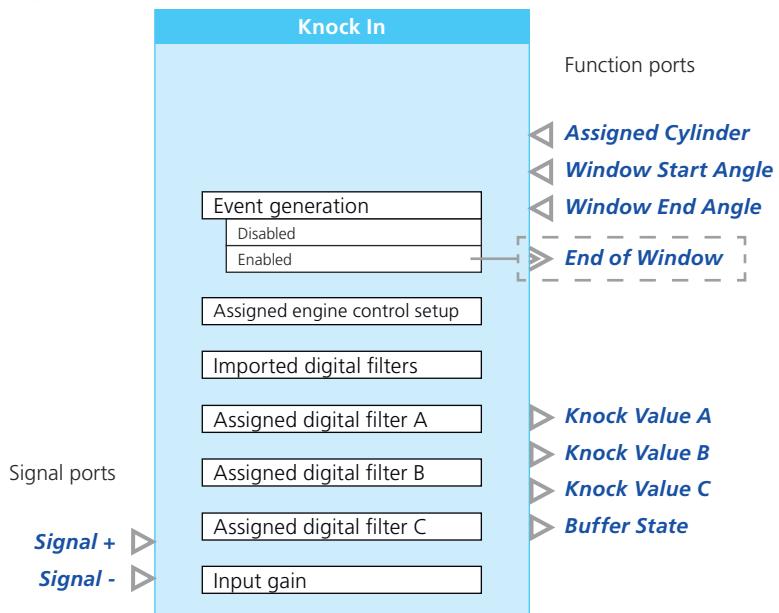
Information in this section

- | | |
|--------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Knock In)..... | 640 |
| Overview of Tunable Properties (Knock In)..... | 643 |

Overview of Ports and Basic Properties (Knock In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Assigned Cylinder

This function import provides the number of the piston engine cylinder, for which the knock signal is analyzed.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the setting of the Number of cylinders property of the assigned Engine Control Setup function block. If the provided value exceeds the value specified at the Number of cylinders property, the function block analyzes the knock signal of the cylinder with the highest number.
Dependencies	–

Window Start Angle

This function import reads the start angle of the knock signal measurement window from within the behavior model. The value is relative to the TDC of the assigned cylinder.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° ▪ The effective range depends on the setting of the Angle range property of the assigned Engine Control Setup function block: <ul style="list-style-type: none"> ▪ 360°: -180° ... +180° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -250° is converted to +110°, +300° is converted to -60°. ▪ 720°: -360° ... +360° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -400° is converted to +320°, +830° is converted to +110°.
Dependencies	–

Window End Angle

This function import reads the end angle of the knock signal measurement window from within the behavior model. The value is relative to the TDC of the assigned cylinder.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° ▪ The effective range depends on the setting of the Angle range property of the assigned Engine Control Setup function block: <ul style="list-style-type: none"> ▪ 360°: -180° ... +180° If you enter values outside the effective range, the system converts them to values matching the effective range. Example: -250° is converted to +110°, +300° is converted to -60°.
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- 720°: -360° ... +360°
If you enter values outside the effective range, the system converts them to values matching the effective range.
Example: -400° is converted to +320°, +830° is converted to +110°.

Dependencies	-
--------------	---

End of Window

This event port provides an I/O event each time a knock signal measurement is completed, i.e., the engine cycle reaches the end angle of the measurement window.

Value range	-
Dependencies	Available only if the Event generation property is set to Enabled.

Knock Value A

This function outport provides the knock values to the behavior model that were derived from the knock signal and passed the digital filter A. The value is updated at the end of each measurement window.

Value range	0 ... 4,294,967,295
Dependencies	-

Knock Value B

This function outport provides the knock values to the behavior model that were derived from the knock signal and passed the digital filter B. The value is updated at the end of each measurement window.

Value range	0 ... 4,294,967,295
Dependencies	-

Knock Value C

This function outport provides the knock values to the behavior model that were derived from the knock signal and passed the digital filter C. The value is updated at the end of each measurement window.

Value range	0 ... 4,294,967,295
Dependencies	-

Buffer State

This function outport indicates, whether the knock values provided at the Knock Value A, B, C function ports were read by the behavior model since the end of the last measurement window or not.

Value range	<ul style="list-style-type: none"> ▪ 1: New knock values are available. The buffer state value is set to 1 at the end of each measurement window. ▪ 0: No new knock values are available. The buffer state value is set to 0 when the Knock Value A, B, C function ports are read.
Dependencies	–

Signal + / Signal -

These signal ports represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (Signal + and Signal -) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Knock In) on page 646.
Dependencies	–

Overview of Tunable Properties (Knock In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
–	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Assigned digital filter A	✓	–
Assigned digital filter B	✓	–
Assigned digital filter C	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Knock In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Knock In).....	644
Configuring Standard Features (Knock In).....	645

Configuring the Basic Functionality (Knock In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Assigning digital filters for knock signal analysis.
- Specifying a gain factor for the knock signal.
- Assigning an Engine Control Setup function block.
- Providing I/O events.

Assigning digital filters for knock signal analysis

The Knock In function block analyzes the knock signal via three digital filters and the results are then provided to the behavior model. The filter coefficients to configure the digital filters are stored in digital filter files that are supplied with ConfigurationDesk. You have to import one or more filter files from a directory with the supplied files and assign one of the imported digital filter file, i.e., a set of filter coefficients, to each of the three digital filters.

The digital filter files are located in the `<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles` folder. The file format is CSV. The import dialog provides a preview of the filter characteristics for each imported file as shown in the following example:

Parameter	Value
Name	Kaiser Window 1
Type	Kaiser Window
Order	49
Lower cutoff frequency (...)	14900 Hz
Upper cutoff frequency (...)	17100 Hz
Center frequency (Fce)	16000 Hz
Number of coefficients	50

You can change the assigned digital filter file via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

Specifying a gain factor for the knock signal	You can specify a gain factor > 1 to amplify the digitized knock signals. This might be necessary to adjust the signal processing, for example, to the sensitivity of the connected knock sensor or to weak knock signals.
Assigning an Engine Control Setup function block	You have to assign an Engine Control Setup function block that provides data about the controlled piston engine (number of cylinders, TDCs, etc.) and references a master APU provider.
Providing I/O events	<p>If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.</p> <p>The Knock In function block can generate an I/O event each time a measurement of the pump current and the probe resistance is completed.</p> <p>To use the I/O events in the behavior model, you have to assign them to a task via the End of Window property. For more information, refer to Modeling Asynchronous Tasks (ConfigurationDesk Real-Time Implementation Guide).</p>

Configuring Standard Features (Knock In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.								
Electrical interface	<p>The electrical interface of the Knock In function block does not provide configurable standard features.</p> <p>Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:</p> <ul style="list-style-type: none"> ▪ Analog In 15 on page 1551 								
Model interface	<p>The interface to the behavior model of the function block provides the following standard features.</p> <table border="1"> <thead> <tr> <th>Configuration Feature</th> <th>Further Information</th> </tr> </thead> <tbody> <tr> <td>Function Ports</td> <td></td> </tr> <tr> <td>Initialization behavior</td> <td>Specifying Initialization and Stop Behavior on page 88</td> </tr> <tr> <td>Test automation support</td> <td>Configuring Test Automation Support on page 92</td> </tr> </tbody> </table>	Configuration Feature	Further Information	Function Ports		Initialization behavior	Specifying Initialization and Stop Behavior on page 88	Test automation support	Configuring Test Automation Support on page 92
Configuration Feature	Further Information								
Function Ports									
Initialization behavior	Specifying Initialization and Stop Behavior on page 88								
Test automation support	Configuring Test Automation Support on page 92								

Hardware Dependencies (Knock In)

MicroAutoBox III Hardware Dependencies (Knock In)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type.

Channel Type	Analog In 15
Hardware	DS1554 Engine Control I/O Module
Event generation	✓
Input voltage range	-5 V ... +5 V
Resolution	16 bit
Sample rate	1 MS/s
Offset error	±1.5 mV (typ.)
Gain error	±0.25 % (typ.)
Edge frequency (-3dB cutoff frequency)	400 kHz
Circuit diagram	Analog In 15 on page 1551
Required channels	1

More hardware data

DS1554 Engine Control I/O Module For more board-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Lambda Probe In

Where to go from here

Information in this section

Introduction (Lambda Probe In).....	647
Overviews (Lambda Probe In).....	652
Configuration of the Function Block (Lambda Probe In).....	657
Hardware Dependencies (Lambda Probe In).....	662

Introduction (Lambda Probe In)

Where to go from here

Information in this section

Introduction to the Function Block (Lambda Probe In).....	647
Introduction to Wideband Lambda Probe Operation (Lambda Probe In).....	648

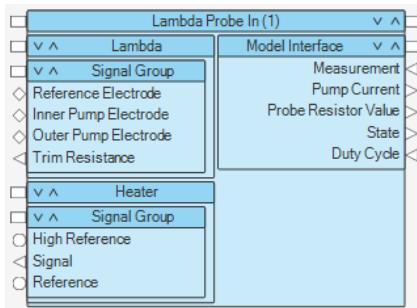
Introduction to the Function Block (Lambda Probe In)

Function block purpose

The Lambda Probe In function block lets you control LSU 4.9 and LSU ADV wideband lambda probes.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

**Main features**

These are the main features:

- Measuring pump cell current and probe resistance of a connected lambda probe and providing the measurement results to the behavior model.
- Supporting LSU 4.9 and LSU ADV wideband lambda probes.
- Generating a configurable PWM signal for the probe heater.
- Generating I/O events and providing them to the behavior model.

Required channel types

The Lambda Probe In function block type requires the following channel types and hardware:

	SCALEXIO	MicroAutoBox III
Hardware	–	DS1554
Channel types	–	Channels of the following channel types are required: ▪ Lambda In 1 ▪ Digital Out 7 or Digital In/Out 8

Introduction to Wideband Lambda Probe Operation (Lambda Probe In)

Basics on lambda measurements

Wideband lambda probes measure the oxygen content in the exhaust gas of a combustion engine so that the engine ECU can determine the amount of fuel required to ensure optimal exhaust gas values.

Definition of lambda The oxygen content in the exhaust gas of a combustion engine is expressed by the variable λ (Greek: lambda):

$$\lambda = \frac{\text{Measured air/fuel mixture}}{\text{Stoichiometric air/fuel mixture}} = \frac{\text{Measured air/fuel mixture}}{14.7 : 1}$$

Combustion is best (causing minimum pollution) if the air/fuel mixture is 14.7 : 1. This equals the stoichiometric air/fuel mixture and $\lambda = 1$.

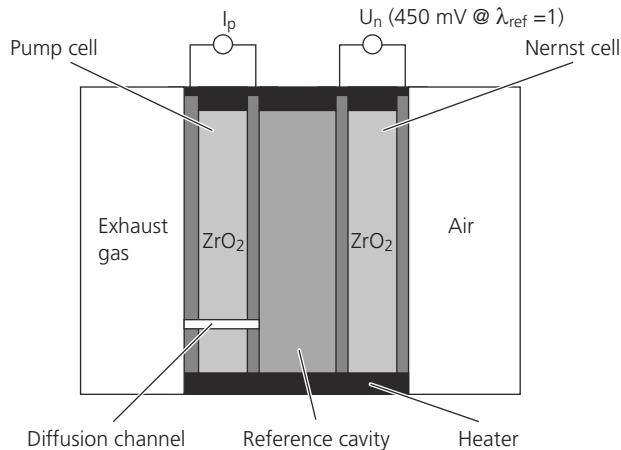
Operation modes of piston engines Modern automotive piston engines such as gasoline direct injection engines have three operation modes:

Operation Mode	Lambda	Engine Characteristic
Rich	$\lambda < 1.0$	Increases engine power
Stoichiometric	$\lambda = 1.0$	Minimum pollution
Lean	$\lambda > 1.0$	Decreases fuel consumption

Wideband lambda probes measure how rich or lean the air/fuel mixture is in the approximate range $\lambda = 0.7 \dots 1.7$.

Wideband lambda probe design and operating principle

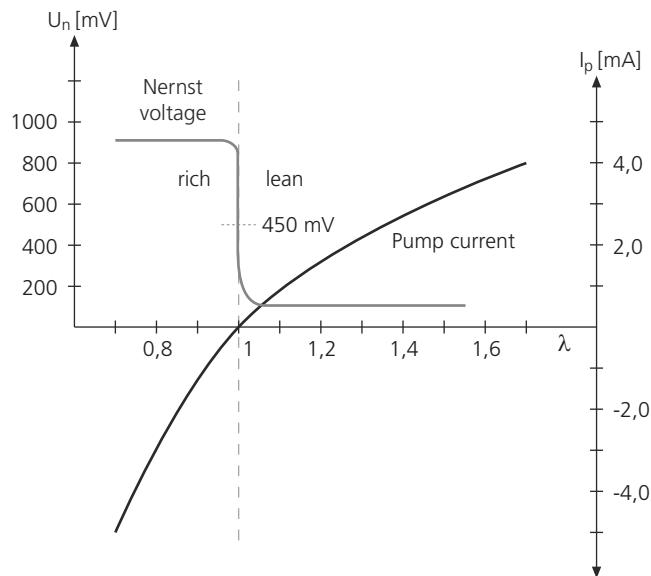
The supported probe types (and most other wideband lambda probe types) consist of a *Nernst cell* and a so-called *pump cell*. There is a reference cavity between the two cells. The exhaust gas passes a diffusion channel and can flow into the reference cavity. The two electrodes of the pump cell are connected to a current supply (ECU) so that a bipolar pump current I_p can be applied to the pump cell. The Nernst voltage U_n is fed to the ECU. To reach the operating temperature quickly, the probe has an integrated heater.



A change in the oxygen content of the exhaust gas (λ) influences the oxygen concentration in the reference cavity (λ_{ref}):

Exhaust Gas	Reference Cavity	U_n
λ is constant	$\lambda_{ref} = 1$	450 mV
λ increases	$\lambda_{ref} = 1$ changes to $\lambda_{ref} > 1$ Oxygen ions leave the exhaust gas and flow into the reference cavity.	Decreases to 100 mV
λ decreases	$\lambda_{ref} = 1$ changes to $\lambda_{ref} < 1$ Oxygen ions leave the reference cavity and flow into the exhaust gas.	Increases to 900 mV

The pump current I_p calculated and generated by the ECU causes oxygen ions to move in the opposite direction through the ZrO_2 layer. This finally results in a steady state with $\lambda_{ref} = 1$ and $U_n = 450$ mV. λ_{ref} does not change until λ of the exhaust gas changes again. The following illustration shows the relationship of λ (exhaust gas) and required pump current I_p . The I_p - λ curve depends on the lambda probe type.



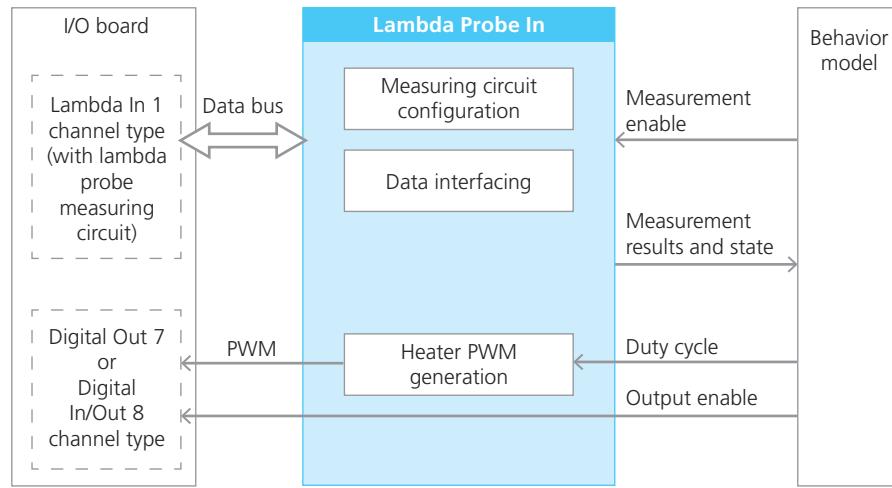
The resistance of the Nernst cell is used to measure the temperature of the probe. The resistance decreases as probe temperatures increase, and it increases as probe temperatures decrease. Typically, the heater is controlled by a PWM signal.

Operating wideband lambda probes with the Lambda Probe In function block

The Lambda Probe In function block contains all the necessary functionalities for operating the supported wideband lambda probes in connection with the probe measuring circuit of the MicroAutoBox III. The function block configures the probe measuring circuit (Bosch CJ135) according to the specified wideband probe type.

When measuring is enabled, the pump current and the probe resistor are measured approx. every 20 ms. At the end of each measurement, up-to-date pump current values, probe resistor values, and measurement status information are provided to the behavior model. At the same time, the function block can generate an I/O event and provide it to the behavior model.

The following illustration shows how the Lambda Probe In function block interacts with other components for operating wideband lambda probes:

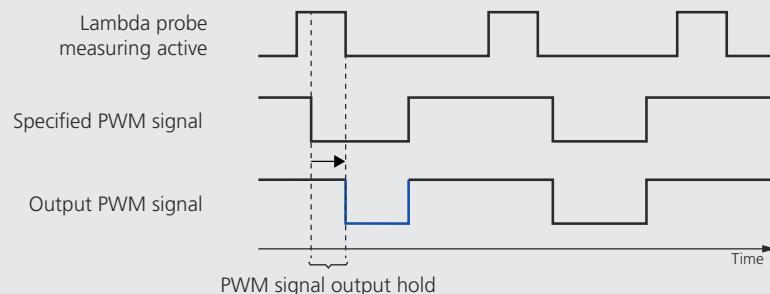


The function block generates a PWM signal for the heater of the connected probe. Heating is required, because wideband lambda probes must be operated above probe-type-specific temperatures to measure pump current and probe resistance reliably. The heater is controlled from within the behavior model, which provides the duty cycle for the generated PWM signal.

Note

Observe the following characteristics of the PWM signal generation:

- Whether measuring is enabled or not, the heater PWM signal is generated continuously. This must be considered when creating the behavior model.
- Measuring intervals vary from approx. 20 ms at typical operating temperatures down to 1 ms when heating up. Each measurement takes approx. 1 ms, during which the current state of the heater PWM signal is held constant to suppress potential interference caused by PWM signal edges. This might modify the output PWM signal with respect to the specified PWM signal. Refer to the following illustration:



Overviews (Lambda Probe In)

Where to go from here

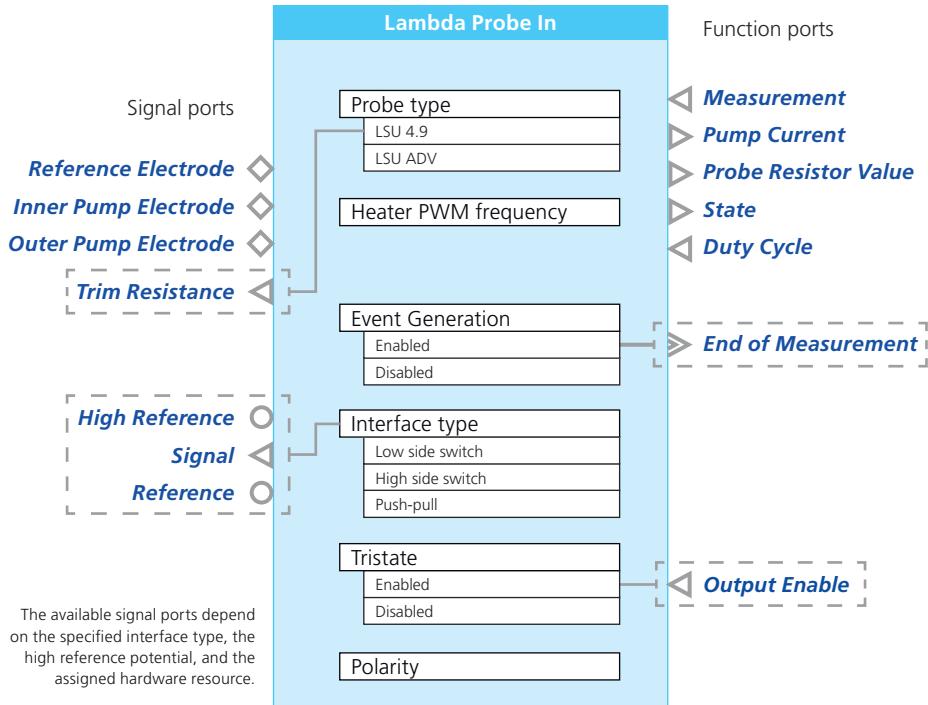
Information in this section

- [Overview of Ports and Basic Properties \(Lambda Probe In\).....652](#)
- [Overview of Tunable Properties \(Lambda Probe In\).....657](#)

Overview of Ports and Basic Properties (Lambda Probe In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Measurement

This function import enables the measurement functionality of the function block from within the behavior model. Each time the signal at this function port changes from Disabled to Enabled, the probe measuring circuit of the MicroAutoBox III is initialized.

Note

Whether measuring is enabled or not, the heater PWM signal is generated continuously.

Value range	<ul style="list-style-type: none"> ▪ 1: Enabled ▪ 0: Disabled
Dependencies	–

Pump Current

This function outport writes the result of the latest measurement of the lambda probe's pump cell current (I_p) to the behavior model. This value is used in the behavior model to evaluate the momentary oxygen content of the exhaust gas (λ).

Value range	<ul style="list-style-type: none"> ▪ -0.5 A ... +0.5 A ▪ < 0 A: The air/fuel mixture is rich ($\lambda < 1$). ▪ > 0 A: The air/fuel mixture is lean ($\lambda > 1$). ▪ 0 A: <ul style="list-style-type: none"> ▪ The air/fuel mixture is stoichiometric ($\lambda = 1$) if the value at the State function port is 2 or 3. ▪ The pump current value is invalid or the measurement is disabled if the value at the State function port is 0, 1, or 4. For more information, refer to State on page 654. <p>For mapping pump current values to λ values, refer to the probe documentation.</p>
Dependencies	–

Probe Resistor Value

This function outport writes the result of the latest measurement of the lambda probe's resistance (R_p) to the behavior model. This value is used in the behavior model to evaluate the momentary probe temperature.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 100 kΩ ▪ 0 kΩ is output in the following cases: <ul style="list-style-type: none"> ▪ The probe resistor value is invalid. ▪ The measurement is disabled. ▪ > 0 kΩ: <ul style="list-style-type: none"> ▪ The probe resistor value is valid. The resistor has a negative temperature coefficient, i.e., the lower the temperature of the probe, the higher the resistance. For specific values, for example, the resistor value at the typical operating temperature, refer to the probe documentation.
Dependencies	–

State	This function outport writes measurement state information to the behavior model.
Value range	<ul style="list-style-type: none"> ▪ 0: Inactive <p>This state is provided if at least one of the following conditions is met:</p> <ul style="list-style-type: none"> ▪ Measuring is disabled at the Measurement function port. ▪ The probe temperature is too low for valid measurements. <p>Note</p> <p>The heater PWM signal is still generated. It is only inactive in the following cases:</p> <ul style="list-style-type: none"> ▪ The value at the Duty Cycle function port is 0 (zero). ▪ The value at the Output Enable function port is 0 (zero). ▪ 1: Valid R_i value <p>The result of the probe resistor (R_i) measurement is valid but the result of the pump current (I_p) measurement is not. The probe temperature is sufficiently high for resistance measurements but too low for valid pump current measurements, which are required for evaluating the oxygen content of the exhaust gas (λ). This is a state when heating up.</p> ▪ 2: Valid I_p value <p>The result of the pump current (I_p) measurement is valid but the result of the probe resistor (R_i) measurement is not. Note that this is not a typical operating state.</p> ▪ 3: Valid R_i and I_p values <p>The results of the probe resistor (R_i) measurement and the pump current (I_p) measurement are valid. The probe temperature is high enough to evaluate the oxygen content of the exhaust gas (λ). This is the normal operating state.</p> ▪ 4: Internal error <p>There is a malfunction in the probe measuring circuit of the MicroAutoBox III. The PWM signal at the Signal signal port changes to the inactive level. To leave this state, you have to change the value at the Measurement function port from Disabled to Enabled. This initializes the probe measuring circuit and usually solves the problem. If not, contact dSPACE Support (www.dspace.com/go/supportrequest).</p>
Dependencies	–

Duty Cycle

This function import reads the duty cycle value for the generation of the heater PWM signal from the behavior model.

Value range	0 ... 100%
Dependencies	-

Output Enable

This function import enables the signal ports of the function block that output the heater PWM signal from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Tristate property is set to Enabled.

End of Measurement

This event port provides an I/O event after each completed measurement of the probe resistance and the pump current.

Value range	-
Dependencies	Available only if the Event generation property is set to Enabled.

Reference Electrode

This signal port serves to measure the lambda probe's Nernst voltage and the probe resistance.

Value range	-
Dependencies	-

Inner Pump Electrode

This signal port represents a virtual ground point for the lambda probe's Nernst cell and pump cell.

Value range	-
Dependencies	-

Outer Pump Electrode

This signal port provides the current for the lambda probe's pump cell.

Value range	-
Dependencies	-

Trim Resistance

This signal outport provides the connection to a trimming resistor that is present on LSU 4.9 wideband lambda probes. The trimming resistor value is specific to each particular probe and compensates for production tolerances. The connection is required for the probe to operate properly.

Value range	–
Dependencies	Available only if the Probe type property is set to LSU 4.9.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal port when the digital outputs are operating as high-side switches or are in push-pull mode.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda Probe In) on page 662.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda Probe In) on page 662.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Lambda Probe In) on page 662.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Lambda Probe In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Heater PWM frequency	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Polarity	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuration of the Function Block (Lambda Probe In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Lambda Probe In).....	657
Configuring Standard Features (Lambda Probe In).....	658
Mapping Signal Ports (Lambda Probe In).....	659

Configuring the Basic Functionality (Lambda Probe In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the type of the connected lambda probe.
- Specifying the heater PWM frequency.
- Providing I/O events.

Specifying the type of the connected lambda probe

You have to specify which type of wideband lambda probe is connected to the function block. The following probe types are supported:

- LSU 4.9
- LSU ADV

With the setting:

- The function is adjusted to the connected probe. For example, the Trim Resistance signal port is additionally provided for LSU 4.9 probes.
- The probe measuring circuit of the MicroAutoBox III is adjusted to the connected probe.

Specifying the heater PWM frequency

You have to specify the frequency of the PWM signal for the lambda probe heater.

Note

You are recommended to specify the value that is given by the probe manufacturer or the lowest possible value (100 Hz) for the heater PWM frequency. For specific values, refer to the probe documentation.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Lambda Probe In function block can generate an I/O event each time a measurement of the pump current and the probe resistance is completed.

To use the I/O events in the behavior model, you have to assign them to a task via the End of Measurement property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Lambda Probe In)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Lambda In 1	Digital Out 7	Digital In/Out 8	More Information
Interface type	–	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–
Polarity of output signals	–	✓	✓	–
High impedance (tristate)	–	Independent of the interface type, the function block can be configured to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.	–	–

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to [Mapping Signal Ports \(Lambda Probe In\)](#) on page 659.

Model interface

The interface to the behavior model of the function block provides the following standard features.

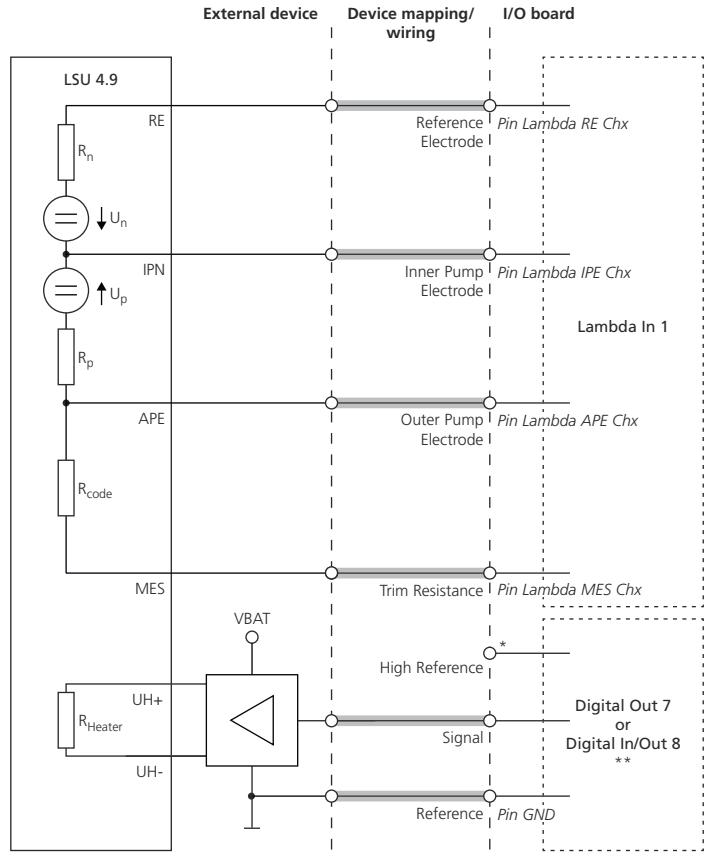
Configuration Feature	More Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Mapping Signal Ports (Lambda Probe In)

Purpose

To map and connect signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the used real-time hardware.

Circuit diagram for Lambda In 1 with LSU 4.9



U_n : Nernst cell voltage

R_n : Nernst cell resistance

U_p : Pump cell voltage

R_p : Pump cell resistance

R_{code} : Compensation resistor

Mapping and wiring

Only wiring

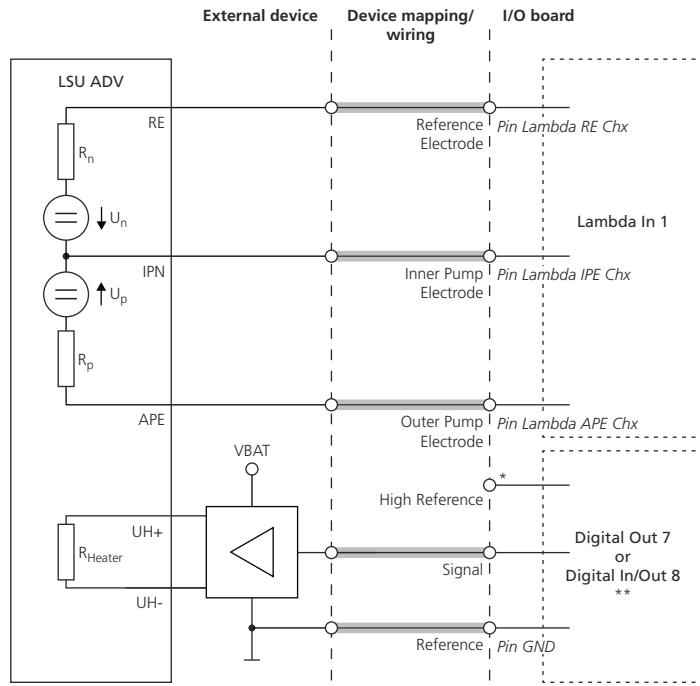
* Availability depends on the assigned channel type and the "Interface type" property setting in ConfigurationDesk

**For details, refer to related circuit diagram

For detailed circuit diagrams of the channel types that can be assigned to the electrical interface for the probe heater, refer to:

- [Digital Out 7](#) on page 1580
- [Digital In/Out 8](#) on page 1589

Circuit diagram for Lambda In 1 with LSU ADV



U_n : Nernst cell voltage
 R_n : Nernst cell resistance
 U_p : Pump cell voltage
 R_p : Pump cell resistance

— Mapping and wiring
 — Only wiring

* Availability depends on the assigned channel type and the "Interface type" property setting in ConfigurationDesk

** For details, refer to related circuit diagram

For detailed circuit diagrams of the channel types that can be assigned to the electrical interface for the probe heater, refer to:

- [Digital Out 7](#) on page 1580
- [Digital In/Out 8](#) on page 1589

Hardware Dependencies (Lambda Probe In)

MicroAutoBox III Hardware Dependencies (Lambda Probe In)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources that are supported by the function block type for the required channel types.

Channel Type		Lambda In 1 with Digital Out 7	Lambda In 1 with Digital In/Out 8	
Hardware		DS1554 Engine Control I/O Module		
Lambda	Channel type	Lambda In 1		
	Event generation	✓		
	Circuit diagram	Mapping Signal Ports (Lambda Probe In) on page 659		
	Required channels	1		
Heater	Channel type	Digital Out 7	Digital In/Out 8	
	Event generation	–		
	PWM frequency range	1 Hz ... 150 kHz		
	Minimum output pulse width	<ul style="list-style-type: none"> ▪ High: 700 ns ▪ Low: 200 ns 	12.5 ns	
	Supported interface types for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull <p>The outputs can be set to high impedance (tristate) at run time.</p>		
	High-side reference voltage	+4.5 V ... +40 V	+5 V	
	Output current range	0 ... ±5 mA	0 ... ±10 mA	
	Circuit diagram	Digital Out 7 on page 1580	Digital In/Out 8 on page 1589	
	Required channels	1	1	

More hardware data

DS1554 Engine Control I/O Module

For more board-specific data, refer to [DS1554 Engine Control I/O Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Wavetable (Voltage Out, Current Sink, Digital Out)

Objective To generate periodic signals by looking up a table with function values.

Where to go from here	Information in this section								
	<table><tr><td>Introduction to Signal Generation Using Time-Coded Wavetables.....</td><td>664</td></tr><tr><td>Wavetable Voltage Out.....</td><td>668</td></tr><tr><td>Wavetable Current Sink.....</td><td>682</td></tr><tr><td>Wavetable Digital Out.....</td><td>696</td></tr></table>	Introduction to Signal Generation Using Time-Coded Wavetables.....	664	Wavetable Voltage Out.....	668	Wavetable Current Sink.....	682	Wavetable Digital Out.....	696
Introduction to Signal Generation Using Time-Coded Wavetables.....	664								
Wavetable Voltage Out.....	668								
Wavetable Current Sink.....	682								
Wavetable Digital Out.....	696								
	<table><tr><th>Information in other sections</th></tr><tr><td>Using a wavetable to simulate the crankshaft/camshaft sensor of a virtual piston engine: Crank/Cam Voltage Out..... 468</td></tr></table>	Information in other sections	Using a wavetable to simulate the crankshaft/camshaft sensor of a virtual piston engine: Crank/Cam Voltage Out..... 468						
Information in other sections									
Using a wavetable to simulate the crankshaft/camshaft sensor of a virtual piston engine: Crank/Cam Voltage Out..... 468									

Introduction to Signal Generation Using Time-Coded Wavetables

Where to go from here

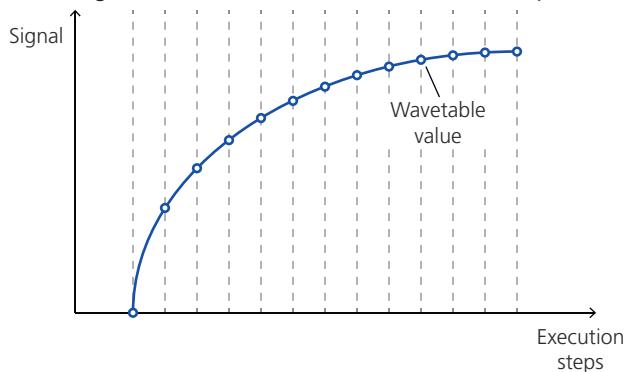
Information in this section

Basics on Signal Generation Using Time-Coded Wavetables.....	664
Specifying Time-Coded Wavetable Files.....	666

Basics on Signal Generation Using Time-Coded Wavetables

The wavetable approach

A wavetable is an array of function values, also called wavetable values. When the wavetable is executed, the wavetable values form the output signal. Executing a wavetable means that its values are output consecutively.



Wavetables can be executed as a function of time (time-coded wavetables) or as a function of angle positions (angular-coded wavetables). In the following, time-coded wavetables are described.

Field of application

Use wavetables if you want to simulate periodic signals that are static, which is contrary to the dynamic waveforms.

Analog output function blocks Use the Wavetable Voltage Out and Wavetable Current Sink function blocks if you want to simulate high-resolution signals, for example, the exact shape of the tooth of a crankshaft wheel.

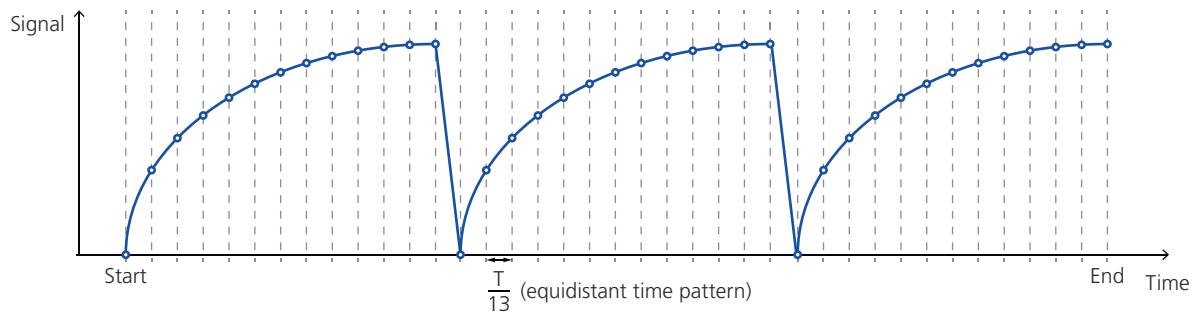
Digital output function block Use the Wavetable Digital Out function block if you want to simulate sensors, for example, a digital crankshaft sensor.

Wavetable execution

Wavetables are executed on the I/O board and the I/O board generates the output signal. This barely affects the CPU, because the wavetable values are not computed at run time, which makes the process highly efficient.

Execution of time-coded wavetables Executing a time-coded wavetable means that all its values are output consecutively at equidistant time intervals to generate an output signal. The equidistant time intervals result from the wavetable period T that is read from within the behavior model and the number of wavetable values.

The example below shows a time-coded wavetable with 13 values that is repeated three times with the wavetable period T .

**Note**

The number of wavetable values affects the lower range limit feasible for the wavetable period, i.e., less values for a shorter (quicker) period. However, a sufficient number of wavetable values are required to maintain a good approximation of the function.

Conditioning the output signal

The wavetable values are normalized values, hence they are not returned as the actual output signal. Conditioning lets you generate output signals with different amplitudes that all refer to the same wavetable. The processing of the wavetable values depends on the function block.

Analog output function blocks The Wavetable Current Sink, Wavetable Voltage Out function blocks multiply the wavetable values by the amplitude. The amplitude is provided by the Amplitude function port.

$$\text{Output Signal} = \text{Wavetable Value} * \text{Amplitude}$$

The table below shows an example wavetable and the resulting output signal for an amplitude of 20 V.

Wavetable Value	Output Signal
0.0	0 V
0.5	10 V
-1.0	-20 V

Digital output function block The Wavetable Digital Out function block supports only the wavetable values 0 and 1.

The table below shows an example wavetable and the resulting output signal.

Wavetable Value	Output Signal
0.0	Low
1	High

Switching between wavetables at run time

You can switch between imported wavetables from the behavior model at run time to use different wavetables. Therefore, imported time-coded wavetables must have the same number of values.

You can import up to 16 time-coded wavetables.

Updating time-coded wavetables

The content of wavetable files can be updated by using the experiment software when the real-time application is stopped. For the changes to take effect, you must restart the real-time application.

Specifying Time-Coded Wavetable Files

Requirements of a wavetable file

A wavetable file is a CSV file with wavetable values. The CSV file must meet the following requirements:

- The file is an ASCII file.
- One CSV file provides the values of only one wavetable.
- Entries are separated by a comma (',').
- New line at the end of a file.
- No additional data or text is included.
- Wavetable values consist of floating-point values (e.g., 1.0).
- The format of the decimal point is a dot ('.') .

- The values are in the range [-1.0 ... 1.0].

Note

The Wavetable Digital Out function block supports only the wavetable values 0 and 1.

Specifying time-coded wavetables

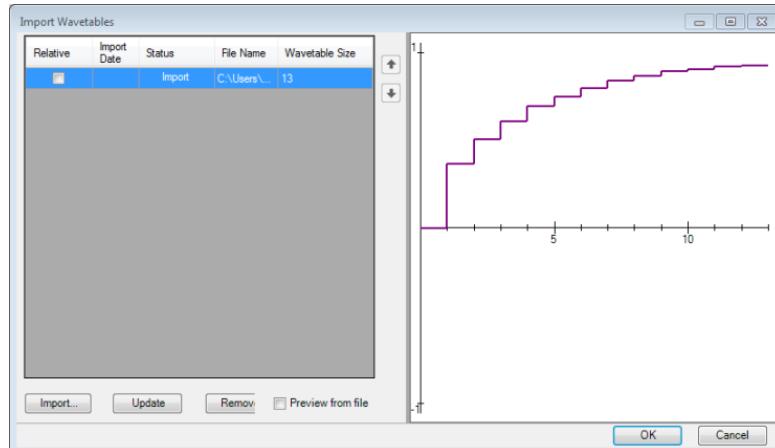
Basically, you can use a text editor to specify the wavetable. The following wavetable characteristics depend on the assigned hardware resource:

- The maximum number of values a wavetable can consist of.
The assigned hardware resource provides a memory to handle a total of 1024 wavetable values for each function block. If a function block uses more than one wavetable, the size of all wavetables must not exceed 1024 wavetable values.
For example: If you use a function block with two wavetables, the size of the wavetables must not exceed 512 wavetable values.
- The accuracy of the analog output signal.
For example, the resolution of the digital-to-analog converter affects the accuracy of the output signal.

The following example is a simple wavetable with 13 entries:

```
0,0.36666,0.50667,0.60667,0.69333,0.74667,0.8,0.84,0.86667,0.89333,0.90667,0.92,0.92667
```

If you edit this example with a text editor and save it with the CSV file name extension, you can import it with the Import Wavetables dialog:



Related topics

Basics

Hardware Dependencies (Wavetable Current Sink)	694
Hardware Dependencies (Wavetable Digital Out)	707
Hardware Dependencies (Wavetable Voltage Out)	679

Wavetable Voltage Out

Where to go from here

Information in this section

Introduction (Wavetable Voltage Out).....	668
Overviews (Wavetable Voltage Out).....	669
Configuring the Function Block (Wavetable Voltage Out).....	674
Hardware Dependencies (Wavetable Voltage Out).....	679

Introduction (Wavetable Voltage Out)

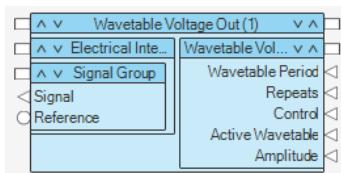
Introduction to the Function Block (Wavetable Voltage Out)

Function block purpose

The Wavetable Voltage Out function block type outputs a voltage signal sequence by looking up a table with time-coded values.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Importing up to 16 wavetable files.
- Controlling wavetable execution (period time, repetitions).
- Selecting different trigger sources for wavetable execution (model-based, angle based, I/O function).
- Switching executed wavetables at runtime.
- Controlling the output signal (amplitude, adding noise and/or offset).
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Wavetable Voltage Out function block supports the following channel types:

	SCALEXIO								MicroAutoBox III
Channel type	Analog Out 1	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 8	Analog Out 9	Analog Out 10	Flexible Out 1	–
Hardware	DS2680			DS6101			DS6241	DS2621	–

Introduction to signal generation based on wavetables

For basic information, refer to [Introduction to Signal Generation Using Time-Coded Wavetables](#) on page 664.

Oversviews (Wavetable Voltage Out)

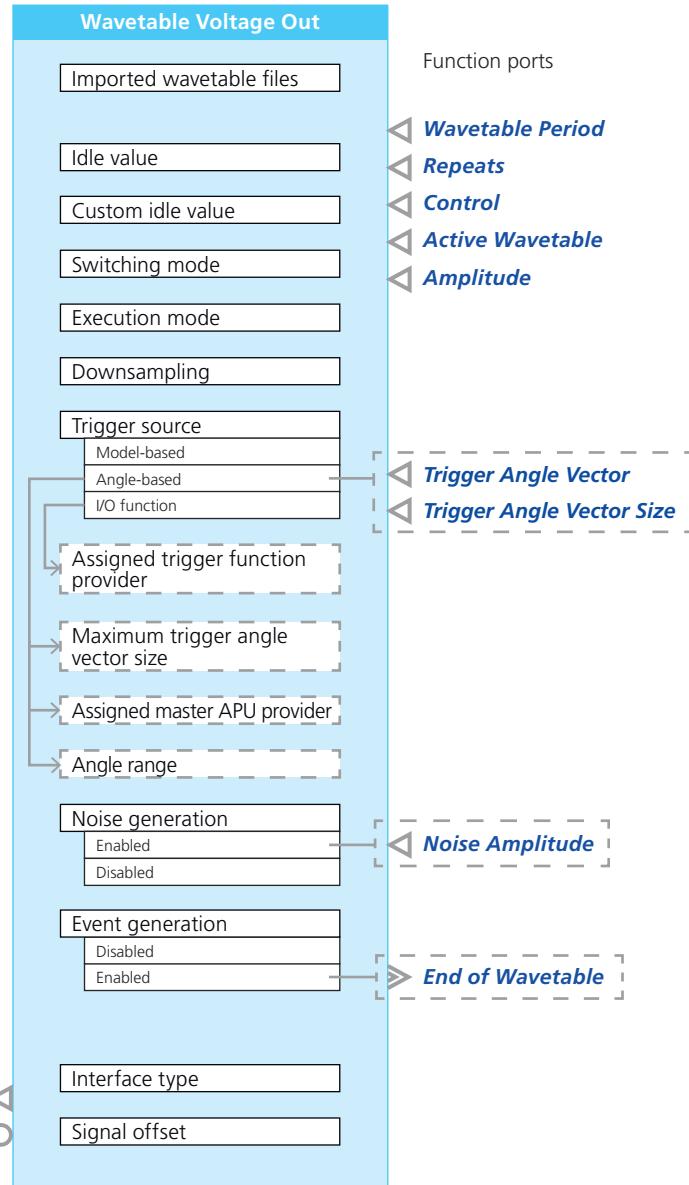
Where to go from here**Information in this section**

- | | |
|---------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Wavetable Voltage Out)..... | 669 |
| Overview of Tunable Properties (Wavetable Voltage Out)..... | 674 |

Overview of Ports and Basic Properties (Wavetable Voltage Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Wavetable Period

This function import lets you define the wavetable period from within the behavior model. The wavetable period is the time for outputting all wavetable values once, i.e., the duration of one wavetable cycle.

Value range	<ul style="list-style-type: none"> $T_{\min} \dots T_{\max}$ (in seconds). The range depends on the following: <ul style="list-style-type: none"> The setting of the Downsampling property. The number of values of the active wavetable. The assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Voltage Out) on page 679.
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Repeats

This function import lets you define from within the behavior model, how often a wavetable is executed. Additionally, you can use this function import to stop an active wavetable execution.

Value range	<ul style="list-style-type: none"> -1 ... 65,534 -1: Infinite repetitions 0: The wavetable is not executed at all. An active wavetable execution is stopped at the end of the currently executed wavetable.
Dependencies	–

Control

This function import lets you enable the start of a wavetable execution from within the behavior model or stop wavetable execution from within the behavior model.

Value range	<ul style="list-style-type: none"> 1: Start When a trigger event occurs <i>and</i> no wavetable is currently executed, wavetable execution starts and is performed as defined at the Repeats function port. Trigger events are ignored if they occur during wavetable execution. For more information, refer to Configuring the Basic Functionality (Wavetable Voltage Out) on page 675. 0: Stop Wavetable execution stops as specified at the Execution mode property. The output changes to the value specified at the Idle value property.
Dependencies	–

Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> 1 ... n n equals the number of imported wavetable files
Dependencies	–

Amplitude

This function import lets you define the amplitude of the output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt).
-------------	---------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ The range depends on the following: ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Voltage Out) on page 679. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... $U_{max\ Noise}$ (in Volt) ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Voltage Out) on page 679. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range. ▪ 0 V: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Noise generation property is set to Enabled. ▪ Supported for Flexible Out 1, Analog Out 3, Analog Out 8, and Analog Out 10 channel types.

Trigger Angle Vector

This function import lets you trigger the execution of the wavetable from within the behavior model. The wavetable is executed whenever an angle value provided at this function port matches an angle value returned by the master APU provider that is assigned to the function block.

To compare the trigger angles that are provided by the behavior model with the angle values of the master APU provider, the function block converts the trigger angles to the angle range that is specified via the **Angle range** property. For example: The function block converts -10° to 350° if the **Angle range** property is set to 360° .

Value range	<ul style="list-style-type: none"> ▪ -3600° ... $+3600^\circ$ for each entry in the vector. ▪ The vector size depends on the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based .

Trigger Angle Vector Size

This function import lets you determine whether all vector entries are considered as trigger angles or only a subset of leading entries from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

End of Wavetable

This event port provides an I/O event each time a wavetable execution has been finished.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Voltage Out) on page 679.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Voltage Out) on page 679.
Dependencies	–

Overview of Tunable Properties (Wavetable Voltage Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Imported Wavetable Files	– ³⁾	–
	Idle Value	✓	–
	Switching Mode	✓	–
	Execution Mode	✓	–
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓
Electrical Interface			
	Signal offset	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ The content of the imported wavetable files can be updated in simulation STOP state.

Configuring the Function Block (Wavetable Voltage Out)

Where to go from here

Information in this section

- | | |
|------------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Wavetable Voltage Out)..... | 675 |
| Configuring Standard Features (Wavetable Voltage Out)..... | 677 |

Configuring the Basic Functionality (Wavetable Voltage Out)

Overview

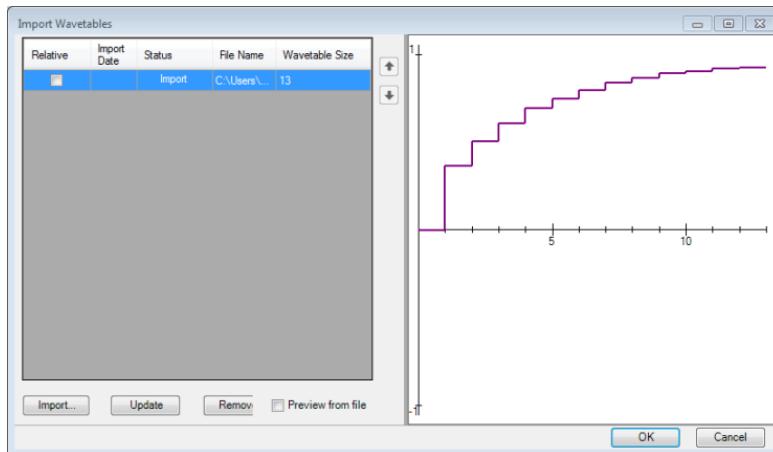
The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Managing wavetable files
- Controlling wavetable execution
- Adding noise to the output signal
- Providing I/O events
- Specifying a signal offset

Managing wavetable files

Wavetable Voltage Out function blocks require at least one wavetable for signal generation. If you do not import a wavetable, the default wavetable is used (all values equal 0).

The Import Wavetables dialog lets you import, update and remove wavetable files.



You open the dialog via the Browse button of the Imported wavetable files property. You can import up to 16 wavetable files.

Controlling wavetable execution

Wavetable execution has the following control options:

Repeated wavetable execution You can execute a wavetable once or repeatedly (repeated n-times or infinitely), depending on the value provided at the Repeats function port. Executing a wavetable repeatedly means that the function block generates a periodic output signal (period = wavetable period T), for example, a sine wave sensor signal.

Changing the wavetable period at run time You can update the wavetable period at run time by changing the value at the Wavetable period function port. The update takes effect immediately.

Switching wavetables at run time You can substitute one wavetable by another one at run time by changing the value at the Active Wavetable

function port. The new wavetable is executed immediately or after the execution of the current wavetable has finished, depending on the **Switching Mode** property.

Stopping wavetable execution You can stop an active wavetable execution by setting the **Repeats** function port to 0 (zero) or by setting the **Control** function port to **Stop**.

- If you set the **Repeats** function port to 0 (zero), wavetable execution stops at the end of the currently executed wavetable.
- If you set the **Control** function port to **Stop**, wavetable execution stops depending on the setting of the **Execution mode** property and the value at the **Repeats** function port.

Execution Mode	Repeats	Stop Behavior
State dependent	1 ... 65,534	Wavetable execution stops after the last repetition.
	-1	Wavetable execution does not stop.
Immediate	Not relevant	Wavetable execution stops immediately.

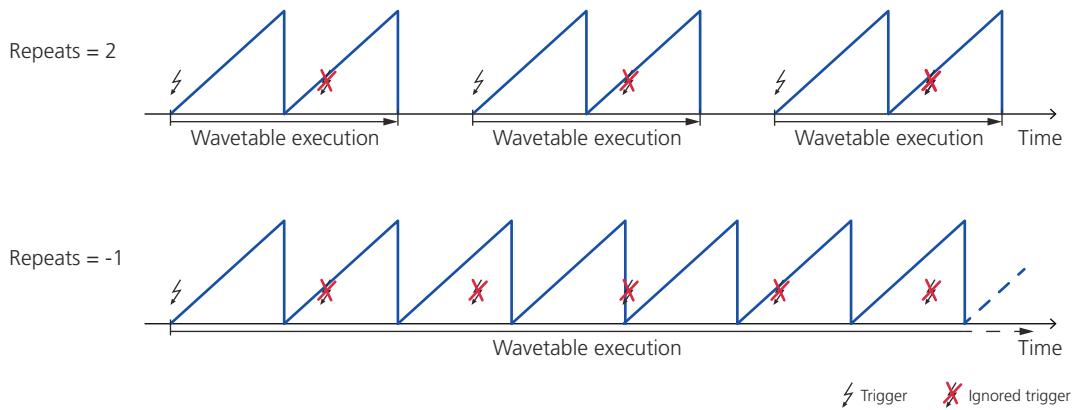
When wavetable execution stops, the function block outputs the specified **Idle** value of the wavetable.

Starting wavetable execution A trigger event from the specified trigger source starts wavetable execution if the following conditions are met:

- The value provided at the **Control** function port is **Start**.
- The value provided at the **Repeats** function port is $<> 0$.
- No wavetable is currently executed.

When wavetable execution starts, it is performed according to the value provided at the **Repeats** function port.

Trigger events are ignored if they occur during wavetable execution. This is shown in the following illustration for **Repeats** values of 2 and -1 (infinite):



You have to specify a trigger source at the **Trigger source** property:

- Model-based: Each model step causes a trigger event.
- Angle-based: A trigger event occurs when the assigned master APU provider returns an angle value that matches the angle value provided at the **Trigger Angle Vector** function port.

As a precondition for angle-based triggering you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

- I/O function: The assigned trigger function triggers the wavetable execution.

Adding noise to the output signal

The Wavetable Voltage Out function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Limitation Noise generation is supported only for the Flexible Out 1, Analog Out 3, Analog Out 8, and Analog Out 10 channel types.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Wavetable Out function blocks can generate an I/O event each time a wavetable execution has been finished.

To use the I/O events in the behavior model, you have to assign them to a task via the End of Wavetable property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Specifying a signal offset

You can apply a voltage offset to the output signal. The offset is channel-specific, hence, all the signals that use the same I/O channel have the same signal offset. This is helpful to generate unipolar output, for example, positive voltages, even with bipolar wavetable values.

Limitation Specifying a signal offset is not supported for the Analog Out 3 and Analog Out 8 channel types.

Configuring Standard Features (Wavetable Voltage Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

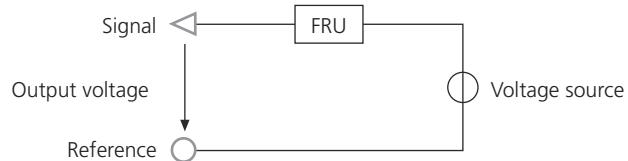
Configuration Feature	Analog Out 1	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 8	Further Information
Interface type	Differential with ground sense	Galvanically isolated	Differential with ground sense	Differential with ground sense	Galvanically isolated	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	✓	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	✓	–	–	Specifying Settings for Failure Simulation on page 123

Configuration Feature	Analog Out 9	Analog Out 10	Flexible Out 1	Further Information
Interface type	Differential with ground sense	Differential with ground sense	Galvanically isolated	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 1 on page 1553](#)
- [Analog Out 3 on page 1554](#)
- [Analog Out 4 on page 1555](#)
- [Analog Out 6 on page 1556](#)
- [Analog Out 8 on page 1557](#)
- [Analog Out 9 on page 1557](#)
- [Analog Out 10 on page 1558](#)
- [Flexible Out 1 on page 1604](#)

Connection example The following illustration shows a connection example and the failure routing unit (FRU) in the Signal branch.



Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Wavetable Voltage Out)

SCALEXIO Hardware Dependencies (Wavetable Voltage Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 1	Analog Out 3	Analog Out 4	Flexible Out 1
Hardware	DS2680 I/O Unit			DS2621 Signal Generation Board
Event generation	✓			✓
Output current range	0 ... +5 mA _{RMS}			0 ... +40 mA _{RMS}
Output voltage range	0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication)
Noise generation	-	✓	-	✓
Noise amplitude range	-	<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication) 	-	<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication)
Resolution	14 bit			16 bit
Gain error	±0.1 % (typ.)			±0.1 % (typ.)
Offset error	±5 mV (typ.)			±2 mV (typ.)
Wavetable period range	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size 		<ul style="list-style-type: none"> ▪ If Downsampling enabled: 4.032 µs ... 292.057 s · Wavetable size 	

Channel Type		Analog Out 1	Analog Out 3	Analog Out 4	Flexible Out 1
		<ul style="list-style-type: none"> ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size 			<ul style="list-style-type: none"> ▪ If Downsampling disabled: 4.032 µs · Wavetable size ... 292.057 s · Wavetable size
Wavetable executions		<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1) 			<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1)
Angular processing unit	Number of slaves	6		1	
	Maximum speed	168,000 °/s			
Maximum trigger angle vector size		<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 			
Configurable fuse		–			<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Circuit diagrams		Analog Out 1 on page 1553	Analog Out 3 on page 1554	Analog Out 4 on page 1555	Flexible Out 1 on page 1604
Required channels		1	1 (additional channels are required for voltage enhancements)	1	1 (additional channels are required for voltage enhancements)

Channel Type	Analog Out 6	Analog Out 8	Analog Out 9	Analog Out 10
Hardware	DS6101 Multi-I/O Board			DS6241 D/A Board
Event generation	✓			
Output current range	0 ... +5 mA _{RMS}			-5 mA ... +5 mA (min.)
Output voltage range	0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	0 ... +10 V	-10 V ... +10 V
Noise generation	–	✓	–	✓
Noise amplitude range	–	<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication) 	–	0 ... +5 V
Resolution	14 bit			16 bit
Gain error	±0.1% (typ.)			±0.03% (typ.)
Offset error	±5 mV (typ.)			±1 mV (typ.)
Wavetable period range	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size 			<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size
Wavetable executions	<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1) 			<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1)

Channel Type		Analog Out 6	Analog Out 8	Analog Out 9	Analog Out 10		
Angular processing unit	Number of slaves	6			6		
	Maximum speed	168,000 °/s		1,200,000 °/s			
Maximum trigger angle vector size	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 		16				
Configurable fuse	–		–				
Circuit diagrams	Analog Out 6 on page 1556		Analog Out 8 on page 1557	Analog Out 9 on page 1557	Analog Out 10 on page 1558		
Required channels	1	1 (additional channels are required for voltage enhancements)		1	1		

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101 on page 1539](#).

General limitation

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6241 D/A Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6241 D/A Board For more board-specific data, refer to [Data Sheet of the DS6241 D/A Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Wavetable Current Sink

Where to go from here

Information in this section

Introduction (Wavetable Current Sink).....	682
Overviews (Wavetable Current Sink).....	683
Configuring the Function Block (Wavetable Current Sink).....	688
Hardware Dependencies (Wavetable Current Sink).....	694

Introduction (Wavetable Current Sink)

Introduction to the Function Block (Wavetable Current Sink)

Function block purpose

The Wavetable Current Sink function block type outputs a current sink signal sequence by looking up a table with time-coded values.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Importing up to 16 wavetable files.
- Controlling wavetable execution (period time, repetitions).
- Selecting different trigger sources for wavetable execution (model-based, angle based, I/O function).
- Switching executed wavetables at runtime.
- Controlling the output signal (amplitude, adding noise and/or offset).
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Wavetable Current Sink function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Analog Out 4	Analog Out 9	Flexible Out 1	-
Hardware	DS2680	DS6101	DS2621	-

Introduction to signal generation based on wavetables

For basic information, refer to [Introduction to Signal Generation Using Time-Coded Wavetables](#) on page 664.

Oversviews (Wavetable Current Sink)

Where to go from here**Information in this section**

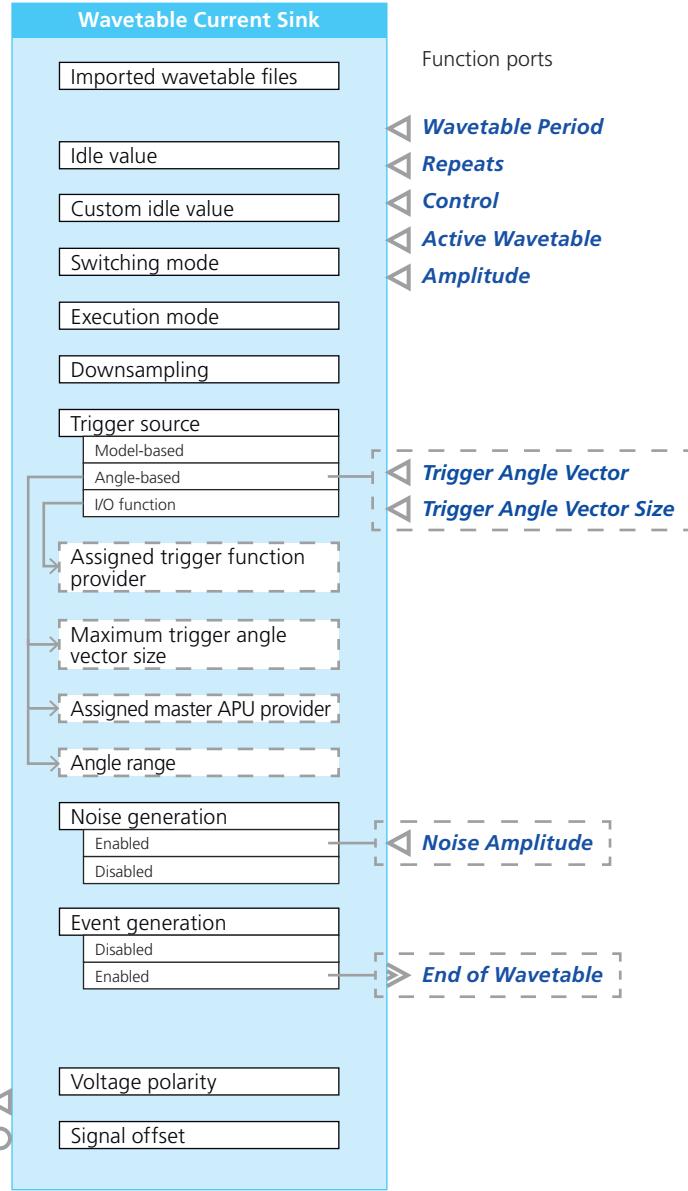
[Overview of Ports and Basic Properties \(Wavetable Current Sink\).....](#) 683

[Overview of Tunable Properties \(Wavetable Current Sink\).....](#) 688

Overview of Ports and Basic Properties (Wavetable Current Sink)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Wavetable Period**

This function import lets you define the wavetable period from within the behavior model. The wavetable period is the time for outputting all wavetable values once, i.e., the duration of one wavetable cycle.

Value range	<ul style="list-style-type: none"> $T_{\min} \dots T_{\max}$ (in seconds). The range depends on the follows: <ul style="list-style-type: none"> The setting of the Downsampling property. The number of values of the active wavetable. The assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Current Sink) on page 694.
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Repeats

This function import lets you define from within the behavior model, how often a wavetable is executed. Additionally, you can use this function import to stop an active wavetable execution.

Value range	<ul style="list-style-type: none"> -1 ... 65,534 -1: Infinite repetitions 0: The wavetable is not executed at all. An active wavetable execution is stopped at the end of the currently executed wavetable.
Dependencies	–

Control

This function import lets you enable the start of a wavetable execution from within the behavior model or stop wavetable execution from within the behavior model.

Value range	<ul style="list-style-type: none"> 1: Start When a trigger event occurs <i>and</i> no wavetable is currently executed, wavetable execution starts and is performed as defined at the Repeats function port. Trigger events are ignored if they occur during wavetable execution. For more information, refer to Configuring the Basic Functionality (Wavetable Current Sink) on page 689. 0: Stop Wavetable execution stops as specified at the Execution mode property. The output changes to the value specified at the Idle value property.
Dependencies	–

Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> 1 ... n n equals the number of imported wavetable files
Dependencies	–

Amplitude

This function import lets you define the amplitude of the output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> I_{min} ... I_{max} (in Ampere).
-------------	----------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ The range depends on the follows: ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Current Sink) on page 694. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... $I_{max\ Noise}$ (in Ampere) ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Current Sink) on page 694. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range. ▪ 0 A: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Noise generation property is set to Enabled. ▪ Supported only for the Flexible Out 1 channel type.

Trigger Angle Vector

This function import lets you trigger the execution of the wavetable from within the behavior model. The wavetable is executed whenever an angle value provided at this function port matches an angle value returned by the master APU provider that is assigned to the function block.

To compare the trigger angles that are provided by the behavior model with the angle values of the master APU provider, the function block converts the trigger angles to the angle range that is specified via the **Angle range** property. For example: The function block converts -10° to 350° if the **Angle range** property is set to 360°.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° for each entry in the vector. ▪ The vector size depends on the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based .

Trigger Angle Vector Size

This function import lets you determine whether all vector entries are considered as trigger angles or only a subset of leading entries from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

End of Wavetable

This event port provides an I/O event each time a wavetable execution has been finished.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port sinks current. It represents the electrical connection point of the logical signal of the function block.

Note

The hardware is passive and adjusts its internal resistance so that the defined current can flow at the signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Current Sink) on page 694.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Current Sink) on page 694.
Dependencies	–

Overview of Tunable Properties (Wavetable Current Sink)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Imported Wavetable Files	– ³⁾	–
	Idle Value	✓	–
	Switching Mode	✓	–
	Execution Mode	✓	–
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓
Electrical Interface			
	Signal offset	✓	–
	Voltage polarity	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ The content of the imported wavetable files can be updated in simulation STOP state.

Configuring the Function Block (Wavetable Current Sink)

Where to go from here

Information in this section

Configuring the Basic Functionality (Wavetable Current Sink).....	689
Configuring Standard Features (Wavetable Current Sink).....	693

Configuring the Basic Functionality (Wavetable Current Sink)

Overview

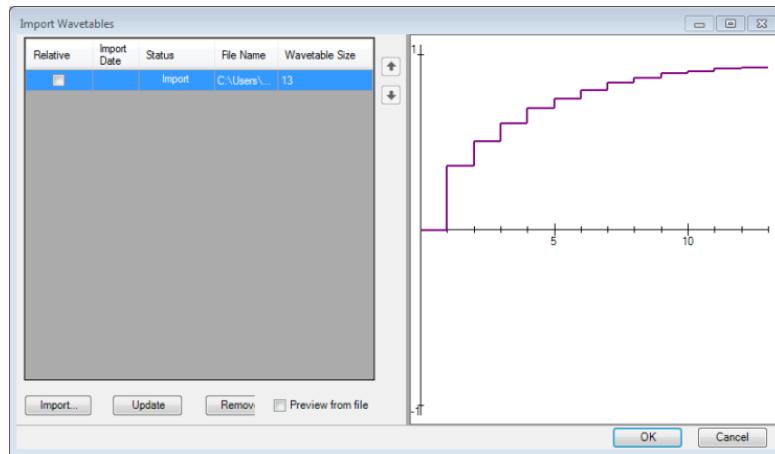
The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Managing wavetable files
- Controlling wavetable execution
- Adding noise to the output signal
- Providing I/O events
- Specifying a signal offset
- Specifying the signal polarity

Managing wavetable files

Wavetable Current Sink function blocks require at least one wavetable for signal generation. If you do not import a wavetable, the default wavetable is used (all values equal 0).

The Import Wavetables dialog lets you import, update and remove wavetable files.



You open the dialog via the Browse button of the Imported wavetable files property. You can import up to 16 wavetable files.

Controlling wavetable execution

Wavetable execution has the following control options:

Repeated wavetable execution You can execute a wavetable once or repeatedly (repeated n-times or infinitely), depending on the value provided at the Repeats function port. Executing a wavetable repeatedly means that the

function block generates a periodic output signal (period = wavetable period T), for example, a sine wave sensor signal.

Changing the wavetable period at run time You can update the wavetable period at run time by changing the value at the **Wavetable period** function port. The update takes effect immediately.

Switching wavetables at run time You can substitute one wavetable by another one at run time by changing the value at the **Active Wavetable** function port. The new wavetable is executed immediately or after the execution of the current wavetable has finished, depending on the **Switching Mode** property.

Stopping wavetable execution You can stop an active wavetable execution by setting the **Repeats** function port to 0 (zero) or by setting the **Control** function port to **Stop**.

- If you set the **Repeats** function port to 0 (zero), wavetable execution stops at the end of the currently executed wavetable.
- If you set the **Control** function port to **Stop**, wavetable execution stops depending on the setting of the **Execution mode** property and the value at the **Repeats** function port.

Execution Mode	Repeats	Stop Behavior
State dependent	1 ... 65,534	Wavetable execution stops after the last repetition.
	-1	Wavetable execution does not stop.
Immediate	Not relevant	Wavetable execution stops immediately.

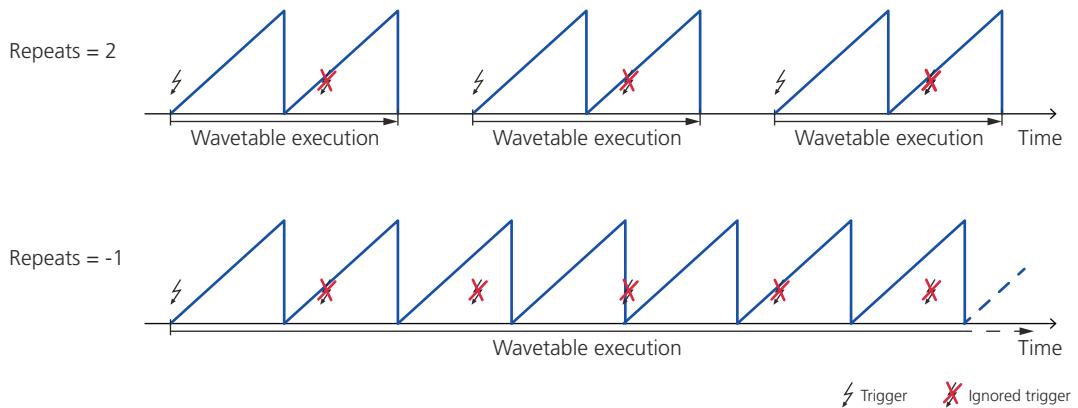
When wavetable execution stops, the function block outputs the specified **Idle** value of the wavetable.

Starting wavetable execution A trigger event from the specified trigger source starts wavetable execution if the following conditions are met:

- The value provided at the **Control** function port is **Start**.
- The value provided at the **Repeats** function port is $<> 0$.
- No wavetable is currently executed.

When wavetable execution starts, it is performed according to the value provided at the **Repeats** function port.

Trigger events are ignored if they occur during wavetable execution. This is shown in the following illustration for **Repeats** values of 2 and -1 (infinite):



You have to specify a trigger source at the **Trigger source** property:

- Model-based: Each model step causes a trigger event.
- Angle-based: A trigger event occurs when the assigned master APU provider returns an angle value that matches the angle value provided at the **Trigger Angle Vector** function port.
As a precondition for angle-based triggering you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.
- I/O function: The assigned trigger function triggers the wavetable execution.

Adding noise to the output signal

The Wavetable Current Sink function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Limitation Noise generation is supported only for the Flexible Out 1 channel type.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Wavetable Out function blocks can generate an I/O event each time a wavetable execution has been finished.

To use the I/O events in the behavior model, you have to assign them to a task via the **End of Wavetable** property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Specifying a signal offset

You can apply a voltage offset to the output signal. The offset is channel-specific, hence, all the signals that use the same I/O channel have the same signal offset. This is helpful to generate unipolar output, for example, positive voltages, even with bipolar wavetable values.

Limitation Specifying a signal offset is not supported for the Analog Out 3 and Analog Out 8 channel types.

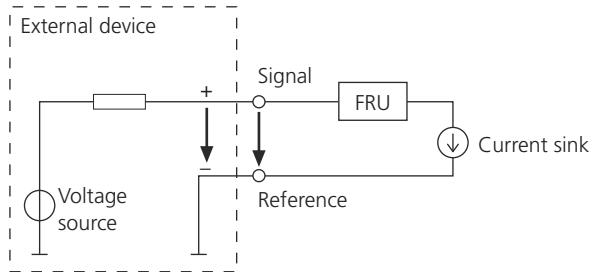
Specifying the signal polarity

The Wavetable Current Sink function block is a current sink and must be fed by an external voltage source. Depending on the selected channel type, the voltage polarity affects the functionality of the function block.

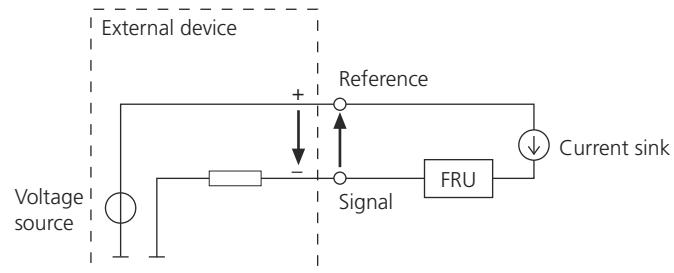
Flexible Out 1 channel type You must specify the voltage polarity of the signal ports with the *Voltage polarity* property to ensure the correct behavior of the function block.

Note that changing the *Voltage polarity* property also changes the signal path that the failure routing unit (FRU) is used on. This means that you can use the FRU on either the one or the other signal path by switching the polarity of the voltage. The following illustration shows a connection example with an FRU on the signal path.

Positive polarity



Negative polarity



For details on the FRU, refer to [Electrical Error Simulation Concept \(SCALEXIO Hardware Installation and Configuration\)](#).

Note

The Wavetable Current Sink function block adjusts only the current. The feed voltage must come from an external source as shown in the illustrations above.

The Voltage polarity property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

Limitation Specifying the signal polarity is not supported for the Analog Out 4 and Analog Out 9 channel types. For these channel types, the voltage polarity does not affect the functionality of the function block. The behavior is correct if the voltage polarity of the signal is negative as well as positive.

Configuring Standard Features (Wavetable Current Sink)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Configuration Feature	Analog Out 4	Analog Out 9	Flexible Out 1	Further Information
Channel multiplication	✓	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 4](#) on page 1555
- [Analog Out 9](#) on page 1557
- [Flexible Out 1](#) on page 1604

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Wavetable Current Sink)

SCALEXIO Hardware Dependencies (Wavetable Current Sink)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 4	Flexible Out 1	Analog Out 9
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board
Event generation	✓	✓	✓
Output current range	<ul style="list-style-type: none"> ▪ +0.1 mA ... +30 mA for 1 channel ▪ 0.24 A for 8 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ +0.2 µA ... +40 mA for 1 channel ▪ 0.4 A for 10 channels (channel multiplication) 	+0.1 mA ... +30 mA
Output voltage range	<p>The following conditions must be met:</p> <ul style="list-style-type: none"> ▪ Range for the lower potential of signal and reference: -2 V ... +18 V ▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V 	-60 V ... +60 V	<p>The following conditions must be met:</p> <ul style="list-style-type: none"> ▪ Range for the lower potential of signal and reference: -2 V ... +18 V ▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V
Noise generation	-	✓	-
Noise amplitude range	-	<ul style="list-style-type: none"> ▪ 0 ... 10 mA for 1 channel 	-

Channel Type		Analog Out 4	Flexible Out 1	Analog Out 9
			<ul style="list-style-type: none"> ▪ 0 ... 100 mA for 10 channels (channel multiplication) 	
Resolution	14 bit	15 bit	14 bit	
Gain error	±0.2 % (typ.)	±0.1 % (typ.)	±0.2 % (typ.)	
Offset error	±20 µA (typ.)	±10 µA (typ.)	±30 µA (typ.)	
Wavetable period range		<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size 	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 4.032 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 4.032 µs · Wavetable size ... 292.057 s · Wavetable size 	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size
Wavetable executions		<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1) 	<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1) 	<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1)
Angular processing unit	Number of slaves	6	1	6
	Maximum speed	168,000 °/s		
Maximum trigger angle vector size		<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 		
Configurable fuse	–		<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–
Circuit diagrams	Analog Out 4 on page 1555		Flexible Out 1 on page 1604	Analog Out 9 on page 1557
Required channels	1 (additional channels are required for current enhancements.)			1

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101](#) on page 1539.

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Wavetable Digital Out

Where to go from here

Information in this section

Introduction (Wavetable Digital Out).....	696
Overviews (Wavetable Digital Out).....	697
Configuring the Function Block (Wavetable Digital Out).....	702
Hardware Dependencies (Wavetable Digital Out).....	707

Introduction (Wavetable Digital Out)

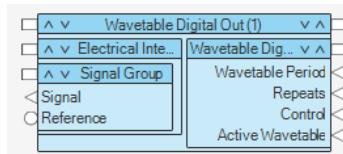
Introduction to the Function Block (Wavetable Digital Out)

Function block purpose

The Wavetable Digital Out function block type outputs a digital signal sequence by looking up a table with time-coded values.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Importing up to 16 wavetable files.
- Controlling wavetable execution (period time, repetitions).
- Selecting different trigger sources for wavetable execution (model-based, angle based, I/O function).
- Switching executed wavetables at runtime.
- Generating I/O events and providing them to the behavior model.

Supported channel types

Wavetable Digital Out The Wavetable Digital Out function block supports the following channel types:

	SCALEXIO				MicroAutoBox III
Channel type	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	–
Hardware	DS2680	DS6101	DS6202	DS2621	–

Introduction to signal generation based on wavetables

For basic information, refer to [Introduction to Signal Generation Using Time-Coded Wavetables](#) on page 664.

Oversviews (Wavetable Digital Out)

Where to go from here**Information in this section**

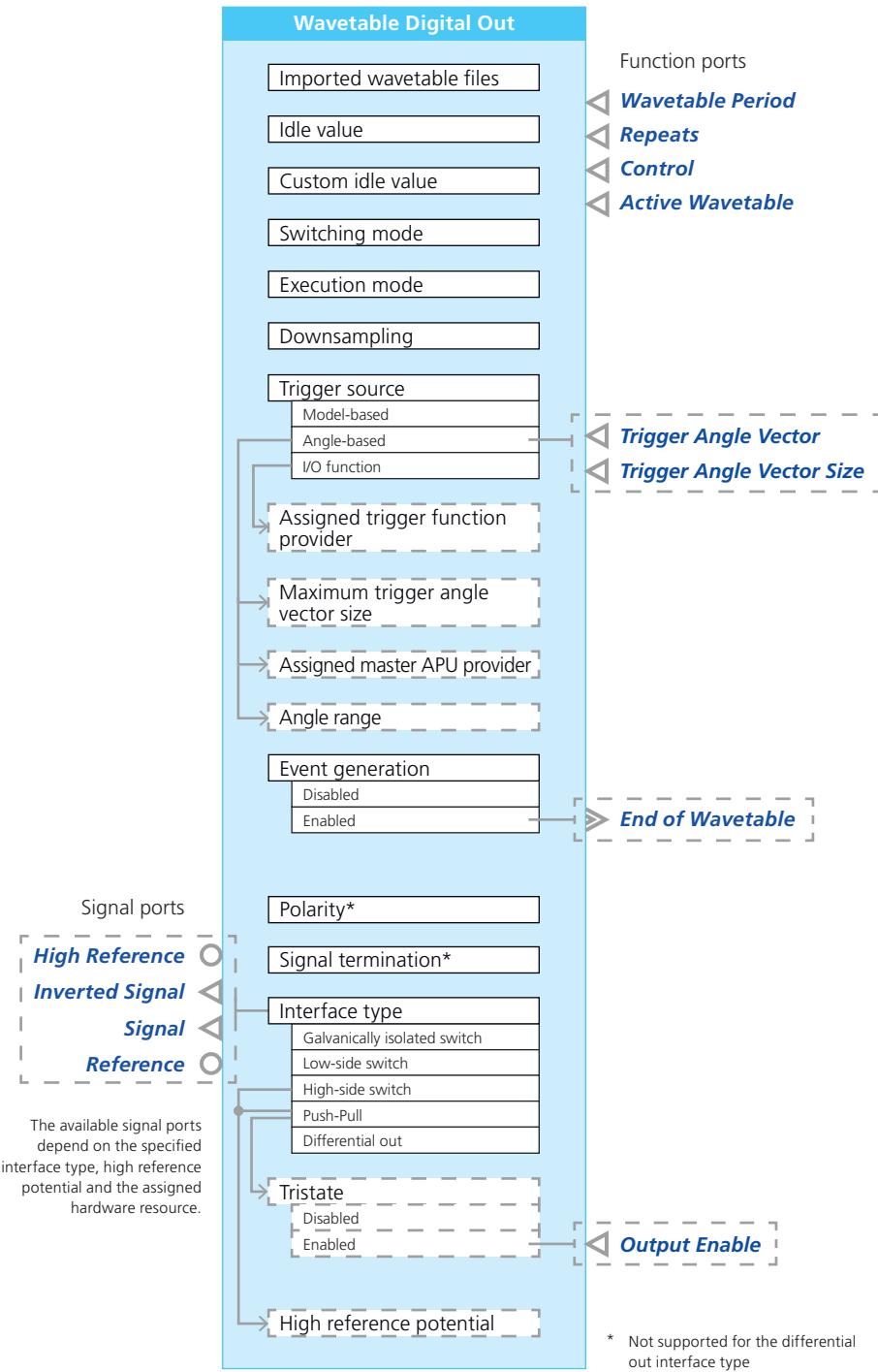
[Overview of Ports and Basic Properties \(Wavetable Digital Out\)](#)..... 697

[Overview of Tunable Properties \(Wavetable Digital Out\)](#)..... 702

Overview of Ports and Basic Properties (Wavetable Digital Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Wavetable Period**

This function import lets you define the wavetable period from within the behavior model. The wavetable period is the time for outputting all wavetable values once, i.e., the duration of one wavetable cycle.

Value range	<ul style="list-style-type: none"> ▪ $T_{\min} \dots T_{\max}$ (in seconds). ▪ The range depends on the follows: <ul style="list-style-type: none"> ▪ The setting of the Downsampling property. ▪ The number of values of the active wavetable. ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Digital Out) on page 707. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Repeats

This function import lets you define from within the behavior model, how often a wavetable is executed. Additionally, you can use this function import to stop an active wavetable execution.

Value range	<ul style="list-style-type: none"> ▪ -1 ... 65,534 ▪ -1: Infinite repetitions ▪ 0: The wavetable is not executed at all. An active wavetable execution is stopped at the end of the currently executed wavetable.
Dependencies	–

Control

This function import lets you enable the start of a wavetable execution from within the behavior model or stop wavetable execution from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1: Start <p>When a trigger event occurs <i>and</i> no wavetable is currently executed, wavetable execution starts and is performed as defined at the Repeats function port.</p> <p>Trigger events are ignored if they occur during wavetable execution. For more information, refer to Configuring the Basic Functionality (Wavetable Digital Out) on page 703.</p> ▪ 0: Stop <p>Wavetable execution stops as specified at the Execution mode property. The output changes to the value specified at the Idle value property.</p>
Dependencies	–

Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the number of imported wavetable files
Dependencies	–

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.

Trigger Angle Vector

This function import lets you trigger the execution of the wavetable from within the behavior model. The wavetable is executed whenever an angle value provided at this function port matches an angle value returned by the master APU provider that is assigned to the function block.

To compare the trigger angles that are provided by the behavior model with the angle values of the master APU provider, the function block converts the trigger angles to the angle range that is specified via the Angle range property. For example: The function block converts -10° to 350° if the Angle range property is set to 360°.

Value range	<ul style="list-style-type: none"> ▪ -3600° ... +3600° for each entry in the vector. ▪ The vector size depends on the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

Trigger Angle Vector Size

This function import lets you determine whether all vector entries are considered as trigger angles or only a subset of leading entries from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

End of Wavetable

This event port provides an I/O event each time a wavetable execution has been finished.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Digital Out) on page 707.
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Digital Out) on page 707.
Dependencies	–

Inverted Signal

This signal port and the Signal signal port together represent the electrical connection points of a logical signal with two complementary potentials carrying the information in the voltage difference between these signals. Both signal ports (Signal and Inverted Signal) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Digital Out) on page 707.
Dependencies	Available only if the Interface type property is set to Differential out.

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wavetable Digital Out) on page 707.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Wavetable Digital Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Imported Wavetable Files	- ³⁾	-
	Idle Value	✓	-
	Switching Mode	✓	-
	Execution Mode	✓	-
Function Ports			
	Initial switch setting (Test Automation)	-	✓
	Initial substitute value (Test Automation)	-	✓
Electrical Interface			
	Polarity	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ The content of the imported wavetable files can be updated in simulation STOP state.

Configuring the Function Block (Wavetable Digital Out)

Where to go from here

Information in this section

- | | |
|------------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Wavetable Digital Out)..... | 703 |
| Configuring Standard Features (Wavetable Digital Out)..... | 705 |

Configuring the Basic Functionality (Wavetable Digital Out)

Overview

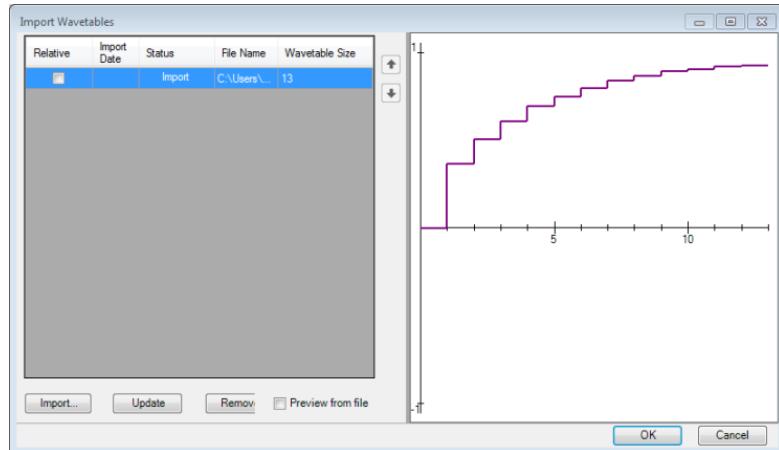
The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Managing wavetable files
- Controlling the wavetable execution
- Reducing voltage level overshoots
- Providing I/O events

Managing wavetable files

Wavetable Digital Out function blocks require at least one wavetable for signal generation. If you do not import a wavetable, the default wavetable is used (all values equal 0).

The Import Wavetables dialog lets you import, update and remove wavetable files.



You open the dialog via the Browse button of the Imported wavetable files property. You can import up to 16 wavetable files.

Controlling wavetable execution

Wavetable execution has the following control options:

Repeated wavetable execution You can execute a wavetable once or repeatedly (repeated n-times or infinitely), depending on the value provided at the Repeats function port. Executing a wavetable repeatedly means that the function block generates a periodic output signal (period = wavetable period T), for example, a sine wave sensor signal.

Changing the wavetable period at run time You can update the wavetable period at run time by changing the value at the Wavetable period function port. The update takes effect immediately.

Switching wavetables at run time You can substitute one wavetable by another one at run time by changing the value at the Active Wavetable function port. The new wavetable is executed immediately or after the execution

of the current wavetable has finished, depending on the **Switching Mode** property.

Stopping wavetable execution You can stop an active wavetable execution by setting the **Repeats** function port to 0 (zero) or by setting the **Control** function port to **Stop**.

- If you set the **Repeats** function port to 0 (zero), wavetable execution stops at the end of the currently executed wavetable.
- If you set the **Control** function port to **Stop**, wavetable execution stops depending on the setting of the **Execution mode** property and the value at the **Repeats** function port.

Execution Mode	Repeats	Stop Behavior
State dependent	1 ... 65,534	Wavetable execution stops after the last repetition.
	-1	Wavetable execution does not stop.
Immediate	Not relevant	Wavetable execution stops immediately.

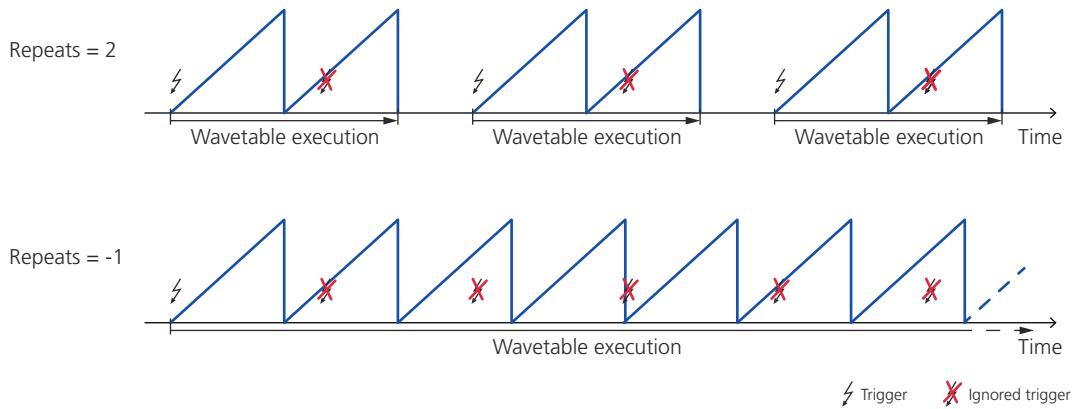
When wavetable execution stops, the function block outputs the specified **Idle** value of the wavetable.

Starting wavetable execution A trigger event from the specified trigger source starts wavetable execution if the following conditions are met:

- The value provided at the **Control** function port is **Start**.
- The value provided at the **Repeats** function port is $<> 0$.
- No wavetable is currently executed.

When wavetable execution starts, it is performed according to the value provided at the **Repeats** function port.

Trigger events are ignored if they occur during wavetable execution. This is shown in the following illustration for **Repeats** values of 2 and -1 (infinite):



You have to specify a trigger source at the **Trigger source** property:

- Model-based: Each model step causes a trigger event.
- Angle-based: A trigger event occurs when the assigned master APU provider returns an angle value that matches the angle value provided at the **Trigger Angle Vector** function port.

As a precondition for angle-based triggering you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

- I/O function: The assigned trigger function triggers the wavetable execution.

Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

Limitation A resistor connected in series to the output driver is supported only by the Digital In/Out 5 channel type. However, signal termination is not supported if you use the Differential out interface type. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Wavetable Out function blocks can generate an I/O event each time a wavetable execution has been finished.

To use the I/O events in the behavior model, you have to assign them to a task via the End of Wavetable property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Standard Features (Wavetable Digital Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<p>Depending on the channel type, refer to:</p> <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112 ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109 ▪ Specifying Digital Output Signals (Flexible Out 1) on page 98
Polarity of output signals	✓	✓	<ul style="list-style-type: none"> ▪ ✓ ▪ Not supported for the differential out interface type. 	✓	–
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.				–
Channel multiplication	✓	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports					
Trigger level of electronic fuse	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 1 on page 1575](#)
- [Digital Out 3 on page 1577](#)
- [Digital In/Out 5 on page 1586](#)
- [Flexible Out 1 on page 1604](#)

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Wavetable Digital Out)

SCALEXIO Hardware Dependencies (Wavetable Digital Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board	DS6202 Digital I/O Board	DS2621 Signal Generation Board
Event generation	✓	✓	✓	✓
Output current range	<ul style="list-style-type: none"> ▪ 0 ... +80 mA_{RMS} for 1 channel ▪ 1.792 A_{RMS} for 28 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +140 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	0 ... ±40 mA (each channel)	<ul style="list-style-type: none"> ▪ 0 ... +40 mA for 1 channel ▪ 0.32 A for 10 channels (channel multiplication)
High side reference voltage	+5 V ... +60 V	+5 V ... +60 V	<ul style="list-style-type: none"> ▪ +3.3 V and +5 V (switchable) ▪ +3.3 V (fix) for the differential out interface type 	-60 V ... +60 V
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch 	<ul style="list-style-type: none"> ▪ Low-side switch 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch

Channel Type	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1
	<ul style="list-style-type: none"> ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch to internal reference ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull
In push-pull configuration the outputs can be set to high impedance (tristate) at run time.				
Wavetable period range	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size 	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size 	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 5.232 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 5.232 µs · Wavetable size ... 292.057 s · Wavetable size 	<ul style="list-style-type: none"> ▪ If Downsampling enabled: 4.032 µs ... 292.057 s · Wavetable size ▪ If Downsampling disabled: 4.032 µs · Wavetable size ... 292.057 s · Wavetable size
Wavetable executions	<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1) 	<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1) 	<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1) 	<ul style="list-style-type: none"> ▪ 0 ... 65,534 ▪ infinite (for Repeats = -1)
Angular processing unit	Number of slaves	6	6	1
	Maximum speed	168,000 °/s	168,000 °/s	1,200,000 °/s
Maximum trigger angle vector size	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 	16	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range
Configurable fuse	–	–	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Signal termination	–	–	<ul style="list-style-type: none"> ▪ 68 Ω (serial termination) ▪ Switchable ▪ Not supported for the differential out interface type 	–
Circuit diagram	Digital Out 1 on page 1575	Digital Out 3 on page 1577	Digital In/Out 5 on page 1586	Flexible Out 1 on page 1604
Required channels	1 (additional channels are required for current enhancement and operation in push/pull mode.)	1	1 (2 channels are required to operate the outputs using the differential interface type.)	1 (additional channels are required for current enhancement and operation in push-pull mode.)

General limitations

Channel multiplication across several I/O boards is not supported.

DS2621 Signal Generation Board The following restrictions apply to the DS2621 Signal Generation Board when the push/pull mode is used:

- Every logical signal requests two channels (primary and auxiliary) on the real-time hardware to be assigned to the function block. These channels must be adjacent to one another on an I/O board. Example: primary channel = n, auxiliary channel = n + 1.
- Channel 10 cannot be used as the primary channel because using channel 1 as the next adjacent channel is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6202 Digital I/O Board The following limitations apply to the DS6202 Digital I/O Board:

- Channel multiplication is not supported in general.
- Only 16 Wavetable Digital Out function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Angular Wavetable (Voltage Out, Digital Out)

Objective To generate periodic signals by looking up a table with function values.

Where to go from here	Information in this section						
	<table><tr><td>Introduction to Signal Generation Using Angular-Coded Wavetables.....</td><td>712</td></tr><tr><td>Angular Wavetable Voltage Out.....</td><td>719</td></tr><tr><td>Angular Wavetable Digital Out.....</td><td>730</td></tr></table>	Introduction to Signal Generation Using Angular-Coded Wavetables.....	712	Angular Wavetable Voltage Out.....	719	Angular Wavetable Digital Out.....	730
Introduction to Signal Generation Using Angular-Coded Wavetables.....	712						
Angular Wavetable Voltage Out.....	719						
Angular Wavetable Digital Out.....	730						

Introduction to Signal Generation Using Angular-Coded Wavetables

Where to go from here

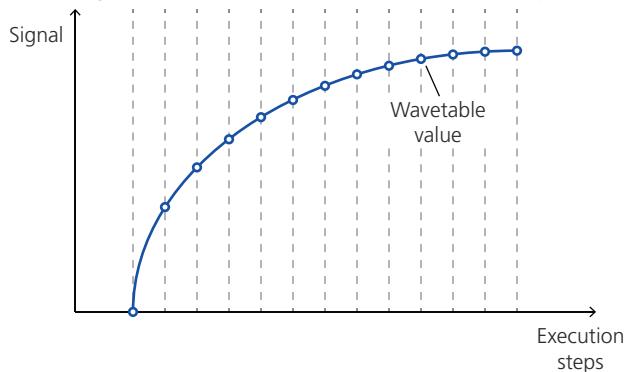
Information in this section

Basics on Signal Generation Using Angular-Coded Wavetables.....	712
Specifying Angular-Coded Wavetable Files.....	715
Replacing Angular-Coded Wavetable Files at Run Time.....	716

Basics on Signal Generation Using Angular-Coded Wavetables

The wavetable approach

A wavetable is an array of function values, also called wavetable values. When the wavetable is executed, the wavetable values form the output signal. Executing a wavetable means that its values are output consecutively.



Wavetables can be executed as a function of time (time-coded wavetables) or as a function of angle positions (angular-coded wavetables). In the following, angular-coded wavetables are described.

Field of application

Use wavetables if you want to simulate periodic signals that are static, which is contrary to the dynamic waveforms.

Analog output function block Use the Angular Wavetable Voltage Out function block if you want to simulate high-resolution signals, for example, the exact shape of the tooth of a crankshaft wheel.

Digital output function block Use the Angular Wavetable Digital Out function block if you want to simulate sensors, for example, a digital crankshaft sensor.

Wavetable execution

Wavetables are executed on the I/O board and the I/O board generates the output signal. This barely affects the CPU, because the wavetable values are not computed at run time, which makes the process highly efficient.

Execution of angular-coded wavetables Executing an angularly coded wavetable means that its values are output consecutively with the angle position of an assigned master angle processing unit (APU) provider.

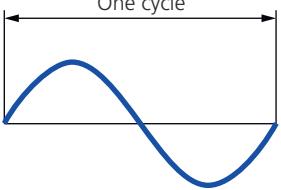
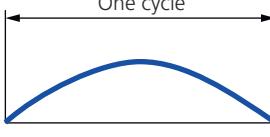
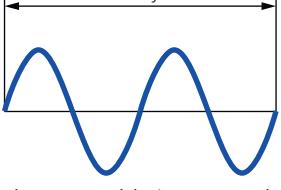
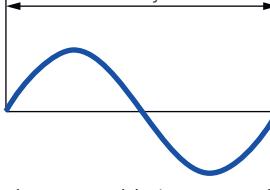
The master APU provider gives access to an APU that is implemented as an angle counter. The steps of the referenced APU always relate to one engine cycle.

However, one engine cycle can cover 360° or 720°. For example: A four-stroke piston engine has a 720° angle range for one engine cycle and an electric motor a 360° angle range.

Therefore, there are two types of angular-coded wavetables that you can execute with the **Angular Wavetable Out** function blocks:

- Wavetables that provide 32,768 angle values for engines with a 360° angle range (360° wavetable).
- Wavetables that provide 65,536 angle values for engines with a 720° angle range (720° wavetable).

The table below describes the signals that are output in the different constellations of the specified angle range and executed wavetable types. The example wavetable includes the values of one complete sine wave.

	360° Wavetable	720° Wavetable
360° Angle Range	 <p>The wavetable is executed once if the function block runs through the 360° angle range.</p>	 <p>Only the first half of the wavetable is executed if the function block runs through the 360° angle range. The function block never outputs the second half.</p>
720° Angle Range	 <p>The wavetable is executed twice if the function block runs through the 720° angle range.</p>	 <p>The wavetable is executed once if the function block runs through the 720° angle range.</p>

The used angle range can be specified for the angular wavetable function blocks. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Conditioning the output signal

The wavetable values are normalized values, hence they are not returned as the actual output signal. Conditioning lets you generate output signals with different amplitudes that all refer to the same wavetable. The processing of the wavetable values depends on the function block.

Analog output function block The Angular Wavetable Voltage Out function block multiplies the wavetable values by the amplitude. The amplitude is provided by the Amplitude function port.

Output Signal = Wavetable Value * Amplitude

The table below shows an example wavetable and the resulting output signal for an amplitude of 20 V.

Wavetable Value	Output Signal
0.0	0 V
0.5	10 V
-1.0	-20 V

Digital output function block The Angular Wavetable Digital Out function block interprets the wavetable values either as 0 or 1.

Signal = 0, if wavetable value ≤ 0.0

Signal = 1, if wavetable value > 0.0

The table below shows an example wavetable and the resulting output signal.

Wavetable Value	Output Signal
0.0	Low
0.5	High
-1.0	Low

Switching between wavetables at run time

You can switch between imported wavetables from the behavior model at run time to use different wavetables. You can also switch between 360° and 720° wavetables.

The number of wavetables you can import depends on the onboard memory of the I/O board. For specific values, refer to:

- [Hardware Dependencies \(Angular Wavetable Voltage Out\) on page 727](#).
 - [Hardware Dependencies \(Angular Wavetable Digital Out\) on page 738](#).
-

Updating angular-coded wavetables

The `downwave.exe` utility replaces one of the wavetables linked to an Angular Wavetable function block when the real-time application is stopped by the experiment tool. The change takes effect when the application status changes from stopped to running.

For more information, refer to [Replacing Angular-Coded Wavetable Files at Run Time](#) on page 716.

Specifying Angular-Coded Wavetable Files

Requirements of a wavetable file A wavetable file is a CSV file with wavetable values. The CSV file must meet the following requirements:

- The file is an ASCII file.
- One CSV file provides the values of only one wavetable.
- Entries are separated by a comma (',').
- New line at the end of a file.
- No additional data or text is included.
- Wavetable values consist of floating-point values (e.g., 1.0).
- The format of the decimal point is a dot ('.') .
- The values are in the range [-1.0 ... 1.0].

Specifying angular-coded wavetables Angular-coded wavetables mainly differ in the supported angle range for one engine cycle (360° or 720°). The angle range defines the number of wavetable entries:

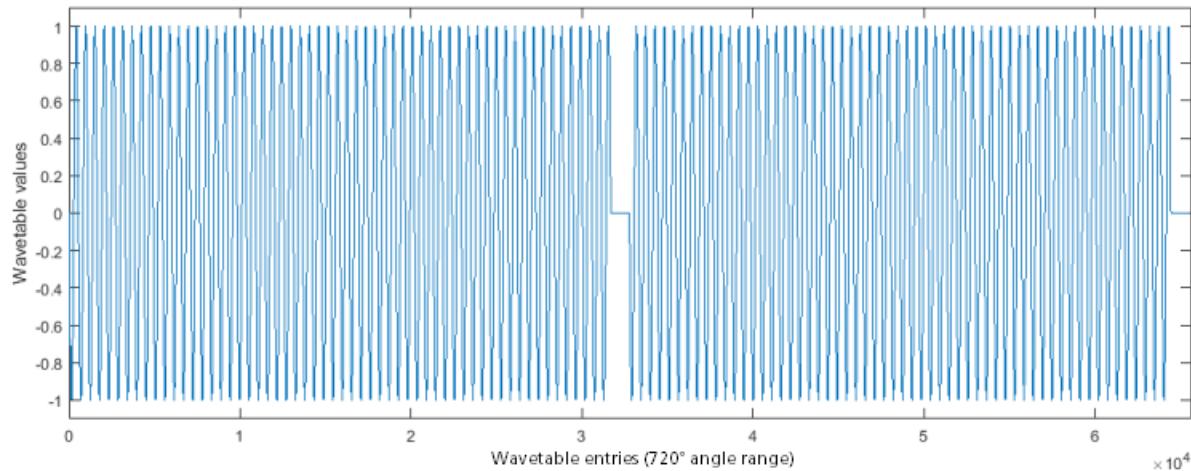
- A 360° wavetable must provide 32,768 values for one engine cycle.
- A 720° wavetable must provide 65,536 values for one engine cycle.

Note

If the executed wavetable supports a different angle range than the referenced master APU provider, the function block outputs the wavetable twice or only the first part with one engine cycle. Refer to [Execution of angular-coded wavetables](#) on page 713.

If you simulate a crankshaft/camshaft sensor with a 720° angle range, you can use MATLAB scripts (M files) to create wavetable files for these sensors. Refer to [Creating Crankshaft/Camshaft Wavetables](#) on page 458.

The following illustration shows the crankshaft example that resides in the `<DocumentsFolder>\EngineConfigurationDemo\EngineConfiguration\EngineExample\Wavetables` folder.



The angular-coded wavetable file of this example looks like the following extraction:

```
-0,-0.011505,-0.023008,-0.034508,-0.046003,-0.057493,  
-0.068974,-0.080447,-0.091909,-0.10336,-0.11479,-0.12622,  
-0.13762,-0.14901,-0.16037,-0.17172,-0.18304,-0.19434,  
-0.20561,-0.21686,-0.22807,-0.23926,-0.25041,-0.26153,-0.27262,  
-0.28367,-0.29469,-0.30566,-0.31659,-0.32749,-0.33833,-0.34914,  
-0.3599,-0.3706,-0.38127, ...
```

The precision of an analog output signal depends on the digital-to-analog converter of the assigned hardware resource.

Related topics

Basics

Hardware Dependencies (Angular Wavetable Digital Out).....	738
Hardware Dependencies (Angular Wavetable Voltage Out).....	727

Replacing Angular-Coded Wavetable Files at Run Time

downwave.exe

The `downwave.exe` utility replaces one of the wavetables linked to an Angular Wavetable function block when the real-time application is stopped by the experiment tool. The change takes effect when the application status changes from stopped to running. The utility is located in `<InstallationFolder>\SCALEXIO\Win32`.

Syntax

The `downwave.exe` utility is invoked by using the following syntax:

```
downwave.exe [general options] [action-specific options]
[download options]
```

The following *general* options are available:

/b IP address of the real-time system or an alias name (refer to the **hosts** file in **%SYSTEM_ROOT%\system32\drivers\etc** which contains the mapping of host names to IP addresses).

/appl Name of the application

/id Identifier of the application

/t Identifier of the target board, e.g., DS2621 (default), DS2680, or DS6101.

/q Quiet mode (only error messages are issued)

/v Display verbose information

? Display available options

The following *action-specific* options are available (select one):

/list List all the wavetables that relate to the application.

/rep Replace an existing wavetable file (default). This requires the *download* options (see below).

/new Download a new wavetable file. Replacing existing wavetables is disabled.

/f Force download of wavetables. Existing wavetables are replaced, missing wavetables are downloaded.

The following *download* options are available:

/block Name of the function block to which the wavetable to be replaced is linked, for example:

Use " " when the string includes blanks.

/idx Index of the wavetable that is to be replaced. To identify a wavetable's index, check the **Imported wavetable files** property of the function block in the **Properties Browser**. The wavetable at the beginning of the list is indexed as 1.

/src Path and name of the replacing wavetable file (CSV file), for example:

You must specify the path if the wavetable file is stored in a different folder than the **downwave** utility. Use " " when the string includes blanks.

Example #1

The command **downwave.exe /b 192.168.1.1 /list** lists all the wavetables that are linked to the application loaded on the real-time system whose IP address is 192.168.1.1. A message like the following is displayed:

```
downwave.exe - Wavetable download tool for SCALEXIO, Ver. 1.1 - 32,
(C) 2010 by dSPACE GmbH
```

```
List of wavetables for application 'Application_001.x86' (ID:
0x7891F7) on target (IP: 192.168.1.1).
```

```
Function block: ' Digital Out 1'  
Index 1 (source file: _1)  
Index 2 (source file: _2)  
Index 3 (source file: _3)  
Index 4 (source file: _4)  
Function block: ' Digital Out 2'  
Index 2 (source file: _0)
```

Example #2

The command

`downwave.exe /b 192.168.1.1 /src _11.csv /block
" Digital Out 1" /idx 1` performs the following replacement:

- Replaced wavetable:

This wavetable is stored on the real-time system whose IP address is 192.168.1.1. The wavetable is linked to the Digital Out 1 function block. It is the 1st entry in the Wavetable files list (`_1.csv`).

- Replacing wavetable:

This is the `_11.csv` wavetable file. It is stored in the same folder as the `downwave.exe` utility.

Angular Wavetable Voltage Out

Where to go from here

Information in this section

Introduction (Angular Wavetable Voltage Out).....	719
Overviews (Angular Wavetable Voltage Out).....	720
Configuring the Function Block (Angular Wavetable Voltage Out).....	723
Hardware Dependencies (Angular Wavetable Voltage Out).....	727

Introduction (Angular Wavetable Voltage Out)

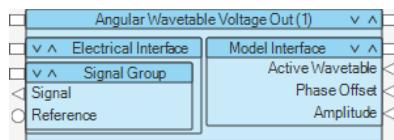
Introduction to the Function Block (Angular Wavetable Voltage Out)

Function block purpose

The Angular Wavetable Voltage Out function block outputs a voltage signal sequence by looking up a table with angular-coded values.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Importing up to 24 wavetable files (hardware dependent).
- Switching executed wavetables at runtime.
- Adjusting the 0° position of the wavetable.
- Controlling the output signal (amplitude, adding noise and/or offset).

Supported channel types

The Angular Wavetable Voltage Out function block supports the following channel types:

	SCALEXIO								MicroAutoBox III
Channel type	Analog Out 1	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 8	Analog Out 9	Analog Out 10	Flexible Out 1	–
Hardware	DS2680			DS6101				DS6241	DS2621

Introduction to signal generation based on wavetables

For basic information, refer to [Introduction to Signal Generation Using Angular-Coded Wavetables](#) on page 712.

Oversviews (Angular Wavetable Voltage Out)

Where to go from here

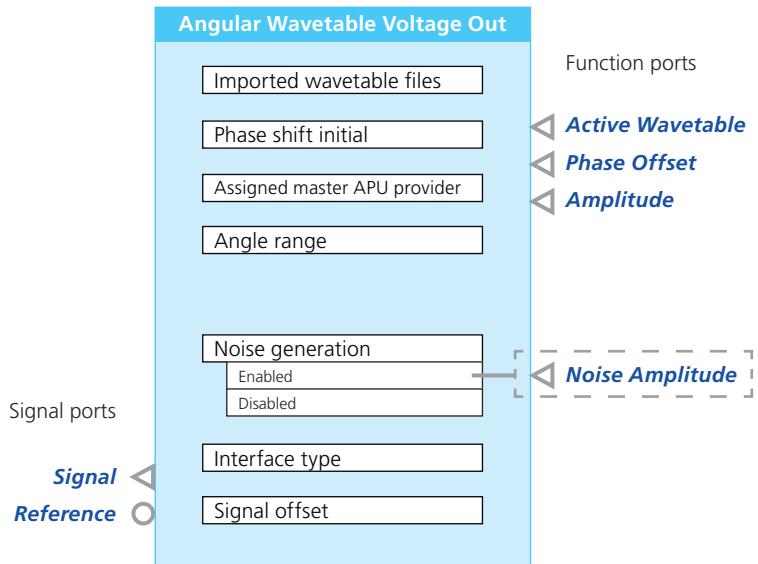
Information in this section

- | | |
|-----------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Angular Wavetable Voltage Out)..... | 720 |
| Overview of Tunable Properties (Angular Wavetable Voltage Out)..... | 723 |

Overview of Ports and Basic Properties (Angular Wavetable Voltage Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the number of imported wavetable files.
Dependencies	–

Phase Offset

This function import lets you define the 0° position independently of the active wavetable from within the behavior model.

The phase offset is set relative to the setting of the Phase shift initial property.

Value range	<ul style="list-style-type: none"> ▪ -360° ... +360° ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	–

Amplitude

This function import lets you define the amplitude of the output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... U_{max} (in Volt) ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Voltage Out) on page 727. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	–

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none">▪ 0 ... $U_{\max \text{ Noise}}$ (in Volt)▪ The range depends on the following:<ul style="list-style-type: none">▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Voltage Out) on page 727.▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.▪ 0 V: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none">▪ Available only if the Noise generation property is set to Enabled.▪ Supported for Flexible Out 1, Analog Out 3, Analog Out 8, and Analog Out 10 channel types.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Voltage Out) on page 727 .
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Voltage Out) on page 727.
Dependencies	–

Overview of Tunable Properties (Angular Wavetable Voltage Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Imported wavetable files	✓ ³⁾	-
Function Ports			
	Initial switch setting (Test Automation)	-	✓
	Initial substitute value (Test Automation)	-	✓
Electrical Interface			
	Phase shift initial	✓	-
	Signal offset	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ The content of the imported wavetable files can be updated when the real-time application is stopped. Wavetable files can be replaced by the [downwave assistance tool](#), refer to [Replacing Angular-Coded Wavetable Files at Run Time](#) on page 716.

Configuring the Function Block (Angular Wavetable Voltage Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (Angular Wavetable Voltage Out).....	724
Configuring Standard Features (Angular Wavetable Voltage Out).....	725

Configuring the Basic Functionality (Angular Wavetable Voltage Out)

Overview

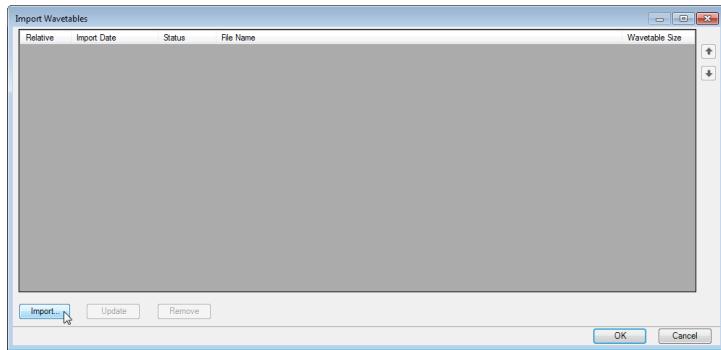
The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Managing wavetable files
- Assigning a master APU provider
- Controlling wavetable execution
- Adding noise to the output signal
- Specifying a signal offset

Managing wavetable files

Angular Wavetable Voltage Out function blocks require at least one wavetable for signal generation.

The Import Wavetables dialog lets you import, update and remove wavetable files.



You open the dialog via the Browse button of the Imported wavetable files property.

Default wavetable ConfigurationDesk adds a default wavetable to the function block only if you do not import a wavetable file. The default wavetable ensures that at least one wavetable is provided for signal generation. All values of the default wavetable are 0.0.

Maximum number of wavetables Wavetables can be referenced by Angular Wavetable and Crank/Cam function blocks. The onboard memory of an I/O board limits the total number of wavetables that can be referenced by all function blocks of both types used in the application. Refer to the following table:

Hardware	Max. Number of Wavetables per Board
DS2680 I/O Unit	24 (for all channel types)
DS2621 Signal Generation Board	12
DS6101 Multi-I/O Board	12 (for all channel types)
DS6241 D/A Board	12

A wavetable that is used n times by Angular Wavetable or Crank/Cam function blocks is stored n times in the wavetable memory. Potentially added default wavetables are also stored n times in the wavetable memory.

Assigning a master APU provider	An Angular Wavetable Voltage Out function block reads the angular value of an angle processing unit (APU) to look-up and output the values of the active wavetable. Therefore, you have to assign a master APU provider. Refer to Using Angular Processing Units (APUs) on page 126.
----------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Controlling wavetable execution	<p>Wavetable execution has the following control options:</p> <p>Initial phase shift You can apply an initial phase shift (φ_{initial}) to the output signal. The actual phase shift φ_{actual} reads as follows:</p> $\varphi_{\text{actual}} = \varphi_{\text{initial}} + \varphi_{\text{Phase offset}}$ <p>$\varphi_{\text{Phase offset}}$ is set from within the behavior model during run time.</p> <p>Switching wavetables at run-time A wavetable can be substituted by another one at run-time by changing the signal at the Active Wavetable function port. The new wavetable is executed immediately.</p>
----------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Adding noise to the output signal	<p>The Angular Wavetable Voltage Out function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.</p> <p>Limitation Noise generation is supported only for the Flexible Out 1, Analog Out 3, Analog Out 8, and Analog Out 10 channel types.</p>
------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Specifying a signal offset	<p>You can apply a voltage offset to the output signal. The offset is channel-specific, hence, all the signals that use the same I/O channel have the same signal offset. This is helpful to generate unipolar output, for example, positive voltages, even with bipolar wavetable values.</p> <p>Limitation Specifying a signal offset is not supported for the Analog Out 3 and Analog Out 8 channel types.</p>
-----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Configuring Standard Features (Angular Wavetable Voltage Out)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

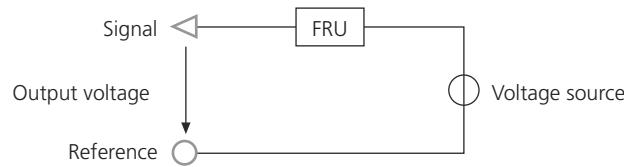
Configuration Feature	Analog Out 1	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 8	Further Information
Interface type	Differential with ground sense	Galvanically isolated	Differential with ground sense	Differential with ground sense	Galvanically isolated	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	✓	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	✓	–	–	Specifying Settings for Failure Simulation on page 123

Configuration Feature	Analog Out 9	Analog Out 10	Flexible Out 1	Further Information
Interface type	Differential with ground sense	Differential with ground sense	Galvanically isolated	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 1 on page 1553](#)
- [Analog Out 3 on page 1554](#)
- [Analog Out 4 on page 1555](#)
- [Analog Out 6 on page 1556](#)
- [Analog Out 8 on page 1557](#)
- [Analog Out 9 on page 1557](#)
- [Analog Out 10 on page 1558](#)
- [Flexible Out 1 on page 1604](#)

Connection example The following illustration shows a connection example and the failure routing unit (FRU) in the Signal branch.



Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Angular Wavetable Voltage Out)

SCALEXIO Hardware Dependencies (Angular Wavetable Voltage Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 1	Analog Out 3	Analog Out 4	Flexible Out 1
Hardware	DS2680 I/O Unit			DS2621 Signal Generation Board
Output current range	0 ... +5 mA _{RMS}			0 ... +40 mA _{RMS}
Output voltage range	0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication)
Noise generation	-	✓	-	✓
Noise amplitude range	-	<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel 	-	<ul style="list-style-type: none"> ▪ 0 ... +15 V for 3 channels (channel multiplication)

Channel Type	Analog Out 1	Analog Out 3	Analog Out 4	Flexible Out 1
		<ul style="list-style-type: none"> ▪ 0 ... +15 V for 3 channels (channel multiplication) 		
Resolution	14 bit			16 bit
Gain error	$\pm 0.1\%$ (typ.)			$\pm 0.1\%$ (typ.)
Offset error	$\pm 5\text{ mV}$ (typ.)			$\pm 2\text{ mV}$ (typ.)
Maximum number of angular-coded wavetable files per board ¹⁾	24			12
Angular processing unit	Number of slaves	6		
	Maximum speed	168,000 °/s		
Configurable fuse	–			<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Circuit diagrams	Analog Out 1 on page 1553	Analog Out 3 on page 1554	Analog Out 4 on page 1555	Flexible Out 1 on page 1604
Required channels	1	1 (additional channels are required for voltage enhancements)	1	1 (additional channels are required for voltage enhancements)

¹⁾ Total number of wavetable files that can be stored on the I/O board. For more information, refer to [Configuring the Basic Functionality \(Angular Wavetable Voltage Out\)](#) on page 724.

Channel Type	Analog Out 6	Analog Out 8	Analog Out 9	Analog Out 10
Hardware	DS6101 Multi-I/O Board			DS6241 D/A Board
Output current range	0 ... +5 mA _{RMS}			-5 mA ... +5 mA (min.)
Output voltage range	0 ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	0 ... +10 V	-10 V ... +10 V
Noise generation	–	✓	–	✓
Noise amplitude range	–	<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication) 	–	0 ... +5 V
Resolution	14 bit			16 bit
Gain error	$\pm 0.1\%$ (typ.)			$\pm 0.03\%$ (typ.)
Offset error	$\pm 5\text{ mV}$ (typ.)			$\pm 1\text{ mV}$ (typ.)
Maximum number of angular-coded wavetable files per board ¹⁾	12			12
Angular processing unit	Number of slaves	6		
	Maximum speed	168,000 °/s		
Configurable fuse	–			–
Circuit diagrams	Analog Out 6 on page 1556	Analog Out 8 on page 1557	Analog Out 9 on page 1557	Analog Out 10 on page 1558

Channel Type	Analog Out 6	Analog Out 8	Analog Out 9	Analog Out 10
Required channels	1	1 (additional channels are required for voltage enhancements)	1	1

¹⁾ Total number of wavetable files that can be stored on the I/O board. For more information, refer to [Configuring the Basic Functionality \(Angular Wavetable Voltage Out\)](#) on page 724.

Synchronous signal update on DS2680 and DS6101 If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101](#) on page 1539.

General limitation	Channel multiplication across several I/O boards is not supported. DS6101 Multi-I/O Board Channel multiplication is not supported in general. DS6241 D/A Board Channel multiplication is not supported in general.
---------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

More hardware data	DS2680 I/O Unit For more board-specific data, refer to Data Sheet of the DS2680 I/O Unit (SCALEXIO Hardware Installation and Configuration) . DS2621 Signal Generation Board For more board-specific data, refer to Data Sheet of the DS2621 Signal Generation Board (SCALEXIO Hardware Installation and Configuration) . DS6101 Multi-I/O Board For more board-specific data, refer to Data Sheet of the DS6101 Multi-I/O Board (SCALEXIO Hardware Installation and Configuration) . DS6241 D/A Board For more board-specific data, refer to Data Sheet of the DS6241 D/A Board (SCALEXIO Hardware Installation and Configuration) .
---------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Angular Wavetable Digital Out

Where to go from here

Information in this section

Introduction (Angular Wavetable Digital Out).....	730
Overviews (Angular Wavetable Digital Out).....	731
Configuring the Function Block (Angular Wavetable Digital Out).....	735
Hardware Dependencies (Angular Wavetable Digital Out).....	738

Introduction (Angular Wavetable Digital Out)

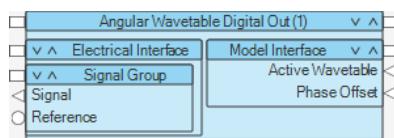
Introduction to the Function Block (Angular Wavetable Digital Out)

Function block purpose

The Angular Wavetable Digital Out function block type outputs a digital signal sequence by looking up a table with angular-coded values.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Importing up to 24 wavetable files (hardware dependent).
- Switching executed wavetables at runtime.
- Adjusting the 0° position of the wavetable.

Supported channel types

The Angular Wavetable Digital Out function block supports the following channel types:

	SCALEXIO				MicroAutoBox III
Channel type	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	–
Hardware	DS2680	DS6101	DS6202	DS2621	–

Introduction to signal generation based on wavetables

For basic information, refer to [Introduction to Signal Generation Using Angular-Coded Wavetables](#) on page 712.

Oversviews (Angular Wavetable Digital Out)

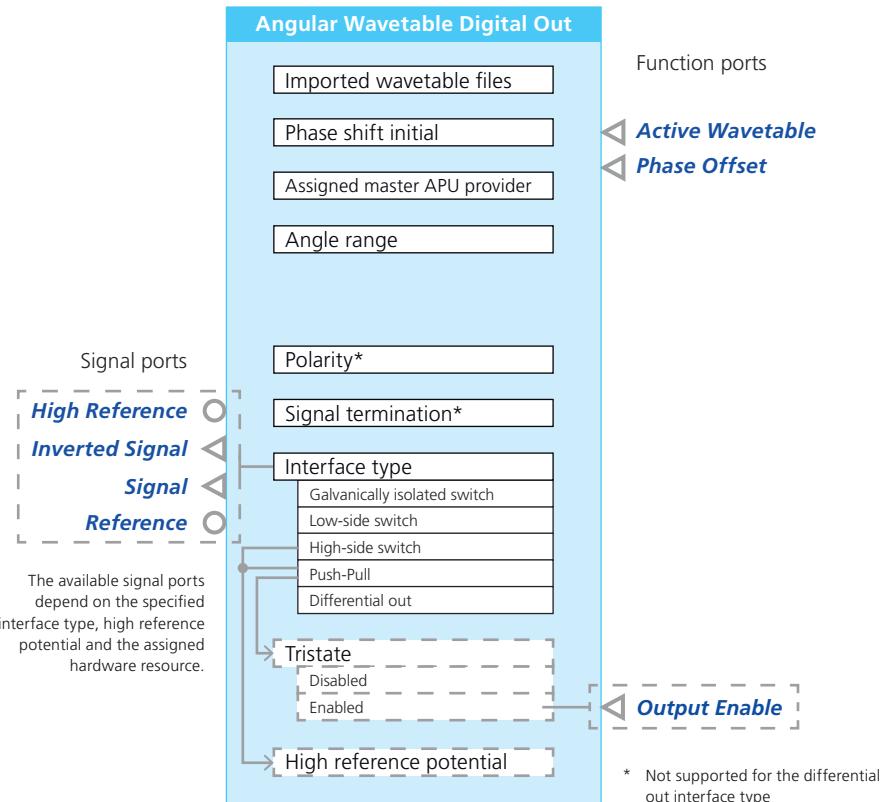
Where to go from here**Information in this section**

Overview of Ports and Basic Properties (Angular Wavetable Digital Out).....	731
Overview of Tunable Properties (Angular Wavetable Digital Out).....	734

Overview of Ports and Basic Properties (Angular Wavetable Digital Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Active Wavetable

This function import lets you select the wavetable that is currently output from within the behavior model.

Value range	<ul style="list-style-type: none"> 1 ... n n equals the number of imported wavetable files.
Dependencies	–

Phase Offset

This function import lets you defines the 0° position independently of the active wavetable from within the behavior model.

The phase offset is set relative to the setting of the Phase shift initial property.

Value range	<ul style="list-style-type: none"> -360° ... +360° If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	–

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance

state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Digital Out) on page 738 .
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Digital Out) on page 738 .
Dependencies	–

Inverted Signal

This signal port and the Signal signal port together represent the electrical connection points of a logical signal with two complementary potentials carrying the information in the voltage difference between these signals. Both signal ports (Signal and Inverted Signal) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Digital Out) on page 738 .
Dependencies	Available only if the Interface type property is set to Differential out.

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Angular Wavetable Digital Out) on page 738 .
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.
--------------	---------------------------------------------------------------------------------------------------------------------------

Overview of Tunable Properties (Angular Wavetable Digital Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Imported wavetable files	✓ ³⁾	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Phase shift initial	✓	-
Polarity	✓	-
Signal termination	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ The content of the imported wavetable files can be updated when the real-time application is stopped. Wavetable files can be replaced by the downwave assistance tool, refer to [Replacing Angular-Coded Wavetable Files at Run Time](#) on page 716.

Configuring the Function Block (Angular Wavetable Digital Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (Angular Wavetable Digital Out).....	735
Configuring Standard Features (Angular Wavetable Digital Out).....	737

Configuring the Basic Functionality (Angular Wavetable Digital Out)

Overview

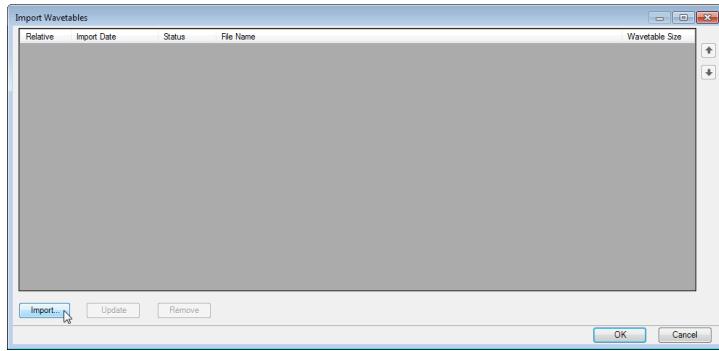
The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Managing wavetable files
- Assigning a master APU provider
- Controlling wavetable execution
- Reducing voltage level overshoots

Managing wavetable files

Angular Wavetable Digital Out function blocks require at least one wavetable for signal generation.

The Import Wavetables dialog lets you import, update and remove wavetable files.



You open the dialog via the Browse button of the Imported wavetable files property.

Default wavetable ConfigurationDesk adds a default wavetable to the function block only if you do not import a wavetable file. The default wavetable

ensures that at least one wavetable is provided for signal generation. All values of the default wavetable are 0.0.

Maximum number of wavetables Wavetables can be referenced by Angular Wavetable and Crank/Cam function blocks. The onboard memory of an I/O board limits the total number of wavetables that can be referenced by all function blocks of both types used in the application. Refer to the following table:

Hardware	Max. Number of Wavetables per Board
DS2680 I/O Unit	24
DS2621 Signal Generation Board	12
DS6101 Multi-I/O Board	12
DS6202 Digital I/O Board	52 (max. 16 function blocks with up to 4 wavetables each)

A wavetable that is used n times by Angular Wavetable or Crank/Cam function blocks is stored n times in the wavetable memory. Potentially added default wavetables are also stored n times in the wavetable memory.

Assigning a master APU provider

An Angular Wavetable Digital Out function block reads the angular value of an angle processing unit (APU) to look-up and output the values of the active wavetable. Therefore, you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Controlling wavetable execution

Wavetable execution has the following control options:

Initial phase shift You can apply an initial phase shift (φ_{initial}) to the output signal. The actual phase shift φ_{actual} reads as follows:

$$\varphi_{\text{actual}} = \varphi_{\text{initial}} + \varphi_{\text{Phase offset}}$$

$\varphi_{\text{Phase offset}}$ is set from within the behavior model during run time.

Switching wavetables at run-time One wavetable can be substituted by another one at run-time by changing the signal at the Active Wavetable function port. The new wavetable is executed immediately.

Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

Limitation A resistor connected in series to the output driver is supported only by the Digital In/Out 5 channel type. However, signal termination is not

supported if you use the Differential out interface type. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586.

Configuring Standard Features (Angular Wavetable Digital Out)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Configuration Feature	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<p>Depending on the channel type, refer to:</p> <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112 ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109 ▪ Specifying Digital Output Signals (Flexible Out 1) on page 98
Polarity of output signals	✓	✓	<ul style="list-style-type: none"> ▪ ✓ ▪ Not supported for the differential out interface type. 	✓	–
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.				–
Channel multiplication	✓	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95

Configuration Feature	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	Further Information
Signal Ports					
Trigger level of electronic fuse	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 1 on page 1575](#)
- [Digital Out 3 on page 1577](#)
- [Digital In/Out 5 on page 1586](#)
- [Flexible Out 1 on page 1604](#)

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Angular Wavetable Digital Out)

SCALEXIO Hardware Dependencies (Angular Wavetable Digital Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 3	Digital In/Out 5
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board	DS6202 Digital I/O Board
Output current range	▪ 0 ... +80 mA _{RMS} for 1 channel	▪ 0 ... +40 mA for 1 channel	▪ 0 ... +140 mA _{RMS} (each channel)	0 ... ±40 mA (each channel)

Channel Type		Digital Out 1	Flexible Out 1	Digital Out 3	Digital In/Out 5
		<ul style="list-style-type: none"> ▪ 1.792 A_{RMS} for 28 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0.32 A for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	
High side reference voltage		+5 V ... +60 V	-60 V ... +60 V	+5 V ... +60 V	<ul style="list-style-type: none"> ▪ +3.3 V and +5 V (switchable) ▪ +3.3 V (fix) for the differential out interface type
Interface type for digital outputs		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull ▪ Differential out
In push-pull configuration the outputs can be set to high impedance (tristate) at run time.					
Configurable fuse		–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–	–
Maximum number of angular-coded wavetable files ¹⁾		24	12	12	52 (max. 16 function blocks with up to 4 wavetables each)
Angular processing unit	Number of slaves	6	1	6	6
	Maximum speed	168,000 °/s	168,000 °/s	168,000 °/s	1,200,000 °/s
Signal termination		–	–	–	<ul style="list-style-type: none"> ▪ 68 Ω (serial termination) ▪ Switchable ▪ Not supported for the differential out interface type
Circuit diagram		Digital Out 1 on page 1575	Flexible Out 1 on page 1604	Digital Out 3 on page 1577	Digital In/Out 5 on page 1586
Required channels		1 (additional channels are required for current enhancement and operation in push/pull mode)		1	1 (2 channels are required to operate the outputs using the differential out interface type.)

¹⁾ Total number of wavetable files that can be stored on the I/O board. For more information, refer to [Configuring the Basic Functionality \(Angular Wavetable Digital Out\)](#) on page 735.

General limitations

Channel multiplication across several I/O boards is not supported.

DS2621 Signal Generation Board The following restrictions apply to the DS2621 Signal Generation Board when the push/pull mode is used:

- Every logical signal requests two channels (primary and auxiliary) on the real-time hardware to be assigned to the function block. These channels must be adjacent to one another on an I/O board. Example: primary channel = n, auxiliary channel = n + 1.
- Channel 10 cannot be used as the primary channel because using channel 1 as the next adjacent channel is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6202 Digital I/O Board The following limitations apply to the DS6202 Digital I/O Board:

- Channel multiplication is not supported in general.
- Only 16 Angular Wavetable Digital Out function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Waveform (Voltage Out, Current Sink, Digital Out)

Objective

To generate periodic signals that commonly depend on two or more independent properties, e.g., speed and time, and that have a complex signal pattern.

Where to go from here

Information in this section

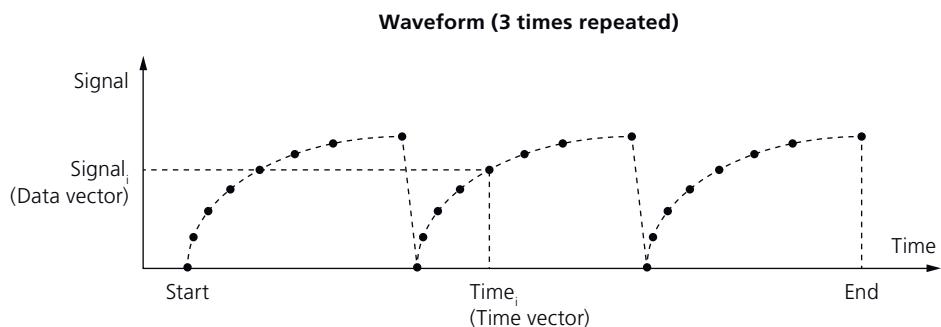
Introduction to Signal Generation Using Waveforms	742
Waveform Voltage Out.....	745
Waveform Current Sink.....	759
Waveform Digital Out.....	772

Introduction to Signal Generation Using Waveforms

Basics on Signal Generation Using Waveforms

The waveform approach

A waveform consists of a distinct number of data points and is repeatedly executed during application execution. The data points vector is computed in the behavior model, and the vector size is determined by the Maximum Vector Size function property. For example, Maximum Vector Size equals 8 in the following illustration, as the waveform consists of 8 data points.



Note

The waveform approach lets you generate highly dynamic periodic signals, as the data points are computed at run time. The data points do not need to be equidistant in time, so only a few are required for curve sections that hardly change with time.

Field of application

Use waveforms if you want to simulate periodic signals that are dynamic, in contrast to static wavetables.

Analog waveforms Use analog waveforms (Waveform Voltage Out, Waveform Current Sink) if you want to simulate complex high-speed signals, for example, the signals of wheel speed sensors and cylinder pressure sensors. You can generate angle-based signals.

Digital waveforms Use digital waveforms (Waveform Digital Out) if you want to simulate simple low-speed signals, for example, plain protocols. You can generate angle-based signals.

Waveform specification

Basically, a waveform is specified by two vectors, a data (= signal) vector and a time vector. You can specify either all the single data and time vector entries, or a

few values that are expanded to the time and data vectors. This depends on the setting of the **Vector Size Mode** property.

The following three **Vector Size Mode** property values are available:

- **Variable vector size:**

You must specify all the single time and data vector entries. This setting allows an arbitrary time pattern. Data vector and time vector contain the same number of entries. Each data vector entry (**Signal_i**) corresponds to a time vector entry (**Time_i**). The vector size is determined by the **Maximum Vector Size** property.

- **Fixed time vector size:**

You must specify one time value, which is expanded to the time vector. This setting leads to an equidistant time pattern. The time vector contains just one entry (**Time**). The size of the data vector is determined by the **Maximum Vector Size** property. The one time vector entry is expanded to a vector that contains the same number of entries as the data vector.

- **Fixed data vector size:**

You must specify two signal values, which are expanded to the data vector. This setting is advantageous for digital signals that switch between two values. The data vector contains just two entries, **Signal₁** and **Signal₂**. The size of the time vector is determined by the **Maximum Vector Size** property. The data vector "pretends" to have the same number of entries. The two data vector entries are output alternately.

Vector Size Mode	Time Vector Specification	Data Vector Specification
Variable vector size	[Time ₁ , ..., Time _n]	[Signal ₁ , ..., Signal _n]
Fixed time vector size	1 entry (Time) is expanded to: [Time, Time, ..., Time]; n values	[Signal ₁ , ..., Signal _n]
Fixed data vector size	[Time ₁ , ..., Time _n]	2 entries (Signal ₁ , Signal ₂) are expanded to: [Signal ₁ , Signal ₂ , Signal ₁ , ...]; n values

Interpretation of the time vector entries The time values (**Time_i**) of the time vector relate either to the first data point of the waveform or to the preceding data point, depending on the setting of the **Timer mode** property:

- **Timer mode = Relative to preceding entry:**

The time values accumulate to the waveform period:

$$\text{Waveform period } T = \sum \text{Time}_i; i = 1 \dots n$$

The single time deviations Δt accumulate to the overall time deviation:

$$\Delta T = n \cdot \Delta t (\Delta t = \pm 34 \text{ ns})$$

- **Timer mode = Relative to waveform start:**

The last time value determines the waveform period:

$$\text{Waveform period } T = \text{Time}_n$$

The overall time deviation reads:

$$\Delta T = \Delta t (\Delta t = \pm 34 \text{ ns}).$$

Note

Relative to waveform start is more precise than Relative to preceding entry.

Waveform signal generation

The time vector and data vector are both calculated in the behavior model, and they are eventually expanded in ConfigurationDesk. The signal values are output at the times specified by the time values.

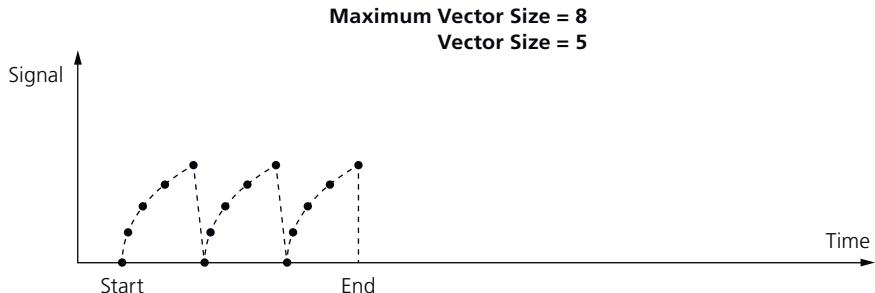
At run time, you can specify via the Vector Size function port whether all the waveform data points or only a subset of waveform data points (Vector Size < Maximum Vector Size) is output:

[Signal₁ ... Signal_{Vector Size}].

Note

The subset always begins with the first vector element, i.e., Signal₁.

If Vector Size was set to 5 during application execution, for example (refer to the previous illustration), only the five leading data points are output:



Implementation details

The maximum number of entries which the time vector and data vector can comprise depends on the used hardware and the settings of Vector Size Mode and Timer Mode. The highest possible number is 169 in all cases.

Waveform Voltage Out

Where to go from here

Information in this section

Introduction (Waveform Voltage Out).....	745
Overviews (Waveform Voltage Out).....	746
Configuring the Function Block (Waveform Voltage Out).....	751
Hardware Dependencies (Waveform Voltage Out).....	755

Introduction (Waveform Voltage Out)

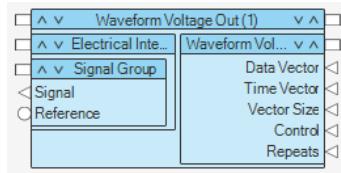
Introduction to the Function Block (Waveform Voltage Out)

Function block purpose

The Waveform Voltage Out function block type can be used to generate periodic voltage output signals, for example, to simulate signals generated by combined oil temperature and pressure sensors in combustion engines.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Selecting different vector modes for waveform specification (variable, fixed time, fixed data).
- Selecting different trigger sources for waveform generation (model-based, angle-based, I/O function).
- Updating the waveform at runtime.
- Controlling the number of waveform executions (repetitions).

- Adding noise to the output signal.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Waveform Voltage Out function block supports the following channel types:

	SCALEXIO										MicroAutoBox III
Channel type	Analog Out 1	Analog Out 2	Analog Out 3	Analog Out 4	Analog Out 6	Analog Out 7	Analog Out 8	Analog Out 9	Analog Out 10	Flexible Out 1	–
Hardware	DS2680				DS6101				DS6241	DS2621	–

Introduction to signal generation based on waveforms

For basic information, refer to [Introduction to Signal Generation Using Waveforms](#) on page 742.

Oversviews (Waveform Voltage Out)

Where to go from here

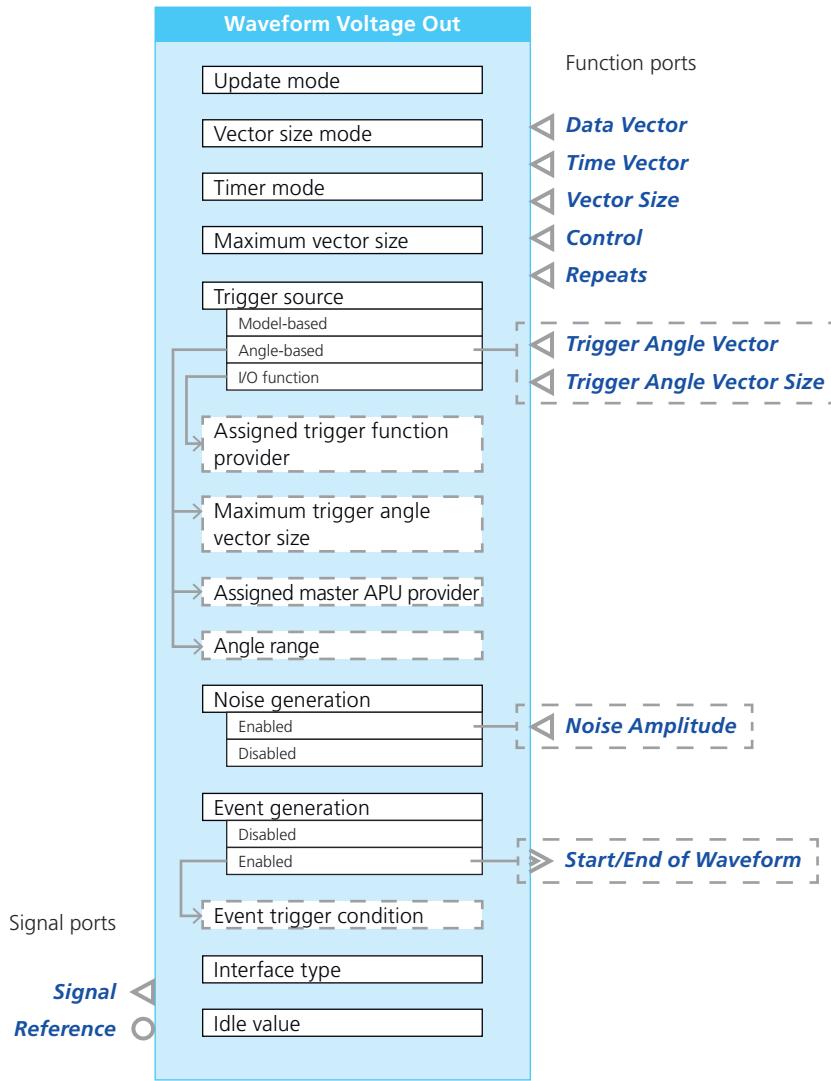
Information in this section

- Overview of Ports and Basic Properties (Waveform Voltage Out)..... 746
- Overview of Tunable Properties (Waveform Voltage Out)..... 751

Overview of Ports and Basic Properties (Waveform Voltage Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Data Vector

This function import lets you define the data values (y-axis) for the waveform output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt) for each entry in the vector. The range depends on the following: <ul style="list-style-type: none"> On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Voltage Out) on page 755. If you use user saturation min/max values for saturation, the specified values can reduce the value range. The vector size depends on the value specified at the Maximum vector size property.
Dependencies	-

Time Vector

This function import lets you define the time values (x-axis) for the waveform output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $T_{\min} \dots T_{\max}$ (in seconds) for each entry in the vector. ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Voltage Out) on page 755. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ The vector size depends on the value specified at the Maximum vector size property.
Dependencies	–

Vector Size

This function import lets you determine from within the behavior model, whether all the waveform data points or only a subset of waveform data points (Vector Size < Maximum vector size) is output.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum Vector Size property.
Dependencies	–

Control

This function import lets you start or stop the waveform execution from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: Stop Stops waveform generation. ▪ 1: Start Starts waveform generation. Changes of the Data Vector, Time Vector, and Vector Size function ports change the waveform. ▪ 2: No update Suppresses waveform updates at run time. The last value of the Control function port is kept. Changes of the Data Vector, Time Vector, and Vector Size function ports are ignored.
Dependencies	–

Repeats

This function import lets you define from within the behavior model, how often a wavetable is output.

Value range	<ul style="list-style-type: none"> ▪ -1: Infinite repetitions
-------------	------------------------------------------------------------------------------

	<p>The waveform is executed repeatedly. This means an output signal is generated periodically, for example, a wheel speed sensor signal.</p> <ul style="list-style-type: none"> ▪ 0: Zero repetitions No waveform is generated. A running waveform execution stops synchronously, regardless of the specified update mode (synchronous or asynchronous). ▪ 1: Single shot The waveform is executed once, i.e., either at each model execution step or when a specific angle occurs (specified via Trigger source property).
Dependencies	–

Note

The Repeats function port ignores new values received from the behavior model if a waveform is already being executed. This behavior might cause problems, if the Repeats function port is set to 1 (= single shot) and a new waveform generation is triggered while the previous waveform generation is still in progress. In this case, the new trigger is ignored and waveform generation will stop after the previous waveform generation is completed. To avoid this effect, you should set the Repeats function port to -1 (= infinite repetitions). In this case, always the last values (received at the function ports) are used immediately for waveform execution.

Trigger Angle Vector

This function import lets you trigger the reading of signals (data/time vectors) from within the behavior model. Signals are read whenever an angle value provided at this function port matches an angle value returned by the master APU provider that is assigned to the function block.

To compare the trigger angles that are provided by the behavior model with the angle values of the master APU provider, the function block converts the trigger angles to the angle range that is specified via the Angle range property. For example: The function block converts -10° to 350° if the Angle range property is set to 360° .

Value range	<ul style="list-style-type: none"> ▪ $-3600^\circ \dots +3600^\circ$ for each entry in the vector. ▪ The vector size depends on the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

Trigger Angle Vector Size

This function import lets you determine whether all vector entries are considered as trigger angles or only a subset of leading entries from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... $U_{\max \text{ Noise}}$ (in Volt) ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Voltage Out) on page 755. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range. ▪ 0 V: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Noise generation property is set to Enabled. ▪ Supported for Flexible Out 1, Analog Out 3, Analog Out 8, and Analog Out 10 channel types.

Start/End of Waveform

This event port provides an I/O event each time a waveform execution starts or has been finished, according to the setting of the Event trigger condition property.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Note

The hardware is passive and adjusts its internal resistance so that the defined current can flow.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Voltage Out) on page 755.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Voltage Out) on page 755.
Dependencies	–

Overview of Tunable Properties (Waveform Voltage Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
Event trigger condition	✓	–	–
Update mode	✓	–	–
Function Ports			
Initial switch setting (Test Automation)	–	✓	–
Initial substitute value (Test Automation)	–	✓	–
Electrical Interface			
Idle Value	✓	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Waveform Voltage Out)

Where to go from here**Information in this section**

- Configuring the Basic Functionality (Waveform Voltage Out)..... 752
- Configuring Standard Features (Waveform Voltage Out)..... 753

Configuring the Basic Functionality (Waveform Voltage Out)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Controlling waveform execution
- Providing I/O events
- Adding noise to the output signal

Controlling waveform execution

Waveform execution has the following control options:

Starting waveform execution Waveform execution can be started by setting the Control function port to Start or No update. No update suppresses any waveform updates at run time.

Specifying waveform repetitions It is possible to execute a waveform once with each trigger event that is provided by the specified trigger source (refer to the Trigger source property) or infinitely. Zero repetitions means that no waveform is generated. Executing a waveform repeatedly means generating a periodic output signal, for example, a wheel speed sensor signal.

Stopping waveform execution Waveform execution can be stopped by setting the Control function port to Stop. Waveform execution stops either immediately, that is, after the current waveform entry (Asynchronous), or after execution of the current waveform finishes (Synchronous), depending on the Update mode property. Execution can also be stopped via the Repeats function port. Zero repetitions (0) always stops a running waveform execution synchronously, regardless of the Update mode property. When stopped, the Idle value is output.

Update Mode	Stop Request Via Leads To
Synchronous	Control = Stop	Synchronous stop
	Repeats = Zero repetitions	Synchronous stop
Asynchronous	Control = Stop	Asynchronous stop
	Repeats = Zero repetitions	Synchronous stop

Updating the waveform at run time The data vector and the time vector can be updated at run time by changing the Data Vector and/or Time Vector function ports. The update takes effect either immediately, that is, after the current waveform entry (Asynchronous) or after the execution of the current waveform finishes (Synchronous), refer to the Update mode property.

Triggering waveform execution The trigger source defines when the function imports read the signals (data/time vectors) computed in the behavior model.

- Model-based: Signals are read with respect to the behavior model's sample rate.

- Angle-based: Signals are read whenever the angle unit (APU) returns an angle value that matches an angle value specified in the Trigger Angle Vector function port.

As a precondition for angle-based triggering you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

- I/O function: Signals are read with each trigger event that is provided by the assigned trigger function provider, for example, the Trigger In function block (via the Assigned trigger function provider property). Trigger function providers have their own defined trigger conditions.

Configuring the waveform specification You can specify either all the single time and data vector entries, or a few values that are expanded to the time and data vectors. This depends on the setting of the Vector Size Mode property.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Waveform Voltage Out function block can generate an I/O event each time a waveform execution starts or has been finished (configurable via the Event trigger condition property).

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Waveform property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Adding noise to the output signal

The Waveform Voltage Out function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Limitation Noise generation is supported only for the Flexible Out 1, Analog Out 3, Analog Out 8, and Analog Out 10 channel types.

Configuring Standard Features (Waveform Voltage Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows

only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

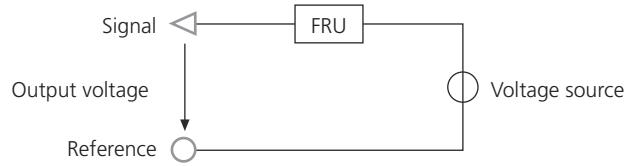
Configuration Feature	Analog Out 1	Analog Out 2	Analog Out 3	Analog Out 4	Analog Out 6	Further Information
Interface type	Differential with ground sense	Galvanically isolated	Galvanically isolated	Differential with ground sense	Differential with ground sense	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	–	✓	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	✓	✓	–	Specifying Settings for Failure Simulation on page 123

Configuration Feature	Analog Out 7	Analog Out 8	Analog Out 9	Analog Out 10	Flexible Out 1	Further Information
Interface type	Galvanically isolated	Galvanically isolated	Differential with ground sense	Differential with ground sense	Galvanically isolated	Specifying the Circuit Type for Analog Output Signals on page 118
Channel multiplication	–	✓	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports						
Trigger level of electronic fuse	–	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 1 on page 1553](#)
- [Analog Out 2 on page 1554](#)
- [Analog Out 3 on page 1554](#)
- [Analog Out 4 on page 1555](#)
- [Analog Out 6 on page 1556](#)
- [Analog Out 7 on page 1556](#)
- [Analog Out 8 on page 1557](#)
- [Analog Out 9 on page 1557](#)
- [Analog Out 10 on page 1558](#)
- [Flexible Out 1 on page 1604](#)

Connection example The following illustration shows a connection example and the failure routing unit (FRU) in the Signal branch.



Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Waveform Voltage Out)

SCALEXIO Hardware Dependencies (Waveform Voltage Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 1	Analog Out 2	Analog Out 3	Analog Out 4	Flexible Out 1
Hardware	DS2680 I/O Unit				DS2621 Signal Generation Board
Event generation	✓				✓
Noise generation	–	✓	–	✓	
Noise amplitude range	–	▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication)	–	▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication)	
Output current range	0 ... +5 mA _{RMS}				0 ... +40 mA _{RMS}

Channel Type		Analog Out 1	Analog Out 2	Analog Out 3	Analog Out 4	Flexible Out 1
Output voltage range		0 ... +10 V for 1 channel	-10 V ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	0 ... +10 V for 1 channel	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication)
Resolution		14 bit				
Gain error		±0.1% (typ.)	±0.5% (typ.)	–	±0.1% (typ.)	±0.1% (typ.)
Offset error		±5 mV (typ.)	±3.5 mV (typ.)	–	±5 mV (typ.)	±2 mV (typ.)
Value range of time vector entries		2.616 µs ... 200.0 s (Resolution: 68 ns)				
Frequency range with maximum output amplitude		–		500 Hz ... 20 kHz	–	–
Angular processing unit	Number of slaves	6				
	Maximum speed	168,000 °/s				
Maximum trigger angle vector size		<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 				
Configurable fuse		–				
Circuit diagrams		Analog Out 1 on page 1553	Analog Out 2 on page 1554	Analog Out 3 on page 1554	Analog Out 4 on page 1555	Flexible Out 1 on page 1604
Required channels		1	1 (additional channels are required for voltage enhancements.)		1	1 (additional channels are required for voltage enhancements.)

Channel Type		Analog Out 6	Analog Out 7	Analog Out 8	Analog Out 9	Analog Out 10
Hardware		DS6101 Multi-I/O Board				
Event generation		✓				
Noise generation		–		✓	–	✓
Noise amplitude range		–		<ul style="list-style-type: none"> ▪ 0 ... +5 V for 1 channel ▪ 0 ... +15 V for 3 channels (channel multiplication) 	–	0 ... +5 V
Output current range		0 ... +5 mA _{RMS}	-5 mA ... +5 mA	0 ... +5 mA _{RMS}		-5 mA ... +5 mA (min.)

Channel Type		Analog Out 6	Analog Out 7	Analog Out 8	Analog Out 9	Analog Out 10
Output voltage range		0 ... +10 V	-10 V ... +10 V	<ul style="list-style-type: none"> ▪ -20 V ... +20 V for 1 channel ▪ -60 V ... +60 V for 3 channels (channel multiplication) 	0 ... +10 V	-10 V ... +10 V
Resolution		14 bit				16 bit
Gain error		±0.1% (typ.)	±0.5% (typ.)	–	±0.1% (typ.)	±0.03% (typ.)
Offset error		±5 mV (typ.)	±3.5 mV (typ.)	–	±5 mV (typ.)	±1 mV (typ.)
Value range of time vector entries		2.616 µs ... 200.0 s (Resolution: 68 ns)	1.020 µs ... 200.0 s (Resolution 68 ns)	2.616 µs ... 200.0 s (Resolution: 68 ns)		2.040 µs ... 200.0 s (Resolution: 68 ns)
Frequency range with maximum output amplitude		–		500 Hz ... 20 kHz	–	–
Angular processing unit	Number of slaves	6				6
	Maximum speed	168,000 °/s				1,200,000 °/s
Maximum trigger angle vector size		<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 				16
Configurable fuse		–				–
Circuit diagrams		Analog Out 6 on page 1556	Analog Out 7 on page 1556	Analog Out 8 on page 1557	Analog Out 9 on page 1557	Analog Out 10 on page 1558
Required channels		1		1 (additional channels are required for voltage enhancements.)	1	1

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101](#) on page 1539.

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6241 D/A Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration](#) (PDF).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration](#) (PDF).

DS6241 D/A Board For more board-specific data, refer to [Data Sheet of the DS6241 D/A Board \(SCALEXIO Hardware Installation and Configuration](#) (PDF).

Waveform Current Sink

Where to go from here

Information in this section

Introduction (Waveform Current Sink).....	759
Overviews (Waveform Current Sink).....	760
Configuring the Function Block (Waveform Current Sink).....	765
Hardware Dependencies (Waveform Current Sink).....	769

Introduction (Waveform Current Sink)

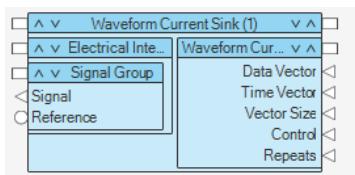
Introduction to the Function Block (Waveform Current Sink)

Function block purpose

The Waveform Current Sink function block type lets you simulate sensors with a current interface, like Hall sensors. The load can be changed at run time without the concurrence of the behavior model, for example, to produce cyclic current profiles.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Selecting different vector modes for waveform specification (variable, fixed time, fixed data).
- Selecting different trigger sources for waveform generation (model-based, angle-based, I/O function).
- Updating the waveform at runtime.
- Controlling the number of waveform executions (repetitions).

- Adding noise to the output signal.
- Generating I/O events and providing them to the behavior model.

Supported channel types	The Waveform Current Sink function block supports the following channel types:		
Channel type	Analog Out 4	Analog Out 9	Flexible Out 1
Hardware	DS2680	DS6101	DS2621

Introduction to signal generation based on waveforms	For basic information, refer to Introduction to Signal Generation Using Waveforms on page 742.
-------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------

Oversviews (Waveform Current Sink)

Where to go from here

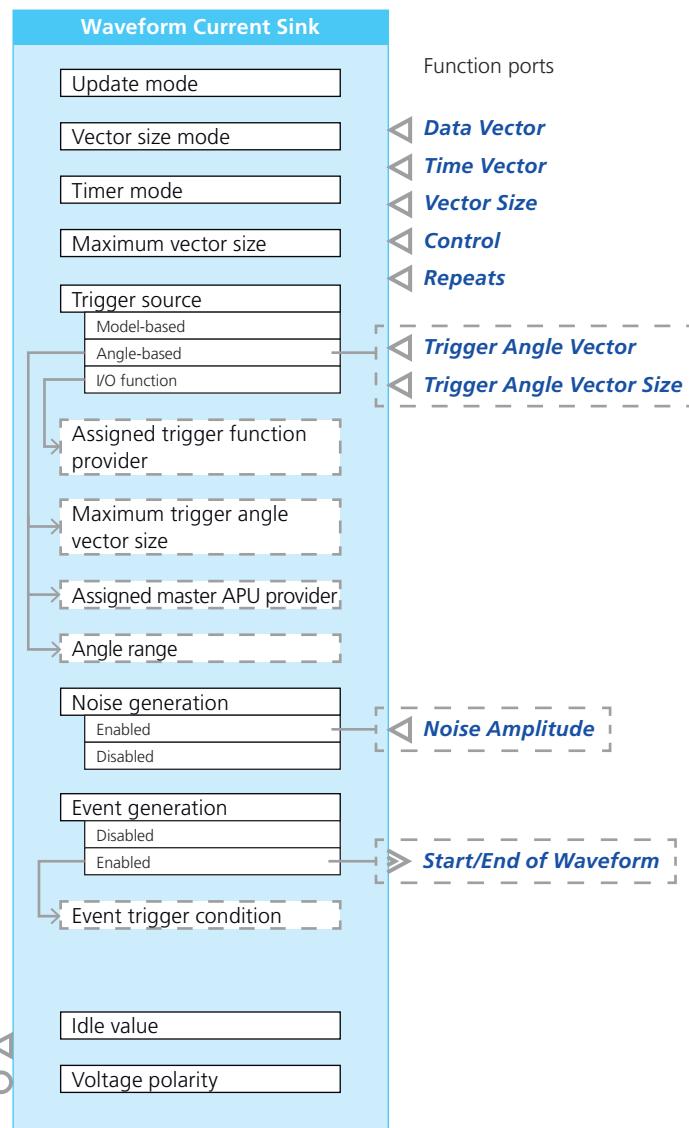
Information in this section

Overview of Ports and Basic Properties (Waveform Current Sink).....	760
Overview of Tunable Properties (Waveform Current Sink).....	765

Overview of Ports and Basic Properties (Waveform Current Sink)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Data Vector

This function import lets you define the data values (y-axis) for the waveform output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... I_{max} (in Ampere) for each entry in the vector. ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Current Sink) on page 769. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ The vector size depends on the value specified at the Maximum vector size property.
Dependencies	-

Time Vector

This function import lets you define the time values (x-axis) for the waveform output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $T_{\min} \dots T_{\max}$ (in seconds) for each entry in the vector. ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Current Sink) on page 769. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ The vector size depends on the value specified at the Maximum vector size property.
Dependencies	–

Vector Size

This function import lets you determine from within the behavior model, whether all the waveform data points or only a subset of waveform data points (Vector Size < Maximum vector size) is output.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum Vector Size property.
Dependencies	–

Control

This function import lets you start or stop the waveform execution from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: Stop Stops waveform generation. ▪ 1: Start Starts waveform generation. Changes of the Data Vector, Time Vector, and Vector Size function ports change the waveform. ▪ 2: No update Suppresses waveform updates at run time. The last value of the Control function port is kept. Changes of the Data Vector, Time Vector, and Vector Size function ports are ignored.
Dependencies	–

Repeats

This function import lets you define from within the behavior model, how often a wavetable is output.

Value range	<ul style="list-style-type: none"> ▪ -1: Infinite repetitions
-------------	------------------------------------------------------------------------------

	<p>The waveform is executed repeatedly. This means an output signal is generated periodically, for example, a wheel speed sensor signal.</p> <ul style="list-style-type: none"> ▪ 0: Zero repetitions No waveform is generated. A running waveform execution stops synchronously, regardless of the specified update mode (synchronous or asynchronous). ▪ 1: Single shot The waveform is executed once, i.e., either at each model execution step or when a specific angle occurs (specified via Trigger source property).
Dependencies	–

Note

The Repeats function port ignores new values received from the behavior model if a waveform is already being executed. This behavior might cause problems, if the Repeats function port is set to 1 (= single shot) and a new waveform generation is triggered while the previous waveform generation is still in progress. In this case, the new trigger is ignored and waveform generation will stop after the previous waveform generation is completed. To avoid this effect, you should set the Repeats function port to -1 (= infinite repetitions). In this case, always the last values (received at the function ports) are used immediately for waveform execution.

Trigger Angle Vector

This function import lets you trigger the reading of signals (data/time vectors) from within the behavior model. Signals are read whenever an angle value provided at this function port matches an angle value returned by the master APU provider that is assigned to the function block.

To compare the trigger angles that are provided by the behavior model with the angle values of the master APU provider, the function block converts the trigger angles to the angle range that is specified via the Angle range property. For example: The function block converts -10° to 350° if the Angle range property is set to 360° .

Value range	<ul style="list-style-type: none"> ▪ $-3600^\circ \dots +3600^\circ$ for each entry in the vector. ▪ The vector size depends on the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

Trigger Angle Vector Size

This function import lets you determine whether all vector entries are considered as trigger angles or only a subset of leading entries from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based.

Noise Amplitude

This function import lets you define the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... $I_{max\ Noise}$ (in Ampere) ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Current Sink) on page 769. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range. ▪ 0 A: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Noise generation property is set to Enabled. ▪ Supported only for the Flexible Out 1 channel type.

Start/End of Waveform

This event port provides an I/O event each time a waveform execution starts or has been finished, according to the setting of the Event trigger condition property.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Signal

This signal port sinks current. It represents the electrical connection point of the logical signal of the function block.

Note

The hardware is passive and adjusts its internal resistance so that the defined current can flow at the signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Current Sink) on page 769.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Current Sink) on page 769.
Dependencies	–

Overview of Tunable Properties (Waveform Current Sink)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
Event trigger condition	✓	–	–
Update mode	✓	–	–
Function Ports			
Initial switch setting (Test Automation)	–	✓	–
Initial substitute value (Test Automation)	–	✓	–
Electrical Interface			
Voltage polarity	✓	–	–
Idle Value	✓	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Waveform Current Sink)

Where to go from here**Information in this section**

- [Configuring the Basic Functionality \(Waveform Current Sink\).....](#) 766
- [Configuring Standard Features \(Waveform Current Sink\).....](#) 768

Configuring the Basic Functionality (Waveform Current Sink)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Controlling waveform execution
- Providing I/O events
- Adding noise to the output signal
- Specifying the signal polarity

Controlling waveform execution

Waveform execution has the following control options:

Starting waveform execution Waveform execution can be started by setting the Control function port to Start or No update. No update suppresses any waveform updates at run time.

Specifying waveform repetitions It is possible to execute a waveform once with each trigger event that is provided by the specified trigger source (refer to the Trigger source property) or infinitely. Zero repetitions means that no waveform is generated. Executing a waveform repeatedly means generating a periodic output signal, for example, a wheel speed sensor signal.

Stopping waveform execution Waveform execution can be stopped by setting the Control function port to Stop. Waveform execution stops either immediately, that is, after the current waveform entry (Asynchronous), or after execution of the current waveform finishes (Synchronous), depending on the Update mode property. Execution can also be stopped via the Repeats function port. Zero repetitions (0) always stops a running waveform execution synchronously, regardless of the Update mode property. When stopped, the Idle value is output.

Update Mode	Stop Request Via Leads To
Synchronous	Control = Stop	Synchronous stop
	Repeats = Zero repetitions	Synchronous stop
Asynchronous	Control = Stop	Asynchronous stop
	Repeats = Zero repetitions	Synchronous stop

Updating the waveform at run time The data vector and the time vector can be updated at run time by changing the Data Vector and/or Time Vector function ports. The update takes effect either immediately, that is, after the current waveform entry (Asynchronous) or after the execution of the current waveform finishes (Synchronous), refer to the Update mode property.

Triggering waveform execution The trigger source defines when the function imports read the signals (data/time vectors) computed in the behavior model.

- Model-based: Signals are read with respect to the behavior model's sample rate.

- Angle-based: Signals are read whenever the angle unit (APU) returns an angle value that matches an angle value specified in the Trigger Angle Vector function port.

As a precondition for angle-based triggering you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

- I/O function: Signals are read with each trigger event that is provided by the assigned trigger function provider, for example, the Trigger In function block (via the Assigned trigger function provider property). Trigger function providers have their own defined trigger conditions.

Configuring the waveform specification You can specify either all the single time and data vector entries, or a few values that are expanded to the time and data vectors. This depends on the setting of the Vector Size Mode property.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Waveform Current Sink function block can generate an I/O event each time a waveform execution starts or has been finished (configurable via the Event trigger condition property).

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Waveform property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Adding noise to the output signal

The Waveform Current Sink function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

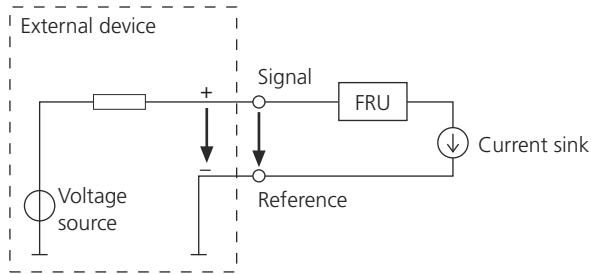
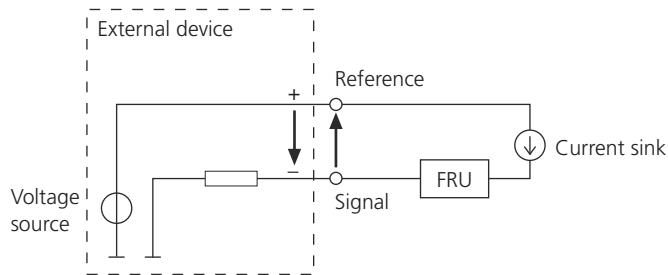
Limitation Noise generation is supported only for the Flexible Out 1 channel type.

Specifying the signal polarity

The Waveform Current Sink function block is a current sink and must be fed by an external voltage source. Depending on the selected channel type, the voltage polarity affects the functionality of the function block.

Flexible Out 1 channel type You must specify the voltage polarity of the signal ports with the Voltage polarity property to ensure the correct behavior of the function block.

Note that changing the Voltage polarity property also changes the signal path that the failure routing unit (FRU) is used on. This means that you can use the FRU on either the one or the other signal path by switching the polarity of the voltage. The following illustration shows a connection example with an FRU on the signal path.

Positive polarity**Negative polarity****Note**

The Waveform Current Sink function block adjusts only the current. The feed voltage must come from an external source as shown in the illustrations above.

The Voltage polarity property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

For details on the FRU, refer to [Electrical Error Simulation Concept \(SCALEXIO Hardware Installation and Configuration\)](#).

Analog Out 4, Analog Out 9 channel types The voltage polarity does not affect the functionality of the function block. The behavior is correct if the voltage polarity of the signal is negative as well as positive. Therefore the Voltage polarity property is not configurable for these channel types.

Configuring Standard Features (Waveform Current Sink)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Analog Out 4	Analog Out 9	Flexible Out 1	Further Information
Channel multiplication	✓	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 4](#) on page 1555
- [Analog Out 9](#) on page 1557
- [Flexible Out 1](#) on page 1604

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Waveform Current Sink)

SCALEXIO Hardware Dependencies (Waveform Current Sink)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 4	Flexible Out 1	Analog Out 9
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board
Event generation	✓	✓	✓
Noise generation	–	✓	–
Noise amplitude range	–	<ul style="list-style-type: none"> ▪ 0 ... 10 mA for 1 channel ▪ 0 ... 100 mA for 10 channels (channel multiplication) 	–
Output current range	<ul style="list-style-type: none"> ▪ +0.1 mA ... +30 mA for 1 channel ▪ 0.24 A for 8 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ +0.2 µA... +40 mA for 1 channel ▪ 0.4 A for 10 channels (channel multiplication) 	+0.1 mA ... +30 mA
Output voltage range	The following conditions must be met: <ul style="list-style-type: none"> ▪ Range for the lower potential of signal and reference: -2 V ... +18 V ▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V 	-60 V ... +60 V	The following conditions must be met: <ul style="list-style-type: none"> ▪ Range for the lower potential of signal and reference: -2 V ... +18 V ▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V
Resolution	14 bit	15 bit	14 bit
Gain error	±0.2% (typ.)	±0.1 % (typ.)	±0.2% (typ.)
Offset error	±20 µA (typ.)	±10 µA (typ.)	±30 µA (typ.)
Value range of time vector entries	2.616 µs ... 200.0 s (Resolution: 68 ns)	1.020 µs ... 200.0 s (Resolution: 68 ns)	2.616 µs ... 200.0 s (Resolution: 68 ns)
Angular processing unit	Number of slaves	6	1
	Maximum speed	168,000 °/s	168,000 °/s
Maximum trigger angle vector size	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range
Configurable fuse	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–
Circuit diagrams	Analog Out 4 on page 1555	Flexible Out 1 on page 1604	Analog Out 9 on page 1557
Required channels	1 (additional channels are required for current enhancements.)		

Synchronous signal update on DS2680 and DS6101

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to [Optimal Selection of Analog Output Channels on DS2680 and DS6101 on page 1539](#).

General limitations

Channel multiplication across several I/O boards is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Waveform Digital Out

Where to go from here

Information in this section

Introduction (Waveform Digital Out).....	772
Overviews (Waveform Digital Out).....	773
Configuring the Function Block (Waveform Digital Out).....	779
Hardware Dependencies (Waveform Digital Out).....	783

Introduction (Waveform Digital Out)

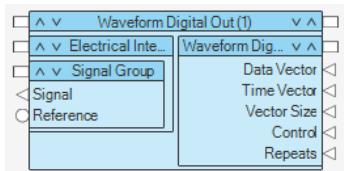
Introduction to the Function Block (Waveform Digital Out)

Function block purpose

The Waveform Digital Out function block type can be used to generate periodic pulses, for example, to simulate low-speed data packages (protocols).

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Selecting different vector modes for waveform specification (variable, fixed time, fixed data).
- Selecting different trigger sources for waveform generation (model-based, angle-based, I/O function).
- Updating the waveform at runtime.
- Controlling the number of waveform executions (repetitions).
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Wavetable Digital Out function block supports the following channel types:

	SCALEXIO				MicroAutoBox III
Channel type	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	–
Hardware	DS2680	DS6101	DS6202	DS2621	–

Introduction to signal generation based on waveforms

For basic information, refer to [Introduction to Signal Generation Using Waveforms](#) on page 742.

Oversviews (Waveform Digital Out)

Where to go from here**Information in this section**

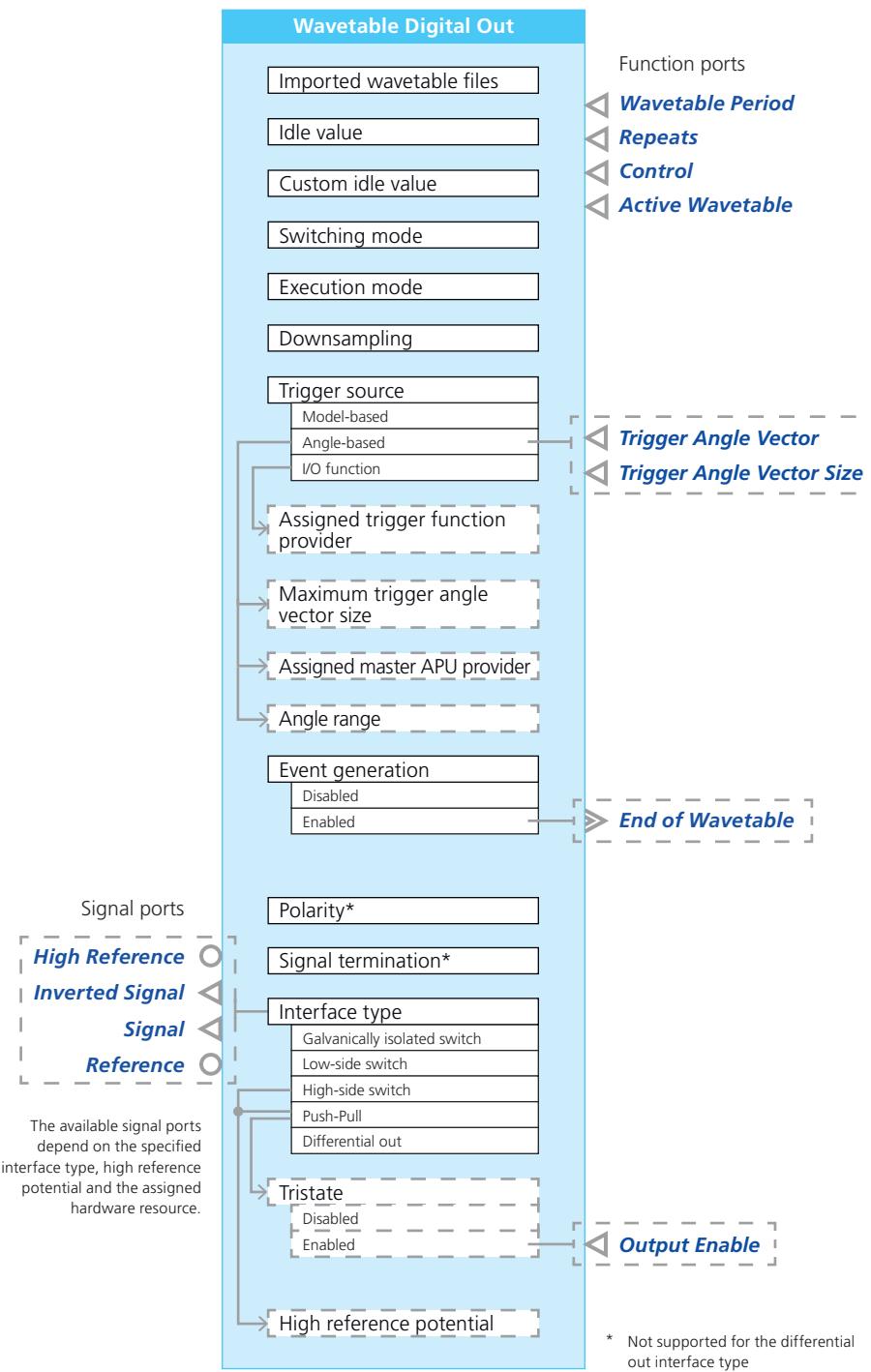
[Overview of Ports and Basic Properties \(Waveform Digital Out\)](#)..... 773

[Overview of Tunable Properties \(Waveform Digital Out\)](#)..... 778

Overview of Ports and Basic Properties (Waveform Digital Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Data Vector**

This function import lets you define the data values (y-axis) for the waveform output signal from within the behavior model.

Value range

▪ 0 or 1 for each entry in the vector.

	<ul style="list-style-type: none"> The vector size depends on the value specified at the Maximum vector size property.
Dependencies	–

Time Vector

This function import lets you define the time values (x-axis) for the waveform output signal from within the behavior model.

Value range	<ul style="list-style-type: none"> $T_{\min} \dots T_{\max}$ (in seconds) for each entry in the vector. The range depends on the following: <ul style="list-style-type: none"> On the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Digital Out) on page 783. If you use user saturation min/max values for saturation, the specified values can reduce the value range. The vector size depends on the value specified at the Maximum vector size property.
Dependencies	–

Vector Size

This function import lets you determine from within the behavior model, whether all the waveform data points or only a subset of waveform data points (Vector Size < Maximum vector size) is output.

Value range	<ul style="list-style-type: none"> 1 ... n n equals the value specified at the Maximum Vector Size property.
Dependencies	–

Control

This function import lets you start or stop the waveform execution from within the behavior model.

Value range	<ul style="list-style-type: none"> 0: Stop Stops waveform generation. 1: Start Starts waveform generation. Changes of the Data Vector, Time Vector, and Vector Size function ports change the waveform. 2: No update Suppresses waveform updates at run time. The last value of the Control function port is kept. Changes of the Data Vector, Time Vector, and Vector Size function ports are ignored.
Dependencies	–

Repeats

This function import lets you define from within the behavior model, how often a wavetable is output.

Value range	<ul style="list-style-type: none"> ▪ -1: Infinite repetitions The waveform is executed repeatedly. This means an output signal is generated periodically, for example, a wheel speed sensor signal. ▪ 0: Zero repetitions No waveform is generated. A running waveform execution stops synchronously, regardless of the specified update mode (synchronous or asynchronous). ▪ 1: Single shot The waveform is executed once, i.e., either at each model execution step or when a specific angle occurs (specified via Trigger source property).
Dependencies	-

Note

The **Repeats** function port ignores new values received from the behavior model if a waveform is already being executed. This behavior might cause problems, if the **Repeats** function port is set to 1 (= single shot) and a new waveform generation is triggered while the previous waveform generation is still in progress. In this case, the new trigger is ignored and waveform generation will stop after the previous waveform generation is completed. To avoid this effect, you should set the **Repeats** function port to -1 (= infinite repetitions). In this case, always the last values (received at the function ports) are used immediately for waveform execution.

Trigger Angle Vector

This function import lets you trigger the reading of signals (data/time vectors) from within the behavior model. Signals are read whenever an angle value provided at this function port matches an angle value returned by the master APU provider that is assigned to the function block.

To compare the trigger angles that are provided by the behavior model with the angle values of the master APU provider, the function block converts the trigger angles to the angle range that is specified via the **Angle range** property. For example: The function block converts -10° to 350° if the **Angle range** property is set to 360° .

Value range	<ul style="list-style-type: none"> ▪ $-3600^\circ \dots +3600^\circ$ for each entry in the vector. ▪ The vector size depends on the value specified at the Maximum trigger angle vector size property.
Dependencies	Available only if the Trigger source property is set to Angle-based .

Trigger Angle Vector Size	<p>This function import lets you determine whether all vector entries are considered as trigger angles or only a subset of leading entries from within the behavior model.</p> <table border="1"> <tr> <td>Value range</td><td> <ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property. </td></tr> <tr> <td>Dependencies</td><td>Available only if the Trigger source property is set to Angle-based.</td></tr> </table>	Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property. 	Dependencies	Available only if the Trigger source property is set to Angle-based.
Value range	<ul style="list-style-type: none"> ▪ 1 ... n ▪ n equals the value specified at the Maximum trigger angle vector size property. 				
Dependencies	Available only if the Trigger source property is set to Angle-based.				
Output Enable	<p>This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.</p> <table border="1"> <tr> <td>Value range</td><td> <ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled </td></tr> <tr> <td>Dependencies</td><td>Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.</td></tr> </table>	Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled 	Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.
Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled 				
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.				
Start/End of Waveform	<p>This event port provides an I/O event each time a waveform execution starts or has been finished, according to the setting of the Event trigger condition property.</p> <table border="1"> <tr> <td>Value range</td><td>–</td></tr> <tr> <td>Dependencies</td><td>Available only if the Event generation property is set to Enabled.</td></tr> </table>	Value range	–	Dependencies	Available only if the Event generation property is set to Enabled.
Value range	–				
Dependencies	Available only if the Event generation property is set to Enabled.				
High Reference	<p>This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.</p> <table border="1"> <tr> <td>Value range</td><td>Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Digital Out) on page 783.</td></tr> <tr> <td>Dependencies</td><td>Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.</td></tr> </table>	Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Digital Out) on page 783 .	Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.
Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Digital Out) on page 783 .				
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.				

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Digital Out) on page 783 .
Dependencies	–

Inverted Signal

This signal port and the **Signal** signal port together represent the electrical connection points of a logical signal with two complementary potentials carrying the information in the voltage difference between these signals. Both signal ports (**Signal** and **Inverted Signal**) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Digital Out) on page 783 .
Dependencies	Available only if the Interface type property is set to Differential out .

Reference

This signal port is a reference port and provides the reference signal for the **Signal** signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Waveform Digital Out) on page 783 .
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Waveform Digital Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Event trigger condition	✓	–
	Update mode	✓	–

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports			
	Initial switch setting (Test Automation)	-	✓
	Initial substitute value (Test Automation)	-	✓
Electrical Interface			
	Polarity	✓	-
	Signal termination	✓	-
	Idle Value	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Waveform Digital Out)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------------|-----|
| Configuring the Basic Functionality (Waveform Digital Out)..... | 779 |
| Configuring Standard Features (Waveform Digital Out)..... | 781 |

Configuring the Basic Functionality (Waveform Digital Out)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Controlling waveform execution
- Providing I/O events
- Reducing voltage level overshoots

Controlling waveform execution

Waveform execution has the following control options:

Starting waveform execution Waveform execution can be started by setting the Control function port to Start or No update. No update suppresses any waveform updates at run time.

Specifying waveform repetitions It is possible to execute a waveform once with each trigger event that is provided by the specified trigger source (refer to the Trigger source property) or infinitely. Zero repetitions means that no waveform is generated. Executing a waveform repeatedly means generating a periodic output signal, for example, a wheel speed sensor signal.

Stopping waveform execution Waveform execution can be stopped by setting the Control function port to Stop. Waveform execution stops either immediately, that is, after the current waveform entry (Asynchronous), or after execution of the current waveform finishes (Synchronous), depending on the Update mode property. Execution can also be stopped via the Repeats function port. Zero repetitions (0) always stops a running waveform execution synchronously, regardless of the Update mode property. When stopped, the Idle value is output.

Update Mode	Stop Request Via Leads To
Synchronous	Control = Stop	Synchronous stop
	Repeats = Zero repetitions	Synchronous stop
Asynchronous	Control = Stop	Asynchronous stop
	Repeats = Zero repetitions	Synchronous stop

Updating the waveform at run time The data vector and the time vector can be updated at run time by changing the Data Vector and/or Time Vector function ports. The update takes effect either immediately, that is, after the current waveform entry (Asynchronous) or after the execution of the current waveform finishes (Synchronous), refer to the Update mode property.

Triggering waveform execution The trigger source defines when the function imports read the signals (data/time vectors) computed in the behavior model.

- Model-based: Signals are read with respect to the behavior model's sample rate.
- Angle-based: Signals are read whenever the angle unit (APU) returns an angle value that matches an angle value specified in the Trigger Angle Vector function port.

As a precondition for angle-based triggering you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

- I/O function: Signals are read with each trigger event that is provided by the assigned trigger function provider, for example, the Trigger In function block (via the Assigned trigger function provider property). Trigger function providers have their own defined trigger conditions.

Configuring the waveform specification You can specify either all the single time and data vector entries, or a few values that are expanded to the time and data vectors. This depends on the setting of the Vector Size Mode property.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Waveform Digital Out function block can generate an I/O event each time a waveform execution starts or has been finished (configurable via the Event trigger condition property).

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Waveform property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

Limitation A resistor connected in series to the output driver is supported only by the Digital In/Out 5 channel type. However, signal termination is not supported if you use the Differential out interface type. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586.

Configuring Standard Features (Waveform Digital Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	Depending on the channel type, refer to: <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals

Configuration Feature	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	Further Information
					<p>(Flexible Out 1) on page 98</p> <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109 ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112
Polarity of output signals	✓	✓	<ul style="list-style-type: none"> ▪ ✓ ▪ Not supported for the Differential out interface type. 	✓	–
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.				–
Channel multiplication	✓	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports					
Trigger level of electronic fuse	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 1](#) on page 1575
- [Digital Out 3](#) on page 1577
- [Digital In/Out 5](#) on page 1586
- [Flexible Out 1](#) on page 1604

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Waveform Digital Out)

SCALEXIO Hardware Dependencies (Waveform Digital Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 3	Digital In/Out 5
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board	DS6202 Digital I/O Board
Event generation	✓	✓	✓	✓
Output current range	<ul style="list-style-type: none"> ▪ 0 ... +80 mA_{RMS} for 1 channel ▪ 1.792 A_{RMS} for 28 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +40 mA for 1 channel ▪ 0.32 A for 10 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +140 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	0 ... ±40 mA (each channel)
High side reference voltage	+5 V ... +60 V	-60 V ... +60 V	+5 V ... +60 V	<ul style="list-style-type: none"> ▪ +3.3 V and +5 V (switchable) ▪ +3.3 V (fix) for the differential out interface type
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch 	<ul style="list-style-type: none"> ▪ Low-side switch

Channel Type		Digital Out 1	Flexible Out 1	Digital Out 3	Digital In/Out 5
		<ul style="list-style-type: none"> ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch to internal reference ▪ Push-pull ▪ Differential out
In push-pull configuration the outputs can be set to high impedance (tristate) at run time.					
Value range of time vector entries		1 µs ... 200 s (Resolution: 68 ns)	1 µs ... 200 s (Resolution: 68 ns)	1 µs ... 200 s (Resolution: 68 ns)	24 ns ... 32 s (Resolution: 8 ns)
Angular processing unit	Number of slaves	6	1	6	6
	Maximum speed	168,000 °/s	168,000 °/s	168,000 °/s	1,200,000 °/s
Maximum trigger angle vector size		<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 	<ul style="list-style-type: none"> ▪ 8 for 360° angle range ▪ 16 for 720° angle range 	16
Configurable fuse		–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–	–
Signal termination		–	–	–	<ul style="list-style-type: none"> ▪ 68 Ω (serial termination) ▪ Switchable ▪ Not supported for the differential out interface type
Circuit diagram		Digital Out 1 on page 1575	Flexible Out 1 on page 1604	Digital Out 3 on page 1577	Digital In/Out 5 on page 1586
Required channels		1 (additional channels are required for current enhancement and operation in push/pull mode.)		1	1 (2 channels are required to operate the outputs using the differential out interface type.)

General limitations

Channel multiplication across several I/O boards is not supported.

DS2621 Signal Generation Board The following restrictions apply to the DS2621 Signal Generation Board when the push/pull mode is used:

- Every logical signal requests two channels (primary and auxiliary) on the real-time hardware to be assigned to the function block. These channels must be adjacent to one another on an I/O board. Example: primary channel = n, auxiliary channel = n + 1.

- Channel 10 cannot be used as the primary channel because using channel 1 as the next adjacent channel is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6202 Digital I/O Board The following limitations apply to the DS6202 Digital I/O Board:

- Channel multiplication is not supported in general.
- Only 16 Waveform Digital Out function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Current Signal Capture

Where to go from here

Information in this section

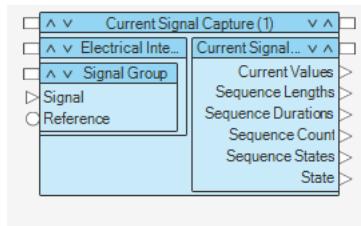
Introduction (Current Signal Capture).....	788
Overviews (Current Signal Capture).....	789
Configuring the Function Block (Current Signal Capture).....	799
Hardware Dependencies (Current Signal Capture).....	808

Introduction (Current Signal Capture)

Introduction to the Function Block (Current Signal Capture)

Function block purpose With the Current Signal Capture function block type, you can measure analog current signals (coming from an external device, e.g., ECU) by capturing signal sequences at configurable sample rates.

Default display The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features These are the main features:

- Capturing sequences of current values at configurable sample rates.
- Supporting different sequence trigger sources and sample trigger sources.
- Providing timestamps and angular timestamps of the captured sequences and values to the behavior model.
- Providing statistical data about the captured sequences to the behavior model.
- Generating I/O events and providing them to the behavior model.

Supported channel types The Current Signal Capture function block supports the following channel types:

	SCALEXIO		MicroAutoBox III
Channel type	Flexible In 1	Flexible In 2	–
Hardware	DS2601	DS2680	–

Overviews (Current Signal Capture)

Where to go from here

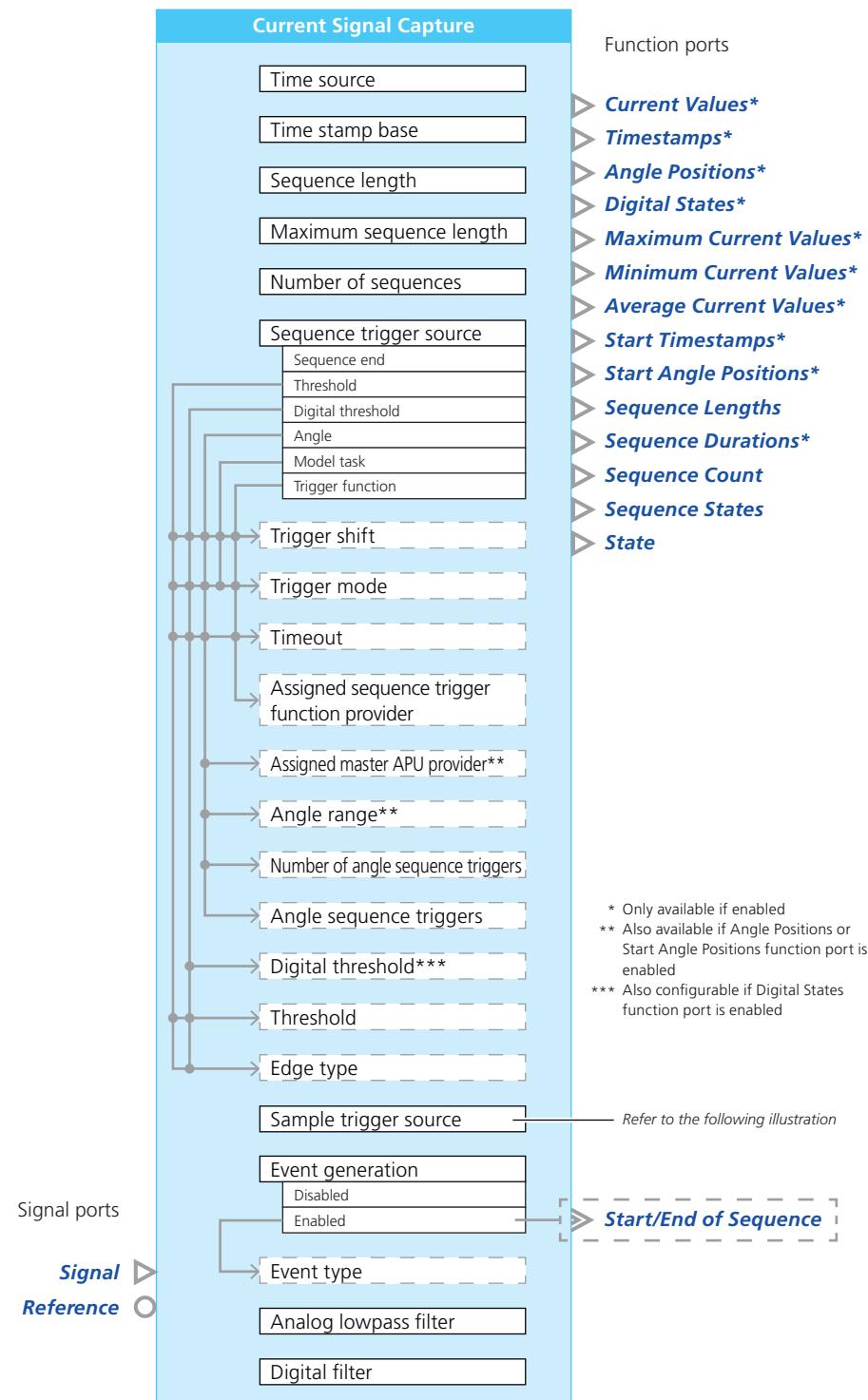
Information in this section

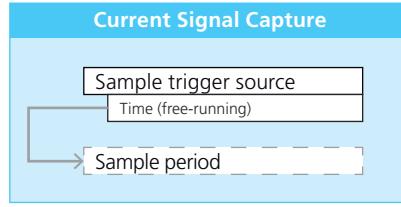
Overview of Ports and Basic Properties (Current Signal Capture).....	789
Overview of Tunable Properties (Current Signal Capture).....	797

Overview of Ports and Basic Properties (Current Signal Capture)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).





To save computation time, you can disable various function ports.

Current Values

This function outport writes a vector with current values for all samples of all the captured sequences to the behavior model.

Value range	<ul style="list-style-type: none"> $I_{min} \dots I_{max}$ (in Ampere) for each entry of the vector. The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Signal Capture) on page 808. The vector size equals the product of the Number of sequences and Maximum sequence length property settings as shown in the following table with value examples.
Dependencies	Available only if the Capture values property is set to True.

The following table shows value examples:

Number of Sequences	Maximum Sequence Length ¹⁾	Values
3	5	$I(1,1), I(1,2), I(1,3), I(1,4), I(1,5); I(2,1), I(2,2), I(2,3), I(2,4), I(2,5); I(3,1), I(3,2), I(3,3), I(3,4), I(3,5)$
4	5	$I(1,1), I(1,2), I(1,3), I(1,4), I(1,5); I(2,1), I(2,2), I(2,3), I(2,4), I(2,5); I(3,1), I(3,2), I(3,3), I(3,4), I(3,5); I(4,1), I(4,2), I(4,3), I(4,4), I(4,5)$

¹⁾ Samples per sequence

Timestamps

This function outport writes a vector with timestamp values for all samples of all the captured sequences to the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... 612,489,500 s for each entry of the vector. The vector size equals the product of the Number of sequences and Maximum sequence length property settings as shown in the following table with value examples.
Dependencies	Available only if the Capture timestamps property is set to True.

The following table shows value examples:

Number of Sequences	Maximum Sequence Length ¹⁾	Values
3	5	T(1,1), T(1,2), T(1,3), T(1,4), T(1,5); T(2,1), T(2,2), T(2,3), T(2,4), T(2,5); T(3,1), T(3,2), T(3,3), T(3,4), T(3,5)
4	5	T(1,1), T(1,2), T(1,3), T(1,4), T(1,5); T(2,1), T(2,2), T(2,3), T(2,4), T(2,5); T(3,1), T(3,2), T(3,3), T(3,4), T(3,5); T(4,1), T(4,2), T(4,3), T(4,4), T(4,5);

¹⁾ Samples per sequence

Angle Positions

This function outport writes a vector with angle position values for all samples of all the captured sequences to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0° ... 360° or 0° ... 720° for each entry of the vector. ▪ The range depends on the Angle range property setting of the function block. ▪ The vector size equals the product of the Number of sequences and Maximum sequence length property settings as shown in the following table with value examples.
Dependencies	Available only if the Capture angle positions property is set to True.

The following table shows value examples:

Number of Sequences	Maximum Sequence Length ¹⁾	Values
3	5	$\varphi(1,1), \varphi(1,2), \varphi(1,3), \varphi(1,4), \varphi(1,5); \varphi(2,1), \varphi(2,2), \varphi(2,3), \varphi(2,4), \varphi(2,5); \varphi(3,1), \varphi(3,2), \varphi(3,3), \varphi(3,4), \varphi(3,5)$
4	5	$\varphi(1,1), \varphi(1,2), \varphi(1,3), \varphi(1,4), \varphi(1,5); \varphi(2,1), \varphi(2,2), \varphi(2,3), \varphi(2,4), \varphi(2,5); \varphi(3,1), \varphi(3,2), \varphi(3,3), \varphi(3,4), \varphi(3,5); \varphi(4,1), \varphi(4,2), \varphi(4,3), \varphi(4,4), \varphi(4,5);$

¹⁾ Samples per sequence

Digital States

This function outport writes a vector with the digital states (0 or 1) of the input voltage to the behavior model for each captured sample in a sequence.

Value range	<ul style="list-style-type: none"> ▪ 0: The captured voltage value is lower than the specified value at the Digital threshold property. ▪ 1: The captured voltage value is higher than the specified value at the Digital threshold property.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> The vector size equals the product of the Number of sequences and Maximum sequence length property settings as shown in the following table with value examples.
Dependencies	Available only if the Capture digital states property is set to True.

The following table shows value examples:

Number of Sequences	Maximum Sequence Length ¹⁾	Values
3	5	S(1,1), S(1,2), S(1,3), S(1,4), S(1,5); S(2,1), S(2,2), S(2,3), S(2,4), S(2,5); S(3,1), S(3,2), S(3,3), S(3,4), S(3,5)
4	5	S(1,1), S(1,2), S(1,3), S(1,4), S(1,5); S(2,1), S(2,2), S(2,3), S(2,4), S(2,5); S(3,1), S(3,2), S(3,3), S(3,4), S(3,5); S(4,1), S(4,2), S(4,3), S(4,4), S(4,5);

¹⁾ Samples per sequence

Maximum Current Values

This function outport writes a vector with the maximum current value for each completed sequence to the behavior model.

Value range	<ul style="list-style-type: none"> $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Signal Capture) on page 808. The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture maximum values property is set to True.

Minimum Current Values

This function outport writes a vector with the minimum current value for each completed sequence to the behavior model.

Value range	<ul style="list-style-type: none"> $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Signal Capture) on page 808. The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture minimum values property is set to True.

Average Current Values

This function outport writes a vector with the mean current value for each completely captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $I_{\min} \dots I_{\max}$ (in Ampere) for each entry of the vector. ▪ The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Signal Capture) on page 808. ▪ The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture average current values property is set to True.

Start Timestamps

This function outport writes a vector with the start time for each completely captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture start timestamps property is set to True.

Start Angle Positions

This function outport writes a vector with the start angle for each completely captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0° ... 360° or 0° ... 720° for each entry of the vector. ▪ The range depends on the Angle range property setting of the function block. ▪ The vector size equals the product of the Number of sequences property setting.
Dependencies	Available only if the Capture start angle positions property is set to True.

Sequence Lengths

This function outport writes a vector with the number of samples for each completely captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 65,535 samples for each entry of the vector. ▪ The vector size equals the Number of sequences property setting.
Dependencies	–

Sequence Durations

This function outport writes a vector with the sequence duration for each completely captured sequence to the behavior model. The durations are not

measured exactly but calculated as the product of the specified values of the Sequence length and Sample period properties.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size equals the Number of sequences property setting.
Dependencies	–

Sequence Count

This function outport writes the number of completely captured sequences to the behavior model.

Value range	0 ... 100
Dependencies	–

Sequence States

This function outport writes a vector with diagnostic information for each captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ For each entry of the vector: <ul style="list-style-type: none"> ▪ 0: No error No error occurred. ▪ 1: Buffer overflow The samples captured in a sequence (via the Sequence Lengths function port) exceed the maximum number of samples specified at the Maximum sequence length property (= values specified at Sequence length property > value specified at Maximum sequence length property). The first measured values are transmitted, all further values are discarded. ▪ 16: Timeout trigger The start of a sequence is triggered by a timeout value. The timeout trigger flag is valid for threshold-based, digital threshold-based and angle-based triggering. ▪ 32: Rising edge trigger The start of a sequence is triggered by a rising edge. The rising edge trigger flag is valid for threshold-based and digital threshold-based triggering. ▪ 64: Falling edge trigger The start of a sequence is triggered by a falling edge. The falling edge trigger flag is valid for threshold-based and digital threshold-based triggering. ▪ The vector size equals the Number of sequences property setting.
Dependencies	–

State This function outport writes diagnostic information for the measured data to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: No error No error occurred. ▪ 1: Data lost Data loss, for example, due to buffer overflow. This occurs if more than 400 samples are captured (for example, when the Sample period property is set to < 5 µs for a task of 2 ms). All measured data is discarded. No sequences are transmitted to the behavior model. ▪ 2: Too many sequences The number of captured sequences exceeds the value specified at the Number of sequences property. One or more of the first captured sequences are discarded.
Dependencies	–

Start/End of Sequence This event port provides an I/O event each time a new sequence starts or at the end of a sequence, according to the setting of the Event type property.

Value range	–
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Event generation property is set to Enabled. ▪ Not supported if the Sequence trigger source property is set to Model task.

Signal This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Signal Capture) on page 808.
Dependencies	–

Reference This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Current Signal Capture) on page 808.
Dependencies	–

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Supported only for the Flexible In 1 channel type.

Overview of Tunable Properties (Current Signal Capture)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Sample period	✓	-
Sequence length	✓	-
Trigger shift	✓	-
Trigger mode	✓	-
Timeout	✓	-
Angle sequence triggers	✓	-
Event type	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Electrical Interface		
Edge type	✓	–
Threshold	✓	–
Digital threshold	✓	–
Analog low-pass filter	✓	–
Assigned digital filter	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Current Signal Capture)

Where to go from here

Information in this section

Configuring the Basic Functionality (Current Signal Capture).....	799
Configuring Trigger Functionality (Current Signal Capture).....	802
Configuring Standard Features (Current Signal Capture).....	807

Configuring the Basic Functionality (Current Signal Capture)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Specifying the sequence length.
- Specifying a sequence trigger source to start the capture of a sequence and a sample trigger source to take samples. Refer to [Configuring Trigger Functionality \(Current Signal Capture\) on page 802](#).
- Specifying the time source for calculating time-dependent values.
- Enabling or disabling the capturing of various data (for example, timestamps, angle positions, min./max. values and average values, digital states (0 or 1) of input voltage) to save computation time.
- Using digital and/or analog filters for the input signal.
- Specifying the generation of I/O events.

Specifying sequence length

You have to specify two types of sequence lengths as follows:

▪ Sequence length

The **Sequence length** property defines the number of samples that are to be captured for each sequence.

- If the value of the **Sequence length** property is nonzero, a sequence ends after the specified number of samples has been captured (= end trigger).
- If the value is set to 0 (= infinite), the **Sequence length** is not used as the end trigger. In this case the sequence is ended with the next sequence trigger. This value is not supported in all trigger modes.

▪ Maximum sequence length

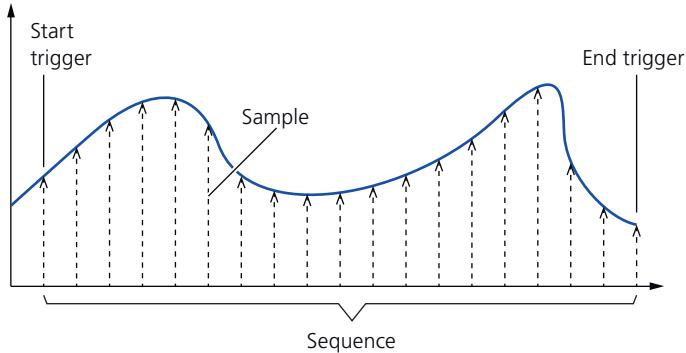
The **Maximum sequence length** defines the maximum number of samples per sequence which can be provided to the behavior model.

The specified value influences the data width of the **Current Values**, the **Timestamps** and the **Angle Positions** function ports.

If the number of captured samples is larger than the specified value of **Maximum sequence length**, the additional values are not provided to the behavior model, but they are still used for calculating the minimum, maximum and average values for a sequence.

Capturing data

Capturing is the process of creating a sequence started by a trigger. A sequence is a list of samples ordered by time. A sample is the digital representation of a snapshot of a continuous signal (value) at a specific time (timestamp). See also the illustration below.



Sequences are completed and then provided to the behavior model when the specified number of samples have been captured or when the sequence is ended by a trigger, which then starts a new sequence.

If more data is captured than can be reported by the function block (for example, due to configuration settings), information about the loss of data is provided and can be accessed by the behavior model. The function block provides a global diagnostic state and diagnostic states for each sequence via specific function ports.

Capturing current, timestamps, and digital states Depending on your requests, you can capture different types of data so you can write it to the behavior model via specific function ports. To save computation time, you are recommended to disable capturing for data that is not required. You can capture the following data:

- Current values for all samples of all the captured sequences
- Timestamps for all samples of all the captured sequences
- Start timestamp values for each complete captured sequence
- Minimum, maximum and average current values for each complete captured sequence
- Digital states (0 or 1) of the input voltage for each captured sample in a sequence. You have to specify a threshold value to differ between the two digital states (via **Digital threshold** property).

For capturing timestamp values, you have to specify, how the timestamp value for each sample is calculated (via **Time stamp base** property). It can be

calculated in relation to the execution start of the real-time application, in relation to the previous sample, or in relation to the sequence start.

Capturing angle positions You can capture the following angle-related data:

- Angle positions for all samples of all the captured sequences
- Start angle positions for each complete captured sequence

As a precondition for capturing angle positions and start angle positions you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Data transfer to the behavior model

The capture result is stored in a buffer. This buffer is updated immediately every time a sequence is completed. However, the behavior model reads the measurement results from the buffer via the function outports, as follows:

- Data transfer depends on the current values available at the Sequence Count and the Sequence Lengths function ports. Data provided by the Current Values and the Timestamps function ports is updated only if the new data is in the range of these two properties.
- Data is not updated if it is outside the range of the current Sequence Count and Sequence Lengths function port values. The existing old values are provided to the behavior model until they are overwritten by new values

The following table shows an example with the following property settings:
Number of sequences = 2, Maximum Sequence Length = 5.

Model Step	Function Port		
	Sequence Count	Sequence Length	Current Values
T1	0	0,0	0, 0, 0, 0; 0, 0, 0, 0
T2	2	4, 4	1, 2, 3, 4, 0; 1, 2, 3, 4, 0
T3	2	2, 5	5, 6, 3, 4, 0; 5, 6, 7, 8, 9
T4	1	3, 5	7, 8, 9, 4, 0; 5, 6, 7, 8, 9

Specifying the time source

For the calculation of time-dependent values, you have to specify the time source (via Time source property). Time-dependent values are, for example, the sample period, timestamps and the timeout value.

The Current Signal Capture function block supports only global time sources. These time sources are provided by a processing unit (for example, the DS6001) in your SCALEXIO system. The time is synchronized with the global system time of your SCALEXIO system.

Using the analog low-pass filter

You can use the analog low-pass filter, for example, to reduce signal noise. The filter characteristics depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Current Signal Capture\)](#) on page 808.

For more basic information on using the filter, refer to [Analog and Digital Filtering with the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Limitation The analog low-pass filter is only supported for the Flexible In 1 channel type.

Using digital filters

The function block supports the usage of digital filters, for example, to smooth the input signal or to reduce signal noise, or to adapt the signal bandwidth to a specific model sample frequency in order to avoid aliasing problems. However, the digital filter properties are only supported for the Time (free-running) sample trigger source. This is because the calculation of the filter coefficients is based on a specific sample frequency.

The filter coefficients are stored in digital filter files which you can import and use them afterwards. The supported filters and filter types depend on the assigned hardware resources. For example, the DS2601 Signal Measurement Board supports various digital filter types, such as, Butterworth filter.

You can change the assigned digital filter file via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

Limitation Importing and using digital filters is only supported for the Flexible In 1 channel type.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Current Signal Capture function block can generate an I/O event each time a sequence is captured. Via the Event type property, you can specify when to trigger the I/O event, either when a new sequence starts or at the end of a sequence.

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Sequence property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Trigger Functionality (Current Signal Capture)

Specifying a sequence trigger source

The sequence trigger source describes the trigger event that is used to start a sequence and where applicable to stop a sequence. You can select one of the following sequence trigger sources:

- Model task
- Sequence end
- Threshold

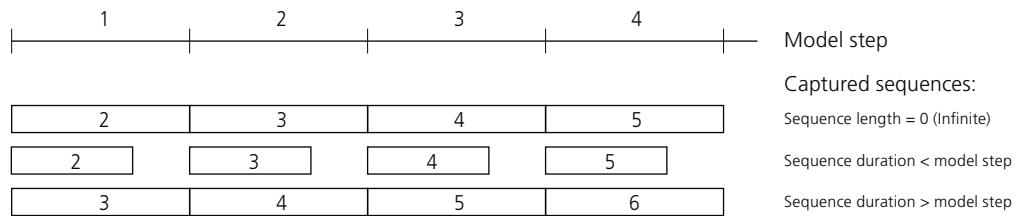
- Angle
- Trigger function
- Digital threshold

Tip

Keep in mind that the value specified at the **Number of sequences** property defines the sequences that can be provided to the behavior model in one model step.

Model task The sequence is triggered synchronously to the model task in which the mapped model port block is executed. The function block starts to capture the sequence if an event triggers the model task, for example, a timer event to execute the next model step. Two cases must be distinguished:

- Case A: The **Sequence length** property is set to 0 (= infinite). Sequences are finished by a new model step. If an event triggers the model task, the function block immediately stops any running capturing process and starts a new sequence capture.
If you set the **Sequence length** property to 0 (= infinite), trigger shifting and the **Ignore** trigger mode are not supported.
- Case B: The **Sequence length** property is set to a nonzero value. The capturing process ends after the specified sequence length has been reached. To provide valid values to the behavior model, a sequence must be captured before the behavior model reads the provided values. Even if the **Immediate** trigger mode is set, the captured sequence is not immediately provided as shown in the following illustration.

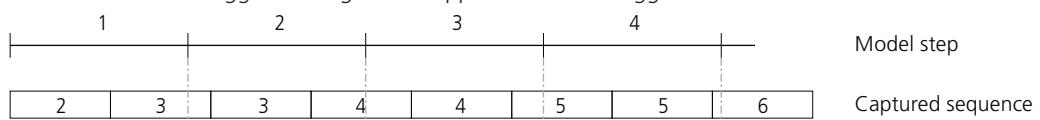


The number in the sequence represents the model step in which the sequence is provided to the model.

Sequence end A new sequence is triggered and therefore started when the current sequence has been completed (= finished).

The first capturing process starts when the execution of the real-time application starts. The value specified at **Sequence length** property is used as the end trigger and also as a start trigger to capture a new sequence.

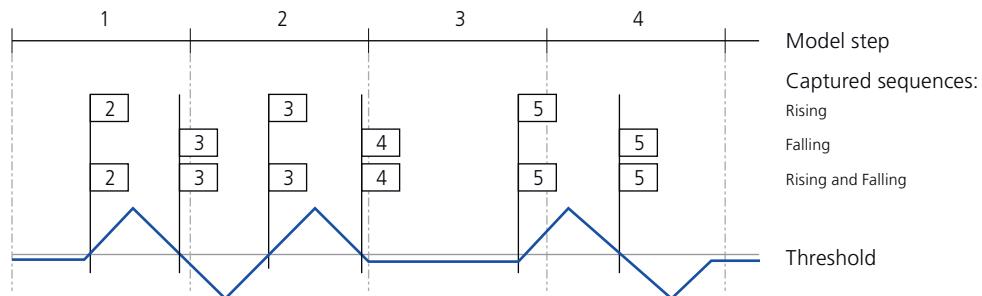
Trigger shifting is not supported for this trigger source.



The number in the sequence represents the model step in which the sequence is provided to the model.

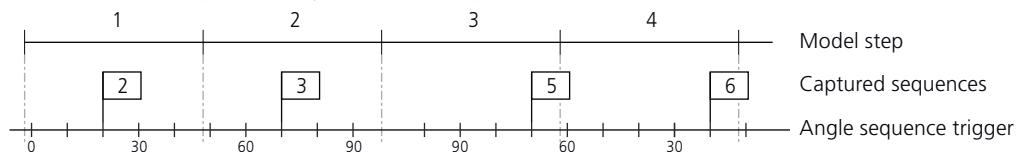
Threshold A sequence is started when the measured current input signal matches the specified trigger condition (threshold and edge direction). You have

to specify a threshold and the edge direction for the trigger via the Threshold and Edge type properties.



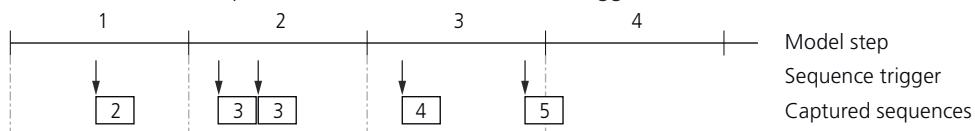
The number in the sequence represents the model step in which the sequence is provided to the model.

Angle A sequence is started when the current angle position is equal to one of the specified Angle sequence triggers positions. The current angle position is provided by a master APU provider.



The number in the sequence represents the model step in which the sequence is provided to the model.

Trigger function A sequence is started by an external trigger which is provided by a specific trigger function, for example, the Trigger In function block (via Assigned sequence trigger function provider property). Trigger function providers have their own defined trigger conditions.

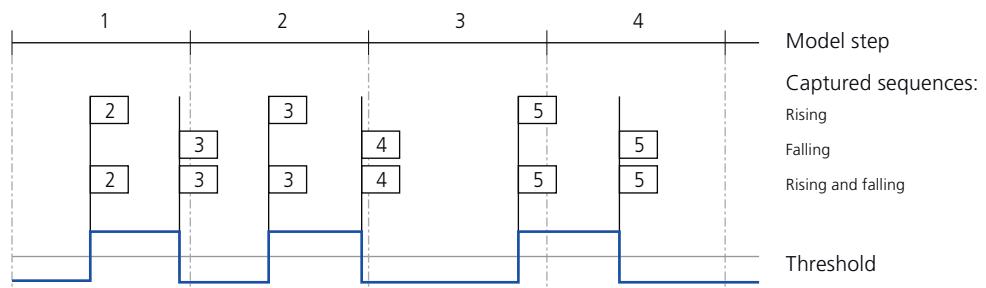


The number in the sequence represents the model step in which the sequence is provided to the model.

Note

To recognize trigger pulses from a trigger function provider the time span between two consecutive trigger pulses must be equal or longer than the reciprocal of the ADC sample rate of the A/D converter on the I/O board. For specific values, refer to [Hardware Dependencies \(Voltage Signal Capture\) on page 834](#) [SCALEXIO Hardware Dependencies \(Current Signal Capture\) on page 808](#).

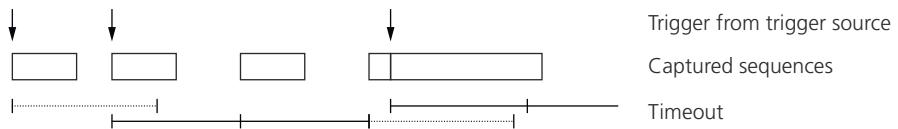
Digital threshold A sequence is started when the measured voltage input signal matches the specified trigger condition (digital threshold and edge direction). You have to specify a threshold and the edge direction for the trigger via the Digital threshold and Edge type properties.



Using timeout values for sequence triggering

You can specify a timeout value (time span) after which a sequence capturing is started even if the specified trigger source does not match the trigger condition. The time span (timeout) begins at the last sequence start triggered by the specified trigger source or by the last timeout trigger.

Example: If you specify a timeout value of 500 ms and you use threshold-based triggering, sequence capturing starts every 500 ms, even if the measured signal value is below the specified threshold value.



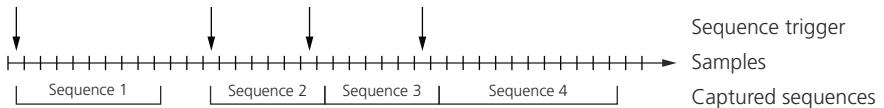
As shown above, a timeout trigger never interrupts a started sequence capture and trigger a new capturing.

Triggering via timeout value is supported for the following sequence trigger sources: Threshold, Digital threshold, Angle, and Trigger function.

Specifying the trigger mode for sequence triggering

The trigger mode declares the behavior of the function block if a trigger occurs while a sequence capture is already in progress. You can specify to ignore (Ignore trigger mode) or to let them stop a running capturing process immediately (Immediate trigger mode).

In the latter case (see illustration below) a trigger stops the current sequence capture and starts a new capturing process. The stopped sequence is treated as complete and the values are provided to the behavior model.



As shown above, the first sample of a sequence that was started by a retrigger will be interpreted as last sample of the sequence that was completed by the retrigger.

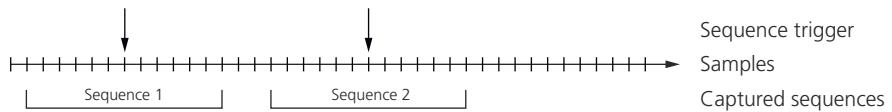
If you use the Sequence end trigger source or you set the Sequence length property to 0 (= infinite), only the Immediate trigger mode is supported.

Specifying trigger shifting for sequence triggers

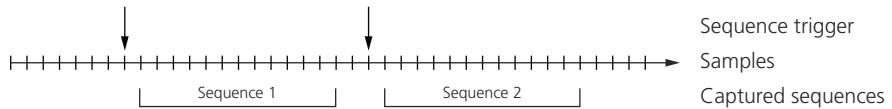
Trigger shifting allows you to adjust the start of sequence capturing relative to the sequence trigger. You can shift the sequence capturing in positive or in negative direction by a specified number of samples.

This can be useful if you have to measure synchronous PWM signals. You can shift the measuring time frame in positive direction so the measurement does not start until the signal has reached a steady state.

Negative shift



Positive shift



Note

The following restrictions apply for trigger shifting:

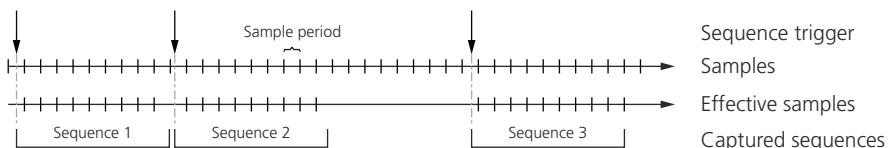
- Sequences triggered by the Sequence end trigger source cannot be shifted.
- Sequences with an infinite sequence length and triggered by the Model task trigger source cannot be shifted.
- Sequence triggered in the Immediate trigger mode cannot be forward shifted.

If you specify a trigger shift even though it is not supported, the function block ignores it and generates a conflict.

Specifying a sample trigger source

The sample trigger source describes the trigger event that is used to take samples. The Current Signal Capture function block supports only the Time (free-running) sample trigger source.

Time (free-running) Samples are taken equidistant at a fixed configurable sample period regardless whether a sequence is captured or not.



Only the Time (free-running) sample trigger source supports digital filtering and the trigger shift functionality.

Configuring Standard Features (Current Signal Capture)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Flexible In 2	Flexible In 1	Further Information
Channel multiplication	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Note: The Current Signal Capture function block type does not support channel multiplication. Therefore the voltage and current ranges of a single hardware channel apply.			
Signal Ports			
Trigger level of electronic fuse	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Flexible In 2 on page 1595](#)
- [Flexible In 1 on page 1593](#)

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Current Signal Capture)

SCALEXIO Hardware Dependencies (Current Signal Capture)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Flexible In 1	Flexible In 2
Hardware	DS2601 Signal Measurement Board	DS2680 I/O Unit
Event generation	✓	✓
Input current range	0 ... +10 A _{RMS}	0 ... +6 A _{RMS}
Input voltage range	-60 V ... +60 V	0 ... +60 V
Measurement range per channel (Threshold)	±30 A	±18 A
Digital threshold	-60 V ... +60 V	0 ... +24 V
Sequence trigger source	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Angle ▪ Threshold ▪ Trigger function ▪ Digital threshold 	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Angle ▪ Threshold ▪ Trigger function ▪ Digital threshold
Sample trigger source	<ul style="list-style-type: none"> ▪ Time (free-running) 	<ul style="list-style-type: none"> ▪ Time (free-running)
Sample period	Range	4.012 µs ... 0.1 s
	Resolution	68 ns
Resolution	16 bit	16 bit
Offset error	±5 mA (typ.)	±15 mA (typ.)
Gain error	±0.1% (typ.)	±0.3% (typ.)
Analog low-pass filter	Filter type	2 nd -order Bessel filter
	Edge frequency (-3dB cutoff frequency)	20 kHz
Using digital filter files	✓ ¹⁾	—
Angular processing unit	Number of slaves	1
	Maximum speed	168,000 °/s
Measurement point	Load side	Load side
Configurable fuse	<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	—
Circuit diagrams	Flexible In 1 on page 1593	Flexible In 2 on page 1595
Required channels	1	1

¹⁾ You only can use digital filter files if the Sample trigger source is set to Time (free-running).

General limitations

The Current Signal Capture function block does not support channel multiplication.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Voltage Signal Capture

Where to go from here

Information in this section

Introduction (Voltage Signal Capture).....	812
Overviews (Voltage Signal Capture).....	813
Configuring the Function Block (Voltage Signal Capture).....	823
Hardware Dependencies (Voltage Signal Capture).....	834

Introduction (Voltage Signal Capture)

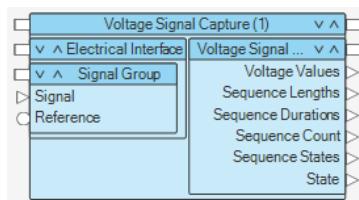
Introduction to the Function Block (Voltage Signal Capture)

Function block purpose

The Voltage Signal Capture function block type measures analog voltage signals from an external device by capturing signal sequences at configurable sample rates.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Capturing sequences of voltage values at configurable sample rates.
- Supporting different sequence trigger sources and sample trigger sources.
- Providing timestamps and angular timestamps of the captured sequences and values to the behavior model.
- Providing statistical data about the captured sequences to the behavior model.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Voltage Signal Capture function block type supports the following channel types:

	SCALEXIO							MicroAutoBox III
Channel type	Flexible In 1	Analog In 1	Analog In 4	Flexible In 3	Analog In 6	Analog In 16	-	
Hardware	DS2601	DS2680	DS6101		DS6221	DS6121	-	

Overviews (Voltage Signal Capture)

Where to go from here

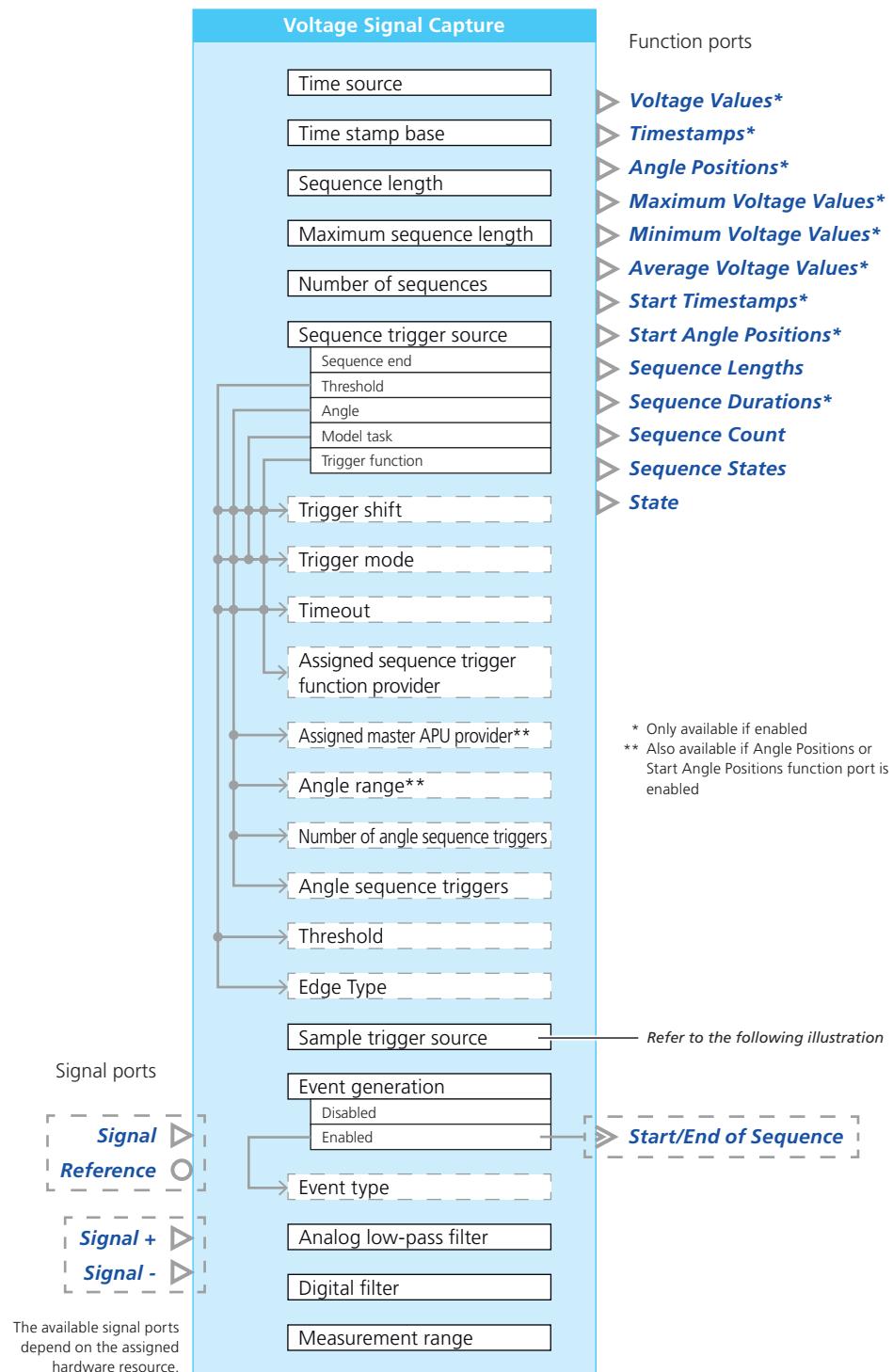
Information in this section

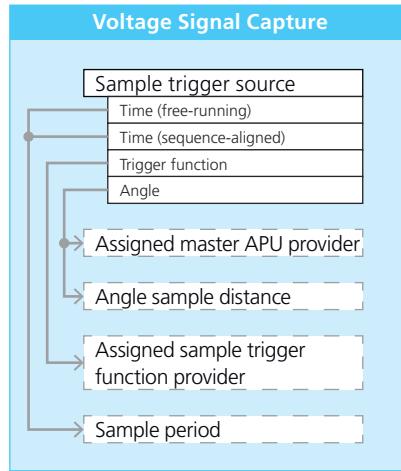
Overview of Ports and Basic Properties (Voltage Signal Capture).....	813
Overview of Tunable Properties (Voltage Signal Capture).....	821

Overview of Ports and Basic Properties (Voltage Signal Capture)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).





To save computation time, you can disable various function ports.

Voltage Values

This function outport writes a vector with voltage values for all samples of all the captured sequences to the behavior model.

Value range	<ul style="list-style-type: none"> $U_{\min} \dots U_{\max}$ (in Volt) for each entry of the vector. The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Signal Capture) on page 834. The vector size equals the product of the Number of sequences and Maximum sequence length property settings as shown in the following table with value examples.
Dependencies	Available only if the Capture values property is set to True.

The following table shows value examples:

Number of Sequences	Maximum Sequence Length ¹⁾	Values
3	5	$U(1,1), U(1,2), U(1,3), U(1,4), U(1,5); U(2,1), U(2,2), U(2,3), U(2,4), U(2,5); U(3,1), U(3,2), U(3,3), U(3,4), U(3,5);$
4	5	$U(1,1), U(1,2), U(1,3), U(1,4), U(1,5); U(2,1), U(2,2), U(2,3), U(2,4), U(2,5); U(3,1), U(3,2), U(3,3), U(3,4), U(3,5); U(4,1), U(4,2), U(4,3), U(4,4), U(4,5);$

¹⁾ Samples per sequence

Timestamps

This function outport writes a vector with timestamp values for all samples of all the captured sequences to the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... 612,489,500 s for each entry of the vector.
-------------	-----------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> The vector size equals the product of the Number of sequences and Maximum sequence length property settings as shown in the following table with value examples.
Dependencies	Available only if the Capture timestamps property is set to True.

The following table shows value examples:

Number of Sequences	Maximum Sequence Length ¹⁾	Values
3	5	T(1,1), T(1,2), T(1,3), T(1,4), T(1,5); T(2,1), T(2,2), T(2,3), T(2,4), T(2,5); T(3,1), T(3,2), T(3,3), T(3,4), T(3,5)
4	5	T(1,1), T(1,2), T(1,3), T(1,4), T(1,5); T(2,1), T(2,2), T(2,3), T(2,4), T(2,5); T(3,1), T(3,2), T(3,3), T(3,4), T(3,5); T(4,1), T(4,2), T(4,3), T(4,4), T(4,5);

¹⁾ Samples per sequence

Angle Positions

This function outport writes a vector with angle position values for all samples of all the captured sequences to the behavior model.

Value range	<ul style="list-style-type: none"> 0° ... 360° or 0° ... 720° for each entry of the vector. The range depends on the Angle range property setting of the function block. The vector size equals the product of the Number of sequences and Maximum sequence length property settings as shown in the following table with value examples.
Dependencies	Available only if the Capture angle positions property is set to True.

The following table shows value examples:

Number of Sequences	Maximum Sequence Length ¹⁾	Values
3	5	$\varphi(1,1), \varphi(1,2), \varphi(1,3), \varphi(1,4), \varphi(1,5); \varphi(2,1), \varphi(2,2), \varphi(2,3), \varphi(2,4), \varphi(2,5); \varphi(3,1), \varphi(3,2), \varphi(3,3), \varphi(3,4), \varphi(3,5)$
4	5	$\varphi(1,1), \varphi(1,2), \varphi(1,3), \varphi(1,4), \varphi(1,5); \varphi(2,1), \varphi(2,2), \varphi(2,3), \varphi(2,4), \varphi(2,5); \varphi(3,1), \varphi(3,2), \varphi(3,3), \varphi(3,4), \varphi(3,5); \varphi(4,1), \varphi(4,2), \varphi(4,3), \varphi(4,4), \varphi(4,5);$

¹⁾ Samples per sequence

Maximum Voltage Values This function outport writes a vector with the maximum voltage value for each completed sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $U_{\min} \dots U_{\max}$ (in Volt) for each entry of the vector. ▪ The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Signal Capture) on page 834. ▪ The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture maximum values property is set to True.

Minimum Voltage Values This function outport writes a vector with the minimum voltage value for each completed sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $U_{\min} \dots U_{\max}$ (in Volt) for each entry of the vector. ▪ The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Signal Capture) on page 834. ▪ The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture minimum values property is set to True.

Average Voltage Values This function outport writes a vector with the mean voltage value for each complete captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $U_{\min} \dots U_{\max}$ (in Volt) for each entry of the vector. ▪ The range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Signal Capture) on page 834. ▪ The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture average current values property is set to True.

Start Timestamps This function outport writes a vector with the start time for each complete captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size equals the Number of sequences property setting.
Dependencies	Available only if the Capture start timestamps property is set to True.

Start Angle Positions

This function outport writes a vector with the start angle for each completely captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0° ... 360° or 0° ... 720° for each entry of the vector. ▪ The range depends on the Angle range property setting of the function block. ▪ The vector size equals the product of the Number of sequences property setting.
Dependencies	Available only if the Capture start angle positions property is set to True.

Sequence Lengths

This function outport writes a vector with the number of samples for each completely captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 65,535 samples for each entry of the vector. ▪ The vector size equals the Number of sequences property setting.
Dependencies	–

Sequence Durations

This function outport writes a vector with the sequence duration for each completely captured sequence to the behavior model. The durations are not measured exactly but calculated as the product of the specified values of the Sequence length and Sample period properties.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 612,489,500 s for each entry of the vector. ▪ The vector size equals the Number of sequences property setting.
Dependencies	–

Sequence Count

This function outport writes the number of completely captured sequences to the behavior model.

Value range	0 ... 100
Dependencies	–

Sequence States

This function outport writes a vector with diagnostic information for each captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ For each entry of the vector: ▪ 0: No error No error occurred. ▪ 1: Buffer overflow The samples captured in a sequence (via the Sequence Lengths function port) exceed the maximum number of
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>samples specified at the Maximum sequence length property (= values specified at Sequence length property > value specified at Maximum sequence length property). The first measured values are transmitted, all further values are discarded.</p> <ul style="list-style-type: none"> ▪ 16: Timeout trigger The start of a sequence was triggered by a timeout value. The timeout trigger flag is valid for threshold-based, digital threshold-based and angle-based triggering. ▪ 32: Rising edge trigger The start of a sequence was triggered by a rising edge. The rising edge trigger flag is valid for threshold-based and digital threshold-based triggering. ▪ 64: Falling edge trigger The start of a sequence was triggered by a falling edge. The falling edge trigger flag is valid for threshold-based and digital threshold-based triggering. ▪ The vector size equals the Number of sequences property setting.
Dependencies	–

State

This function outport writes diagnostic information for the measured data to the behavior model.

	<p>Value range</p> <ul style="list-style-type: none"> ▪ 0: No error No error occurred. ▪ 1: Data lost Data loss, for example, due to buffer overflow. All measured data is discarded. No sequences are transmitted to the behavior model. Valid for Analog in 6 channel type (on DS6221): Together with the data lost value (=1) one of the following flags is written to the behavior model to specify the data loss more precisely: <ul style="list-style-type: none"> ▪ 65,536: Communication error ▪ 131,072: I/O buffer overflow Buffer overflows occur if the required bandwidth for transferring the data is permanently too high. ▪ 2: Too many sequences The number of captured sequences exceeds the value specified at the Number of sequences property. One or more of the first captured sequences are discarded.
Dependencies	–

Start/End of Sequence

This event port provides an I/O event each time a new sequence starts or at the end of a sequence, according to the setting of the Event type property.

Value range	–
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Event generation property is set to Enabled. ▪ Not supported if the Sequence trigger source property is set to Model task.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Signal Capture) on page 834
Dependencies	Not supported for the Analog In 6 channel type.

Signal + / Signal -

These signal ports represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (Signal + and Signal -) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Signal Capture) on page 834
Dependencies	Supported only for the Analog In 6 channel type.

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Voltage Signal Capture) on page 834
Dependencies	Not supported for the Analog In 6 channel type.

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load.

- Supported only for Analog In 1 and Flexible In 1 channel types.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Supported only for the Flexible In 1 channel type.

Overview of Tunable Properties (Voltage Signal Capture)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Sample period	✓	–
Sequence length	✓	–
Trigger shift	✓	–
Trigger mode	✓	–
Timeout	✓	–
Angle sequence triggers	✓	–
Angle sample distance	✓	–
Event type	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Edge type	✓	–
Threshold	✓	–
Measurement point	✓	–
Analog low-pass filter	✓	–
Assigned digital filter	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
----------	------------------------------------------	------------------------------------------

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Note

In addition to the properties listed above, the Number of angle sequence triggers parameter can also be accessed in the experiment software via a variable description file (TRC). However, to avoid malfunctions do not change the value of this parameter in your experiment software.

Configuring the Function Block (Voltage Signal Capture)

Where to go from here

Information in this section

Configuring the Basic Functionality (Voltage Signal Capture).....	823
Configuring Trigger Functionality (Voltage Signal Capture).....	827
Configuring Standard Features (Voltage Signal Capture).....	832

Configuring the Basic Functionality (Voltage Signal Capture)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the sequence length.
- Specifying a sequence trigger source to start the capture of a sequence and a sample trigger source to take samples. Refer to [Configuring Trigger Functionality \(Voltage Signal Capture\)](#) on page 827.
- Specifying the time source for calculating time-dependent values.
- Enabling or disabling the capturing of various data (for example, timestamps, angle positions, min./max. values, and average values) to save computation time.
- Using digital and/or analog filters for the input signal.
- Selecting the voltage measurement range.
- Specifying the generation of I/O events.

Specifying sequence length

You have to specify two types of sequence lengths as follows:

▪ Sequence length

The Sequence length property defines the number of samples that are to be captured for each sequence.

- If the value of the Sequence length property is nonzero, a sequence ends after the specified number of samples has been captured (= end trigger).
- If the value is set to 0 (= infinite), the Sequence length is not used as the end trigger. In this case the sequence is ended with the next sequence trigger. This value is not supported in all trigger modes.

▪ Maximum sequence length

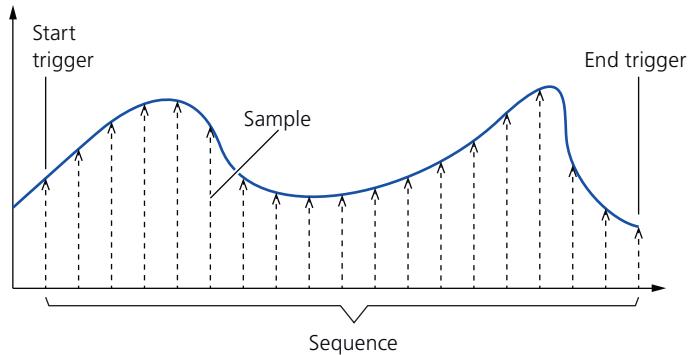
The Maximum sequence length defines the maximum number of samples per sequence which can be provided to the behavior model.

The specified value influences the data width of the Voltage Values, the Timestamps and the Angle Positions function ports.

If the number of captured samples is larger than the specified value of **Maximum sequence length**, the additional values are not provided to the behavior model, but they are still used for calculating the minimum, maximum and average values for a sequence.

Capturing data

Capturing is the process of creating a sequence started by a trigger. A sequence is a list of samples ordered by time. A sample is the digital representation of a snapshot of a continuous signal (value) at a specific time (timestamp). See also the illustration below.



Sequences are completed and then provided to the behavior model when the specified number of samples have been captured or when the sequence is ended by a trigger, which then starts a new sequence.

If more data is captured than can be reported by the function block (for example, due to configuration settings), information about the loss of data is provided and can be accessed by the behavior model. The function block provides a global diagnostic state and diagnostic states for each sequence via specific function ports.

Capturing voltage and timestamps Depending on your requests, you can capture different types of data so you can write it to the behavior model via specific function ports. To save computation time, you are recommended to disable capturing for data that is not required. You can capture the following data:

- Voltage values for all samples of all the captured sequences
- Timestamps for all samples of all the captured sequences
- Start timestamp values for each complete captured sequence
- Minimum, maximum and average voltage values for each complete captured sequence

For capturing timestamp values, you have to specify, how the timestamp value for each sample is calculated (via **Time stamp base** property). It can be calculated in relation to the execution start of the real-time application, in relation to the previous sample, or in relation to the sequence start.

Capturing angle positions You can capture the following angle-related data:

- Angle positions for all samples of all the captured sequences
- Start angle positions for each complete captured sequence

As a precondition for capturing angle positions and start angle positions you have to assign a master APU provider. Refer to [Using Angular Processing Units \(APUs\) on page 126](#).

Data transfer to the behavior model

The capture result is stored in a buffer. This buffer is updated immediately every time a sequence is completed. However, the behavior model reads the measurement results from the buffer via the function outports, as follows:

- Data transfer depends on the current values available at the Sequence Count and the Sequence Lengths function ports. Data provided by the Voltage Values and the Timestamps function ports is updated only if the new data is in the range of these two properties.
- Data is not updated if it is outside the range of the current Sequence Count and Sequence Lengths function port values. The existing old values are provided to the behavior model until they are overwritten by new values

The following table shows an example with the following property settings:
Number of sequences = 2, Maximum Sequence Length = 5.

Model Step	Function Port		
	Sequence Count	Sequence Length	Current Values
T1	0	0,0	0, 0, 0, 0; 0, 0, 0, 0
T2	2	4, 4	1, 2, 3, 4, 0; 1, 2, 3, 4, 0
T3	2	2, 5	5, 6, 3, 4, 0; 5, 6, 7, 8, 9
T4	1	3, 5	7, 8, 9, 4, 0; 5, 6, 7, 8, 9

Specifying the time source

For the calculation of time-dependent values, you have to specify the time source (via Time source property). Time-dependent values are, for example, the sample period, timestamps and the timeout value. You can select between:

- Global: The time source is provided by a processing unit (for example, the DS6001) in your SCALEXIO system. The time is synchronized with the global system time of your SCALEXIO system.
- Local: The time source is provided by the I/O board (for example, the DS6221) which is assigned to the function block.

The local time is not synchronized, but the jitter is significantly lower than with the global time source. You should use this setting on A/D boards if you want to achieve the full signal-to-noise ratio (SNR) performance for the input signals.

Only the Analog In 6 and the Analog In 16 channel types support local time sources.

Using the analog low-pass filter

You can use the analog low-pass filter, for example, to reduce signal noise. The filter characteristics depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Voltage Signal Capture\) on page 834](#).

For more basic information on using the filter, refer to [Analog and Digital Filtering with the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Limitation The analog low-pass filter is only supported for the Flexible In 1 channel type.

Using digital filters

The function block supports the usage of digital filters, for example, to smooth the input signal or to reduce signal noise, or to adapt the signal bandwidth to a specific model sample frequency in order to avoid aliasing problems. However, the digital filter properties are only supported for the Time (free-running) sample trigger source. This is because the calculation of the filter coefficients is based on a specific sample frequency.

The filter coefficients are stored in digital filter files which you can import and use them afterwards. The supported filters and filter types depend on the assigned hardware resources. For example, the DS2601 Signal Measurement Board supports various digital filter types, such as, Butterworth filter.

You can change the assigned digital filter file via experiment software when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

Limitation Importing and using digital filters is only supported for the Flexible In 1 channel type.

Selecting the measurement range

Depending on the assigned hardware resource, you have to select the measurement range for the voltage input signals, so that the range covers the expected input voltage level.

The voltage values for the low and high ranges depend on the hardware assigned to the function block. For specific values, refer to [Hardware Dependencies \(Voltage Signal Capture\)](#) on page 834.

Limitation Selecting the measurement range is only supported for the Analog In 16 channel type.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Voltage Signal Capture function block can generate an I/O event each time a sequence is captured. Via the Event type property, you can specify when to trigger the I/O event, either when a new sequence starts or at the end of a sequence.

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Sequence property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring Trigger Functionality (Voltage Signal Capture)

Specifying a sequence trigger source

The sequence trigger source describes the trigger event that is used to start a sequence and where applicable to stop a sequence. You can select one of the following sequence trigger sources:

- Model task
- Sequence end
- Angle
- Threshold (not supported for the Analog In 6 and Analog In 16 channel types)
- Trigger function

Tip

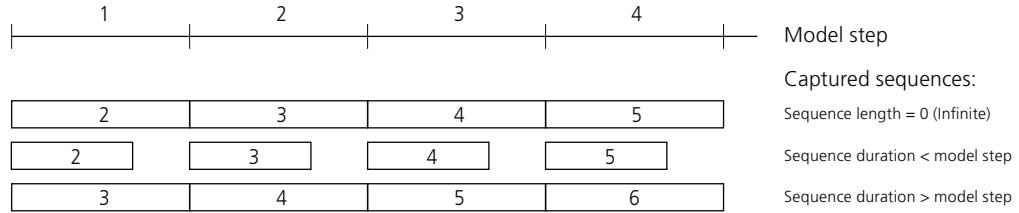
Keep in mind that the value specified at the **Number of sequences** property defines the sequences that can be provided to the behavior model in one model step.

Model task The sequence is triggered synchronously to the model task in which the mapped model port block is executed. The function block starts to capture the sequence if an event triggers the model task, for example, a timer event to execute the next model step. Two cases must be distinguished:

- Case A: The **Sequence length** property is set to 0 (= infinite). Sequences are finished by a new model step. If an event triggers the model task, the function block immediately stops any running capturing process and starts a new sequence capture.

If you set the **Sequence length** property to 0 (= infinite), trigger shifting and the **Ignore** trigger mode are not supported.

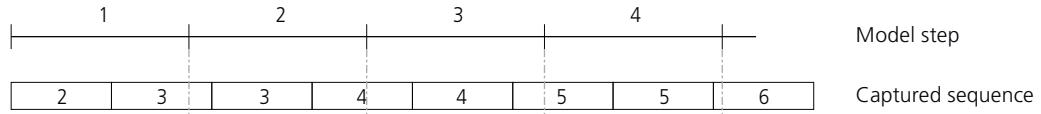
- Case B: The **Sequence length** property is set to a nonzero value. The capturing process ends after the specified sequence length has been reached. To provide valid values to the behavior model, a sequence must be captured before the behavior model reads the provided values. Even if the **Immediate** trigger mode is set, the captured sequence is not immediately provided as shown in the following illustration.



Sequence end A new sequence is triggered and therefore started when the current sequence has been completed (= finished).

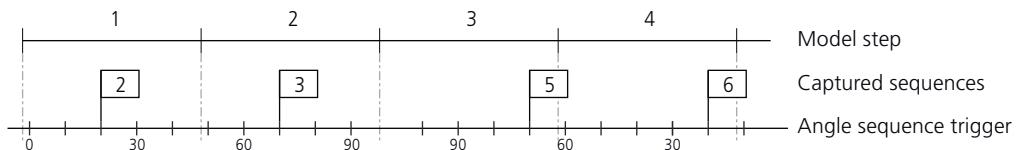
The first capturing process starts when the execution of the real-time application starts. The value specified at Sequence length property is used as the end trigger and also as a start trigger to capture a new sequence.

Trigger shifting is not supported for this trigger source.



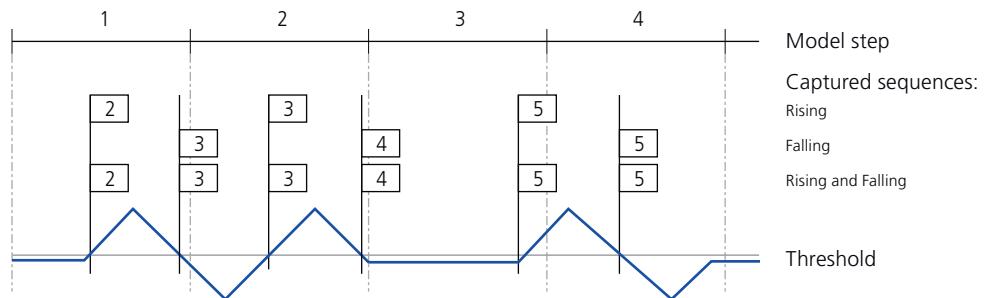
The number in the sequence represents the model step in which the sequence is provided to the model.

Angle A sequence is started when the current angle position is equal to one of the specified Angle sequence triggers positions. The current angle position is provided by a master APU provider.



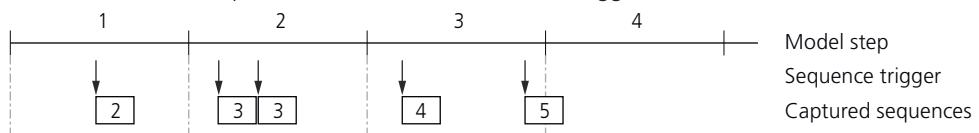
The number in the sequence represents the model step in which the sequence is provided to the model.

Threshold A sequence is started when the measured current input signal matches the specified trigger condition (threshold and edge direction). You have to specify a threshold and the edge direction for the trigger via the Threshold and Edge type properties.



The number in the sequence represents the model step in which the sequence is provided to the model.

Trigger function A sequence is started by an external trigger which is provided by a specific trigger function, for example, the Trigger In function block (via Assigned sequence trigger function provider property). Trigger function providers have their own defined trigger conditions.



The number in the sequence represents the model step in which the sequence is provided to the model.

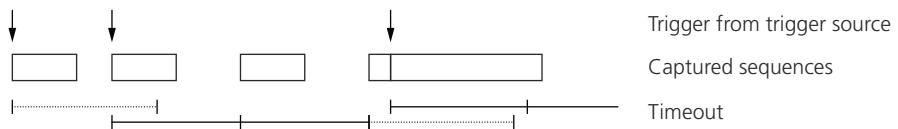
Note

To recognize trigger pulses from a trigger function provider the time span between two consecutive trigger pulses must be equal or longer than the reciprocal of the ADC sample rate of the A/D converter on the I/O board. For specific values, refer to [Hardware Dependencies \(Voltage Signal Capture\) on page 834](#).

Using timeout values for sequence triggering

You can specify a timeout value (time span) after which a sequence capturing is started even if the specified trigger source does not match the trigger condition. The time span (timeout) begins at the last sequence start triggered by the specified trigger source or by the last timeout trigger.

Example: If you specify a timeout value of 500 ms and you use threshold-based triggering, sequence capturing starts every 500 ms, even if the measured signal value is below the specified threshold value.



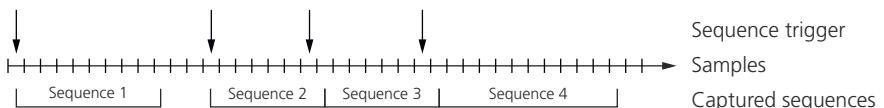
As shown above, a timeout trigger never interrupts a started sequence capture and trigger a new capturing.

Triggering via timeout value is supported for the following sequence trigger sources: Threshold, Angle, and Trigger function.

Specifying the trigger mode for sequence triggering

The trigger mode declares the behavior of the function block if a trigger occurs while a sequence capture is already in progress. You can specify to ignore (Ignore trigger mode) or to let them stop a running capturing process immediately (Immediate trigger mode).

In the latter case (see illustration below) a trigger stops the current sequence capture and starts a new capturing process. The stopped sequence is treated as complete and the values are provided to the behavior model.



As shown above, the first sample of a sequence that was started by a retrigger will be interpreted as last sample of the sequence that was completed by the retrigger.

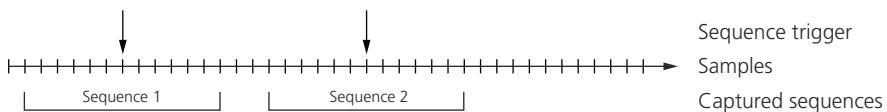
If you use the Sequence end trigger source or you set the Sequence length property to 0 (= infinite), only the Immediate trigger mode is supported.

Specifying trigger shifting for sequence triggers

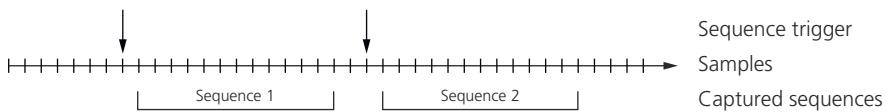
Trigger shifting allows you to adjust the start of sequence capturing relative to the sequence trigger. You can shift the sequence capturing in positive or in negative direction by a specified number of samples.

This can be useful if you have to measure synchronous PWM signals. You can shift the measuring time frame in positive direction so the measurement does not start until the signal has reached a steady state.

Negative shift



Positive shift



Note

The following restrictions apply for trigger shifting:

- Sequences triggered by the Sequence end trigger source cannot be shifted.
- Sequences with an infinite sequence length and triggered by the Model task trigger source cannot be shifted.
- Sequence triggered in the Immediate trigger mode cannot be forward shifted.

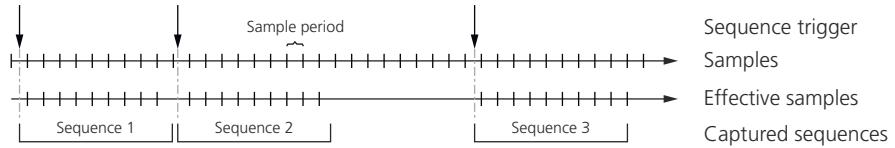
If you specify a trigger shift even though it is not supported, the function block ignores it and generates a conflict.

Specifying a sample trigger source

The sample trigger source describes the trigger event that is used to take samples. You can select one of the following sources:

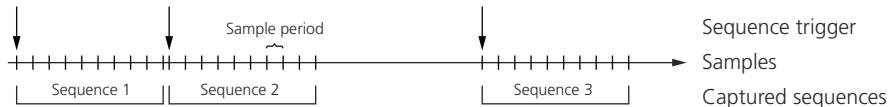
- Time (free-running)
- Time (sequence-aligned) (supported only for Analog In 6 and Analog In 16 channel types)
- Trigger function (supported only for Analog In 6 and Analog In 16 channel types)
- Angle (supported only for Analog In 6 and Analog In 16 channel types)

Time (free-running) Samples are taken equidistant at a fixed configurable sample period regardless whether a sequence is captured or not.



Only the Time (free-running) sample trigger source supports digital filtering and the trigger shift functionality.

Time (sequence-aligned) Only if a sequence is captured, samples are taken equidistant at a fixed configurable sample period. The first sample is generated synchronously to the sequence start.

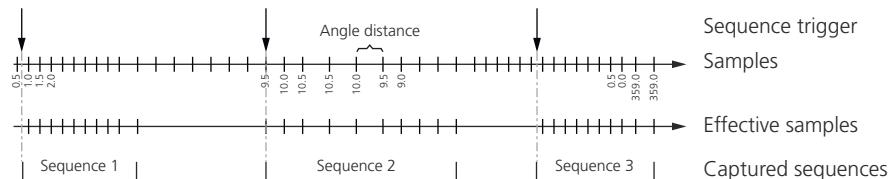


Via the Sample period property you have to specify the time between two samples. The higher the sample rate, the more precise is the signal measurement.

Angle Samples are taken equidistant from an angle unit with a fixed configurable angle distance. The angle range relates to one engine cycle. The engine cycle of a four-stroke piston engine, for example, covers 720°.

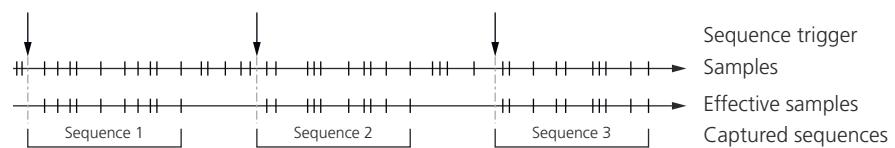
The angle unit is referenced in the assigned master APU provider that resides in your active ConfigurationDesk application, for example, an Angular Clock Setup function block.

Samples that are taken when no sequence is captured are ignored



Trigger function A sample is taken from an external signal via a specific trigger function, for example, the Trigger In function block. These trigger function providers have their own defined trigger conditions.

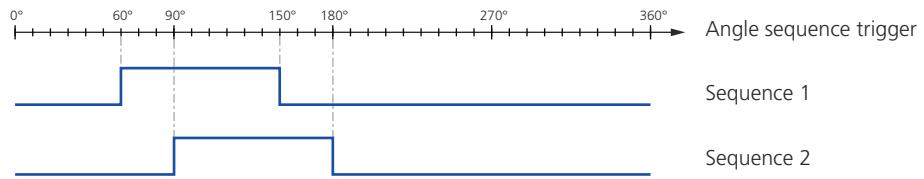
Samples that are taken when no sequence is captured are ignored as shown below.



Using angle-based sequence triggers and angle-based sample triggers together

Because only one APU provider can be assigned to the function block, the same angle unit has to be used as the Angle sequence trigger source and the Angle sample trigger source.

The following example shows an invalid configuration with two overlapping sequences overlapping between 90° and 150°.



Angle sample distance = 1°
Sequence length $\leq 90^\circ$

To avoid overlaps, the difference between two angle sequence trigger positions must not drop below the following value:

Minimum angle position difference = (Sequence length - 1) * Angle sample distance + Minimum angle sample distance

With minimum angle sample distance = $2 * 720^\circ / 2^{16}$

Configuring Standard Features (Voltage Signal Capture)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SCALEXIO							
Configuration Feature	Analog In 1	Analog In 4	Analog In 6	Analog In 16	Flexible In 1	Flexible In 3	Further Information
Measurement point	Load side	–	–	–	▪ Load side ▪ ECU side	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	Differential with ground sense	Differential with ground sense	▪ Differential with ground sense ▪ Differential	Differential with ground sense	▪ Galvanically isolated ▪ Differential with ground sense	Differential with ground sense	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	✓	–	–	–	✓	–	Specifying Current and Voltage Values

SCALEXIO							
Configuration Feature	Analog In 1	Analog In 4	Analog In 6	Analog In 16	Flexible In 1	Flexible In 3	Further Information
							for Channel Multiplication on page 95
Signal Ports							
Trigger level of electronic fuse	–	–	–	–	✓	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	–	✓	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	–	–	–	✓	–	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog In 1](#) on page 1544
- [Analog In 4](#) on page 1546
- [Analog In 6](#) on page 1547
- [Analog In 16](#) on page 1552
- [Flexible In 1](#) on page 1593
- [Flexible In 3](#) on page 1603

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Voltage Signal Capture)

SCALEXIO Hardware Dependencies (Voltage Signal Capture)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog In 1	Flexible In 1	Analog In 4	Flexible In 3	Analog In 6	Analog In 16
Hardware	DS2680 I/O Unit	DS2601 Signal Measurement Board	DS6101 Multi-I/O Board		DS6221 A/D Board	DS6121 Multi-I/O Board
Event generation	✓	✓	✓		✓	✓
Input current range	0 ... +6 A _{RMS}	0 ... +10 A _{RMS}	–		–	–
Measurement range per channel	±18 A	±30 A	–		–	–
Input voltage range	0 ... +60 V	-60 V ... +60 V	0 ... +60 V	-10 V ... +10 V	-10 V ... +10 V	-60 V ... +60 V
Measurement range	0 ... +60 V	-60 V ... +60 V	0 ... +60 V	-10 V ... +10 V	-10 V ... +10 V	<ul style="list-style-type: none"> ▪ Low: -10 V ... +10 V ▪ High: -60 V ... +60 V
Sequence trigger source	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Threshold ▪ Angle ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Threshold ▪ Angle ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Threshold ▪ Angle ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Threshold ▪ Angle ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Threshold ▪ Angle ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Threshold ▪ Angle ▪ Trigger function
Sample trigger source	<ul style="list-style-type: none"> ▪ Time (free-running) 	<ul style="list-style-type: none"> ▪ Time (free-running) 	<ul style="list-style-type: none"> ▪ Time (free-running) 	<ul style="list-style-type: none"> ▪ Time (free-running) ▪ Time (sequence-aligned) ▪ Angle ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) ▪ Time (sequence-aligned) ▪ Angle ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Time (free-running) ▪ Time (sequence-aligned) ▪ Angle ▪ Trigger function
Sample period	Range	11.664 µs ... 0.1 s	4.012 µs ... 0.1 s	11.664 µs ... 0.1 s	250 ns ... 0.1 s	500 ns ... 0.1 s
	Resolution	11.664 µs	68 ns	11.664 µs	<ul style="list-style-type: none"> ▪ 8.5 ns for Time source set to Global ▪ 8 ns for Time source set to Local 	<ul style="list-style-type: none"> ▪ 8.5 ns for Time source set to Global ▪ 8 ns for Time source set to Local
Resolution		16 bit				

Channel Type		Analog In 1	Flexible In 1	Analog In 4	Flexible In 3	Analog In 6	Analog In 16
Sample rate		85.7 kS/s (fixed, multiplexed)	0 ... 250 kS/s	85.7 kS/s (fixed, multiplexed)		0 ... 5 MS/s	0 ... 2 MS/s
Offset error		±10 mV (typ.)	±10 mV (typ.)	±5 mV (typ.)	±2 mV (typ.)	±1 mV (typ.)	Depending on the selected measurement range: ▪ Low: ±2 mV (typ.) ▪ High: ±5 mV (typ.)
Gain error		±0.2 % (typ.)	±0.1 % (typ.)	±0.1 % (typ.)	±0.1 % (typ.)	±0.1 % (typ.)	±0.1 % (typ.)
Analog low-pass filter	Filter type	–	2 nd -order Bessel filter	–		2 nd -order (always active)	–
	Edge frequency (-3dB cutoff frequency)		20 kHz			1.9 MHz	
Using digital filter files		–	✓ ¹⁾	–	–	–	–
Angular processing unit	Number of slaves	6	1	6	6	6	6
	Maximum speed	168,000 °/s	168,000 °/s	168,000 °/s		168,000 °/s	1,200,000 °/s
Supported interface types		Differential with ground sense	▪ Galvanically isolated ▪ Differential with ground sense	Differential with ground sense		▪ Differential with ground sense ▪ Differential	Differential with ground sense
Measurement point		Load side	▪ Load side ▪ ECU side	–	–	–	–
Configurable fuse		–	▪ Load side ▪ 0.5 A _{RMS} ... 10 A _{RMS}	–	–	–	–
Circuit diagrams		Analog In 1 on page 1544	Flexible In 1 on page 1593	Analog In 4 on page 1546	Flexible In 3 on page 1603	Analog In 6 on page 1547	Analog In 16 on page 1552
Required channels		1	1	1	1	1	1

¹⁾ You only can use digital filter files if the Sample trigger source is set to Time (free-running).

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6221 A/D Board For more board-specific data, refer to [Data Sheet of the DS6221 A/D Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Voltage Signal Capture (ADC Type 4)

Where to go from here

Information in this section

Introduction (Voltage Signal Capture - ADC Type 4).....	838
Overviews (Voltage Signal Capture - ADC Type 4).....	839
Configuring the Function Block (Voltage Signal Capture - ADC Type 4).....	843
Hardware Dependencies (Voltage Signal Capture - ADC Type 4).....	849

Introduction (Voltage Signal Capture - ADC Type 4)

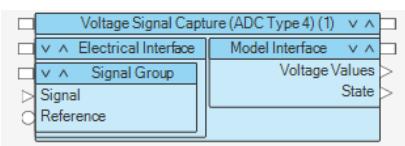
Introduction to the Function Block (Voltage Signal Capture - ADC Type 4)

Function block purpose

With the Voltage Signal Capture (ADC Type 4) function block, you can measure analog voltage signals (coming from an external device) by capturing signal sequences, for example, at configurable sample rates. The function block type is exclusively designed to be used for the ADC Type 4 module of the DS1511, DS1511B1, and DS1513 Multi-I/O Boards.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Capturing a sequence of voltage values at different samples
- Specifying a sequence trigger source and a sample trigger source
- Generating I/O events and providing them to the behavior model
- Providing diagnostic information to the behavior model

Supported channel types

The Voltage Signal Capture (ADC Type 4) function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III	
Channel type	-	Analog In 7	Analog In 8
Hardware	-	DS1511	<ul style="list-style-type: none">▪ DS1511B1▪ DS1513

Overviews (Voltage Signal Capture - ADC Type 4)

Where to go from here

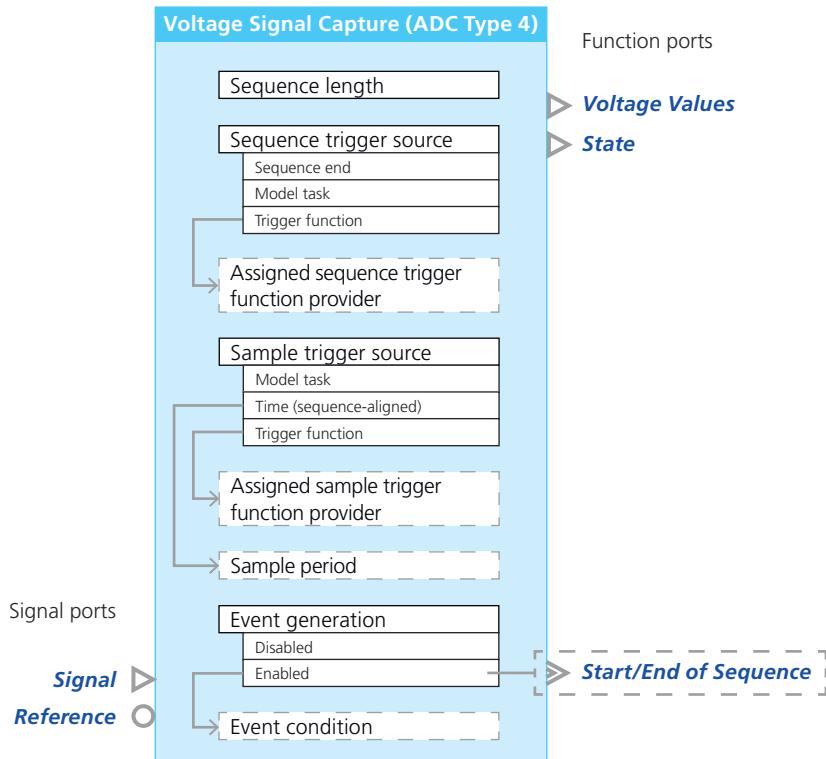
Information in this section

Overview of Ports and Basic Properties (Voltage Signal Capture - ADC Type 4).....	839
Overview of Tunable Properties (Voltage Signal Capture - ADC Type 4).....	842

Overview of Ports and Basic Properties (Voltage Signal Capture - ADC Type 4)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Voltage Values

This function outport writes a vector with the voltage values for all samples of the last captured sequence to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $U_{\min} \dots U_{\max}$ (in Volt) for each entry of the vector. ▪ The range depends on the assigned hardware resource. For concrete values, refer to Hardware Dependencies (Voltage Signal Capture - ADC Type 4) on page 849. ▪ The number of entries in the vector equals the Sequence length property setting.
Dependencies	–

State

This function outport provides diagnostic information for the captured data via bits in a diagnostic word.

Value range	<ul style="list-style-type: none"> ▪ 0: No error occurred. ▪ > 0: See table below.
Dependencies	–

The following table shows the values and descriptions of the diagnostic flags:

Bit (Flag)	Value	Description
2	2	<p>Sequence overflow. A new sequence was captured completely before the data of the previous one was read from the behavior model. Data of the previous sequence is discarded. Only the measured data of the newest sequence is provided to the behavior model. To avoid a sequence overflow, the behavior model must read the data at least after each captured sequence. This you can achieve by one of the following measures:</p> <ul style="list-style-type: none"> ▪ If the I/O function is executed in a periodic task in the behavior model, the model step size must be in a range that <i>at most one</i> sequence is captured between two model steps. ▪ You can execute the I/O function in a function-call subsystem, that is triggered by a sequence end trigger.
25	16,777,216	<p>Conversion overflow. A sequence start trigger is received before the preceding sequence capturing has been finished. The newest sequence start trigger is ignored. A conversion overflow can occur only if you use a trigger function as a sequence trigger source, for example, the Trigger In function block. To avoid conversion overflow, the time span between two consecutive trigger pulses must be longer than the reciprocal of the ADC sample rate of the A/D converter on the hardware. For specific values, refer to Hardware Dependencies (Voltage Signal Capture - ADC Type 4) on page 849.</p>
26	33,554,432	<p>Old data. Data available at the Voltage Values function port did not change since the last reading.</p>

Bit (Flag)	Value	Description
		<p>To avoid reading old data, the behavior model must not read the data more than once after each captured sequence. This you can achieve by one of the following measures:</p> <ul style="list-style-type: none"> ▪ If the I/O function is executed in a periodic task in the behavior model, the model step size must be in a range that <i>at least one</i> sequence is captured between two model steps. ▪ You can execute the I/O function in a function-call subsystem, that is triggered by a sequence end trigger. <p>For some trigger combinations, timing conditions must be met to ensure that the triggering works as desired. If the time conditions are not met, the function block might enter an error state in which data capturing is not performed and the old data is not overwritten. For more information on the affected combinations and time conditions, refer to Specifying Trigger Sources (Voltage Signal Capture - ADC Type 4) on page 844.</p>

If required, two or more pieces of diagnostic information are provided in one diagnostic word. For example, if a message has a sequence overflow (bit 2, value 2) and a conversion overflow (bit 25, value 16,777,216), the diagnostic word value is 16,777,218. To get the specific diagnostic information, you must evaluate the provided diagnostic word in the behavior model.

Start/End of Sequence

This event port provides an I/O event each time a new sequence starts or at the end of a sequence, according to the setting of the Event condition property.

Value range	–
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Event generation property is set to Enabled. ▪ Not supported if the Sequence trigger source property is set to Model task.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For concrete values, refer to Hardware Dependencies (Voltage Signal Capture - ADC Type 4) on page 849
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For concrete values, refer to Hardware Dependencies (Voltage Signal Capture - ADC Type 4) on page 849
Dependencies	–

Overview of Tunable Properties (Voltage Signal Capture - ADC Type 4)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Sample period	✓	–
Sequence length	✓	–
Event condition	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
–	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Voltage Signal Capture - ADC Type 4)

Where to go from here

Information in this section

Configuring the Basic Functionality (Voltage Signal Capture - ADC Type 4).....	843
Specifying Trigger Sources (Voltage Signal Capture - ADC Type 4).....	844
Configuring the Standard Features (Voltage Signal Capture - ADC Type 4).....	848

Configuring the Basic Functionality (Voltage Signal Capture - ADC Type 4)

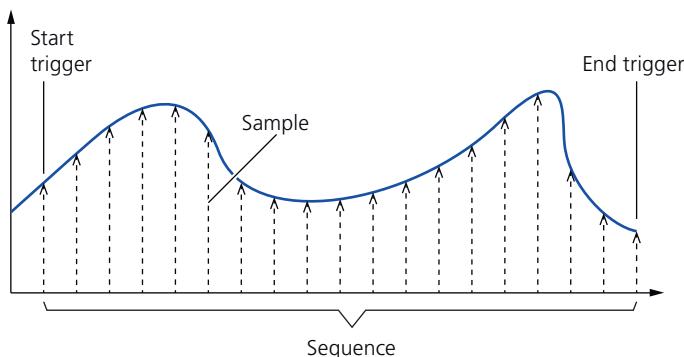
Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Specifying the sequence length
- Specifying a sequence trigger source to start the capture of a sequence and a sample trigger source to take samples. Refer to [Specifying Trigger Sources \(Voltage Signal Capture - ADC Type 4\) on page 844](#).
- Specifying event generation

Specifying sequence length

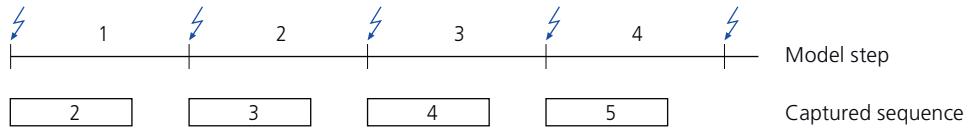
You have to specify the number of samples that are to be captured for each sequence (via Sequence length property). A sequence is started by a trigger. It is a list of samples ordered by time. A sample is the digital representation of a snapshot of a continuous signal (value) at a specific time (timestamp). See also the illustration below.



Sequences are completed and then provided to the behavior model only when the specified number of samples (= sequence length) have been captured.

Providing valid values to the behavior model

The capture result first is stored in a buffer. This buffer is updated immediately every time a sequence is completed. However, the captured sequence is not immediately provided to the behavior model as shown in the following illustration:



The number in the sequence represents the model step in which the seqence is provided to the model.

To provide valid values to the behavior model, a sequence capturing process must be finished before the behavior model reads the value. This you can achieve by one of the following measures:

- If the I/O function is executed in a periodic task in the behavior model, the model step size must be in a range that at least one sequence is captured between two model steps.
- You can execute the I/O function in a function-call subsystem, that is triggered by a sequence end trigger.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Voltage Signal Capture (ADC Type 4) function block can generate an I/O event each time a sequence is captured. Via the Event condition property, you can specify when to trigger the I/O event, either when a new sequence starts or at the end of a sequence.

To use the I/O events in the behavior model, you have to assign them to a task via the Start/End of Sequence property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Specifying Trigger Sources (Voltage Signal Capture - ADC Type 4)

Introduction

You have to specify a sequence trigger source to start the capture of a sequence and a sample trigger source to take samples.

Specifying a sequence trigger source

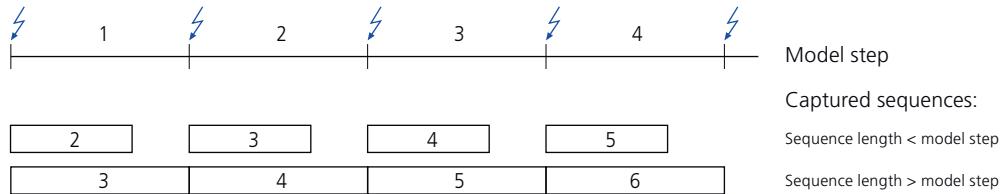
The sequence trigger source describes the trigger event that is used to start a sequence and where applicable to stop a sequence. You can select one of the following sequence trigger sources:

- Model task
- Sequence end
- Trigger function

Model task The sequence is triggered synchronously to the model task in which the mapped model port block is executed. The function block starts to capture the sequence if an event triggers the model task, for example, a timer event to execute the next model step.

The capturing process ends after the specified sequence length has been reached.

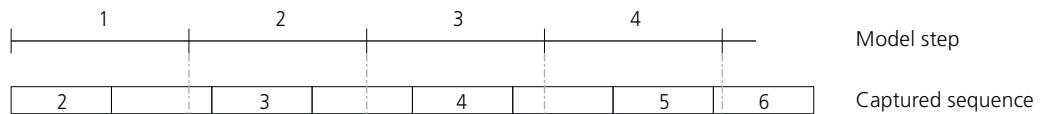
To provide valid values to the behavior model, a sequence must be captured before the behavior model reads the provided values. The captured sequence is not immediately provided as shown in the following illustration.



The number in the sequence represents the model step in which the sequence is provided to the model.

Sequence end A new sequence is triggered and therefore started when the current sequence has been completed (= finished).

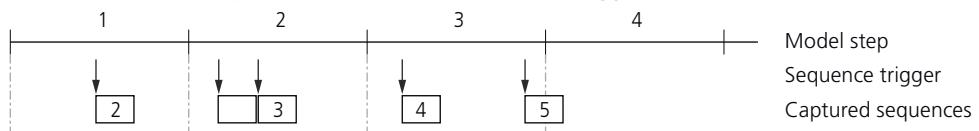
The first capturing process starts when the execution of the real-time application starts. The value specified at Sequence length property is used as the end trigger and also as a start trigger to capture a new sequence.



The number in the sequence represents the model step in which the sequence is provided to the model.

The value of sequences without a number are not provided to the behavior model. They are overwritten by the next sequence before they can be read by the behavior model.

Trigger function A sequence is started by an external trigger which is provided by a specific trigger function, for example, the Trigger In function block (via Assigned sequence trigger function provider property). Trigger function providers have their own defined trigger conditions.



The number in the sequence represents the model step in which the sequence is provided to the model.

The value of sequences without a number are not provided to the behavior model. They are overwritten by the next sequence before they can be read by the behavior model.

Note

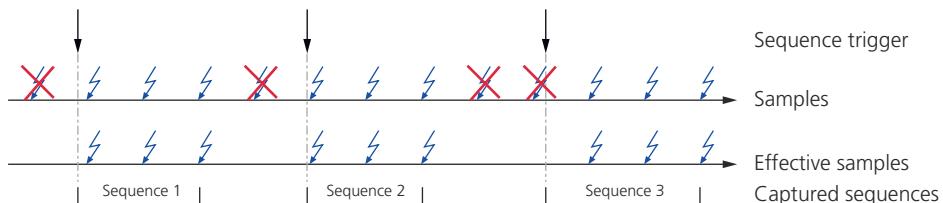
To recognize trigger pulses from a trigger function provider the time span between two consecutive trigger pulses must be equal or longer than the reciprocal of the ADC sample rate of the A/D converter on the I/O board. For specific values, refer to [Hardware Dependencies \(Voltage Signal Capture\) on page 834](#)[Hardware Dependencies \(Voltage Signal Capture - ADC Type 4\) on page 849](#).

Specifying a sample trigger source

The sample trigger source describes the trigger event that is used to take samples. You can select one of the following sources:

- Model task
- Time (sequence-aligned)
- Trigger function

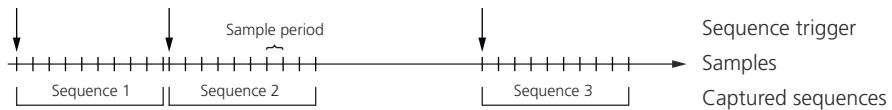
Model task Samples are taken synchronously to the model task in which the mapped model port block is executed.



Note

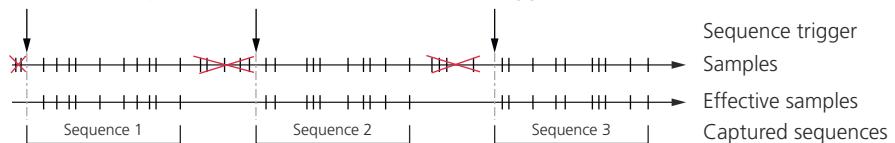
If samples are taken when no sequence is captured, the function block might enter an error state in which data capturing is terminated and the old data is not overwritten. In this case, the function block provides the old data diagnostic information to the behavior model via the State function port.

Time (sequence-aligned) Only if a sequence is captured, samples are taken equidistant at a fixed configurable sample period. The first sample is generated synchronously to the sequence start.



Via the Sample period property you have to specify the time between two samples. The higher the sample rate, the more precise is the signal measurement.

Trigger function A sample is taken from an external signal via a specific trigger function, for example, the Trigger In function block. These trigger function providers have their own defined trigger conditions.



Note

If samples are taken when no sequence is captured the function block might enter an error state in which data capturing is terminated and the old data is not overwritten. In this case, the function block provides the old data diagnostic information to the behavior model via the State function port.

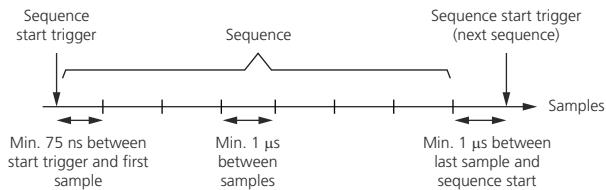
Unsupported and limited combination of trigger sources

Some combinations of trigger sources are not supported by the function block as a whole. Other combinations have unsupported features or must meet specific timing conditions.

		Sample Trigger Source		
		Model Task	Time (Sequence-aligned)	Trigger Function
Sequence Trigger Source	Model task	Combination is not supported. ¹⁾	Event generation is not supported. ¹⁾	<ul style="list-style-type: none"> ▪ Timing conditions must be met. ▪ Event generation is not supported.¹⁾
	Sequence end	Combination without limitations.	Combination without limitations.	Combination without limitations.
	Trigger function	Timing conditions must be met.	Combination without limitations.	<ul style="list-style-type: none"> ▪ Timing conditions must be met. ▪ The assigned trigger function provider must be different from each other.¹⁾

¹⁾ No code is generated and a conflict is displayed in the Conflicts Viewer.

Timing conditions For some trigger source combinations (see table above), timing conditions must be met. Refer to the following illustration.



If the timing conditions are not met, the function block might enter an error state in which all other sequence and sample triggers are ignored until the real-time application is downloaded again to the hardware. In this case, data capturing is terminated and the function block provides the old data diagnostic information to the behavior model via the State function port.

Configuring the Standard Features (Voltage Signal Capture - ADC Type 4)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The electrical interface of the Voltage Signal Capture (ADC Type 4) function block does not provide configurable standard features.
-----------------------------	-------------------------------------------------------------------------------------------------------------------------------------

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog In 7](#) on page 1548
- [Analog In 8](#) on page 1548

Model interface	The interface to the behavior model of the function block provides the following standard features.
------------------------	-----------------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Voltage Signal Capture - ADC Type 4)

MicroAutoBox III Hardware Dependencies (Voltage Signal Capture - ADC Type 4)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Characteristics		Analog In 7	Analog In 8
Board		DS1511 Multi-I/O Board	<ul style="list-style-type: none"> ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board
Input voltage range		0 ... +5 V	-10 V ... +10 V
Sequence trigger source		<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Model task ▪ Sequence end ▪ Trigger function
Sample trigger source		<ul style="list-style-type: none"> ▪ Model task ▪ Time (sequence-aligned) ▪ Trigger function 	<ul style="list-style-type: none"> ▪ Model task ▪ Time (sequence-aligned) ▪ Trigger function
Sample period	Range	1 µs ... 1 s	1 µs ... 1 s
	Resolution	10 ns	10 ns
Resolution		16 bit	16 bit
Sample rate		0 ... 1 MS/s	0 ... 1 MS/s
Offset error		±0.5 mV	±3.0 mV
Gain error		±0.25 %	±0.25 %
Circuit diagrams		Analog In 7 on page 1548	Analog In 8 on page 1548
Required channels		1	1

More hardware data

DS1511 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Digital Incremental Encoder In

Where to go from here

Information in this section

Introduction (Digital Incremental Encoder In).....	852
Overviews (Digital Incremental Encoder In).....	858
Configuring the Function Block (Digital Incremental Encoder In).....	864
Hardware Dependencies (Digital Incremental Encoder In).....	874

Introduction (Digital Incremental Encoder In)

Where to go from here

Information in this section

Introduction to the Function Block (Digital Incremental Encoder In).....	852
Basics on Digital Incremental Encoders.....	853
Basics on Measuring the Position and Speed (Digital Incremental Encoder In).....	855

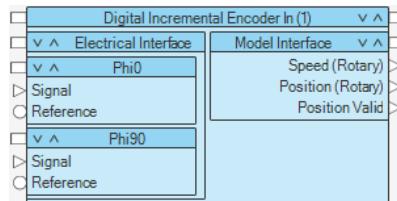
Introduction to the Function Block (Digital Incremental Encoder In)

Function block purpose

The Digital Incremental Encoder In function block provides access to rotary or linear digital incremental encoders. The function block can be used, for example, to measure the angular position and the speed of an electric motor.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Adjusting the function block to the connected encoder.
- Determining and providing angular positions and speed values to the behavior model.
- Calibrating the position measurement via index signal of the connected encoder and/or from within the behavior model.
- Enabling the master APU functionality to provide the position values to be used by other function blocks.

Available demo project

ConfigurationDesk provides the *EDrivesControlDemo* project. This project is an example for electric motor control using the DS6121 Multi-I/O Board. You can use the project as a template, for example, to copy parts of it to your application.

For a description of the project, refer to [EDrivesControlDemo Project: Example of Electric Motor Control Using the DS6121 Multi-I/O Board \(ConfigurationDesk Demo Projects\)](#).

Supported channel types

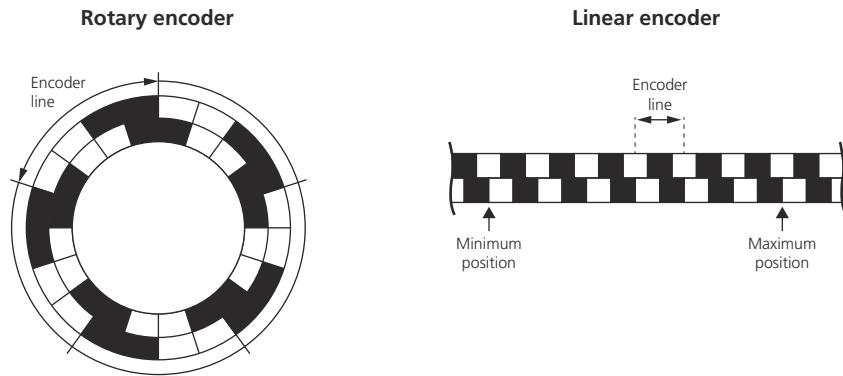
The Digital Incremental Encoder In function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Digital In/Out 9	Flexible In/Out 1	Digital In/Out 5	Digital In 4
Hardware	DS6121		DS6202	<ul style="list-style-type: none"> ▪ DS1511 ▪ DS1511B1 ▪ DS1513

Basics on Digital Incremental Encoders

Introduction

A digital incremental encoder is a sensor that converts the motion of a shaft (rotary encoder) or a linear movement (linear encoder) to a digital code. The following illustration shows examples of coded pattern used for incremental encoders.

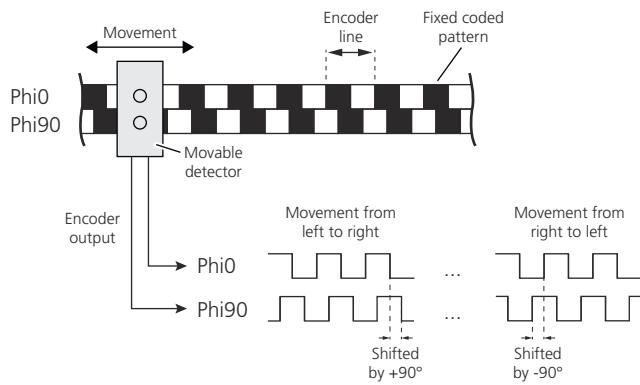


Incremental encoders are commonly used in applications that require precise measurement and control of position and speed.

Measurement principle

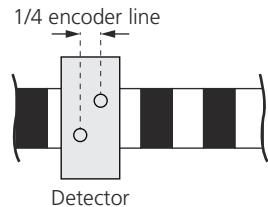
Incremental encoders usually provide three signals: The Phi0 and Phi90 signals and an optional index signal.

Phi0 and Phi90 signals The Phi0 and Phi90 signals are series of square-wave pulses (increments) that are used to analyze position changes. The signals are generated by a detector that moves over a coded pattern with equally spaced lines (encoder lines). Depending on the encoder, either the detector is movable or the coded pattern. In the example below, the detector is movable.



The Phi0 and Phi90 signals are shifted by 90° against each other. This shift can be used to detect the direction of movement.

Instead of coding the pattern with two tracks for Phi0 and Phi90 as shown in the example above, it is common to shift the detector for a quarter encoder line as shown in the illustration below. This results in the same output signals.



Index signal Incremental encoders usually provide an index signal as the third signal. The index signal can be used to calibrate the position measurement to measure the absolute position.

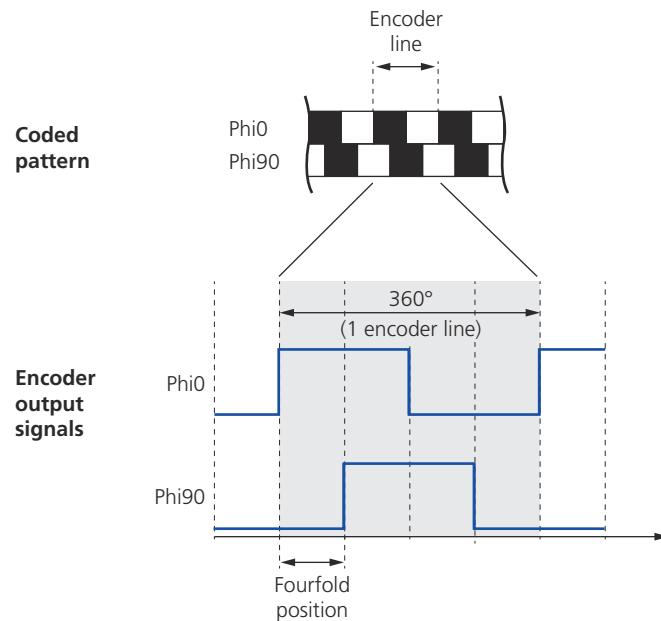
Rotary encoders generate the index signal periodically with each rotation. Linear encoders generate an index signal at a certain position only.

Measurement precision

The number of encoder lines per revolution (rotary encoder) or encoder lines between the minimum position and the maximum position (linear encoder) determines the resolution of an encoder. The more encoder lines are available, the more precise is the position and speed measurement.

Encoder lines The following illustration shows the shapes of the generated Phi0 and Phi90 digital signals. The gray-shaded area represents one encoder line. An encoder line starts with any edge of the Phi0 or Phi90 signals and finishes four edges later (two edges of the Phi0 signal and two edges of the Phi90 signal).

To describe the 90° phase shift of the Phi90 signal, one encoder line is represented by 360° .

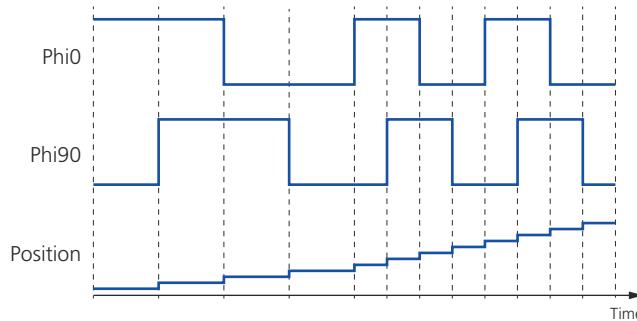


Fourfold position One encoder line is represented by four fourfold positions. If one fourfold position is passed, the detector (or coded pattern) is moved for a quarter encoder line. This method is used to increase the resolution.

Basics on Measuring the Position and Speed (Digital Incremental Encoder In)

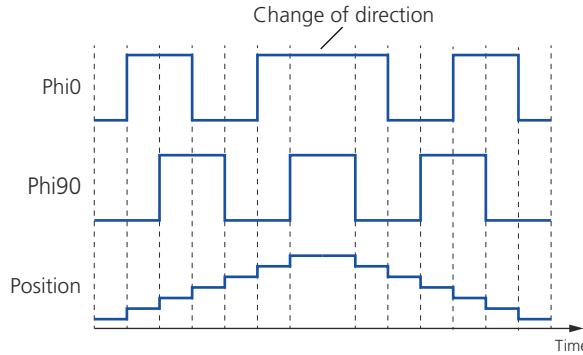
Position measurement

The Digital Incremental Encoder In function block counts the number of passed fourfold positions to measure the position. When the Phi0 signal leads the Phi90 signal, the function block increments the position value after each detected edge as shown below.



The function block converts the counted position value by using the resolution of the connected encoder and outputs the position value in degree (rotary encoder) or meter (linear encoder).

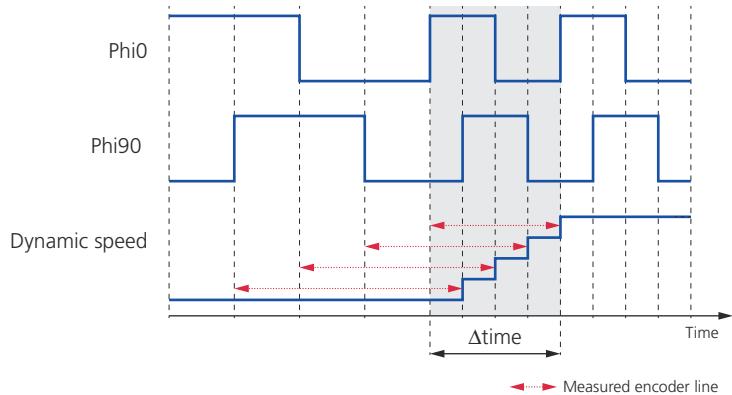
The function block decrements the position value after each detected edge when the Phi0 signal follows the Phi90 signal. The following illustration shows the position value when a change of direction occurs.



Speed measurement

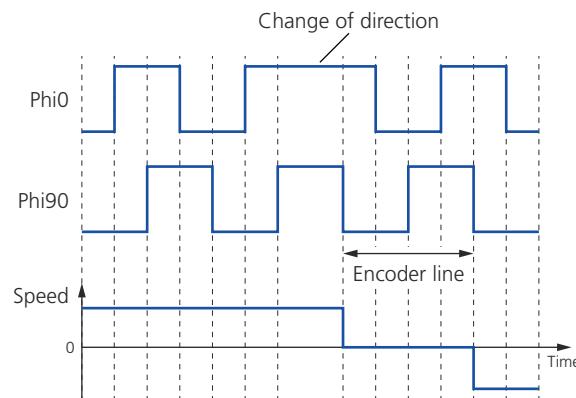
The function block can determine the speed of the connected encoder on the basis of the measured position value. The speed is calculated as the derivative of the position: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode (dynamic or average).

Dynamic speed measurement The dynamic speed is the calculated speed for one encoder line. $\Delta\text{position}$ is the position difference that is covered by one encoder line. Δtime is the difference between the time stamps of the related signal edges. Refer to the following example.



The example above also shows you the run of the speed curve. It takes one encoder line until the measured speed reaches its final value. Keep in mind that an encoder line starts with any edge of the **Phi0** or **Phi90** signals and finishes four edges later.

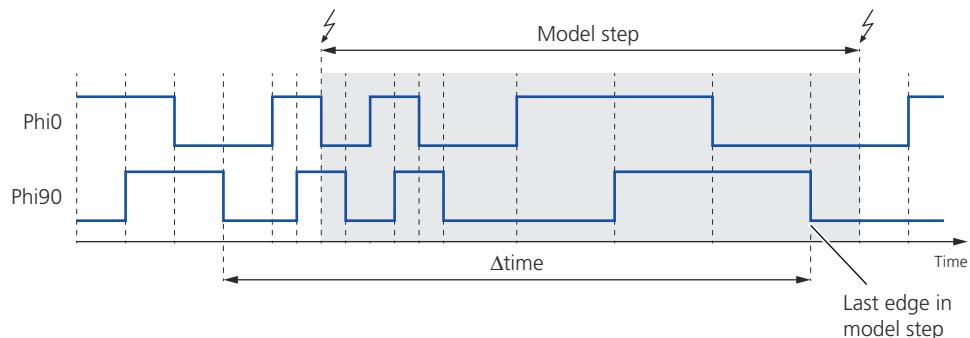
When the direction changes, new dynamic speed values are available after the first entire encoder line has been detected. The example below shows an immediate change of direction.



The dynamic measurement mode provides more accurate speed values at rapidly changing rotation speeds.

Average speed measurement The average speed is the calculated speed in a model step. When a new model step starts, the function block looks for the last edge that occurs in the past model step. In the example below, this is the falling edge of Phi90. For the speed calculation similar edges are always used. In the example below, this is the falling edge of Phi90 that occurred shortly before the last model step starts.

Δt ime is the difference between the time stamps of these signal edges. Δp osition is the position difference within the related signal edges.



If no encoder line ends in the past model step, the function block outputs the last value of the average speed measurement.

The average measurement mode provides more accurate speed values at less rapid changes or at constant rotation speeds. For most applications, it is recommended and useful to select this mode.

Oversviews (Digital Incremental Encoder In)

Where to go from here

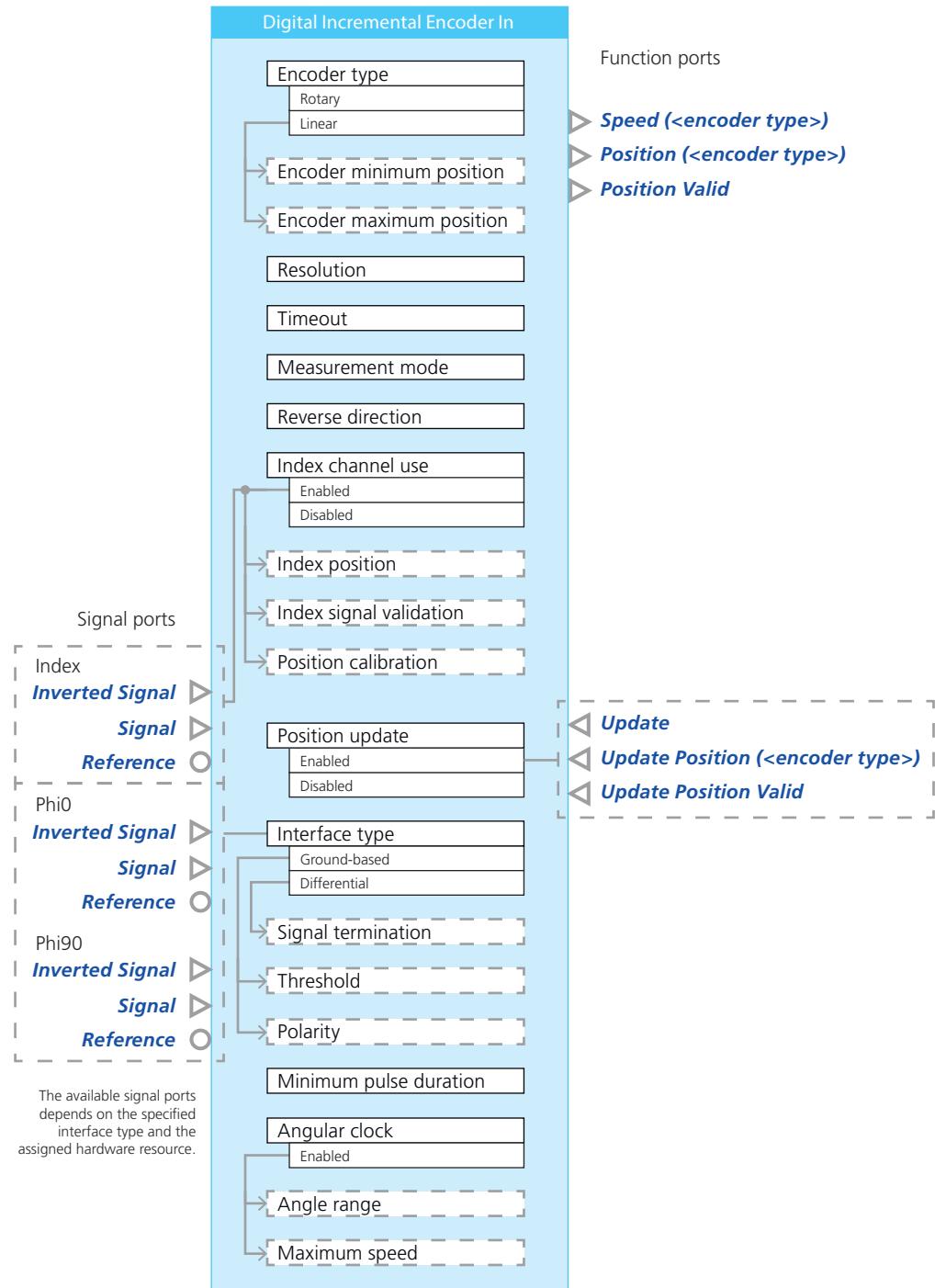
Information in this section

Overview of Ports and Basic Properties (Digital Incremental Encoder In).....	858
Overview of Tunable Properties (Digital Incremental Encoder In).....	862

Overview of Ports and Basic Properties (Digital Incremental Encoder In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Speed (<encoder type>)**

This function outport writes the measured speed of the connected encoder to the behavior model. The sign (+ or -) of the speed value depends on the direction of rotation/movement.

Value range	[Double.min] ... [Double.max] in °/s or m/s (depending on the specified encoder type)
Dependencies	—

Position (<encoder type>)

This function outport writes the measured position of the connected encoder to the behavior model.

Value range	<p>Depends on the specified encoder type:</p> <ul style="list-style-type: none"> ▪ Rotary encoder: 0° ... 360°. At 360° the position counter restarts from 0°. ▪ Linear encoder: [Encoder minimum position] ... [Encoder maximum position] in meter <p>Positive and negative values are possible.</p>
Dependencies	—

Position Valid

This function outport provides the status to the behavior model, indicating whether the measured position value is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid The measured position value is stated as invalid if one the following points apply: <ul style="list-style-type: none"> ▪ The position measurement is not calibrated with an index signal. ▪ The measured position is set to invalid from the behavior model via the Update Position Valid function port. ▪ 1: Valid The measured position value is stated as valid if one the following points apply: <ul style="list-style-type: none"> ▪ The position measurement is calibrated with the index signal of the connected encoder. ▪ The measured position is set to valid from the behavior model via the Update Position Valid function port.
Dependencies	—

Update

This function import triggers the following activities from within the behavior model:

- The calibration (update) of the position measurement with the value of the Update Position function port.
- The update of the Position Valid function outport with the value of the Update Position Valid function import.

Value range	<ul style="list-style-type: none"> ▪ 0: False There is no trigger signal from the behavior model.
-------------	----------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 1: True <p>The calibration of the position measurement and the update of the Position Valid function output is triggered from the behavior model.</p>
Dependencies	Available only if the Position update property is set to Enabled.

Update Position (<encoder type>)

This function import reads the position value for the calibration of the position measurement from the behavior model. The calibration is performed when it is triggered by the Update function port.

Value range	<p>Depends on the specified encoder type:</p> <ul style="list-style-type: none"> ▪ Rotary encoder: <ul style="list-style-type: none"> ▪ -3600° ... +3600° ▪ 0° ... +360° (effective range) ▪ If you enter values outside the effective range, the system converts the entered value to a valid value. Examples: -10° is converted to 350°. 1430° is also converted to 350°. ▪ Linear encoder: [Encoder minimum position] ... [Encoder maximum position] in meter <p>Positive and negative values are possible.</p>
Dependencies	Available only if the Position update property is set to Enabled.

Update Position Valid

This function import reads the value for the update of the Position Valid function port from the behavior model. The update is performed when it is triggered by the Update function port.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid An update sets the Position Valid function port to Invalid. ▪ 1: Valid An update sets the Position Valid function port to Valid.
Dependencies	Available only if the Position update property is set to Enabled.

Inverted Signal

This signal port and the Signal signal port together represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (Signal and Inverted Signal) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Incremental Encoder In) on page 874 .
Dependencies	Available only if the Interface type property is set to Differential.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Incremental Encoder In) on page 874 .
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For concrete values, refer to Hardware Dependencies (Digital Incremental Encoder In) on page 874 .
Dependencies	Available only if the Interface type property is set to Ground-based.

Overview of Tunable Properties (Digital Incremental Encoder In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Properties	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
Timeout	✓	–	–
Encoder minimum position	✓	–	–
Encoder maximum position	✓	–	–
Index position	✓	–	–
Index signal validation	✓	–	–
Position calibration	✓	–	–
Function Ports			
Initial switch setting (Test Automation)	–	✓	–
Initial substitute value (Test Automation)	–	✓	–

	Properties	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Electrical Interface			
Measurement mode	✓	–	
Threshold	✓	–	
Polarity	✓	–	
Signal termination	✓	–	
Minimum pulse duration	✓	–	

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Digital Incremental Encoder In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Digital Incremental Encoder In).....	864
Using Calibration for Position Measurement (Digital Incremental Encoder In).....	869
Configuring Standard Features (Digital Incremental Encoder In).....	873

Configuring the Basic Functionality (Digital Incremental Encoder In)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Adjusting the function block to the connected encoder (encoder type, resolution, rotation direction)
- Configuring calibration for the position measurement. Refer to [Using Calibration for Position Measurement \(Digital Incremental Encoder In\)](#) on page 869.
- Specifying speed measurement (measurement mode, standstill detection).
- Specifying the trigger threshold, pulse filter, and the polarity for the signal measurement.
- Terminating the input signal.
- Enabling the master APU functionality to provide the position values to be used by other function blocks.

Adjusting the function block to the connected encoder

You have to define some characteristics of the connected encoder to adjust it to the function block to work properly.

Selecting the encoder type You have to select the type of the connected encoder via the Encoder type property. You can measure the position of a rotary or a linear digital incremental encoder with the function block.

- Rotary: Position values are provided in degrees.
- Linear: Position values are provided in meters. For the linear encoder type, you have to specify the Encoder minimum position and the Encoder maximum position.

You can change the minimum and the maximum position via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying the resolution You have to specify the resolution of the connected encoder. The specified value must match the resolution value of the connected encoder. Refer to the data sheet of the encoder. The higher the resolution, the more precise is the position measurement.

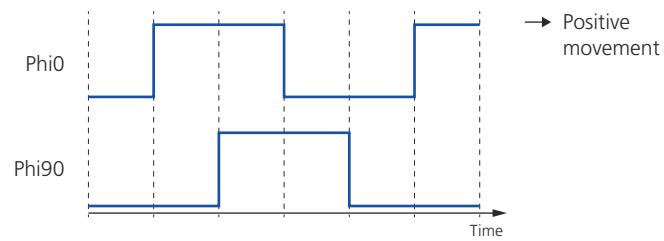
- For rotary encoders, you have to enter the number of encoder lines for one revolution.
- For linear encoders, you have to enter the total number of encoder lines that can be measured between the minimum position (Encoder minimum position property) and the maximum position (Encoder maximum position property).

Adjusting rotational and movement direction To adjust the rotational direction (rotary encoder) or the movement of a linear encoder to the direction required in the behavior model, you can change the direction by enabling the Reverse direction property. This is useful if the encoder must be mounted in the wrong direction, for example, on the rotor shaft.

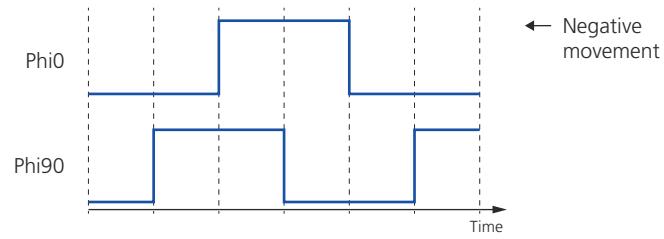
If you enable the Reverse direction property, the reverse order is interpreted as a forward rotation. As a result, the speed values provided at the Speed function port are inverted and the values provided at the Position function port move in the reverse direction.

The following illustration shows the Phi0 and Pi90 signals and how the Digital Incremental Encoder In function block interprets the phase shift of these signals with respect to the rotation and linear direction.

Rotation direction: Forward



Rotation direction: Reverse



The measured values are interpreted as a forward rotation if the Phi0 signal leads the Phi90 signal. Refer to the data sheet of the encoder to find out, in which mounting position this applies.

Specifying speed measurement

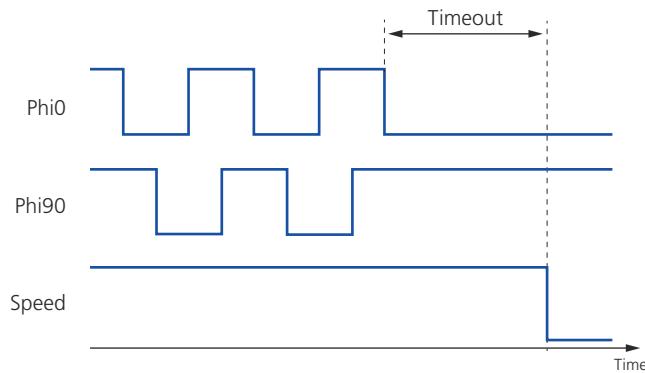
You can measure the dynamic speed or the average speed of the connected digital incremental encoder. The measured values are provided to the behavior model via the Speed function port.

The speed is calculated as follows: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode.

Dynamic speed The dynamic speed is the instantaneous speed value determined on the basis of one encoder line. For details, refer to [Basics on Measuring the Position and Speed \(Digital Incremental Encoder In\) on page 855](#).

Average speed The average speed is the measured speed in a model step. For details, refer to [Basics on Measuring the Position and Speed \(Digital Incremental Encoder In\) on page 855](#).

Detecting standstill The Digital Incremental Encoder In function block detects a standstill if the Phi0 and Phi90 signals have no zero crossings for a time span that you specify with the Timeout property. Refer to the illustration below.

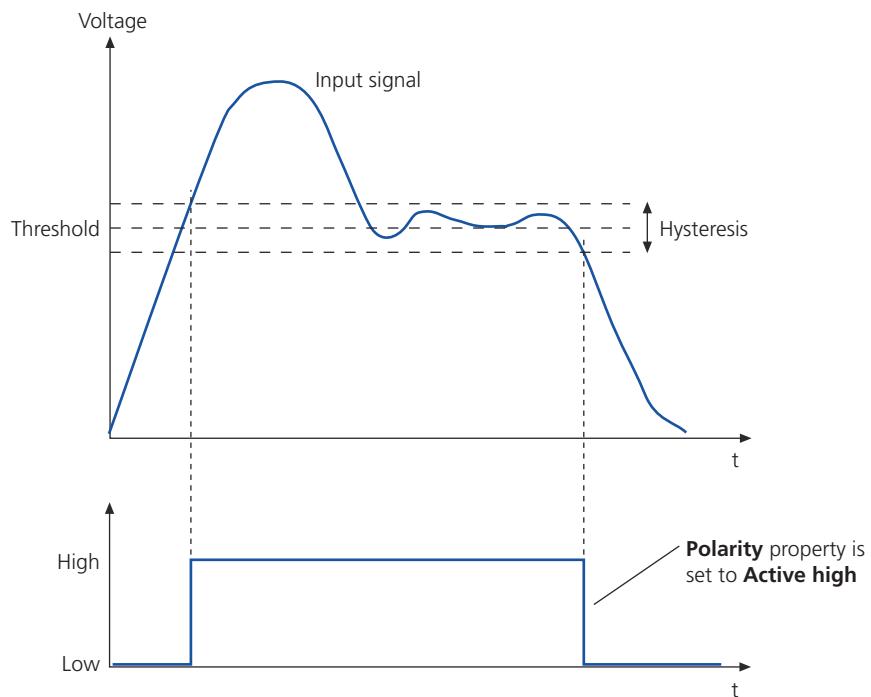


If standstill is detected, the value of the Speed function port is set to 0 immediately. A detected standstill does not affect the position measurement.

You can change the measurement mode setting and the timeout value via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying the trigger threshold

If you use the ground-based interface type, you can specify a threshold value to transform the input signal into binary values. The hysteresis value is fixed and depends on the assigned hardware resource. A rising edge of the input signal must exceed the threshold plus half the hysteresis value to be detected as high, and a falling edge of the input signal must fall below the threshold minus the half hysteresis value to be detected as low.



The value range of the threshold and the hysteresis constant depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Digital Incremental Encoder In\) on page 874](#).

Filtering the input signal

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the pulse duration depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Digital Incremental Encoder In\) on page 874](#).

You can also change the value of the **Minimum pulse duration** property via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying polarity of the signal

To adapt the real-time hardware to signals of different polarity without having to use inverters on the hardware, you can switch the **Polarity** of the input signal.

- Active high

The higher voltage value of the signal represents a binary 1 and the lower value represents a binary 0.

- Active low

The lower voltage value of the signal represents a binary 1 and the higher value represents a binary 0.

Note

Switching the polarity is only supported if you use the ground-based interface type.

Limitation The polarity setting is not supported for the following channel types:

- SCALEXIO hardware: Flexible In/Out 1, Digital In/Out 5 (for the differential interface type setting)

Terminating the input signal

You can activate signal termination for differential input signals to prevent signal reflections at the line end.

Note

You cannot activate a signal termination for ground-based input signals.

Therefore, signal termination is supported only by the Digital In/Out 5 and Flexible In/Out 1 channel types. A $120\ \Omega$ resistor is connected in series to a 5 nF capacitor between the input lines. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586 or [Flexible In/Out 1](#) on page 1608.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Enabling master APU functionality

You can enable the master APU functionality for the function block to provide the measured angle positions to be used by other function blocks as the angle source, such as the Block-Commutated PWM Out function block.

To do this, you have to assign a master APU to the Digital Incremental Encoder In function block, which is located on the same dSPACE board as the I/O channels that are assigned to the function block.

The angular speed of the assigned master APU might exceed the angular speed that can be processed by the slave APU of the hardware resource of the connected function block. Therefore, you can specify a maximum speed (via the Maximum speed property) that fulfills the requirements of your application.

ConfigurationDesk uses the specified maximum speed to check if the hardware resources support the maximum speed.

For more information on the APU functionality, refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Limitation The Master APU functionality is not supported for MicroAutoBox III hardware.

Using Calibration for Position Measurement (Digital Incremental Encoder In)

Motivation

The calibration of the position measurement is necessary if you want to measure and to provide the absolute position at the Position function port. An incremental encoder provides only relative positions via its Phi0 and Phi90 signals.

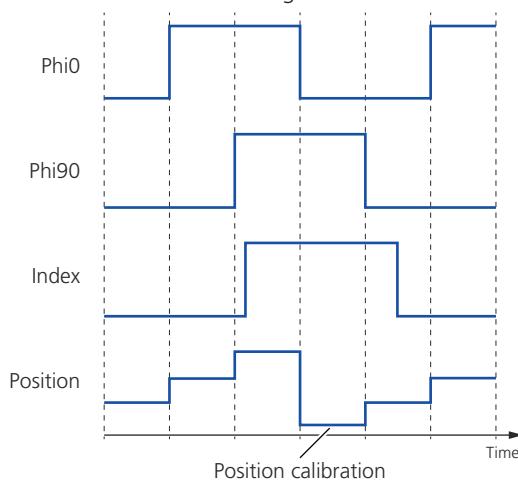
Configuring calibration for the position measurement

Methods for calibration There are two methods to calibrate the position measurement: You can calibrate the position with the index signal or from the behavior model.

You can use both methods in parallel. For example: Until the index signal with its accurate position is not measured, you can calibrate the signal measurement with a less accurate position from the behavior model.

Calibrating with the index signal The index signal gives an absolute position of the connected encoder that can be used for calibration.

If the index signal becomes active, the position measurement will be calibrated with the value of the Index position property when the next edge of the Phi0 or Phi90 signal is detected. The position will also be calibrated if the index becomes inactive before the next edge is reached.



You enable and specify the calibration of the position measurement as follows:

- You enable the index signal measurement with the **Index channel use** property.
- You enter the position of the index in the **Index position** property. The entered position is internally rounded to a fourfold position.
- You specify if only the first index is used to calibrate the position of the incremental encoder or every index with the **Position calibration** property.
- You can increase the accuracy of the position measurement if you enable and specify the gating of the index signal with the **Index signal validation** property.

When you calibrate the position measurement with a gated index, the position is calibrated at a certain fourfold position.

For more information, refer to [Gating the index signal](#) on page 870.

Calibrating from the behavior model You enable the calibration of the position measurement from the behavior model via the **Position update** property.

If enabled, the following function ports for calibration are added to the function block:

Function Port	Description
Update	Triggers the calibration of the position measurement and updates the Position and Position Valid function ports with the values of the Update Position and Update Position Valid function ports.
Update Position	Specifies the position used to calibrate the position measurement. If the Update function port is set to True, the position measurement is calibrated with this position.
Update Position Valid	Specifies the value that the Position Valid function port provides if the Update function port is set to True.

Identifying calibrated position values

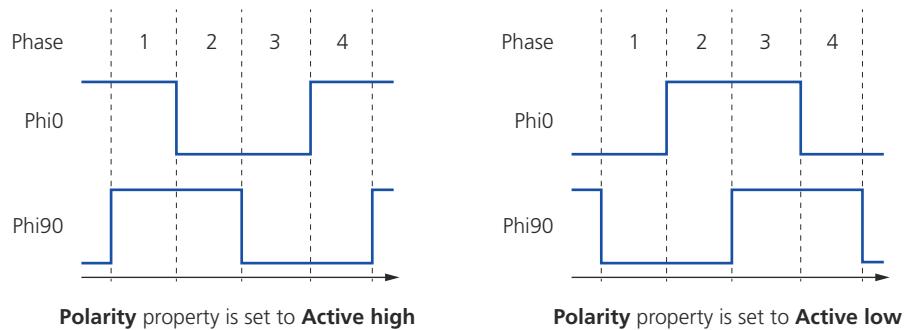
Independent from the used calibration method, you identify calibrated position values with the Position Valid function port. This function port provides the information whether the measured position is valid, i.e., the position is measured with a calibrated position measurement.

If you calibrate the position measurement from the behavior model, you have to validate explicitly that the position measurement is calibrated. This is useful if you have to switch between the calibration methods. For example: You use a measured position in your project only if it is calibrated with the index signal. However, position values that are calibrated from the behavior model are used to detect the index position for the first time. In this use scenario, the position value must be valid only if the index signal is used to calibrate the position measurement. Otherwise, you cannot switch between the calibration from the behavior model and the calibration with the index signal.

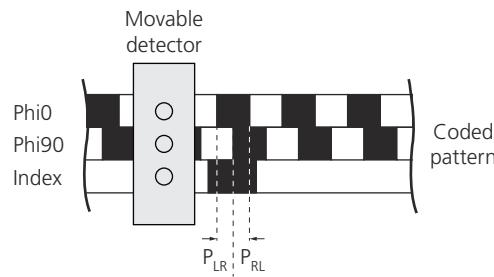
When the position measurement is calibrated with an index signal, the measured position is automatically valid.

Gating the index signal

To specify the gating of the index signal, the **Index signal validation** property provides four phases that represent certain fourfold positions. The illustration below shows you which phase represents which fourfold position.



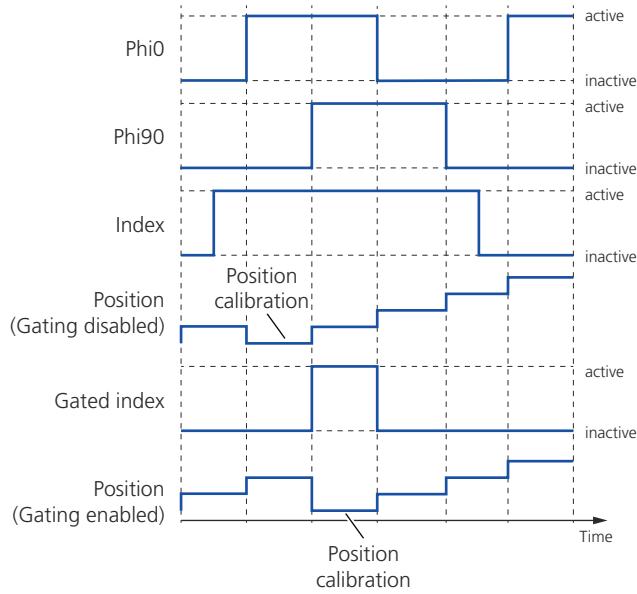
Example of an inconsistent calibration In the example below, the coded pattern provides a wide index. If the detector of the encoder moves from left to right, the position is calibrated at the position P_{LR} . If the detector of the encoder moves from right to left, the position is calibrated at the position P_{RL} .



Calibrating with a gated index The Index signal validation property lets you specify a certain fourfold position to gate the index signal.

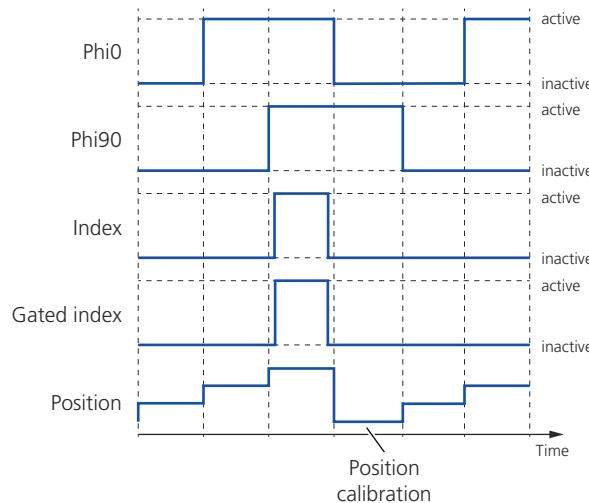
When the index signal becomes active, the position measurement will not be calibrated until the gated index becomes active. The gated index is active only in the specified fourfold position. This calibration of the position measurement is independent from the direction of movement.

In the following example, the index is gated to the fourfold position when the Phi0 signal and the Phi90 signal are active.



Exceptions:

- When the index signal becomes active in the specified fourfold position for index gating, the position is calibrated when the next Phi0 or Phi90 edge is detected. The position will also be calibrated if the index signal becomes inactive before the next edge is reached as shown in the following illustration.



- When the index signal is active only outside the specified fourfold position for index gating, the position measurement will never be calibrated.

Configuring Standard Features (Digital Incremental Encoder In)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO				
Configuration Feature	Digital In/Out 5	Digital In/Out 9	Flexible In/Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ Ground-based ▪ Differential 	Ground-based	Differential	Specifying the Circuit Type for Voltage Input Signals on page 116

MicroAutoBox III		
Configuration Feature	Digital In 4	Further Information
Interface type	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Digital In/Out 5 on page 1586 ▪ Digital In/Out 9 on page 1590 ▪ Flexible In/Out 1 on page 1608 	<ul style="list-style-type: none"> ▪ Digital In 4 on page 1572

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Digital Incremental Encoder In)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (Digital Incremental Encoder In).....	874
MicroAutoBox III Hardware Dependencies (Digital Incremental Encoder In).....	875

SCALEXIO Hardware Dependencies (Digital Incremental Encoder In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	Digital In/Out 5	Digital In/Out 9	Flexible In/Out 1
Hardware	DS6202 Digital I/O Board	DS6121 Multi-I/O Board	
Resolution	<ul style="list-style-type: none"> ▪ 1 ... 4,194,303 lines/revolution (rotary encoder) ▪ 1 ... 4,194,303 lines (linear encoder) 	<ul style="list-style-type: none"> ▪ 1 ... 4,194,303 lines/revolution (rotary encoder) ▪ 1 ... 4,194,303 lines (linear encoder) 	
Supporting master APU functionality ¹⁾	✓	✓	
Supported interface type for input signals	<ul style="list-style-type: none"> ▪ Ground-based ▪ Differential 	Ground-based	Differential
Input voltage range (for ground-based inputs)	0 V ... +30 V	0 V ... +60 V	–
Common mode voltage range for differential inputs ²⁾	-5 V ... +10 V	–	±25 V
Threshold voltage for differential inputs	±300 mV	–	±125 mV (typ.)
Threshold for ground-based signals	Voltage Range	0 V ... +12 V	0 V ... +12 V
	Hysteresis	600 mV (typ.)	600 mV (typ.)
	DAC resolution	8 bit	8 bit
Pulse filter	Minimum pulse duration	8 ns ... 10 ms	0 ... 10 ms
Signal termination		<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (only for differential input signals, parallel termination) ▪ Switchable 	<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (parallel termination) ▪ Switchable

Channel type	Digital In/Out 5	Digital In/Out 9	Flexible In/Out 1
Circuit diagrams	Digital In/Out 5 on page 1586	Digital In/Out 9 on page 1590	Flexible In/Out 1 on page 1608
Required channels	<ul style="list-style-type: none"> ▪ 2 + 1 for optional index measurement (for ground-based signals) ▪ 4 + 2 for optional index measurement (for differential signals) 	2 + 1 for optional index measurement	

¹⁾ The master APU functionality provides the angular position values to be used by other function blocks.

²⁾ Common mode voltage (V_{CM}) = $0.5 \cdot (V_{Signal} + V_{Inverted\ Signal})$ measured to the ground potential of the dSPACE real-time hardware.

General limitations

DS6202 Digital I/O Board The following limitations apply to the DS6202 Digital I/O Board:

- Channel multiplication is not supported in general.
- Differential input signals can be processed only by two subsequent channels, starting with an odd channel number. The odd channel must be connected to the inverted signal.
- Only 6 Digital Incremental Encoder In function blocks can be assigned to one DS6202 Digital I/O Board.

DS6121 Multi-I/O Board Channel multiplication is not supported in general.

More hardware data

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (Digital Incremental Encoder In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	Digital In 4
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board
Resolution	<ul style="list-style-type: none"> ▪ 1 ... 2,097,152 lines/revolution (rotary encoder) ▪ 1 ... 2,097,152 lines (linear encoder)
Supported interface type for input signals	Ground-based

Channel type		Digital In 4
Input voltage range		0 V ... +40 V
Threshold	Input threshold voltage	+2 V (typ.)
	Input hysteresis voltage	1 V (typ.)
Pulse filter	Minimum pulse duration	0 ns ... 12.75 µs
Signal termination		–
Circuit diagrams		Digital In 4 on page 1572
Required channels		2 + 1 for optional index measurement

More hardware data

DS1511 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Sine Encoder In

Where to go from here

Information in this section

Introduction (Sine Encoder In).....	878
Overviews (Sine Encoder In).....	884
Configuring the Function Block (Sine Encoder In).....	890
Hardware Dependencies (Sine Encoder In).....	900

Introduction (Sine Encoder In)

Where to go from here

Information in this section

Introduction to the Function Block (Sine Encoder In).....	878
Basics on Analog Incremental Encoders (Sine Encoder In).....	879
Basics on Measuring the Position and Speed (Sine Encoder In).....	880

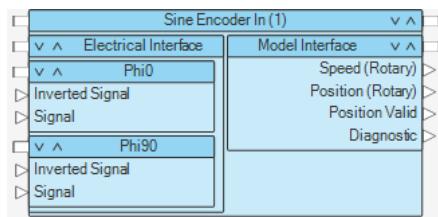
Introduction to the Function Block (Sine Encoder In)

Function block purpose

The Sine Encoder In function block provides access to rotary or linear analog incremental encoders that provide sinusoidal output signals. The function block can be used, for example, to measure the angular position and the speed of an electric motor.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Adjusting the function block to the connected encoder.
- Determining and providing position and speed values to the behavior model.
- High-resolution position calculation to provide exact position values at any time.
- Calibrating the position measurement via index signal of the connected encoder and/or from within the behavior model.
- Enabling the master APU functionality to provide the position values to be used by other function blocks.

Supported channel types

The Sine Encoder In function block type supports the following channel types:

	SCALEXIO		MicroAutoBox III
Channel type	Analog In 6	Flexible In/Out 1	-
Hardware	DS6121		-

Basics on Analog Incremental Encoders (Sine Encoder In)

Introduction

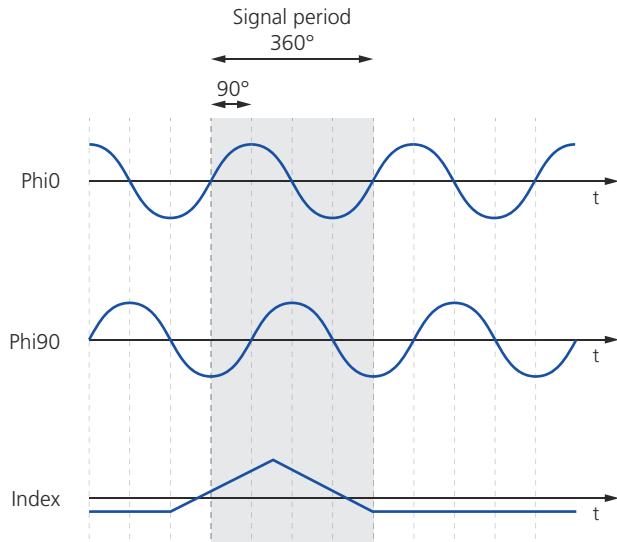
An analog incremental encoder is a sensor that converts the motion of a shaft (rotary encoder) or a linear movement (linear encoder) to sinusoidal signals.

Analog incremental encoders are commonly used in applications that require high resolution position measurement. The high quality of the sinusoidal signals permits high interpolation factors for speed control.

Measurement principle

Typically, two different sensing methods are implemented with analog encoders, either based on optical or inductive sensing. With an optical rotary encoder, the encoder disc modulates a light beam whose intensity is sensed by photo-electrical cells. These produce two 90 degree phase-shifted sinusoidal incremental signals Φ_0 and Φ_{90} .

Many incremental encoders provide an index signal as the third signal. The index signal can be used to calibrate the position measurement.



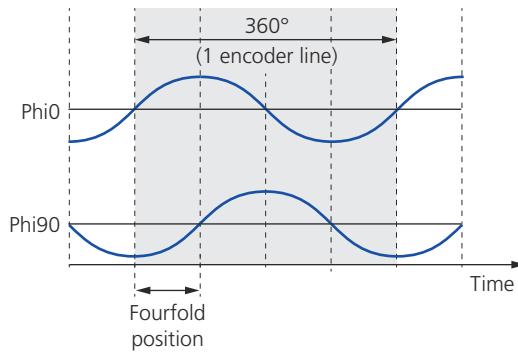
The Φ_0 and Φ_{90} signals are shifted by 90° against each other. This shift can be used to detect the direction of movement.

Rotary encoders generate the index signal periodically with each rotation. Linear encoders generate an index signal at a certain position only.

Measurement precision

The number of encoder lines per revolution (rotary encoder) or encoder lines between the minimum position and the maximum position (linear encoder) determines the resolution of an encoder. The more encoder lines are available, the more precise is the position and speed measurement.

The following illustration shows the shapes of the generated Phi0 and Phi90 analog signals. The gray-shaded area represents one encoder line. An encoder line starts with any zero crossing of the Phi0 or Phi90 signals and finishes four zero crossings later (two zero crossings of the Phi0 signal and two zero crossings of the Phi90 signal).



To describe the 90° phase shift of the Phi90 signal, one encoder line is represented by 360° . One encoder line is represented by four fourfold positions.

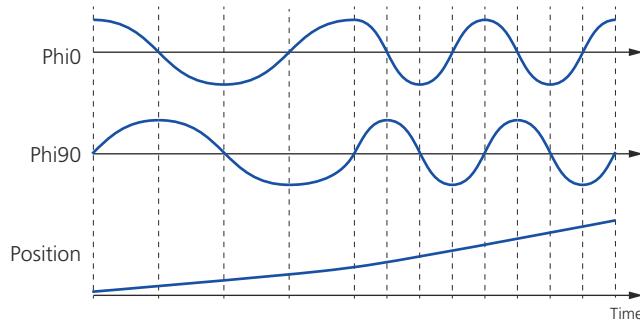
Basics on Measuring the Position and Speed (Sine Encoder In)

Position measurement

The Sine Encoder In function block measures the position using the following methods:

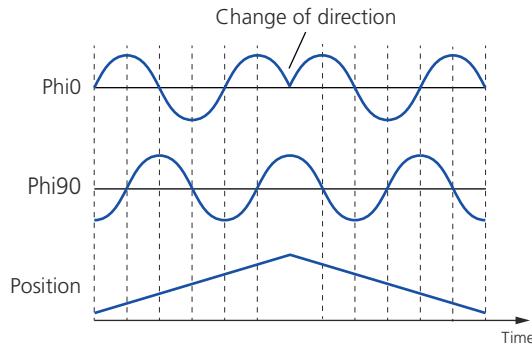
- The function block counts the number of passed fourfold positions (after each detected zero crossing).
- Between two zero crossings, the function block uses the ratio of the value of Phi90 and Phi0 to calculate the position.

The combination of these two methods provides continuous and very exact position values. If the Phi0 signal leads the Phi90 signal, the function block increments the position value as shown in the following illustration.



The function block converts the calculated position value by using the resolution of the connected encoder and outputs the position value in degree (rotary encoder) or meter (linear encoder).

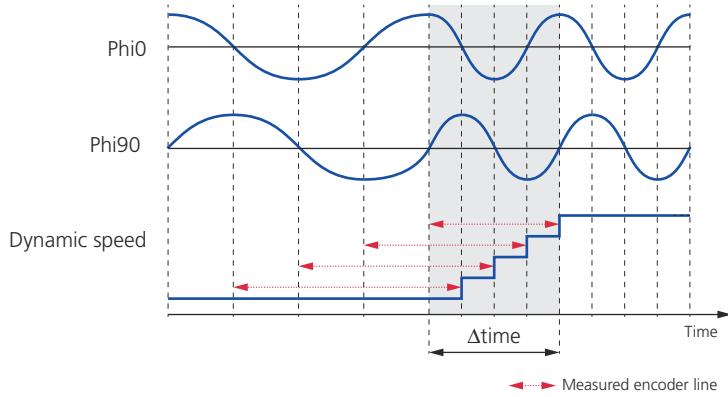
The function block decrements the position value when the Φ_{90} signal follows the Φ_{0} signal. The following illustration shows the position value when a change of direction occurs.



Speed measurement

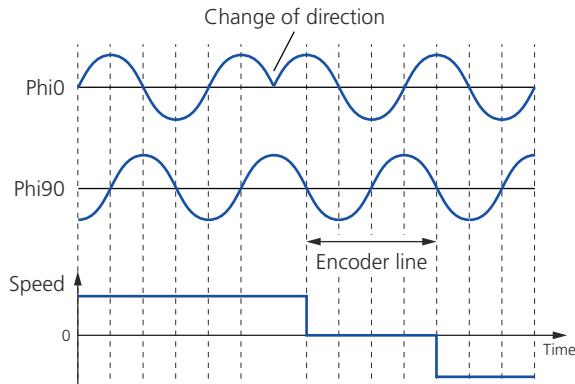
The function block can determine the speed of the connected encoder on the basis of the measured position value. The speed is calculated as the derivative of the position: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode (dynamic or average).

Dynamic speed measurement The dynamic speed is the calculated speed for one encoder line. $\Delta\text{position}$ is the position difference that is covered by one encoder line. Δtime is the difference between the time stamps of the related signal zero crossing. Refer to the following example.



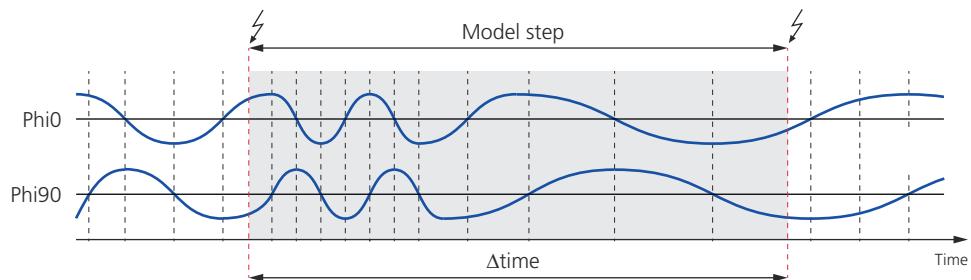
The example above also shows you the run of the speed curve. It takes one encoder line until the measured speed reaches its final value. Keep in mind that an encoder line starts with any zero crossing of the Phi0 or Phi90 signals and finishes four zero crossings later.

When the direction changes, new dynamic speed values are available after the first entire encoder line has been detected. The example below shows an immediate change of direction.



The dynamic measurement mode provides more accurate speed values at rapidly changing rotation speeds.

Average speed measurement The average speed is the measured speed in a model step. $\Delta position$ is the position difference in a model step. $\Delta time$ is the difference between the start and the end of a model step. Refer to the following example.



Note, that in this measurement mode the function block immediately detects a standstill if Δ position = 0. Furthermore, a change of direction cannot be detected.

The average measurement mode provides more accurate speed values at less rapid changes or at constant rotation speeds. For most applications, it is recommended and useful to select this mode.

Overviews (Sine Encoder In)

Where to go from here

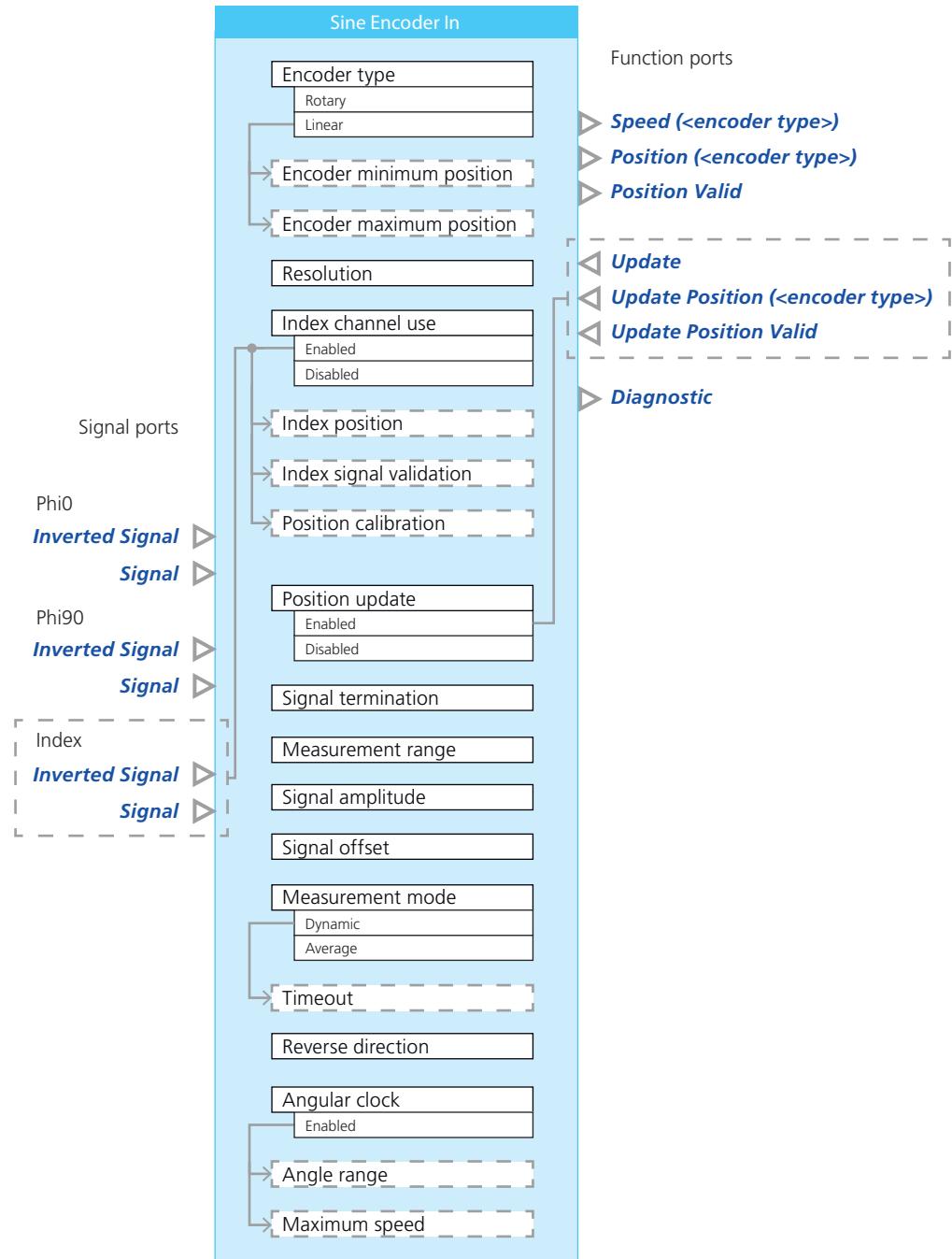
Information in this section

Overview of Ports and Basic Properties (Sine Encoder In).....	884
Overview of Tunable Properties (Sine Encoder In).....	889

Overview of Ports and Basic Properties (Sine Encoder In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Speed (<encoder type>)**

This function outport writes the measured speed of the connected encoder to the behavior model. The sign (+ or -) of the speed value depends on the direction of rotation/movement.

Value range	[Double.min] ... [Double.max] in °/s or m/s (depending on the specified encoder type)
Dependencies	—

Position (<encoder type>)

This function outport writes the measured position of the connected encoder to the behavior model.

Value range	<p>Depends on the specified encoder type:</p> <ul style="list-style-type: none"> ▪ Rotary encoder: 0° ... 360°. At 360° the position counter restarts from 0°. ▪ Linear encoder: [Encoder minimum position] ... [Encoder maximum position] in meter <p>Positive and negative values are possible.</p>
Dependencies	—

Position Valid

This function outport provides the status to the behavior model, indicating whether the measured position value is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid The measured position value is stated as invalid if one the following points apply: <ul style="list-style-type: none"> ▪ The position measurement is not calibrated with an index signal. ▪ The measured position is set to invalid from the behavior model via the Update Position Valid function port. ▪ 1: Valid The measured position value is stated as valid if one the following points apply: <ul style="list-style-type: none"> ▪ The position measurement is calibrated with the index signal of the connected encoder. ▪ The measured position is set to valid from the behavior model via the Update Position Valid function port.
Dependencies	—

Update

This function import triggers the following activities from within the behavior model:

- The calibration (update) of the position measurement with the value of the Update Position function port.
- The update of the Position Valid function outport with the value of the Update Position Valid function import.

Value range	<ul style="list-style-type: none"> ▪ 0: False There is no trigger signal from the behavior model.
-------------	----------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 1: True <p>The calibration of the position measurement and the update of the Position Valid function output is triggered from the behavior model.</p>
Dependencies	Available only if the Position update property is set to Enabled.

Update Position (<encoder type>)

This function import reads the position value for the calibration of the position measurement from the behavior model. The calibration is performed when it is triggered by the Update function port.

Value range	<p>Depends on the specified encoder type:</p> <ul style="list-style-type: none"> ▪ Rotary encoder: <ul style="list-style-type: none"> ▪ -3600° ... +3600° ▪ 0° ... +360° (effective range) ▪ If you enter values outside the effective range, the system converts the entered value to a valid value. Examples: - 10° is converted to 350°. 1430° is also converted to 350°. ▪ Linear encoder: [Encoder minimum position] ... [Encoder maximum position] in meter <p>Positive and negative values are possible.</p>
Dependencies	Available only if the Position update property is set to Enabled.

Update Position Valid

This function import reads the value for the update of the Position Valid function port from the behavior model. The update is performed when it is triggered by the Update function port.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid An update sets the Position Valid function port to Invalid. ▪ 1: Valid An update sets the Position Valid function port to Valid.
Dependencies	Available only if the Position update property is set to Enabled.

Diagnostic

This function output port writes diagnostic information to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: No error ▪ 1: Phi0 amplitude too low <p>The measured value for the Phi0 signal is more than 20% lower than the expected signal amplitude (peak-to-peak)</p>
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>voltage, Vpp) specified with the Signal amplitude property. A possible cause can be a broken or disconnected wire.</p> <ul style="list-style-type: none"> ▪ 2: Phi0 amplitude too high The measured value for the Phi0 signal is more than 20% higher than the expected signal amplitude (peak-to-peak voltage, Vpp) specified with the Signal amplitude property. ▪ 4: Phi90 amplitude too low The measured value for the Phi90 signal is more than 20% lower than the expected signal amplitude (peak-to-peak voltage, Vpp) specified with the Signal amplitude property. A possible cause can be a broken or disconnected wire. ▪ 8: Phi90 amplitude too high The measured value for the Phi90 signal is more than 20% higher than the expected signal amplitude (peak-to-peak voltage, Vpp) specified with the Signal amplitude property. ▪ 16: Mismatch of Phi0 and Phi90 amplitudes The difference between the amplitude of the Phi0 signal and the amplitude of the Phi90 signal exceeds 12,5%. A possible cause can be a fault in the cabling.
Dependencies	–

Inverted Signal

This signal port and the **Signal** signal port together represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (**Signal** and **Inverted Signal**) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Sine Encoder In) on page 900.
Dependencies	–

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Sine Encoder In) on page 900.
Dependencies	–

Overview of Tunable Properties (Sine Encoder In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Timeout ³⁾	✓	–
	Encoder minimum position	✓	–
	Encoder maximum position	✓	–
	Index position	✓	–
	Index signal validation	✓	–
	Position calibration	✓	–
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓
Electrical Interface			
	Signal amplitude	✓	–
	Measurement mode	✓	–
	Signal termination	✓	–
	Signal offset	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ Accessible only if the Measurement mode property is set to Dynamic.

Configuring the Function Block (Sine Encoder In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Sine Encoder In).....	890
Using Calibration for Position Measurement (Sine Encoder In).....	894
Configuring Standard Features (Sine Encoder In).....	899

Configuring the Basic Functionality (Sine Encoder In)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Adjusting the function block to the connected encoder (encoder type, resolution, rotation direction, signal amplitude, signal offset).
- Configuring calibration for the position measurement. Refer to [Using Calibration for Position Measurement \(Sine Encoder In\)](#) on page 894.
- Specifying speed measurement (measurement mode, standstill detection).
- Selecting the measurement range.
- Terminating the input signal.
- Enabling the master APU functionality to provide the position values to be used by other function blocks.

Adjusting the function block to the connected encoder

You have to define some characteristics of the connected encoder to adjust it to the function block to work properly.

Selecting the encoder type You have to select the type of the connected encoder via the Encoder type property. You can measure the position of a rotary or a linear analog incremental encoder with the function block.

- Rotary: Position values are provided in degrees.
- Linear: Position values are provided in meters. For the linear encoder type, you have to specify the Encoder minimum position and the Encoder maximum position.

You can change the minimum and the maximum position via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying the resolution You have to specify the resolution of the connected encoder. The specified value must match the resolution value of the connected encoder. Refer to the data sheet of the encoder. The higher the resolution, the more precise is the position measurement.

- For rotary encoders, you have to enter the number of encoder lines for one revolution.
- For linear encoders, you have to enter the total number of encoder lines that can be measured between the minimum position (Encoder minimum position property) and the maximum position (Encoder maximum position property).

Specifying the signal amplitude You have to specify the expected signal amplitude (peak-to-peak voltage, V_{pp}) of the Phi0 and Phi90 signals provided by the connected encoder. The specified value must match the value of the connected encoder. Refer to the data sheet of the encoder.

The value range depends on the setting of the Measurement range property and on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Sine Encoder In\)](#) on page 900.

The function block checks if the measured amplitudes for the Phi0 and Phi90 signals are in the specified range. If the measured values are outside a fixed tolerance range (-20% ... +20%) from the specified signal amplitude, a diagnostic flag is reported to the behavior model via the Diagnostic function port. For more information, refer to [Overview of Ports and Basic Properties \(Sine Encoder In\)](#) on page 884.

You can change the signal amplitude value via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying a signal offset Most single-ended encoders provide a DC offset voltage for their Phi0 and Phi90 output signals. In this case, you have to specify a signal offset to eliminate the DC voltage component from the incoming signal. The specified value must match the DC offset voltage value of the connected encoder. Refer to the data sheet of the encoder.

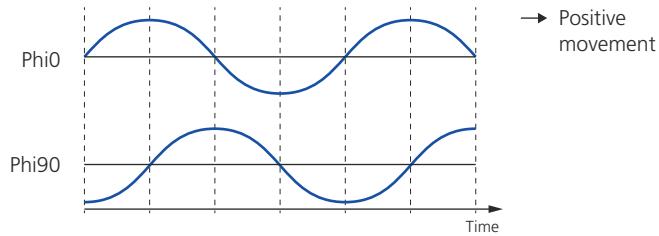
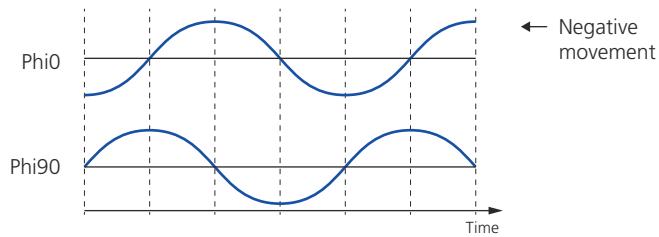
The value range depends on the setting of the Measurement range and Signal amplitude properties and on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Sine Encoder In\)](#) on page 900.

You can change the signal offset value via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Adjusting rotational and movement direction To adjust the rotational direction (rotary encoder) or the movement of a linear encoder to the direction required in the behavior model, you can change the direction by enabling the Reverse direction property. This is useful if the encoder must be mounted in the wrong direction, for example, on the rotor shaft.

If you enable the Reverse direction property, the reverse order is interpreted as a forward rotation. As a result, the speed values provided at the Speed function port are inverted. The values provided at the Position function port move in the reverse direction.

The following illustration shows the Phi0 and Pi90 signals and how the Sine Encoder In function block interprets the phase shift of these signals with respect to the rotation and linear direction.

Rotation direction: Forward**Rotation direction: Reverse**

The measured values are interpreted as a forward rotation if the $\text{Phi}0$ signal leads the $\text{Phi}90$ signal. Refer to the data sheet of the encoder to find out, in which mounting position this applies.

Specifying speed measurement

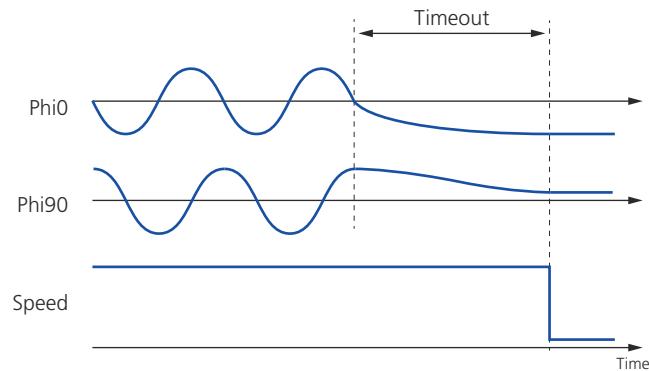
You can measure the dynamic speed or the average speed of the connected analog incremental encoder. The measured values are provided to the behavior model via the Speed function port.

The speed is calculated as follows: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode:

▪ **Dynamic speed**

The dynamic speed is the instantaneous speed value determined on the basis of one electrical signal period. For details, refer to [Basics on Measuring the Position and Speed \(Sine Encoder In\)](#) on page 880.

In this measurement mode the function block detects a standstill if the $\text{Phi}0$ and $\text{Phi}90$ signals have no zero crossings for a time span. Refer to the illustration below.



You have to specify the time span with the **Timeout** property. If standstill is detected, the value of the **Speed** function port is set to 0 immediately. A detected standstill does not affect the position measurement.

You can change the timeout value via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

- Average speed

The average speed is the measured speed in a model step. For details, refer to [Basics on Measuring the Position and Speed \(Sine Encoder In\) on page 880](#).

You can change the measurement mode setting via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application. Note: If you change the setting from average speed to dynamic speed measurement in the experiment software, the timeout value is still not accessible there.

Selecting the measurement range

You have to select the measurement range (low or high) for the **Phi0** and **Phi90** signals. If possible, use the **Low** setting, to achieve highest accuracy for position and speed measurement.

The selected measurement range determines the voltage range for the following properties: **Signal amplitude** and **Signal offset**. For specific values, refer to [Hardware Dependencies \(Sine Encoder In\) on page 900](#).

Note

Valid only for the **Flexible In/Out 1** channel type: All **Sine Encoder In** function blocks which use channels from the **Flexible In/Out 1** channel type from the same I/O board should have the same setting for the **Measurement range** property. The assigned I/O board supports only a common setting for the **Flexible In/Out 1** channel type. In case of different settings, ConfigurationDesk uses the high measurement range as a common default for all affected **Sine Encoder In** function blocks. In addition, a conflict is generated and displayed in the **Conflicts Viewer**.

Terminating the input signal

You can activate signal termination for differential input signals to prevent signal reflections at the line end.

Signal termination is supported only by the **Flexible In/Out 1** channel type. A 120 Ω resistor is connected in series to a 5 nF capacitor between the input lines. For a circuit diagram, refer to [Flexible In/Out 1](#) on page 1608.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Limitation Signal termination is not supported for the **Analog In 16** channel type.

Enabling master APU functionality

You can enable the master APU functionality for the function block to provide the measured angle positions to be used by other function blocks as the angle source, such as the Block-Commutated PWM Out function block.

To do this, you have to assign a master APU to the Sine Encoder In function block, which is located on the same dSPACE board as the I/O channels that are assigned to the function block.

The angular speed of the assigned master APU might exceed the angular speed that can be processed by the slave APU of the hardware resource of the connected function block. Therefore, you can specify a maximum speed (via the Maximum speed property) that fulfills the requirements of your application.

ConfigurationDesk uses the specified maximum speed to check if the hardware resources support the maximum speed.

For more information on the APU functionality, refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Using Calibration for Position Measurement (Sine Encoder In)

Motivation

The calibration of the position measurement is necessary if you want to measure and to provide the absolute position at the Position function port. An incremental encoder provides only relative positions via its Phi0 and Phi90 signals.

Configuring calibration for the position measurement

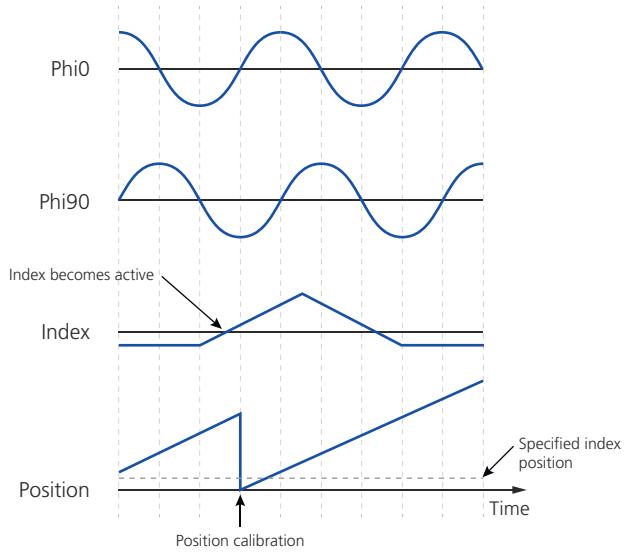
Methods for calibration There are two methods to calibrate the position measurement: You can calibrate the position with the index signal or from the behavior model.

You can use both methods in parallel. For example: Until the index signal with its accurate position is not measured, you can calibrate the signal measurement with a less accurate position from the behavior model.

Calibrating with the index signal The index signal gives an absolute position of the connected encoder that can be used for calibration.

If the index signal becomes active (positive voltage value), the position measurement will be calibrated with the value of the Index position property when the next zero crossing of the Phi0 or Phi90 signal is detected. The position will also be calibrated if the index becomes inactive before the next zero crossing is reached.

The specified index position is always achieved at the center of a fourfold position. Therefore, at a zero crossing of the Phi0 or Phi90 signal, the index position is set to a position value below the specified value (in forward direction). Refer to the following illustration.



You enable and specify the calibration of the position measurement as follows:

- You enable the index signal measurement with the **Index channel use** property.
- You enter the position of the index in the **Index position** property. The entered position is internally rounded to a fourfold position.
- You specify if only the first index is used to calibrate the position of the incremental encoder or every index with the **Position calibration** property.
- You can increase the accuracy of the position measurement if you enable and specify the gating of the index signal with the **Index signal validation** property.

When you calibrate the position measurement with a gated index, the position is calibrated at a certain fourfold position.

For more information, refer to [Gating the index signal](#) on page 896.

Calibrating from the behavior model You enable the calibration of the position measurement from the behavior model via the **Position update** property.

If enabled, the following function ports for calibration are added to the function block:

Function Port	Description
Update	Triggers the calibration of the position measurement and updates the Position and Position Valid function ports with the values of the Update Position and Update Position Valid function ports.
Update Position	Specifies the position used to calibrate the position measurement. If the Update function port is set to True , the position measurement is calibrated with this position.
Update Position Valid	Specifies the value that the Position Valid function port provides if the Update function port is set to True .

Identifying calibrated position values

Independent from the used calibration method, you identify calibrated position values with the Position Valid function port. This function port provides the information whether the measured position is valid, i.e., the position is measured with a calibrated position measurement.

If you calibrate the position measurement from the behavior model, you have to validate explicitly that the position measurement is calibrated. This is useful if you have to switch between the calibration methods.

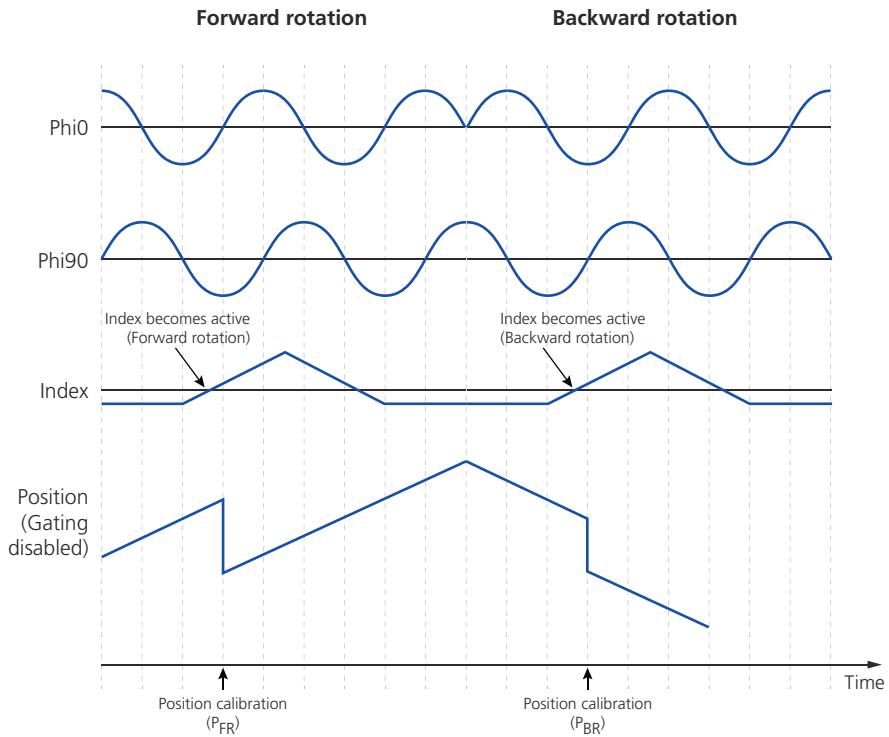
For example: You use a measured position in your project only if it is calibrated with the index signal. However, position values that are calibrated from the behavior model are used to detect the index position for the first time. In this use scenario, the position value must be valid only if the index signal is used to calibrate the position measurement. Otherwise, you cannot switch between the calibration from the behavior model and the calibration with the index signal.

When the position measurement is calibrated with an index signal, the measured position is automatically valid.

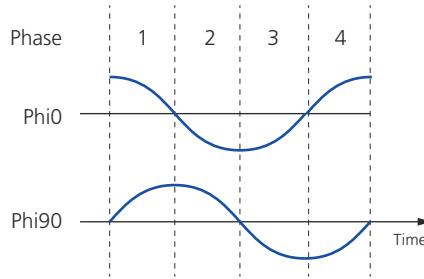
Gating the index signal

The index line is normally wider than a fourfold line. By gating, you can increase the accuracy and prevent an inconsistent calibration of the position measurement.

Example of an inconsistent calibration In the example below, the connected encoder provides a wide index. If the motor rotation is a forward rotation (or a movement from left to right) the position is calibrated at the position P_{FR} . If the rotation is a backwards rotation (or a movement from right to left), the position is calibrated at the position P_{BR} .

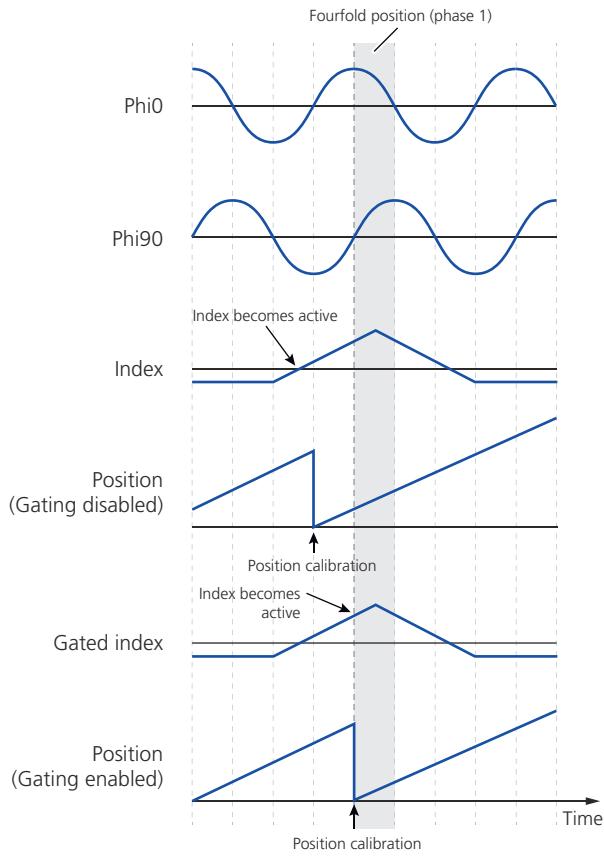


Calibrating with a gated index To specify the gating of the index signal, the Index signal validation property provides four phases that represent certain fourfold positions. The illustration below shows you which phase represents which fourfold position.



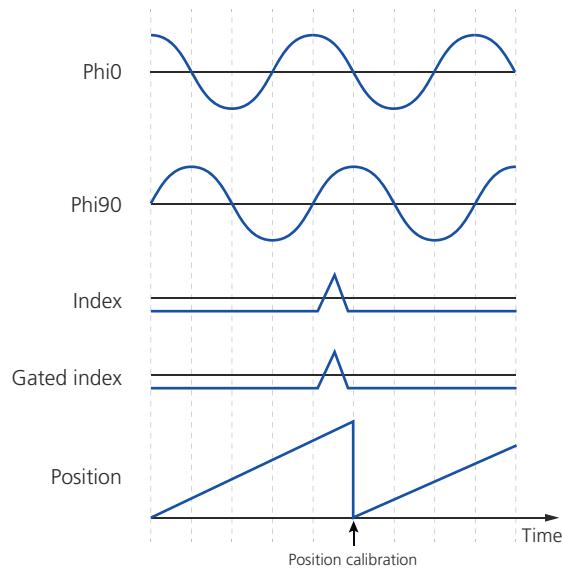
When the index signal becomes active, the position measurement will not be calibrated until the gated index becomes active. The gated index is active only in the specified fourfold position. This calibration of the position measurement is independent from the direction of movement.

In the following example, the index is gated to the fourfold position (phase 1) when the Phi0 and Phi90 amplitudes both are positive.



Exceptions:

- When the index signal becomes active in the specified fourfold position for index gating, the position is calibrated when the next zero crossing of the Phi0 or Phi90 signals is detected. The position will also be calibrated if the index signal becomes inactive before the next zero crossing is reached as shown in the following illustration.



- When the index signal is active only outside the specified fourfold position for index gating, the position measurement will never be calibrated.

Configuring Standard Features (Sine Encoder In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The electrical interface of the function block does not provide configurable standard features.
-----------------------------	-------------------------------------------------------------------------------------------------

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog In 16](#) on page 1552
- [Flexible In/Out 1](#) on page 1608

Model interface	The interface to the behavior model of the function block provides the following standard features.
------------------------	-----------------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Sine Encoder In)

SCALEXIO Hardware Dependencies (Sine Encoder In)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type		Analog In 16	Flexible In/Out 1
Hardware		DS6121 Multi-I/O Board	
Resolution		<ul style="list-style-type: none"> ▪ 1 ... 4,194,303 lines/revolution (rotary encoder) ▪ 1 ... 4,194,303 lines (linear encoder) 	
Supporting master APU functionality ¹⁾		✓, max. 6 each I/O board	
Measurement range	Low ²⁾	±10 V (20 Vpp)	±0.625 V (1,25 Vpp)
	High ²⁾	±30 V (60 Vpp)	±1.25 V (2,5 Vpp)
Signal amplitude ³⁾	Low ²⁾	0 ... 20 Vpp	0 ... 1.25 Vpp
	High ²⁾	0 ... 60 Vpp	0 ... 2.5 Vpp
Signal offset ³⁾	Low ²⁾	-10 V ... +10 V	-0.625 V ... +0.625 V
	High ²⁾	-30 V ... +30 V	-1.25 V ... +1.25 V
Input high voltage (for zero crossing detection) ⁴⁾	Phi0, Phi90 signals	+ 0.1 x Signal amplitude	+ 0.1 x Signal amplitude
	Index signal	0 V	0 V
Input low voltage (for zero crossing detection) ⁴⁾	Phi0, Phi90 signals	- 0.1 x Signal amplitude	- 0.1 x Signal amplitude
	Index signal	- 0.1 x Signal amplitude	- 0.1 x Signal amplitude
Signal termination		–	<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (parallel termination) ▪ Switchable
Supported interface type		Differential with ground sense	Differential
Circuit diagrams		Analog In 16 on page 1552	Flexible In/Out 1 on page 1608
Required channels		2 + 1 for optional index measurement	

¹⁾ The master APU functionality provides the angular position values to be used by other function blocks.

²⁾ Setting of the Measurement range property.

³⁾ The sum of Signal amplitude (in Vp) and Signal offset cannot exceed the value range of the selected measurement range.

⁴⁾ Without signal offset voltage.

More hardware data

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Hall Encoder In

Where to go from here

Information in this section

Introduction (Hall Encoder In).....	902
Overviews (Hall Encoder In).....	906
Configuring the Function Block (Hall Encoder In).....	910
Hardware Dependencies (Hall Encoder In).....	918

Introduction (Hall Encoder In)

Where to go from here

Information in this section

Introduction to the Function Block (Hall Encoder In).....	902
Introduction to Hall Encoders.....	903

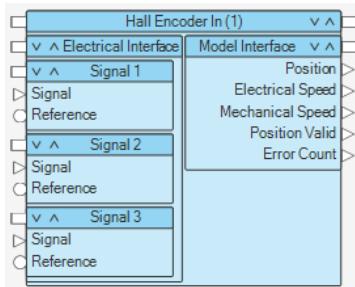
Introduction to the Function Block (Hall Encoder In)

Function block purpose

The Hall Encoder In function block provides access to Hall encoders with differential and single-ended signals. The function block can be used to determine the angular position and the speed of an electric motor.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Specifying the configuration of the Hall encoder (for example, angle position values for signal edges, angular offset, number of pole pairs).
- Determining and providing angular position and speed values to the behavior model.
- Checking the validity of the measured input signals and providing the result to the behavior model.
- Enabling master APU functionality to provide the position values to be used by other function blocks.

Available demo project

ConfigurationDesk provides the *EDrivesControlDemo* project. This project is an example for electric motor control using the DS6121 Multi-I/O Board. You can

use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to [EDrivesControlDemo Project: Example of Electric Motor Control Using the DS6121 Multi-I/O Board \(ConfigurationDesk Demo Projects\)](#).

Supported channel types

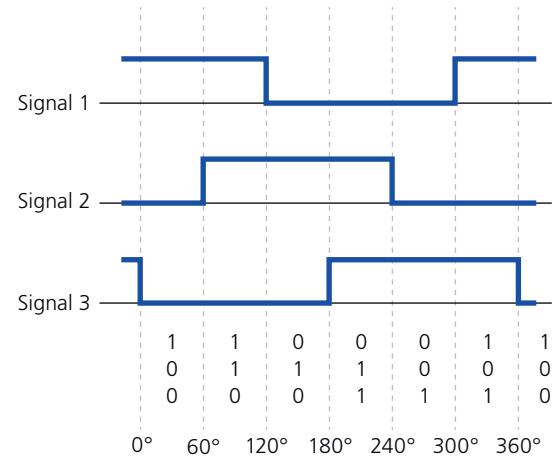
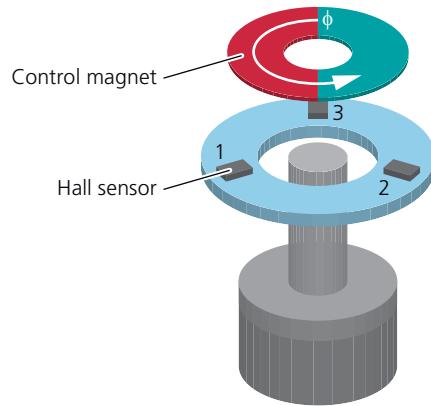
The Hall Encoder In function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	Digital In/Out 9	Flexible In/Out 1
Board	DS6121	–

Introduction to Hall Encoders

Basics on Hall encoders

A Hall encoder provides access to Hall sensors that respond to the magnetic field of the motor's rotor, for example, via a control magnet to determine the motor position.



A Hall encoder usually consists of three Hall sensors, called 1, 2 and 3, that are mounted with an angular distance of 120° between each other. Each Hall sensor converts the energy stored in a magnetic field to an electrical signal by means of the Hall effect. The output of the Hall sensor is an analog signal. A Schmitt trigger is used to convert it to a digital signal. Therefore, the information on the rotor position is provided by rising and falling edges of this digital signal. The combination of the three signals (Signal 1, Signal 2, Signal 3) represents the angular position of the motor.

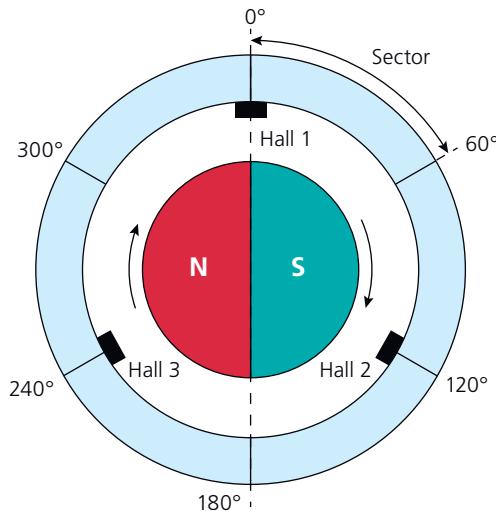
A Hall encoder can detect the position of a motor immediately after power on. However, the precision of a Hall encoder is very low compared to other rotor position sensors.

Example of a typical Hall encoder configuration

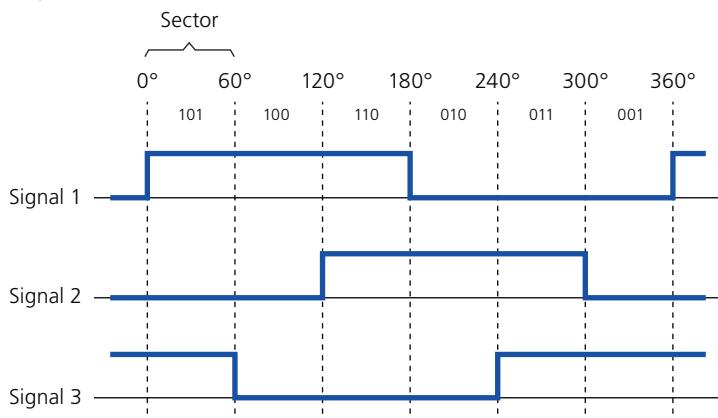
The Hall encoder detects the rotor position by sectors. Sectors are defined for each Hall sensor by specifying the angular position of a sensor's rising edge and falling edge. The following table shows a typical configuration example for three Hall sensors spaced 120° apart.

Sensor	Rising Edge	Falling Edge
Hall 1	0°	180°
Hall 2	120°	300°
Hall 3	240°	60°

The described configuration of the Hall encoder is shown in the following illustration.



Each sector with a size of 60° provides a unique identifier, that corresponds to an angular position of the motor.



An invalid value occurs if the Hall encoder provides a signal pattern that cannot be interpreted as a unique rotor position, for example, if all signals are low as a result of a broken wire.

Position measurement

The Hall encoder provides the electrical position as the mean value of the two angles that limit the particular sector (e.g., 150° for the sector 120° – 180°), as shown in the following table.

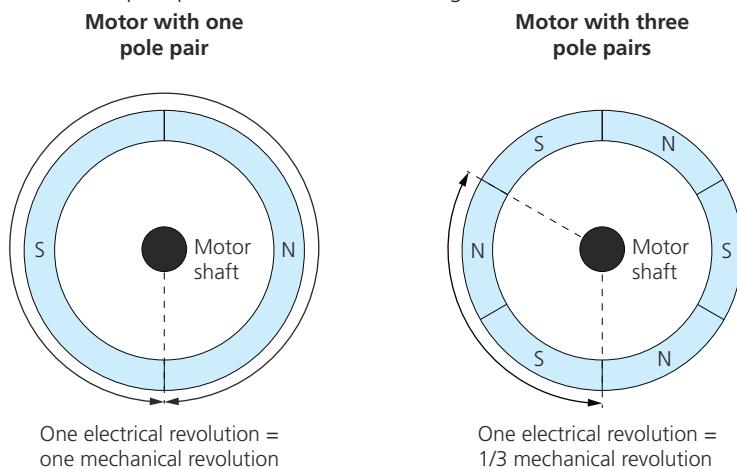
Sector	Provided Position Value
0° - 60°	30°
60° - 120°	90°
120° - 180°	150°
180° - 240°	210°
240° - 300°	270°
300° - 360°	330°

To achieve more accurate position values continuously in the entire sector, you can use position extrapolation. In this case, the position value is extrapolated on the basis of the current rotation speed.

Speed measurement

You can measure the electrical speed of the connected Hall encoder. The speed is calculated as the derivative of the position: $\Delta\text{position} / \Delta\text{time}$.

To determine the mechanical speed, the number of pole pairs of the electric motor or the control magnet is used. Mechanical speed = Electrical speed / Number of pole pairs. Refer to the following illustration.



Overviews (Hall Encoder In)

Where to go from here

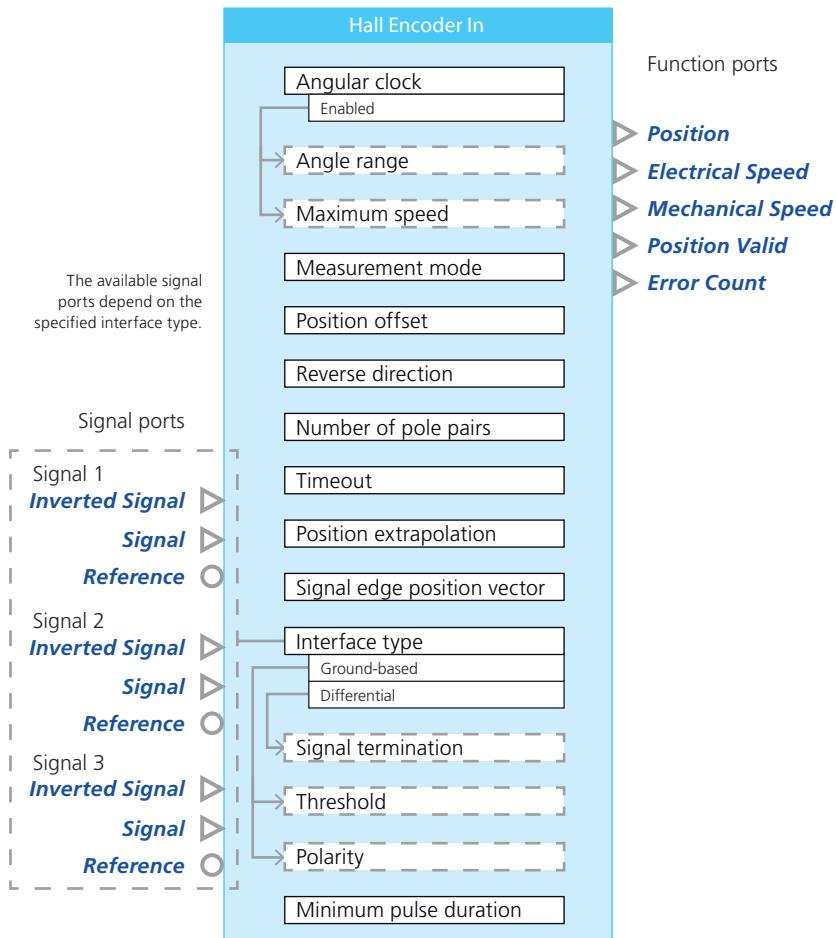
Information in this section

- [Overview of Ports and Basic Properties \(Hall Encoder In\).....](#) 906
- [Overview of Tunable Properties \(Hall Encoder In\).....](#) 909

Overview of Ports and Basic Properties (Hall Encoder In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Position

This function outport writes the measured angular position of the connected Hall encoder to the behavior model.

Value range	0° ... +360°. At 360° the position counter restarts from 0°.
Dependencies	—

Electrical Speed

This function outport writes the electrical speed of the connected Hall encoder to the behavior model. The sign (+ or -) of the speed value depends on the direction of rotation.

Value range	[Double.min] °/s ... [Double.max] °/s
Dependencies	—

Mechanical Speed

This function outport writes the mechanical speed of the connected Hall encoder to the behavior model. The sign (+ or -) of the speed value depends on the direction of rotation.

Value range	[Double.min] °/s ... [Double.max] °/s
Dependencies	—

Position Valid

This function outport provides the status to the behavior model, indicating whether the measured position value is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid The signal pattern of the measured encoder signals (Signal 1, Signal 2 and Signal 3) is not valid, if the function block cannot determine an angle position from the signal pattern. This may be the case, for example, if all three encoder signals are in the high state or if all signals are in the low state. If an invalid value is detected, the position value provided at the Position function port is not updated with the new measured value. Therefore the last valid position value is still provided. ▪ 1: Valid The signal pattern of the measured encoder signals is stated as valid. Therefore the position value provided at the Position function port is updated with the new value.
Dependencies	—

Error Count

This function outport provides the number of errors that occurred in a model step to the behavior model. The error count value is incremented each time a signal edge is detected, which leads to an invalid signal pattern. The value is reset for each model step and saturated at 255.

Value range	0 ... 255
Dependencies	–

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Hall Encoder In) on page 918.
Dependencies	–

Inverted Signal

This signal port and the **Signal** signal port together represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (**Signal** and **Inverted Signal**) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Hall Encoder In) on page 918.
Dependencies	Available only if the Interface type property is set to Differential.

Reference

This signal port is a reference port and provides the reference signal for the **Signal** signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Hall Encoder In) on page 918.
Dependencies	Available only if the Interface type property is set to Ground-based.

Overview of Tunable Properties (Hall Encoder In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Measurement mode	✓	–
Position offset	✓	–
Timeout	✓	–
Threshold	✓	–
Polarity	✓	–
Signal termination	✓	–
Minimum pulse duration	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Hall Encoder In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Hall Encoder In).....	910
Configuring the Standard Features (Hall Encoder In).....	916

Configuring the Basic Functionality (Hall Encoder In)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Specifying the configuration of the Hall encoder (angle position values for signal edges, angular offset, number of pole pairs, rotation direction).
- Enabling extrapolation for determining position values.
- Specifying speed measurement (measurement mode, standstill detection).
- Specifying the trigger threshold, pulse filter, and polarity of the signal measurement.
- Terminating the input signal.
- Enabling master APU functionality to provide the position values to be used by other function blocks.

Specifying the configuration of the Hall encoder

You have to define some characteristics of the geometry of the Hall encoder to get position values that are valid and as accurate as possible.

Specifying angle positions of signal edges You have to specify the angle positions (via the Signal edge position vector property) at which the digital signals of the Hall encoder (Signal 1, Signal 2, Signal 3) change to the high level (= rising edge of the signal). The falling edge angles are automatically calculated from the specified rising edge angle: Falling edge angle = rising edge angle + 180°.

The function block supports any combination of angle positions, for example, angle distances of 60° [0, 60, 120]. However, the following exception applies: The distances between the angle positions for both the rising and falling edges must be at least 1°. Otherwise a conflict is generated and displayed in the Conflicts Viewer.

Specifying number of pole pairs You have to specify the number of pole pairs of the control magnet of the connected Hall encoder. Refer to the data sheet of the encoder. The specified value is used to calculate the output value of

the Mechanical Speed function port as follows: Mechanical speed = Electrical speed/Number of pole pairs. Refer to [Introduction to Hall Encoders](#) on page 903.

Specifying position offset To adjust the measured values of the Hall encoder to the real rotor position, you can specify an offset angle via the Position offset property. Angular errors can be caused by an offset between the rotor shaft and the encoder mounted on the shaft. Positive and negative offset values are possible.

The specified value is added to the measured position value of the Hall encoder and the sum is provided to the behavior model via the Position function port. The following table shows an example with an offset angle of +5°.

Sector	Mean Angle	Offset Angle	Provided Position Value
0° - 60°	30°	+5°	35°
60° - 120°	90°	+5°	95°
120° - 180°	150°	+5°	155°
180° - 240°	210°	+5°	215°
240° - 300°	270°	+5°	275°
300° - 360°	330°	+5°	335°

You can also change the value of the position offset via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

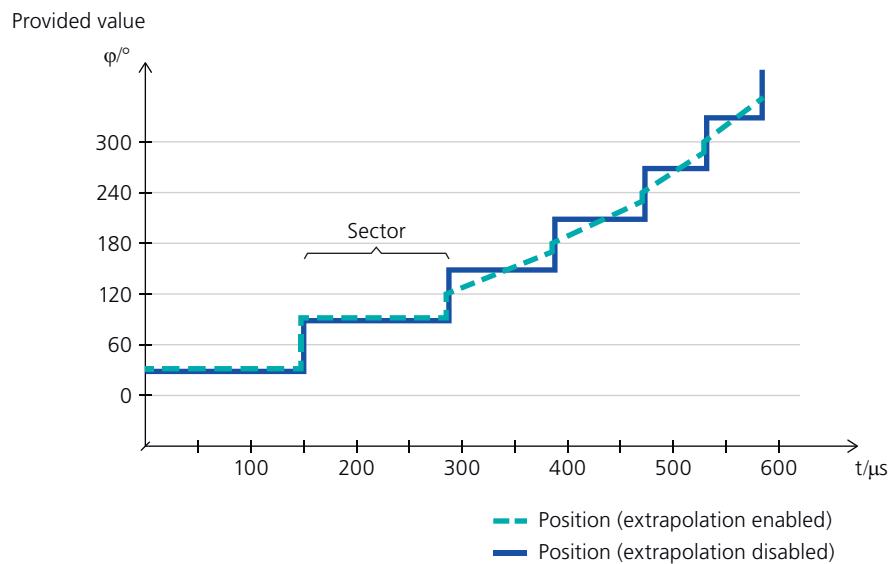
Adjusting rotational direction To adjust the rotational direction to the direction required in the behavior model without changing the cabling, you can change the direction by enabling the Reverse direction property.

If you enable this property, the reverse order of the measured Hall encoder signals (Signal 1, Signal 2, Signal 3) is interpreted as a forward rotation. As a result, the speed values provided at the Electrical Speed and Mechanical Speed function ports are inverted and the value provided at the Position function port moves in the reverse direction.

Enabling extrapolation for determining position values

To achieve more accurate position values continuously in the entire sector you have to enable extrapolation. In this case, the provided position value is extrapolated on the basis of the current rotation speed.

The following illustration shows the principle of extrapolation with an example. An encoder accelerates from standstill. When the encoder starts, there is not enough data for speed measurement. Therefore, the mean value of the two angles that limit the particular sector is provided. Extrapolated speed values are only available after passing the second sector edge.



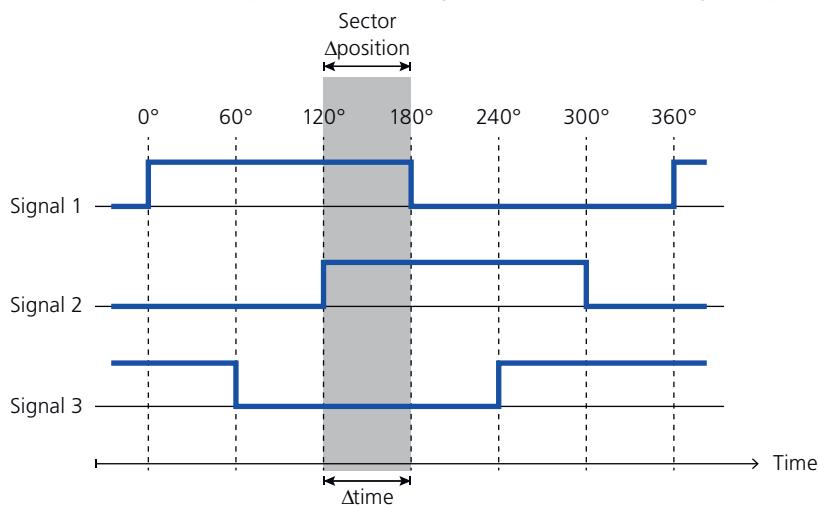
The extrapolated position does not exceed the end position of the current sector. The moment a signal edge occurs, the position is immediately updated to the sector start position (or the sector end position for reverse rotation).

If position extrapolation is disabled, the provided position value is the mean of the two angles that limit the particular sector (e.g., 150° for the sector 120° – 180°). This also applies if a standstill is detected when extrapolation is enabled.

Specifying speed measurement

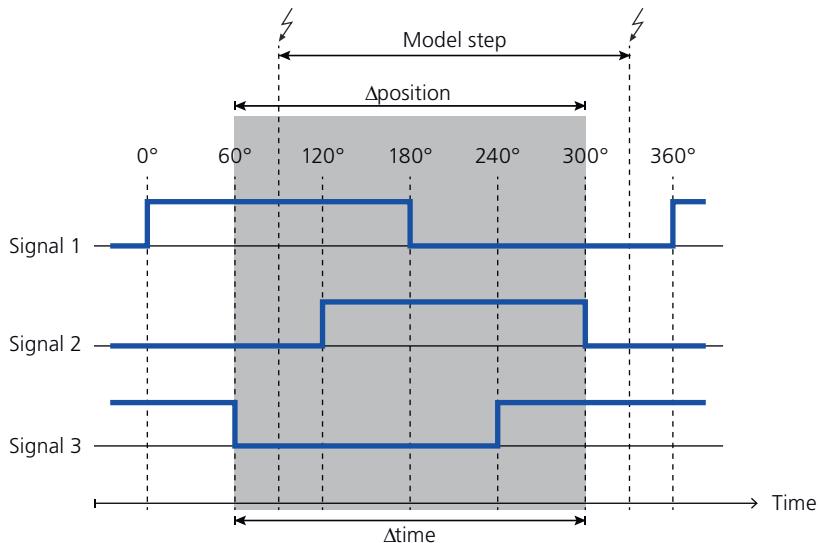
You can measure the dynamic speed or the average speed of the connected Hall encoder. The speed is calculated as follows: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode (dynamic or average).

Dynamic speed measurement The dynamic speed is the calculated speed in a sector. $\Delta\text{position}$ is the angle difference of the sector. Δtime is the difference between the time stamps of the sector edges. Refer to the following example.



The dynamic measurement mode provides more accurate speed values at rapidly changing rotation speeds.

Average speed measurement The average speed is the calculated speed in a model step. $\Delta time$ is the difference of the time stamps of the sector edges that occur just before the start of a model step and just before the end of a model step. $\Delta position$ is the angle difference of the related sector edges. Refer to the following example.



If no sector edge occurs in a model step, the function block outputs the last calculated average speed values.

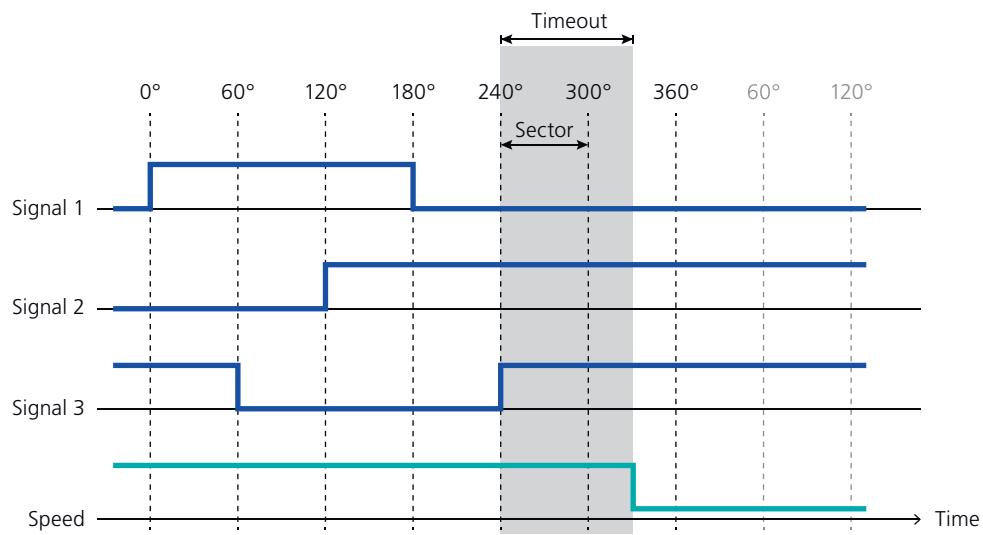
To get valid speed values in the average measurement mode, the following conditions must apply:

- The duration of a model step must not exceed 10 ms.
- The position difference in a model step must not exceed $\pm 1800^\circ$ ($= \pm 5$ encoder revolutions).

The average measurement mode provides more accurate speed values at less rapid changes or at constant rotation speeds. For most applications, it is recommended and useful to select this mode.

You can also change the speed measurement mode via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Detecting standstill The Hall Encoder In function block detects a standstill if the measured sector does not change within the timeout that you specified with the Timeout property.



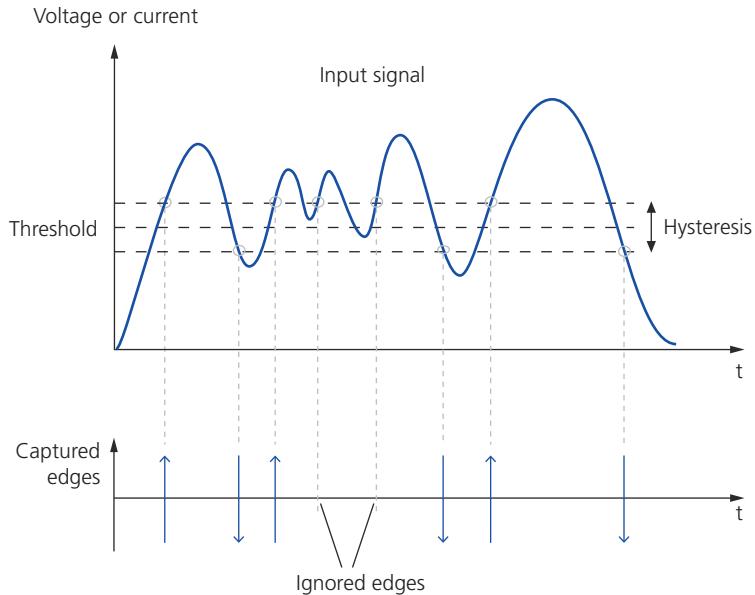
If a standstill is detected, the following applies:

- The values of the Electrical Speed and Mechanical Speed function ports are immediately set to 0 %/s .
- The mean value of the two positions that limit the current sector (e.g. 270° for the sector 240° - 300°, see illustration above) is provided to the Position function port as position value. This also applies if position extrapolation is enabled.

You can change the timeout value via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying the trigger threshold

The adjustable threshold value lets you compensate for any voltage/current offsets on the input signal. You should set the threshold value so as to capture all relevant edges of the waveform. The hysteresis value is fixed.



Edges of the input signal can be ignored for three reasons:

- The signal does not pass through the threshold
- The signal does not reach the threshold plus half the hysteresis
- The signal does not reach the threshold minus half the hysteresis

The value range of the threshold and the hysteresis constant depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Hall Encoder In\) on page 918](#).

Specifying polarity of the signal

To adapt the real-time hardware to signals of different polarity without the need to use inverters on the hardware side, you can switch the **Polarity** of the input signal.

- Active high

The rising edges of the input signals are used for the sectors angle start positions (specified at the **Signal edge position vector** property).

- Active low

The falling edges of the input signals are used for the sectors angle start positions (specified at the **Signal edge position vector** property).

Note

Switching the polarity is only supported if you use the ground-based interface type.

Filtering the input signal

You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the **Minimum pulse duration** property are ignored.

The value range of the pulse duration depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Hall Encoder In\)](#) on page 918.

You can also change the value of the **Minimum pulse duration** property via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Terminating the input signal

You can activate signal termination for differential input signals to prevent signal reflections at the line end.

Note

You cannot activate a signal termination for ground-based input signals.

Therefore, signal termination is supported only by the **Flexible In/Out 1** channel type. A $120\ \Omega$ resistor is connected in series to a 5 nF capacitor between the input lines. For a circuit diagram, refer to [Flexible In/Out 1](#) on page 1608.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Enabling master APU functionality

You can enable the master APU functionality for the function block to provide the measured angle positions to be used by other function blocks as the angle source, such as the **Block-Commutated PWM Out** function block.

To do this, you have to assign a master APU to the **Hall Encoder In** function block, which is located on the same dSPACE board as the I/O channels that are assigned to the function block.

The angular speed of the assigned master APU might exceed the angular speed that can be processed by the slave APU of the hardware resource of the connected function block. Therefore, you can specify a maximum speed (via the **Maximum speed** property) that fulfills the requirements of your application.

ConfigurationDesk uses the specified maximum speed to check if the hardware resources support the maximum speed.

For more information on the APU functionality, refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Configuring the Standard Features (Hall Encoder In)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital In/Out 9	Flexible In/Out 1	Further Information
Interface type	Ground-based	Differential	Specifying the Circuit Type for Voltage Input Signals on page 116

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital In/Out 9 on page 1590](#)
- [Flexible In/Out 1 on page 1608](#)

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Hall Encoder In)

SCALEXIO Hardware Dependencies (Hall Encoder In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Characteristics		Digital In/Out 9	Flexible In/Out 1
Hardware		DS6121 Multi-I/O Board	
Input voltage range		0 ... +60 V	±25 V
Supporting master APU functionality ¹⁾		✓	
Supported interface type		Ground-based	Differential
Threshold	Input threshold voltage	0 ... +12 V	±125 mV (typ.)
	Input hysteresis voltage	600 mV (typ.)	250 mV (typ.)
	DAC resolution	8 bit	—
Pulse filter	Minimum pulse duration	0 ... 10 ms	0 ... 10 ms
Signal termination		—	<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (parallel termination) ▪ Switchable
Circuit diagrams		Digital In/Out 9 on page 1590	Flexible In/Out 1 on page 1608
Required channels		3	3

¹⁾ The master APU functionality provides the angular position values to be used by other function blocks.

More hardware data

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Resolver In

Where to go from here

Information in this section

Introduction (Resolver In).....	920
Overviews (Resolver In).....	924
Configuring the Function Block (Resolver In).....	928
Hardware Dependencies (Resolver In).....	934

Introduction (Resolver In)

Where to go from here

Information in this section

Introduction to the Function Block (Resolver In).....	920
Introduction to Resolvers.....	921

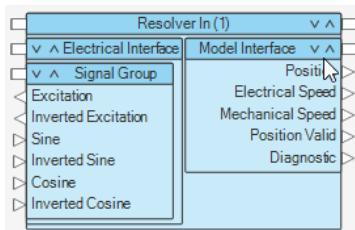
Introduction to the Function Block (Resolver In)

Function block purpose

The Resolver In function block type provides access to resolvers to determine the angular position and the speed, for example, of an electric motor.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Adjusting the function block to the connected resolver.
- Determining and providing angular position and speed values to the behavior model.
- Providing diagnostic information to the behavior model.
- Enabling master APU functionality to provide the position values to be used by other function blocks.

Supported resolver

The Resolver In function block supports the use of resolvers with the following interface characteristics:

Excitation Voltage	Transformation Ratio	Resulting Induction (Sine, Cosine) Voltage Setting	Excitation Frequency	Maximum Speed
3 V _{RMS}	0.37 ... 0.63	1.5 V _{RMS}	2 ... 20 kHz, adjustable in steps of 250 Hz	150,000 rpm
	0.87 ... 1.46	3.5 V _{RMS}		
	1.24 ... 2.1	5 V _{RMS}		
7 V _{RMS}	0.16 ... 0.27	1.5 V _{RMS}		
	0.38 ... 0.62	3.5 V _{RMS}		
	0.53 ... 0.9	5 V _{RMS}		
10 V _{RMS}	0.11 ... 0.19	1.5 V _{RMS}		
	0.26 ... 0.44	3.5 V _{RMS}		
	0.37 ... 0.63	5 V _{RMS}		

Supported channel types

The Resolver In function block supports the following channel types:

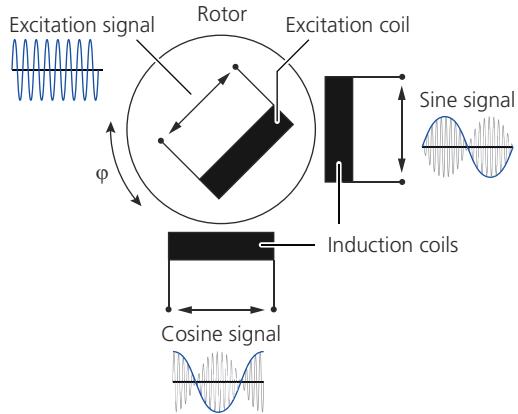
	SCALEXIO	MicroAutoBox III
Channel type	Resolver In 2	–
Board	DS6121	–

Introduction to Resolvers

Basics on resolvers

Resolvers are absolute encoders that measure angular positions, for example, of electric motors. They can detect the angle position with high precision, directly after power on.

Resolvers are generally mounted on the rotor shaft of a motor. Their construction is similar to that of an electric motor. They operate as variable coupling transformers in which the magnetic force of the coupling between an excitation coil and two induction coils changes with the position of the rotor shaft.

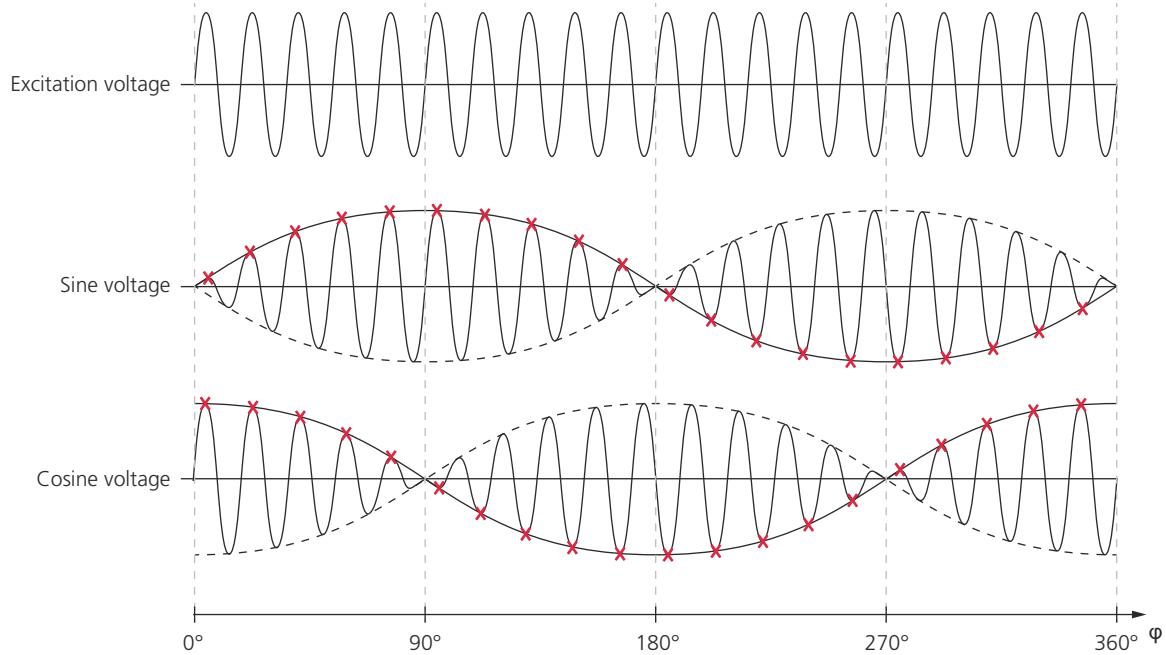


The excitation coil is excited with a sinusoidal signal. This signal causes position-dependent induction signals in the induction coils. The induction coils are arranged in a right angle to one another, so that the envelope of the resulting signals are a sine signal and a cosine signal.

Position measurement

The dSPACE hardware manages the generation of the excitation signal and it determines the position angle values from the induced signals (sine and cosine).

The following illustration shows the shape of the excitation signal and the resulting sine and cosine induction signals for one revolution. An exact position value can be always determined.



Speed measurement

You can measure the electrical speed of the rotor shaft of the connected resolver. The speed is calculated as the derivative of the position: $\Delta \text{position} / \Delta \text{time}$.

The mechanical speed is calculated from the electrical speed as follows:
Mechanical speed = Electrical speed / Number of pole pairs.

Overviews (Resolver In)

Where to go from here

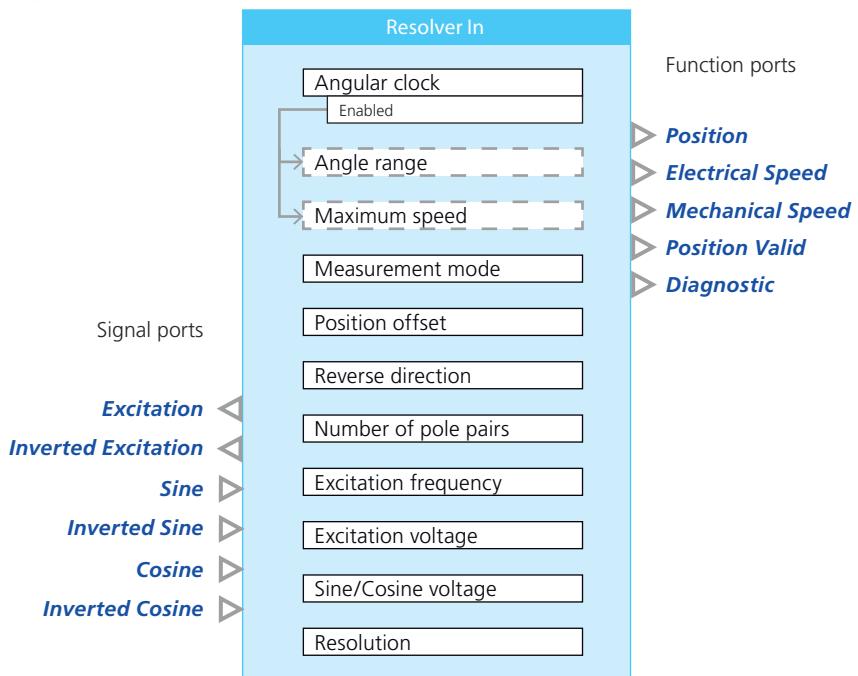
Information in this section

Overview of Ports and Basic Properties (Resolver In).....	924
Overview of Tunable Properties (Resolver In).....	927

Overview of Ports and Basic Properties (Resolver In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Position

This function outport writes the measured angular position of the rotor shaft to the behavior model.

Value range	0° ... +360°. At 360° the position counter restarts from 0°.
Dependencies	–

Electrical Speed

This function outport writes the electrical speed of the rotor shaft to the behavior model. The sign (+ or -) of the speed value depends on the direction of rotation.

Value range	[Double.min] °/s ... [Double.max] °/s
Dependencies	–

Mechanical Speed

This function outport writes the mechanical speed of the rotor shaft to the behavior model. The sign (+ or -) of the speed value depends on the direction of rotation.

Value range	[Double.min] °/s ... [Double.max] °/s
Dependencies	–

Position Valid

This function outport provides the status to the behavior model, indicating whether the measured position value is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid No valid position has been read since the start of the real-time application. ▪ 1: Valid The measured position value is stated as valid. Therefore the position value provided at the Position function port is updated with the new value.
Dependencies	–

Diagnostic

This function outport writes diagnostic information to the behavior model, for example, if an error occurs.

Value range	<ul style="list-style-type: none"> ▪ 1: Phase lock error The difference between the phase of the excitation signal and the phase of the sine and cosine signals exceeds the phase lock range of the dSPACE hardware. The max. value on the DS6121 is 45°. A phase shift can occur, for example, if the connecting cables for the different signals are of different lengths. ▪ 2: Speed too high The current speed of the resolver's rotor shaft is higher than the function block is able to track. The maximum speed depends on the setting of the Resolution property. Specify a lower resolution, to increase the maximum speed. Refer to
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

[Configuring the Basic Functionality \(Resolver In\)](#) on page 928.

- 4: Loss of tracking

The function block cannot determine the position. The difference between the measured position and the internally calculated reference position exceeds the tolerance value.

The tolerance value depends on the specified measurement resolution as follows:

- 10 bit -> 12.5° max. tolerance
- 12 bit -> 5.0° max. tolerance
- 14 bit -> 2.5° max. tolerance
- 16 bit -> 2.5° max. tolerance

If the selected measurement mode allows for a low resolution, you can specify a lower value to increase the tolerance level. Refer to [Configuring the Basic Functionality \(Resolver In\)](#) on page 928.

- 8: Input signal too low

The measured values for the sine and/or cosine input signals are below the expected voltage level specified at the Sine/Cosine voltage property (minus a tolerance value of 30 %). Possible cause can be a broken or disconnected wire.

- 16: Input signal too high

The measured values for the sine and/or cosine input signals exceed the expected voltage level specified at the Sine/Cosine voltage property (plus a tolerance value of 30 %).

Dependencies	-
--------------	---

Excitation / Inverted Excitation

These signal ports represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential output. Both signal ports (Excitation and Inverted Excitation) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Resolver In) on page 934.
Dependencies	-

Sine / Inverted Sine

These signal ports represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (Sine and Inverted Sine) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Resolver In) on page 934.
Dependencies	–

Cosine / Inverted Cosine

These signal ports represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input. Both signal ports (Sine and Inverted Sine) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Resolver In) on page 934.
Dependencies	–

Overview of Tunable Properties (Resolver In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Measurement mode	✓	–
Position offset	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
–	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Resolver In)

Where to go from here

Information in this section

Configuring the Basic Functionality (Resolver In).....	928
Configuring the Standard Features (Resolver In).....	933

Configuring the Basic Functionality (Resolver In)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Adjusting the function block to the connected resolver (excitation voltage and frequency, angular offset, number of pole pairs, rotation direction).
- Selecting the resolution of the position measurement.
- Specifying the speed measurement mode.
- Enabling master APU functionality to provide the position values to be used by other function blocks.

Adjusting the function block to the connected resolver

You have to define some characteristics of the connected resolver to adjust it to the function block to work properly.

Selecting excitation voltage You have to select the RMS output voltage (3 Vrms, 5 Vrms or 10 Vrms) required for the excitation coil of the connected resolver. Refer to the data sheet of the resolver.

Specifying excitation frequency You have to specify the frequency (2 kHz ... 20 kHz) for the excitation voltage required by the connected resolver. Refer to the data sheet of the resolver.

The excitation frequency has a resolution of 250 Hz. The specified value is rounded to the nearest valid frequency. The actual frequency generated by the hardware is displayed via the Calculated excitation frequency property.

Selecting expected sine and cosine input voltage From the excitation voltage and the transformation ratio of the resolver, you can calculate the expected input voltage that is induced at the sine and cosine coils of the connected resolver: $V_{Sine/Cosine} = V_{Excitation} \times \text{transformation ratio}$. From this value select the input voltage (1.5 Vrms, 3.5 Vrms, 5.0 Vrms) that matches best.

A typical transformation ratio is 0.5. Refer to the data sheet of the resolver.

Example: An excitation voltage of 10 Vrms produces a 5.0 Vrms sine/cosine voltage at a transformation ratio of 0.5. For supported transformation ratios and

the resulting induction voltages, refer to [Introduction to the Function Block \(Resolver In\)](#) on page 920.

The function block checks if the voltage level of the signals are in a valid range. If not, related status information is written to the behavior model via the Diagnostic function port.

Specifying position offset To adjust the measured values of the resolver to the real rotor position, you can specify an offset angle via the Position offset property. Angular errors can be caused by an offset between the rotor shaft and the encoder mounted on the shaft. Positive and negative offset values are possible.

The specified value is added to the measured position value of the resolver and the sum is provided to the behavior model via the Position function port.

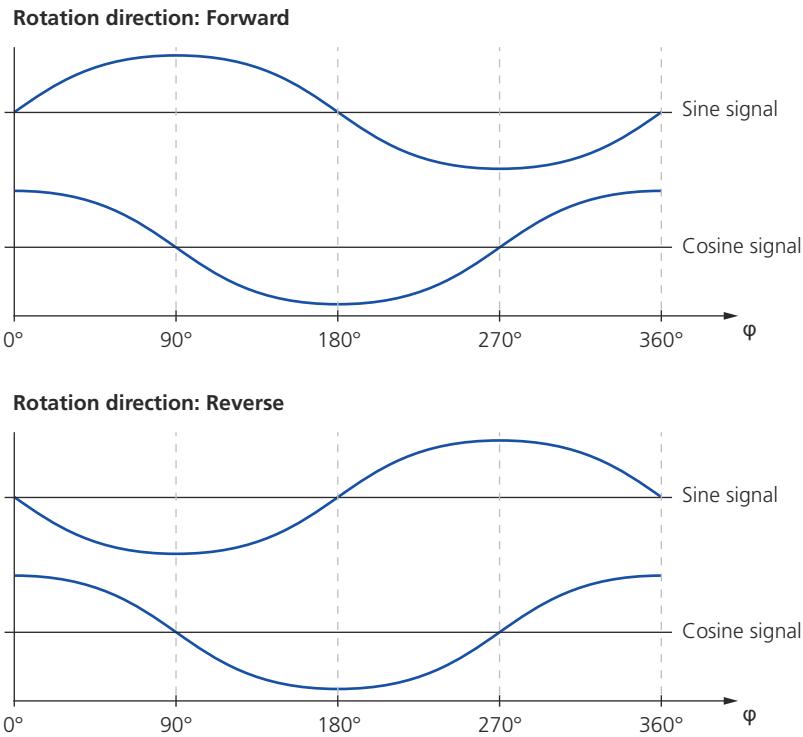
You can also change the value of the position offset via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying number of pole pairs You have to specify the number of pole pairs of the connected resolver. Refer to the data sheet of the connected resolver. The specified value is used to calculate the output value of the Mechanical Speed function port as follows: Mechanical speed = Electrical speed/Number of pole pairs.

Adjusting rotational direction To adjust the rotational direction to the direction required in the behavior model, you can change the direction by enabling the Reverse direction property. This is useful, for example, if the resolver must be mounted in the wrong direction on the rotor shaft.

If you enable the Reverse direction property, the reverse order is interpreted as a forward rotation. As a result, the speed values provided at the Electrical Speed and Mechanical Speed function ports are inverted and the value provided at the Position function port moves in the reverse direction.

The following illustration shows the envelope signal curves of the sine and cosine signals and how the Resolver In function block interprets these curves with respect to the rotation direction.



Refer to the data sheet of the resolver to find out, which rotation direction is supported in which mounting position.

Selecting the resolution of the position measurement

The maximum speed that the function block can process depends on the resolution of the position measurement. Therefore, you have to select the resolution to adjust the function block to the expected maximum speed of the resolver's rotor shaft via the Resolution property. If you increase the resolution, the maximum speed decreases as follows:

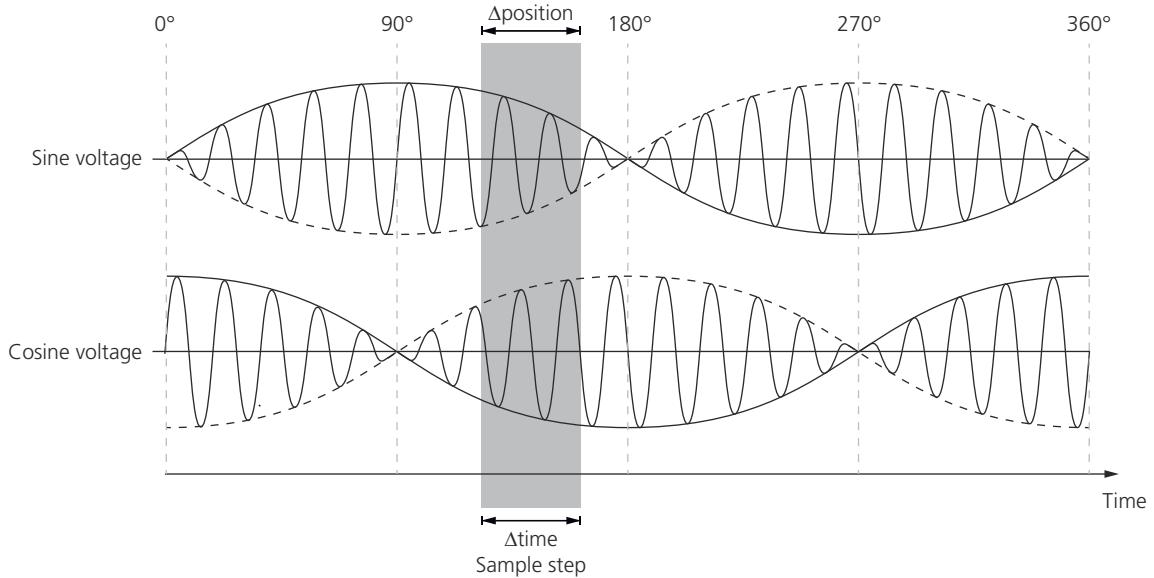
Resolution Setting	Maximum Speed		Recommended Excitation Frequency Range
10 bit	900,000 %/s	150,000 rpm	10 kHz ... 20 kHz
12 bit	360,000 %/s	60,000 rpm	6 kHz ... 20 kHz
14 bit	180,000 %/s	30,000 rpm	3 kHz ... 12 kHz
16 bit	45,000 %/s	7,500 rpm	2 kHz ... 10 kHz

The resolution setting also influences the accuracy of the measured angular position. To achieve the best accuracy, select the resolution under consideration of the recommended excitation frequency range for the respective setting. Refer to the table above.

Specifying speed measurement

You can measure the dynamic speed or the average speed of the connected resolver. The speed is calculated as follows: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode (dynamic or average).

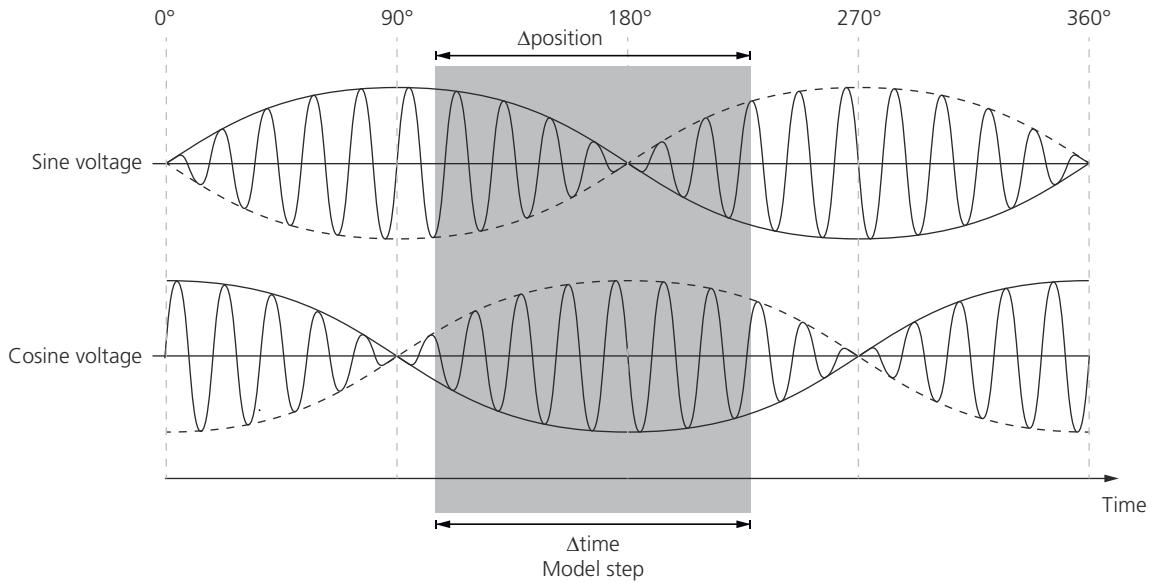
Dynamic speed measurement The dynamic speed is determined from a fixed sample rate provided by the dSPACE hardware. The $\Delta\text{position}$ is the angle difference in a sample step, Δtime is the difference between the start and the end of a sample step.



Use the dynamic speed mode only if you select a high resolution (14 bit or 16 bit) for the position measurement. Otherwise, the calculated speed values are very inaccurate.

The dynamic measurement mode provides more accurate speed values at rapidly changing rotation speeds.

Average speed measurement The average speed is the calculated speed in a model step. $\Delta\text{position}$ is the angle difference in a model step, Δtime is the difference between the start and the end of a model step. Refer to the following example.



Use the average speed mode if you have to reduce the effect of noise on input signals or if you want to obtain speed values with higher accuracy.

To get valid speed values in this mode, the following conditions must apply:

- The duration of a model step must not exceed 10 ms.
- The position difference in a model step must not exceed +/- 1800° (= +/- 5 encoder revolutions).

The average measurement mode provides more accurate speed values at less rapid changes or at constant rotation speeds. For most applications, it is recommended and useful to select this mode.

You can also change the speed measurement mode via the experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Enabling master APU functionality

You can enable the master APU functionality for the function block to provide the measured angle positions to be used by other function blocks as the angle source, such as the Block-Commutated PWM Out function block.

To do this, you have to assign a master APU to the Resolver In function block, which is located on the same dSPACE board as the I/O channels that are assigned to the function block.

The angular speed of the assigned master APU might exceed the angular speed that can be processed by the slave APU of the hardware resource of the connected function block. Therefore, you can specify a maximum speed (via the Maximum speed property) that fulfills the requirements of your application.

ConfigurationDesk uses the specified maximum speed to check if the hardware resources support the maximum speed.

For more information on the APU functionality, refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Configuring the Standard Features (Resolver In)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The electrical interface of the Resolver In function block does not provide configurable standard features.

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Resolver In 2](#) on page 1621

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Resolver In)

SCALEXIO Hardware Dependencies (Resolver In)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type.

Characteristics	Resolver In 2
Hardware	DS6121 Multi-I/O Board
Input voltage range	0 V ... +6.3 V _{RMS}
Output voltage range	0 V ... +10 V _{RMS}
Output current	0 V ... +160 mA _{RMS}
Output frequency	2 ... 20 kHz
Supported interface type	Differential
Resolution (of position values)	10, 12, 14, 16 bit
Supporting master APU functionality ¹⁾	✓
Circuit diagrams	Resolver In 2 on page 1621
Required channels	1

¹⁾ The master APU functionality provides the angular position values to be used by other function blocks.

General limitations

Only one Resolver In function block can be used with one DS6121 Multi-I/O Board.

More hardware data

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

EnDat Master

Where to go from here

Information in this section

Introduction (EnDat Master).....	936
Overviews (EnDat Master).....	939
Configuring the Function Block (EnDat Master).....	944
Hardware Dependencies (EnDat Master).....	950

Introduction (EnDat Master)

Where to go from here

Information in this section

Introduction to the Function Block (EnDat Master).....	936
Introduction to the EnDat Interface.....	937

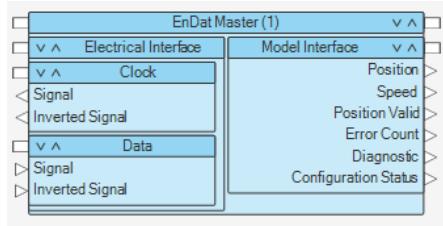
Introduction to the Function Block (EnDat Master)

Function block purpose

The EnDat Master function block provides access to absolute encoders (single-turn and multi-turn encoders) via the EnDat 2.1 or EnDat 2.2 protocol. The function block can be used to determine the angular position, the revolution count, and the speed, for example, of an electric motor.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Adjusting the function block to the connected absolute encoder (single-turn or multi-turn encoder).
- Supporting the EnDat 2.1 and EnDat 2.2 protocol.
- Determining and providing angular positions, revolution counts, and speed values to the behavior model.
- Checking the validity of the received position values and providing the result to the behavior model.
- Evaluation of received diagnostic information from the encoder (such as configuration and transmission errors) and providing the result to the behavior model.
- Executing data transmission continuously or triggered by an I/O event, for example, from another function block.

- Enabling the master APU functionality to provide the position values to be used by other function blocks.

Supported channel types

The EnDat Master function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	Flexible In/Out 1	-
Board	DS6121	-

Introduction to the EnDat Interface

Basics on the EnDat interface

The EnDat (Encoder Data) interface is a digital, bidirectional interface developed by Heidenhain GmbH in Germany. It provides access to encoders that determine the mechanical rotor position.

The interface is realized by a point-to-point connection with one slave (encoder) and one master (subsequent electronic) based on the physical differential RS485 standard. Therefore only four signal lines are required for the two differential encoder signals *Clock* and *Data*. The data is transmitted synchronously with the clock signal that is provided from the EnDat master.

Encoders supporting the EnDat interface contain an internal non-volatile memory that can be read and written by the master. The EnDat interface can transmit position values from encoders as well as read out information stored in the encoders memory to update or store new information. Furthermore, the EnDat protocol provides diagnostic capabilities, such as a CRC error check. This lets you evaluate the validity of the transmitted values.

For more information on the EnDat protocol refer to the Heidenhain website at www.heidenhain.com.

Supported encoders

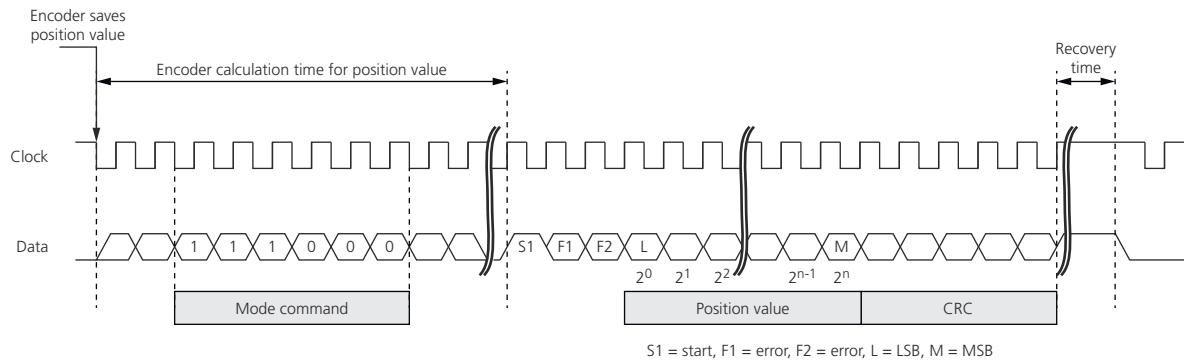
The EnDat interface supports single-turn rotary encoders and multi-turn rotary encoders. A single-turn encoder provides the absolute position for one revolution, a multi-turn encoder also provides the current revolution count.

Absolute encoders output the angular position value as a digital numerical value. Because this numerical value is unique across the entire resolution range of the absolute encoder, no initial reference run is required, as is the case with incremental encoders, for example. The position of the encoder is available immediately after power on.

Transmission of position values

The data transmission is synchronized by a clock signal. The data signal is used to get the measured data from the connected encoder but also to write commands

and data to the encoder. The following illustration shows a data transmission using the EnDat 2.2 protocol.



The transmission cycle begins with the first falling clock edge. The measured values are saved and the position value is calculated by the encoder. After two clock pulses the EnDat master transmits the mode command. This mode command determines which data is to be transmitted (position values, parameters, diagnostics, etc.).

After the encoder's calculation time has expired, the encoder transmits a start bit (S1). If position data is to be transmitted, the encoder first transmits the error flags and then the position data and a CRC checksum to the EnDat master.

After a recovery time, a new data transmission can begin by starting the clock.

EnDat protocol versions The EnDat protocol is available in two versions: The EnDat 2.1 version and the extended EnDat 2.2 version. The version supported by the connected encoder is stored in the encoder's memory. The EnDat 2.2 mode command set includes all EnDat 2.1 mode commands. In addition, EnDat 2.2 permits more mode commands and supports higher clock frequencies.

Speed measurement

The EnDat Master function block can determine the electrical speed of the connected encoder on the basis of the transmitted position value. The speed is calculated as the derivative of the position: $\Delta\text{position} / \Delta\text{time}$.

Overviews (EnDat Master)

Where to go from here

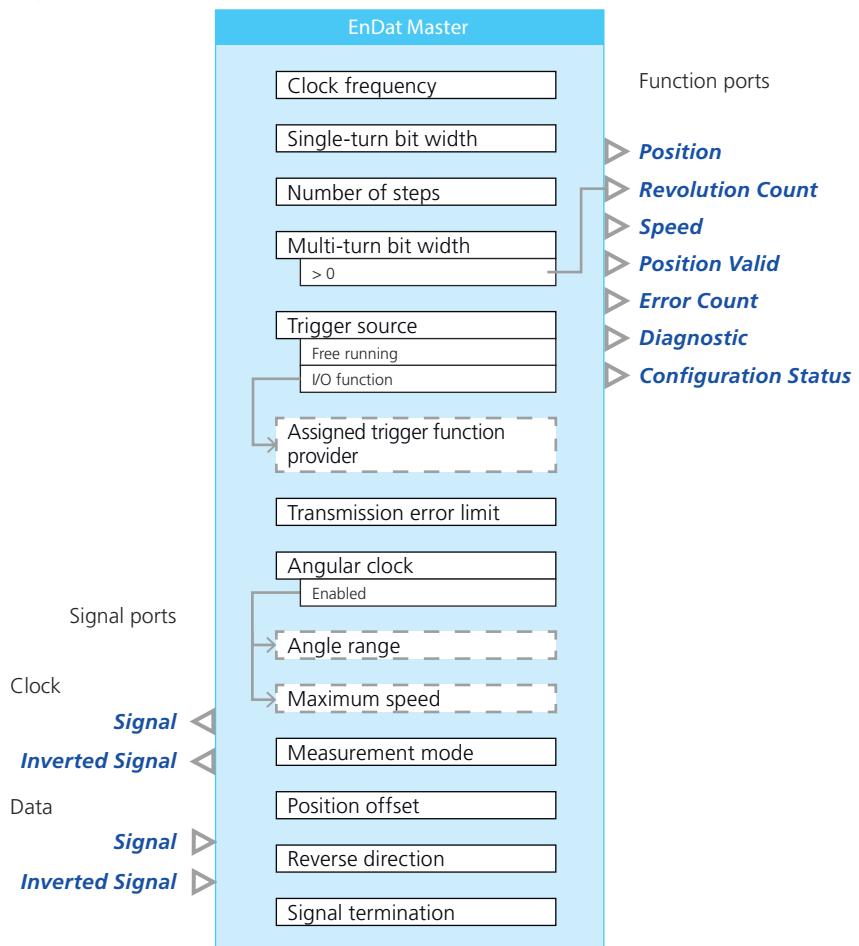
Information in this section

- | | |
|------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (EnDat Master)..... | 939 |
| Overview of Tunable Properties (EnDat Master)..... | 943 |

Overview of Ports and Basic Properties (EnDat Master)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Position

This function outport writes the transmitted position value of the connected absolute encoder to the behavior model.

Value range	$0^\circ \dots 360^\circ$. At 360° , the position counter restarts from 0° .
Dependencies	–

Revolution Count

This function outport writes the transmitted revolution count of the connected absolute encoder to the behavior model.

Value range	$0 \dots 2^{[\text{Multi-turn bit width}]} - 1$
Dependencies	Available only if the Multi-turn bit width property is set to 1 or higher.

Speed

This function outport writes the electrical speed of the connected absolute encoder to the behavior model. The sign (+ or –) of the speed value depends on the direction of rotation.

Value range	[Double.min] °/s ... [Double.max] °/s
Dependencies	–

Position Valid

This function outport provides the status to the behavior model, indicating whether the position value received from the connected encoder is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid The received position value is stated as invalid, if at least one of the following conditions occur: <ul style="list-style-type: none"> ▪ The number of consecutive transmission errors (CRC checksum error) exceeds the specified transmission error limit. ▪ No valid message has been read since the start of the real-time application. ▪ The connected encoder does not respond. ▪ The F1 error bit is active (high) in the EnDat protocol. ▪ The F2 error bit is active (low) in the EnDat protocol (only valid for EnDat 2.2 encoders). If an invalid value is detected, the position value provided at the Position function port is not updated with the new received value. Therefore the last valid position value is still provided. ▪ 1: Valid The received position value is stated as valid. Therefore, the position value provided at the Position function port is updated with the new received value.
Dependencies	–

Error Count

This function outport provides the number of errors that occurred in a model step to the behavior model. The error count value is incremented each time a transmission error (CRC checksum error) is detected. The value is reset for each model step and saturated at 255.

Value range	0 ... 255
Dependencies	–

Diagnostic

This function outport writes diagnostic information to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1: The encoder did not respond. No encoder is connected to the EnDat Master function block, or the connected encoder does not respond. ▪ 2: A CRC checksum error occurred. The transmitted CRC checksum indicates corrupted measurement data. ▪ 16: Depending on the used EnDat protocol: <ul style="list-style-type: none"> ▪ EnDat 2.1: The F1 error bit is active (high). ▪ EnDat 2.2: The F1 error bit is active (high) or the F2 error bit is active (low).
Dependencies	–

Configuration Status

This function outport writes status information related to the configuration settings of the function blocks to the behavior model.

For the function block to work properly, the configuration of the function block must match the connected encoder. The consistency check of the configuration requires communication with the connected encoder. This status information can therefore be detected only while a real-time application is running.

Value range	<ul style="list-style-type: none"> ▪ 0: No error. No error occurred. ▪ 1: Encoder data read error. An error occurred while reading EnDat parameters from the connected encoder.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>If you are using an EnDat 2.2 encoder and a "position resolution error" (value = 16) occurs, this flag is also set because some of the encoder data can only be read with the correct resolution setting.</p> <ul style="list-style-type: none"> ▪ 16: Position resolution error. The sum of the specified values for the Single-turn bit width and Multi-turn bit width properties does not match the value stored in the memory of the connected encoder. ▪ 32: Revolution count error. The specified value for the Multi-turn bit width property does not match the value stored in the memory of the connected encoder. ▪ 64: Number of steps error. The specified value for the Number of steps property does not match the value stored in the memory of the connected encoder. ▪ 128: Clock frequency error (available only for encoders that support the EnDat 2.2 protocol). The calculated value for the clock frequency exceeds the maximum value for the clock signal frequency that is stored in the memory of the connected encoder.
Dependencies	–

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (EnDat Master) on page 950.
Dependencies	–

Inverted Signal

This signal port and the Signal signal port together represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input or output. Both signal ports (Signal and Inverted Signal) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (EnDat Master) on page 950.
Dependencies	–

Overview of Tunable Properties (EnDat Master)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Transmission error limit	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Measurement Mode	✓	-
Position offset	✓	-
Signal termination (for data signal)	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (EnDat Master)

Where to go from here

Information in this section

Configuring the Basic Functionality (EnDat Master).....	944
Configuring the Standard Features (EnDat Master).....	949

Configuring the Basic Functionality (EnDat Master)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Adjusting the function block to the connected encoder (resolution for position and revolution values, position offset, rotation direction).
- Specifying the clock frequency.
- Specifying speed measurement (measurement mode).
- Specifying a trigger source for starting data transmission.
- Specifying a limit for transmission errors.
- Terminating the input signal.
- Enabling the master APU functionality to provide the position values to be used by other function blocks.

Adjusting the function block to the connected encoder

You have to define some characteristics of the connected encoder to adjust it to the function block to guarantee that the function block determine correct position and speed values.

Specifying the resolution for position and revolution counts The resolution settings of the function block must match the values stored in the memory of the connected encoder. Refer to the data sheet of the encoder.

After starting the real-time application, the consistency of the settings is checked. If the function block detects a mismatch, related status information is provided to the behavior model via the Configuration Status function port.

- You have to specify the resolution of the position value via Single-turn bit width property. That is the number of bits provided by the connected encoder to represent the position value for one revolution. A maximum value of 48 bits is supported by the function block.
If no single-turn data is used in your application, you can set the value to zero.
- If the connected single-turn encoder does not support the full range of the encoder's position resolution, you have to explicitly specify the number of steps. For example, the encoder has a resolution of 12 bits (0 ... 4095 position values), but the encoders code disc provides only 3,000 steps.

If you specify zero, the number of steps is derived from the value specified at the **Single-turn bit width** property. In this case, the full resolution of the connected encoder is used. Most encoders support this setting.

- If you use a multi-turn encoder, you have to specify the resolution of the revolution counter via the **Multi-turn bit width** property. That is the number of bits provided by the connected encoder to represent the revolution number. A maximum value of 48 bits is supported by the function block.

The value implicitly defines the maximum number of revolutions that can be counted by the encoder.

Note

If a single-turn encoder is connected to the function block, you have to set the value of the **Multi-turn bit width** property to 0.

The sum of the values specified for the **Single-turn bit width** and the **Multi-turn bit width** properties must be greater than 0 and not exceed the max. value of 48 bits. Otherwise, a conflict is generated and displayed in the **Conflicts Viewer**.

Specifying position offset To adjust the measured values of the encoder to the real rotor position, you can specify an offset angle via the **Position offset** property. Angular errors can be caused by an offset between the rotor shaft and the encoder mounted on the shaft. Positive and negative offset values are possible.

The specified value is added to the received position value from the encoder and the sum is provided to the behavior model via the **Position** function port.

You can also change the value of the position offset via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Adjusting rotational direction To adjust the rotational direction to the direction required in the behavior model, you can change the direction by enabling the **Reverse direction** property. This is useful, for example, if the encoder must be mounted in the wrong direction on the rotor shaft.

If you enable the **Reverse direction** property, the reverse order is interpreted as a forward rotation, i.e., the measured values are interpreted as a forward rotation if the position values received from the encoder are descending. Refer to the data sheet of the encoder to find out, in which mounting position this applies.

As a result, the speed values provided at the **Speed** function port are inverted and the values provided at the **Position** and **Revolution Count** function ports move in the reverse direction.

Specifying clock frequency

Data transmission is driven by a clock signal. You have to specify the frequency of the clock signal in the range 45 kHz ... 16 MHz to be generated from the dSPACE hardware. Due to technical issues the generated clock frequency might differ from the specified frequency. The actual frequency generated by the hardware is displayed via the **Calculated clock frequency** property.

Note

The specified clock frequency should not exceed the maximum clock frequency value that is stored in the memory of the connected encoder. Refer to the data sheet of the encoder.

Effects on specifying a clock frequency of 1 MHz or higher for EnDat 2.2 encoders The default waiting time between two data transmissions (also called recovery time) is in the range 10 ... 30 µs. If you specify a clock frequency of 1 MHz or higher for an EnDat 2.2 encoder, the encoder is automatically configured to the shorter waiting time of 1.25 ... 3.75 µs. This is useful for higher clock frequencies, because more time can be used for data transmission and less is spent on waiting. This behavior corresponds to the EnDat 2.2 specification.

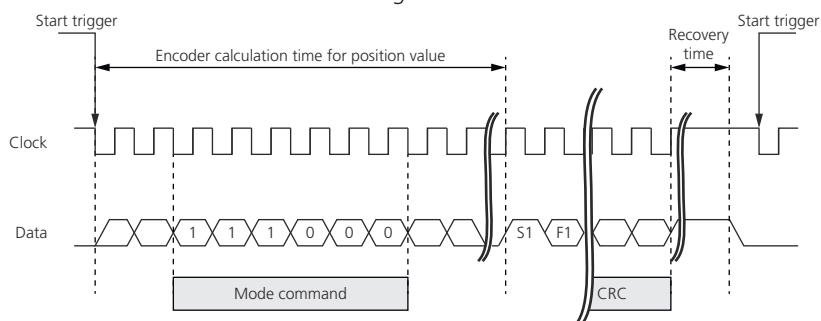
Note

The configuration of the shorter waiting time is stored in the encoder's non-volatile memory. After its activation, EnDat 2.2 commands cannot be used with a clock frequency of less than 1 MHz.

If you want to use an encoder with another hardware system that uses a clock frequency below 1 MHz but is unable to configure the waiting time, you have to clear the shorter waiting time configuration. To do so, specify a clock frequency of less than 1 MHz and start the real-time application with the connected encoder once.

Specifying a trigger source

You have to specify a trigger source for starting the data transmission. At each trigger, the clock signal is activated (signal changes from high to low level) and then the position value is transmitted to the function block and provided to the behavior model. Refer to the following illustration.



You can choose between the following sources:

- If you select the Free-running trigger source the transmission of position data from the encoder occurs repeatedly at a nearly constant sample rate. The sample rate defines the distance between two messages and depends on various values, for example, the specified clock frequency, the single-turn bit width and the multi-turn bit width.

- If you use an I/O function as trigger source, the transmission of position data from the encoder starts by an external trigger, which is provided by a specific trigger function. This can be, for example, the Trigger In function block. Trigger function providers have their own defined trigger conditions.
- If a trigger occurs during data transmission or during the recovery time, this trigger is discarded.

Specifying a limit for transmission errors

You have to specify how many consecutive transmission errors (CRC checksum errors) the function block accepts without changing the state at the Position Valid function port to Invalid. In real environments, a transmission error must be expected from time to time. This feature makes it possible not to mark the position value as invalid if a single or few transmission errors occur.

The following table shows the possible values, some example settings and their effects on the Position Valid function port.

Possible Values for the Transmission Error Limit	Description	Example Values		
		Specified Value for Limit	Number of Transmission Errors That Occurred	Value at Position Valid function port
-1	Any number of consecutive transmission errors is accepted.	-1	1	Valid
			6	
0	No error is tolerated.	0	1	Invalid
			6	
1 ... 254	The specified number of consecutive transmission errors is accepted without invalidating the position value.	5	1	Valid
			5	Valid
			6	Invalid

Tip

The Error Count function outport provides the number of errors that occurred in a model step independently of the transmission error limit. The error count value is incremented each time a transmission error is detected in a model step.

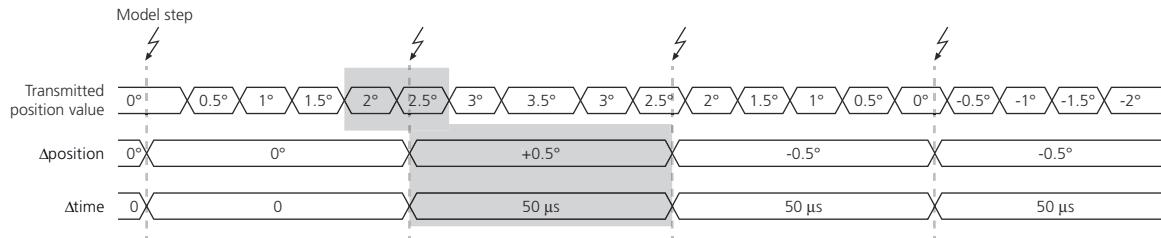
You can also change the setting of the error limit via the experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Specifying speed measurement

You can measure the dynamic speed or the average speed of the connected encoder. The speed is calculated as follows: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode (dynamic or average).

Dynamic speed measurement The dynamic speed is determined each time a new position value is available. The Δ position is the angle difference between the two position values, Δ time is the difference between the corresponding time stamps.

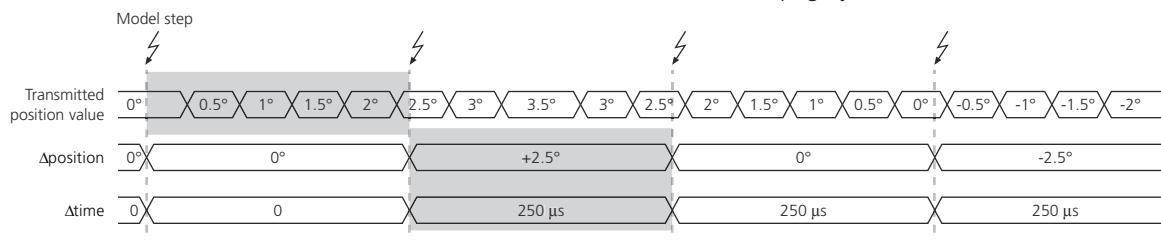
At each model step the newest determined speed is provided at the Speed function port. The following illustration is an example (with a task sample time of 250 μ s) and shows the relevant values for one model step (gray shaded area).



The dynamic measurement mode provides more accurate speed values at rapidly changing rotation speeds.

Average speed measurement The average speed is the calculated speed in a model step. All transmitted position values that end in the past model step are used for the average speed measurement. Δ position is the angle difference in a model step, Δ time is the difference between the corresponding time stamps. Refer to the following example.

Δ time is the difference of the time stamps of the position value that occur just before the start of a model step and just before the end of a model step. The following illustration is an example (with a task sample time of 250 μ s) and shows the relevant values for one model step (gray shaded area).



The average measurement mode provides more accurate speed values at less rapid changes or at constant rotation speeds. For most applications, it is recommended and useful to select this mode.

Terminating the input signal

You can activate signal termination for differential input signals to prevent signal reflections at the line end.

Signal termination is realized with a 120Ω resistor connected in series to a 5 nF capacitor between the input lines for the data signal. For a circuit diagram, refer to [Flexible In/Out 1](#) on page 1608.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Enabling master APU functionality

You can enable the master APU functionality for the function block to provide the measured angle positions to be used by other function blocks as the angle source, such as the Block-Commutated PWM Out function block.

To do this, you have to assign a master APU to the EnDat Master function block, which is located on the same dSPACE board as the I/O channels that are assigned to the function block.

The angular speed of the assigned master APU might exceed the angular speed that can be processed by the slave APU of the hardware resource of the connected function block. Therefore, you can specify a maximum speed (via the Maximum speed property) that fulfills the requirements of your application.

ConfigurationDesk uses the specified maximum speed to check if the hardware resources support the maximum speed.

For more information on the APU functionality, refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Configuring the Standard Features (EnDat Master)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The electrical interface of the function block does not provide configurable standard features.

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Flexible In/Out 1](#) on page 1608

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (EnDat Master)

SCALEXIO Hardware Dependencies (EnDat Master)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type.

Characteristics		Flexible In/Out 1
Hardware		DS6121 Multi-I/O Board
Supporting master APU functionality ¹⁾		✓
Clock, Data (Out)	Output voltage range	<ul style="list-style-type: none"> ▪ +1.5 ... +3.3 V ▪ +3.0 V typ. @RL = ∞
	Common mode voltage range ²⁾	+2 V (typ.) ... +3 V (max.)
	Supported interface type	Differential
	Frequency	45 kHz ... 16 MHz
Data (In)	Input voltage range	± 25 V
	Threshold voltage for differential inputs	± 125 mV (typ.)
	Supported interface type	Differential
	Signal termination	<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (parallel termination) ▪ Switchable
Circuit diagrams		Flexible In/Out 1 on page 1608
Required channels		2

¹⁾ The master APU functionality provides the angular position values to be used by other function blocks.

²⁾ Common mode voltage (V_{CM}) = $0.5 \cdot (V_{Signal} + V_{Inverted\ Signal})$ measured to the ground potential of the dSPACE real-time hardware.

General limitations

The channels for the clock and data signals must be located on the same I/O board.

More hardware data

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

SSI Master

Where to go from here

Information in this section

Introduction (SSI Master).....	952
Overviews (SSI Master).....	956
Configuring the Function Block (SSI Master).....	963
Hardware Dependencies (SSI Master).....	977

Introduction (SSI Master)

Where to go from here

Information in this section

Introduction to the Function Block (SSI Master).....	952
Introduction to the SSI/BiSS-C Interfaces (SSI Master).....	953

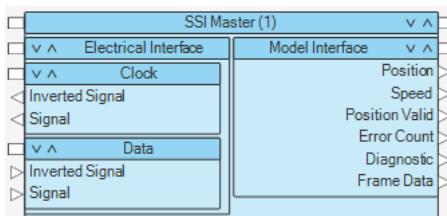
Introduction to the Function Block (SSI Master)

Function block purpose

The SSI Master function block provides access to absolute encoders (single-turn and multiturn encoders) that support the SSI or the BiSS-C interface. The function block can be used to determine the angular position, the revolution count, and the speed of an electric motor, for example.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Supporting SSI, BiSS-C and BiSS BP1 (predefined BiSS-C encoder standard) protocols
- Adjusting the function block to the connected absolute encoder and to the used protocol (for example, configuration of protocol-specific characteristics).
- Determining and providing angular positions, revolution counts, and speed values to the behavior model.
- Checking the validity of the received position values and providing the result to the behavior model.
- Evaluating of received diagnostic information from the encoder (such as connection and transmission errors) and providing the result to the behavior model.
- Executing data transmission continuously or triggered by an I/O event, for example, from another function block.

- Providing the received data as a raw bitstream to the behavior model.
- Enabling the master APU functionality to provide the position values to be used by other function blocks.

Supported channel types

The SSI Master function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	Digital In/Out 9	Flexible In/Out 1
Board	DS6121	-

Introduction to the SSI/BiSS-C Interfaces (SSI Master)

Introduction

The synchronous serial interface (SSI) and the bidirectional serial synchronous (BiSS) interface provide access to absolute encoders that determine information such as the mechanical motor position.

The SSI Master function block supports both interface types in a point-to-point configuration. That means that only one encoder (slave) is operated on the master. Connecting multiple encoders (slaves) to the function block is not supported.

Note

The BiSS interface is hardware-compatible with the SSI interface in the point-to-point configuration.

The BiSS interface is designed in different modes. The SSI Master function block supports the BiSS-C mode.

Supported encoders

The SSI Master function block supports single-turn and multiturn rotary encoders that are compatible with SSI, BiSS-C and BiSS BP1 (predefined BiSS-C encoder standard protocols).

A single-turn encoder provides the absolute position for one revolution, a multiturn encoder also provides the current revolution count.

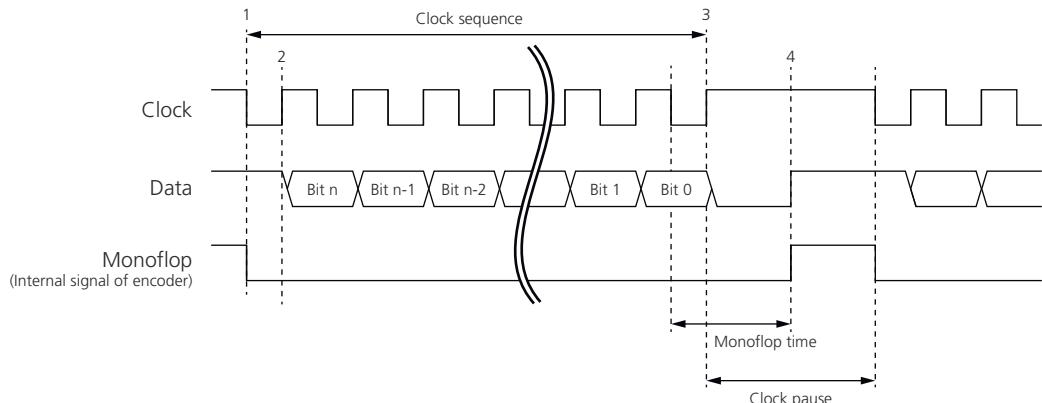
Absolute encoders output the angular position value as a digital numerical value. Because this numerical value is unique across the entire resolution range of the absolute encoder, no initial reference run is required, as is the case with incremental encoders, for example. The position of the encoder is available immediately after power-up.

Transmission of position and diagnostic values

The SSI and BiSS-C interfaces can transmit position values from encoders as well as diagnostic capabilities, such as a CRC checksum, warning and error flags. This lets you, for example, evaluate the validity of the transmitted position values.

Two unidirectional encoder signals, *Clock* and *Data*, are used. The data is transmitted synchronously with the clock signal that is provided by the SSI master function block. The data signal is used to get the measured data from the connected encoder. The first falling edge of the clock signal triggers the data transmission.

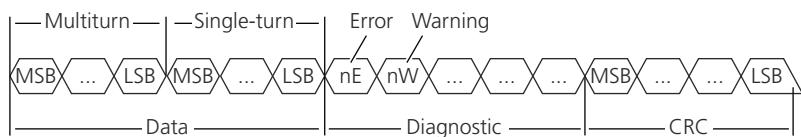
Example for the SSI protocol The following illustration shows an example of data transmission using the SSI protocol.



When no data is transmitted, the clock and data signals are at the high level. The transmission is controlled via a monoflop. The first falling edge of the clock signal triggers the monoflop (position 1 in the illustration above). With the first rising edge of the clock signal, the transmission starts with the most significant bit (position 2). When all bits are transmitted (position 3), the clock signal remains at the high level for the specified pause duration. If the monoflop is not triggered again by falling edges of the clock signal, it falls back to the low level after a monoflop-specific waiting time (position 4). As long as the monoflop is inactive (high signal), the register of the encoder is updated with the measured position.

Example of data format

The following illustration shows an example for a data format based on the SSI or BiSS-C protocol. The transmitted frame contains a data area, a diagnostic area, and a CRC area.



The SSI/BiSS-C protocols have no strict specifications regarding the data format, most of the bits are optional. The position of some bits, for example, the error and the warning bits in the bitstream and the evaluation of these bits is not standardized. The transmission of the position data might also differ between various manufacturers.

Note

It is strongly recommended to read the data sheet of the connected encoder to get the correct configuration settings for the SSI Master function block.

Speed measurement

The SSI Master function block can determine the electrical speed of the connected encoder on the basis of the transmitted position value. The speed is calculated as the derivative of the position: Δposition / Δtime.

Providing data as a raw bitstream to the behavior model

The SSI Master function block provides the Frame Data function port. This port provides the entire content of a transmission as a raw bitstream. You can evaluate this bitstream in the behavior model by your own data processing method.

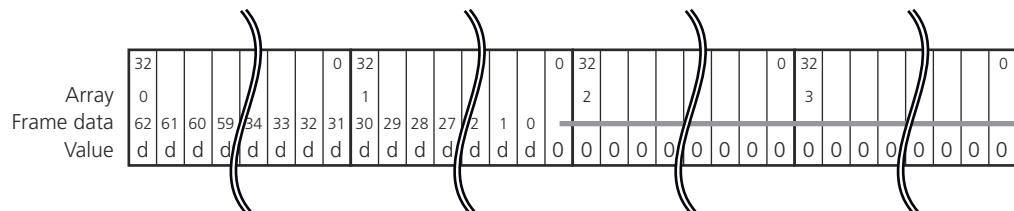
Note

When using the BiSS-C protocol, the CDS bit is not included in the raw bitstream.

The data bits of the frame are stored in an array with four elements, each 32 bits in size. The following order is applied:

- The most significant bit of the first element in the array (index 0) holds the first transmitted bit of the data frame transmitted by the encoder. Usually, this is the most significant bit of the position data when you use a single-turn encoder, or of the revolution data when you use a multturn encoder.
- The 32nd bit of the data frame fills the least significant bit of the first array element.
- Unused bits in the array elements are set to zero by default.

The following example shows an array with a data frame of 63 bits.



Oversviews (SSI Master)

Where to go from here

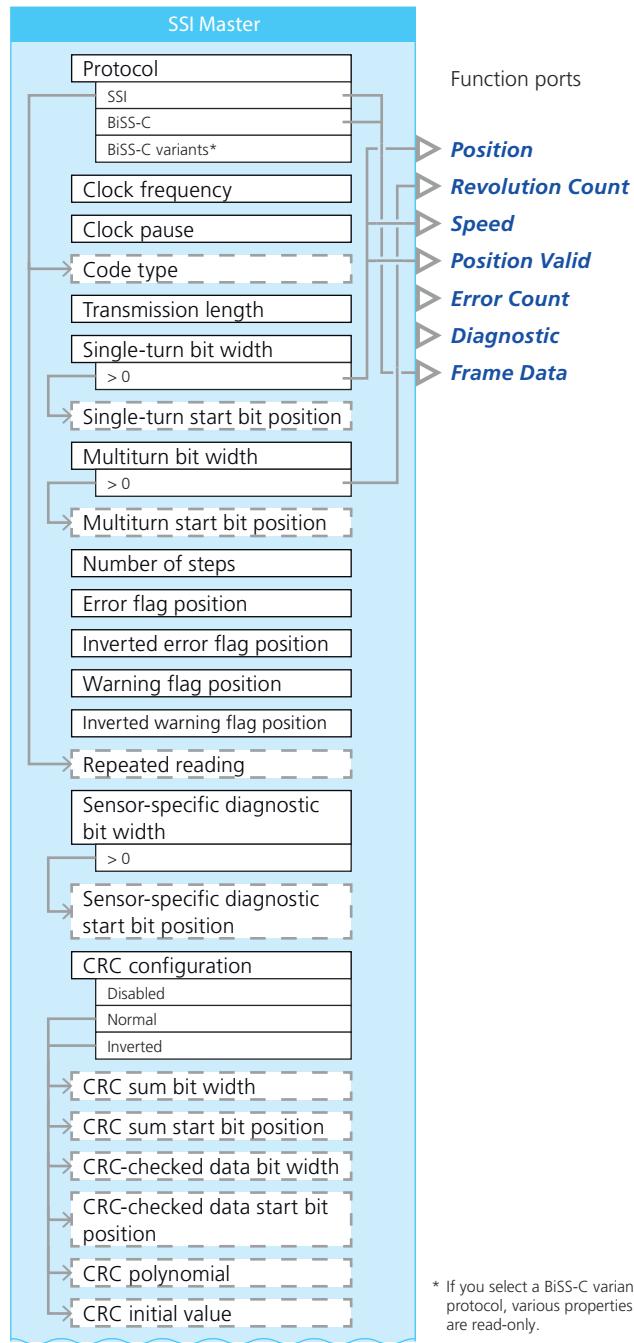
Information in this section

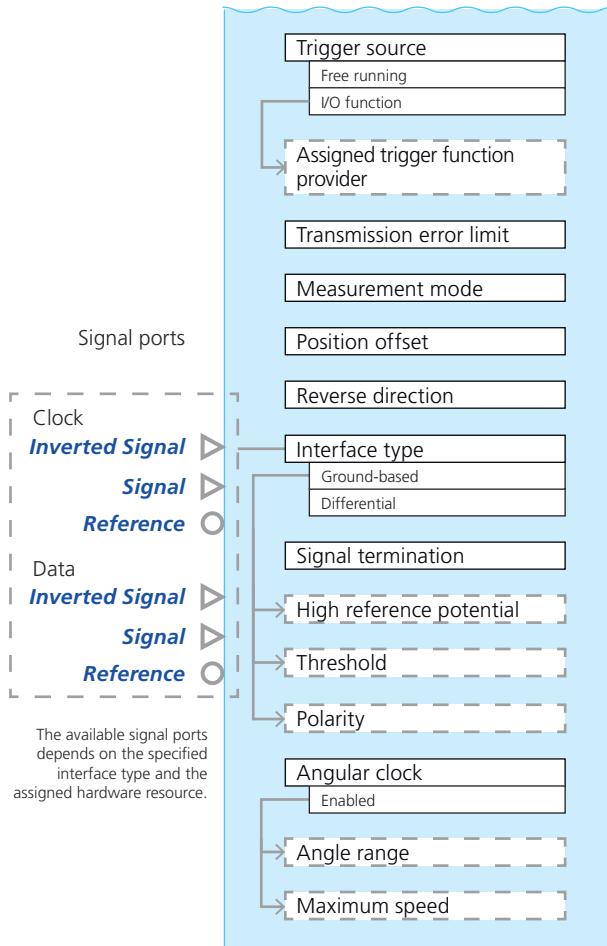
Overview of Ports and Basic Properties (SSI Master).....	956
Overview of Tunable Properties (SSI Master).....	961

Overview of Ports and Basic Properties (SSI Master)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



**Position**

This function outport writes the transmitted position value of the connected absolute encoder to the behavior model.

Value range	$0^\circ \dots 360^\circ$. At 360° , the position counter starts again at 0° .
Dependencies	Available only if the Single-turn bit width property is set to 1 or higher.

Revolution Count

This function outport writes the transmitted revolution count of the connected absolute encoder to the behavior model.

Value range	$0 \dots 2^{[\text{Multiturn bit width}]} - 1$
Dependencies	Available only if the Multiturn bit width property is set to 1 or higher.

Speed

This function outport writes the speed of the connected absolute encoder to the behavior model. The sign (+ or –) of the speed value depends on the direction of rotation.

Value range	[Double.min] °/s ... [Double.max] °/s
Dependencies	Available only if the Single-turn bit width property is set to 1 or higher.

Position Valid

This function outport provides the status to the behavior model, indicating whether the position value received from the connected encoder is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid The received position value is stated as invalid, if at least one of the following conditions occur: <ul style="list-style-type: none"> ▪ The number of consecutive transmission errors (CRC checksum errors and repeated reading errors) exceeds the specified transmission error limit. ▪ No valid message has been read since the start of the real-time application. ▪ The connected encoder does not respond. ▪ The error bit is active (high) or the inverted error bit is active (low). If an invalid value is detected, the position value provided at the Position function port is not updated. Therefore, the last valid position value is still provided. ▪ 1: Valid The received position value is stated as valid. Therefore, the position value provided at the Position function port is updated with the new received value.
Dependencies	Available only if the Single-turn bit width property is set to 1 or higher.

Error Count

This function outport provides the number of errors that occurred in a model step to the behavior model. The error count value is incremented each time a transmission error (CRC checksum error or repeated reading error) is detected. The value is reset for each model step and saturated at 255.

Value range	0 ... 255
Dependencies	–

Diagnostic

This function outport writes diagnostic information to the behavior model.

Value range	The provided decimal value represents a 24-bit binary value. The following diagnostic information is available: <ul style="list-style-type: none"> ▪ 1 (bit 0 = 1): The encoder did not respond.
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>No encoder is connected to the SSI Master function block, or the connected encoder does not respond within a timeout value (often referred to as monoflop time or transfer timeout). The timeout is reached if there is no rising edge of the data signal within the specified clock pause of the clock signal. This can happen if the specified clock pause is shorter than the timeout value required by the encoder.</p> <ul style="list-style-type: none"> ▪ 2 (bit 1 = 1): A transmission error occurred. <p>Data verification by repeated reading detected a data inconsistency or the transmitted CRC checksum indicates a transmission error.</p> <ul style="list-style-type: none"> ▪ 16 (bit 4 = 1): The error bit is active (high). ▪ 32 (bit 5 = 1): The inverted error bit is active (low). ▪ 64 (bit 6 = 1): The warning bit is active (high). ▪ 128 (bit 7 = 1): The inverted warning bit is active (low). ▪ 256, 512, 1024, etc. (bit 16 ... bit 23, 0x00FF0000): Additional customer-specified (sensor-specific) diagnostic data is provided. <p>Eight additional bits are used for this customer-specified diagnostic value. If, for example, your diagnostic value has a width of 5 bits, bits 16 to 20 are used for the value. Combinations of bits are set to indicate more than one diagnostic information at a time. To get the specific information, you must evaluate the value of the port in the behavior model.</p>
Dependencies	—

Frame Data

This function output port writes an array with a raw bitstream for transmitted SSI or BiSS-C frames of the connected encoder to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... [UInt32.Max] for each element in the array. ▪ The array is divided into four elements, each 32 bits in size. For more information on the bit order, refer to Introduction to the SSI/BiSS-C Interfaces (SSI Master) on page 953.
Dependencies	Available only if the Protocol property is set to SSI or BiSS-C.

Inverted Signal

This signal port and the Signal signal port together represent the electrical connection points of a logical signal with two complementary potentials. The electrical interface of the function block works as a fully differential input or output. Both signal ports (Signal and Inverted Signal) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SSI Master) on page 977.
Dependencies	Available only if the Interface type property is set to Differential.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SSI Master) on page 977.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SSI Master) on page 977.
Dependencies	Available only if the Interface type property is set to Ground-based.

Overview of Tunable Properties (SSI Master)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Measurement mode	✓	–
Transmission error limit	✓	–
Position offset	✓	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Threshold	✓	–
Polarity	✓	–
Signal termination	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (SSI Master)

Where to go from here

Information in this section

Configuring the Basic Functionality (SSI Master).....	963
Configuring the Detection and Handling of Transmission Errors (SSI Master).....	969
Configuring Signal Conditioning for Data and Clock Signals (SSI Master).....	972
Predefined Settings for BiSS-C Variant Protocols (SSI Master).....	974
Configuring the Standard Features (SSI Master).....	976

Configuring the Basic Functionality (SSI Master)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Selecting the transmission protocol and specifying the content of the protocol (for example, transmission length, resolution for position and revolution counts, coding scheme for transmitted data, position of diagnostic bits in the transmission frame).
- Specifying the characteristics of the clock (clock frequency and clock pause, i.e., the pause time between two transmissions).
- Adjusting the function block to the mounting position of the encoder (position offset, rotation direction).
- Specifying speed measurement (measurement mode).
- Specifying a trigger source for starting data transmission.
- Enabling the master APU functionality to provide the position values to be used by other function blocks.
- Configuring the detection of transmission errors, for example, via CRC. Refer to [Configuring the Detection and Handling of Transmission Errors \(SSI Master\)](#) on page 969.
- Configuring signal conditioning (trigger threshold, polarity, signal termination). Refer to [Configuring Signal Conditioning for Data and Clock Signals \(SSI Master\)](#) on page 972.

Selecting the protocol to be used for data transmission

You can select one of the following protocols for data transmission:

- SSI
- BiSS-C

- BiSS-C variant 0-12, BiSS-C variant 12-12, BiSS-C variant 24-12, BiSS-C variant 0-24, BiSS-C variant 12-24, BiSS-C variant 24-24

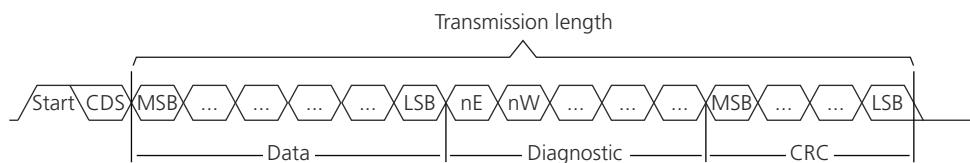
These predefined protocols can be used for BiSS BP1-compatible encoders. Compatible encoders can also be used if the single-turn bit width and the multiturn bit width are not a multiple of 12.

The name is Variant <Multiturn bit width>-<Single-turn bit width> and shows the setting for the multturn and single-turn resolutions in the protocol. In addition, various other properties are fixed to protocol-specific values. These predefined settings are then read-only.

For an overview of the predefined protocol-specific settings, refer to [Predefined Settings for BiSS-C Variant Protocols \(SSI Master\)](#) on page 974.

Specifying the transmission length

You have to specify the number of bits that are used for each transmission. The transmission length includes the position data (single-turn bits), revolution data (multiturn bits), diagnostic bits, and CRC bits. The following illustration shows an example for the BiSS-C protocol.



As shown above, the CDS bit is not part of the transmission length.

Note

To avoid transmission errors, the specified value must match the configuration of the connected encoder. Refer to the data sheet of the encoder.

Specifying the resolution for position and revolution counts

- You have to specify the resolution of the position value via the Single-turn bit width property. This is the number of bits provided by the connected encoder to represent the position value for one revolution. A maximum value of 28 bits is supported by the function block.
If no single-turn data is transmitted or used in your application, you must set the value to zero.
- If the connected single-turn encoder does not support the full range of the encoder's position resolution, you have to explicitly specify the number of steps. For example, the encoder has a resolution of 12 bits (i.e., 4,096 steps), but the encoders code disc provides only 3,000 steps.
If you specify zero, the number of steps is derived from the value specified at the Single-turn bit width property. In this case, the full resolution of the connected encoder is used. Most encoders support this setting.
- If you use a multturn encoder, you have to specify the resolution of the revolution counter via the Multiturn bit width property. This is the number

of bits provided by the connected encoder to represent the revolution number. A maximum value of 28 bits is supported by the function block.

The value implicitly defines the maximum number of revolutions that can be counted by the encoder.

If a single-turn encoder is connected to the function block, set the value of the **Multiturn bit width** property to 0.

- If you specify a single-turn bit width, you must specify the position of the start bit in the bit sequence of a transmission. This applies even if you specify a multturn bit width.

Note

To get correct position data, the resolution settings of the function block must match the configuration of the connected encoder. Refer to the data sheet of the encoder.

Specifying the characteristics of the clock signal

Specifying the clock frequency Data transmission is synchronized by a clock signal. You have to specify the frequency of the clock signal to be generated by the dSPACE hardware. The SSI Master function block supports frequencies in the range of 45 kHz ... 2 MHz for the SSI protocol, and frequencies of 45 kHz ... 10 MHz for the BiSS-C protocol. Due to technical constraints the generated clock frequency might differ from the specified frequency. The actual frequency generated by the hardware is displayed via the **Calculated clock frequency** property.

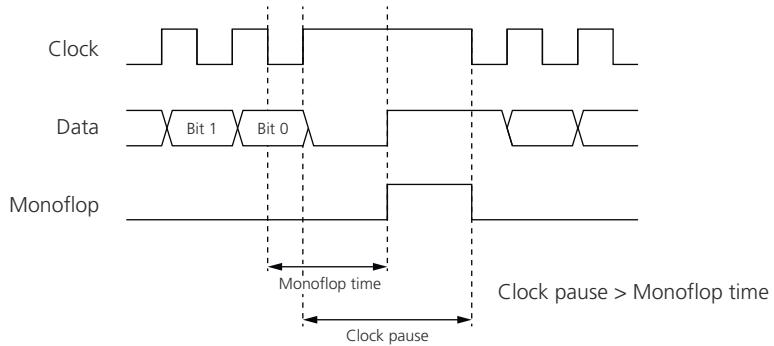
Note

To get correct position data, the specified clock frequency must not exceed the maximum clock frequency supported by the connected encoder. Refer to the data sheet of the encoder.

Specifying a clock pause You have to specify a pause time for the clock output signal. This is the minimum time that must elapse between two data transmissions.

Note

To avoid a connection error, the specified clock pause must be longer than the timeout value required by the connected encoder. This timeout value is often referred to as monoflop time or transfer timeout. Refer to the data sheet of the encoder.



If the specified clock pause is shorter than the required timeout, a diagnostic flag (The encoder did not respond) is provided at the Diagnostic function port.

Selecting a coding scheme for transmitted data

If you use the SSI protocol, you have to select a coding scheme for the transmitted data. You can select between **Binary code** or **Gray code** via the **Code type** property. All other protocols support only the binary code.

Note

To get correct position data, the specified setting must match the configuration of the connected encoder. Refer to the data sheet of the encoder.

If you use the **Gray code** setting, the multturn data bits must precede the single-turn data bits in the bit sequence of a transmission. If you use a different sequence in the frame, a conflict is generated and displayed in the **Conflicts Viewer**.

Specifying the position of diagnostic bits in the transmission frame

The SSI and BiSS-C protocols support the transmission of specific diagnostic data. You have to specify the following characteristics:

- If the connected encoder provides sensor-specific diagnostic data, you have to specify the number of bits (bit width) that are used for this diagnostic data and its start position in the bit sequence of a transmission.
- If the connected encoder provides error or warning status information (either high active or low active or both), you have to specify the position of the corresponding status information in the bit sequence of a transmission.

If no warning and/or error bits are transmitted or used in your application, set the corresponding position bit to zero. If no sensor-specific diagnostic data is transmitted or used in your application, set the **Sensor-specific diagnostic bit width** property to zero.

Note

To get correct diagnostic data, the specified value must match the configuration of the connected encoder. Refer to the data sheet of the encoder.

The transmitted diagnostic data is evaluated by the function block and can be provided as diagnostic flag to the behavior model via the Diagnostic function port. For more information, refer to [Overview of Ports and Basic Properties \(SSI Master\)](#) on page 956.

Adjustments due to the mounting position of the encoder

In some cases you might have to make some adjustments due to the mounting of the encoder on the motor shaft to obtain correct position values.

Specifying position offset To adjust the measured values of the encoder to the real rotor position, you can specify an offset angle via the Position offset property. Angular errors can be caused by an offset between the rotor shaft and the encoder mounted on the shaft. Positive and negative offset values are possible.

The specified value is added to the received position value from the encoder and the sum is provided to the behavior model via the Position function port.

You can also change the value of the position offset via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

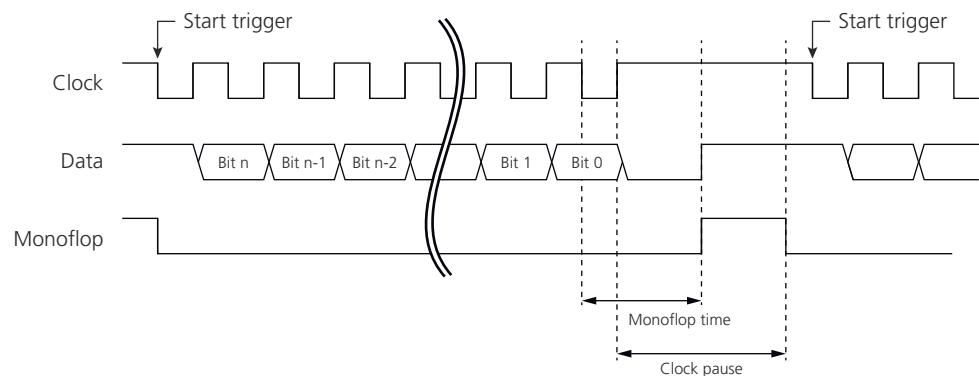
Adjusting rotational direction To adjust the rotational direction to the direction required in the behavior model, you can change the direction by enabling the Reverse direction property. This is useful, for example, if the encoder must be mounted in the wrong direction on the rotor shaft.

If you enable the Reverse direction property, the reverse order is interpreted as a forward rotation, i.e., the measured values are interpreted as a forward rotation if the position values received from the encoder are descending. Refer to the data sheet of the encoder to find out, in which mounting position this applies.

As a result, the speed values provided at the Speed function port are inverted and the values provided at the Position and Revolution Count function ports move in the reverse direction.

Specifying a trigger source

You have to specify a trigger source for starting the data transmission. At each trigger, the clock signal is activated (signal changes from high to low level) and then the position value is transmitted to the function block and provided to the behavior model. Refer to the following illustration.



You can choose between the following sources:

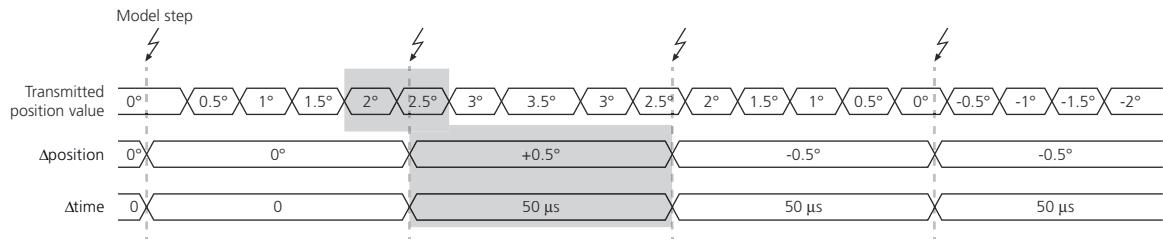
- If you select the Free-running trigger source, the transmission of position data from the encoder occurs repeatedly at a nearly constant sample rate. The sample rate defines the distance between two messages and depends on various values, for example, the specified clock frequency, the single-turn bit width and the multiturn bit width.
 - If you use an I/O function as the trigger source, the transmission of position data from the encoder is started by an external trigger, which is provided by a specific trigger function. This can be the Trigger In function block. Trigger function providers have their own defined trigger conditions.
- If a trigger occurs during data transmission or during the clock pause, this trigger is discarded.

Specifying speed measurement

You can measure the dynamic speed or the average speed of the connected encoder. The speed is calculated as follows: $\Delta\text{position} / \Delta\text{time}$. $\Delta\text{position}$ and Δtime are determined on the basis of the selected measurement mode (dynamic or average).

Dynamic speed measurement The dynamic speed is determined each time a new position value is available. The $\Delta\text{position}$ is the angle difference between the two position values, Δtime is the difference between the corresponding time stamps.

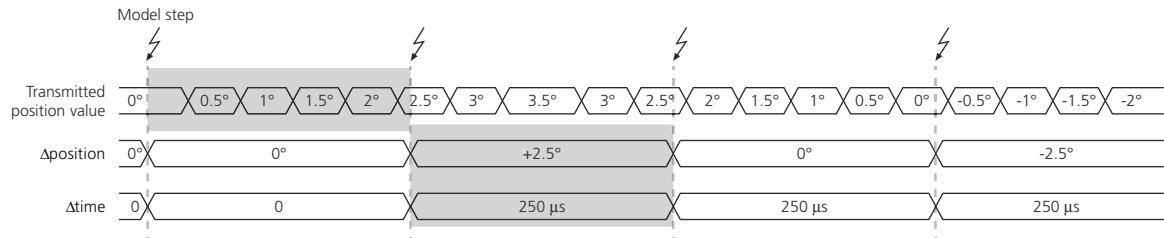
At each model step, the newest determined speed is provided at the Speed function port. The following illustration is an example (with a task sample time of 250 μs) and shows the relevant values for one model step (gray shaded area).



The dynamic measurement mode provides more accurate speed values at rapidly changing rotation speeds.

Average speed measurement The average speed is the calculated speed in a model step. All transmitted position values that end in the past model step are used for the average speed measurement. $\Delta\text{position}$ is the angle difference in a model step, Δtime is the difference between the corresponding time stamps. Refer to the following example.

Δtime is the difference of the time stamps of the position value that occur just before the start of a model step and just before the end of a model step. The following illustration is an example (with a task sample time of 250 μs) and shows the relevant values for one model step (gray shaded area).



The average measurement mode provides more accurate speed values at less rapid changes or at constant rotation speeds. For most applications, it is recommended and useful to select this mode.

Enabling master APU functionality

You can enable the master APU functionality for the function block to provide the measured angle positions to be used by other function blocks as the angle source, such as the Block-Commutated PWM Out function block.

To do this, you have to assign a master APU to the SSI Master function block, which is located on the same dSPACE board as the I/O channels that are assigned to the function block.

The angular speed of the assigned master APU might exceed the angular speed that can be processed by the slave APU of the hardware resource of the connected function block. Therefore, you can specify a maximum speed (via the Maximum speed property) that fulfills the requirements of your application.

ConfigurationDesk uses the specified maximum speed to check if the hardware resources support the maximum speed.

For more information on the APU functionality, refer to [Using Angular Processing Units \(APUs\)](#) on page 126.

Configuring the Detection and Handling of Transmission Errors (SSI Master)

Overview

The function block supports the following configuration features to detect and handle transmission errors:

- Enabling the repeated reading method (only for the SSI protocol).
- Enabling and configuring the CRC.
- Specifying a limit value before a transmission error becomes effective

Enabling the repeated reading method

Using the SSI protocol for data transmission, you can enable a data consistency check to detect transmission errors. If you use this check, the position data is read twice from the encoder and then both values are compared. This also is referred to as repeated reading method.

Note

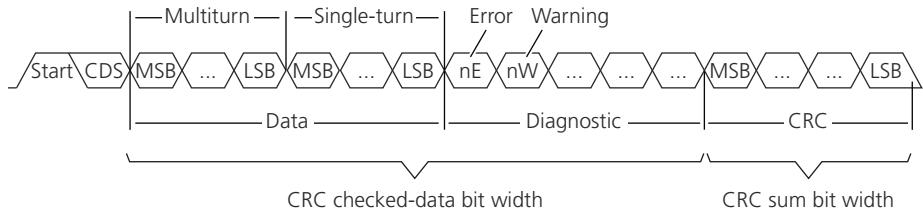
- The repeated reading method must be supported by the connected encoder. Reading the position data twice must be done without pause to ensure that the encoder transmits the same position value. Not all encoders support this feature. Refer to the data sheet of the encoder.
- If you activated the verification of the position data, the effective bandwidth of the communication between the SSI interface and the connected encoder is reduced, because every read access is performed twice.
- The BiSS-C protocol (including the variants) does not support the repeated reading method.

If the two position values differ from one another the following applies:

- A transmission error is reported via the Diagnostic function port. For more information, refer to [Overview of Ports and Basic Properties \(SSI Master\)](#) on page 956.
- The error count value is incremented and provided at the Error Count function port.
- The position value provided at the Position function port is not updated. Therefore the last valid position value is still provided.

Using and configuring the CRC

The BiSS-C protocol and some encoders using the SSI protocol provide a Cyclic Redundancy Check (CRC) to detect transmission errors. If used, a CRC checksum value is contained in the transmitted frame to increase the transmission safety. Refer to the following illustration.



The CRC checksum is generated with a specified start value and a specified generator polynomial.

Note

To avoid transmission errors, all values specified for the CRC must match the configuration of the connected encoder. Refer to the data sheet of the encoder.

If a CRC checksum error is detected the following applies:

- A transmission error is reported via the Diagnostic function port. For more information, refer to [Overview of Ports and Basic Properties \(SSI Master\)](#) on page 956.

- The error count value is incremented and provided at the Error Count function port.
- The position value provided at the Position function port is not updated with the new received value. Therefore the last valid position value is still provided.

For the CRC you have to specify the following characteristics:

- The CRC configuration property allows you to specify whether the CRC checksum is transmitted as is (not inverted) or whether it is transmitted after it was inverted.
- You have to specify the number of bits (CRC sum bit width property) that are used for the CRC checksum (= resulting CRC value) in each transmission and its start position in the bit sequence of a transmission (CRC sum start bit position property).
- You have to specify the number of bits (CRC checked data bit width property) that are used for the CRC-checked data in each transmission and its start position in the bit sequence of a transmission (CRC checked data start bit property).
- You have to specify the start value (CRC initial value property) and the generator polynomial value (CRC polynomial property), that is used for the calculation of the CRC checksum. You have to enter the polynomial as decimal value. For the calculation you have to use the MSB. Example: The polynomial $x^6 + x^1 + x^0$ leads to a decimal value of 67 (0x0043).

The following rules apply to the configuration. If they are not fulfilled, conflicts are generated and displayed in the Conflicts Viewer.

- The CRC sum bit width must not overlap with the CRC checked-data bit width.
- The CRC-checked data must precede the CRC checksum.

Tip

If you disable the CRC and the connected encoder nevertheless transmits CRC checksum values, this values are not evaluated by the function block. However, the CRC bits are contained in the raw bitstream that can be provided to the behavior model via the Frame Data function port.

Checking the data transmission using a parity bit Some SSI-compatible encoders use a parity bit for error detection in the transmitted data. In this case, you can check the received bit sum by using the following property settings:

Property	Setting
CRC polynomial	3 (decimal value)
CRC initial value	<ul style="list-style-type: none"> ▪ 0 (even parity) ▪ 1 (odd parity)
CRC sum bit width	1
CRC sum start bit position	Bit position of the parity bit.

Specifying a limit value for transmission errors

You have to specify how many consecutive transmission errors (CRC checksum errors and repeated reading errors) the function block accepts without changing the state at the Position Valid function port to Invalid. In real environments, a transmission error must be expected from time to time. This feature makes it possible not to mark the position value as invalid if a single or few transmission errors occur.

The following table shows the possible values, some example settings and their effects on the Position Valid function port.

Possible Values for the Transmission Error Limit	Description	Example Values		
		Specified Value for Limit	Number of Transmission Errors That Occurred	Value at Position Valid function port
-1	Any number of consecutive transmission errors is accepted.	-1	1	Valid
			6	
0	No error is tolerated.	0	1	Invalid
			6	
1 ... 254	The specified number of consecutive transmission errors is accepted without invalidating the position value.	5	1	Valid
			5	Valid
			6	Invalid

Tip

The Error Count function output provides the number of errors that occurred in a model step independently of the transmission error limit. The error count value is incremented each time a transmission error is detected in a model step.

You can also change the setting of the error limit via the experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Configuring Signal Conditioning for Data and Clock Signals (SSI Master)

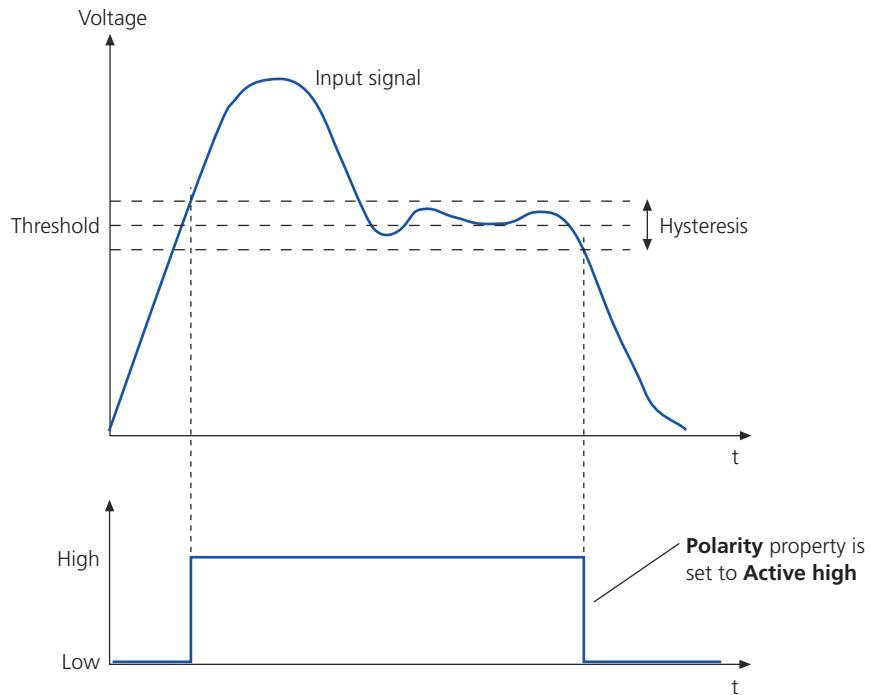
Overview

Depending on the used interface type (ground-based, differential) the function block provides the following configuration features to condition the data and the clock signal:

- Specifying a threshold (for the data signal)
- Specifying the polarity of the signals (together for the clock and data signal).
- Selecting the high-level voltage (for the clock signal).
- Enabling signal termination.

Specifying the threshold for the data signal

If you use the ground-based interface type, you can specify a threshold value to transform the input signal into binary values. The hysteresis value is fixed and depends on the assigned hardware resource. A rising edge of the input signal must exceed the threshold plus half the hysteresis value to be detected as high, and a falling edge of the input signal must fall below the threshold minus the half hysteresis value to be detected as low.



The value range of the threshold and the hysteresis constant depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(SSI Master\) on page 977](#).

Specifying polarity of the signal

To adapt the real-time hardware to signals of different polarity without the need to use inverters on the hardware side, you can switch the **Polarity** of the input signal (data) and the output signal (clock). The switching takes place together for both signals.

- **Active high**

The higher voltage value of the signal represents a binary 1, and the lower value represents a binary 0.

- **Active low**

The lower voltage value of the signal represents a binary 1, and the higher value represents a binary 0.

Note

Switching the polarity is only supported if you use the ground-based interface type.

Selecting the high level voltage for the clock signal

You have to select the high level voltage for the generated clock signal via the High reference potential property as follows:

- Shared:

The internal 5 V supply of the dSPACE real-time hardware is used as voltage for the high level.

- Shared 2:

The internal 3.3 V supply of the dSPACE real-time hardware is used as voltage for the high level.

Note

Selecting the high level voltage is only supported if you use the ground-based interface type.

Enabling signal termination

The setting of the Interface type property determines the termination resistor that is switched as follows:

- Ground-based: The termination resistor for the clock signal is switched.

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable. For a circuit diagram, refer to [Digital In/Out 9](#) on page 1590.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

- Differential: The termination resistor for the data signal is switched.

To prevent signal reflections at the line end, a $120\ \Omega$ resistor connected in series to a 5 nF capacitor is activated between the input lines. For a circuit diagram, refer to [Flexible In/Out 1](#) on page 1608.

You can also enable/disable signal termination via experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

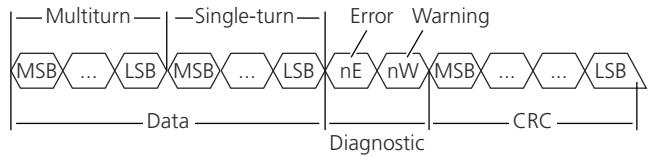
Predefined Settings for BiSS-C Variant Protocols (SSI Master)

Introduction

The SSI Master function block provides predefined protocols that can be used for BiSS BP1-compatible encoders.

Data format

The following illustration shows the data format of BiSS BP1-compatible encoders.

**Predefined settings**

The following table shows the properties that are fixed to protocol-specific values. These predefined settings are then read-only in the Properties Browser.

Property	BiSS-C variant 0-12	BiSS-C variant 12-12	BiSS-C variant 24-12	BiSS-C variant 0-24	BiSS-C variant 12-24	BiSS-C variant 24-24
Transmission length	20	32	44	32	44	56
Single-turn bit width	12	12	12	24	24	24
Single-turn start bit position	1	13	25	1	13	25
Multiturn bit width	0	12	24	0	12	24
Multiturn start bit position	–	1	1	–	1	1
Error flag position	0	0	0	0	0	0
Inverted error flag position	13	25	37	25	37	49
Warning flag position	0	0	0	0	0	0
Inverted warning flag position	14	26	38	26	38	50
Sensor-specific diagnostic bit width	0	0	0	0	0	0
Sensor-specific diagnostic start bit position	–	–	–	–	–	–
CRC configuration	Inverted	Inverted	Inverted	Inverted	Inverted	Inverted
CRC sum bit width	6	6	6	6	6	6
CRC sum start bit position	15	27	39	27	39	51
CRC-checked data bit width	14	26	38	26	38	50
CRC-checked data start bit position	1	1	1	1	1	1
CRC polynomial	67	67	67	67	67	67
CRC initial value	0	0	0	0	0	0

Configuring the Standard Features (SSI Master)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital In/Out 9	Flexible In/Out 1	Further Information
Interface type	Ground-based	Differential	Specifying the Circuit Type for Voltage Input Signals on page 116

To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital In/Out 9](#) on page 1590
- [Flexible In/Out 1](#) on page 1608

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (SSI Master)

SCALEXIO Hardware Dependencies (SSI Master)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Characteristics		Digital In/Out 9	Flexible In/Out 1
Hardware		DS6121 Multi-I/O Board	
Supporting master APU functionality ¹⁾		✓	
Clock (Out)	Frequency	<ul style="list-style-type: none"> ▪ SSI protocol: 45 kHz ... 2 MHz ▪ BiSS-C protocol: 45 kHz ... 10 MHz 	
	Output voltage range	<p>Depending on high side reference voltage:</p> <ul style="list-style-type: none"> ▪ +3.3 V (typ.) at $RL = \infty$ ▪ +5.0 V (typ.) at $RL = \infty$ 	<ul style="list-style-type: none"> ▪ +1.5 ... +3.3 V ▪ +3.0 V (typ.) at $RL = \infty$
	Supported interface type for digital outputs	Ground-based	Differential
	High side reference voltage	+3.3 V and +5 V	–
	Signal termination	<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable 	–
	Common mode voltage range ²⁾	–	+2 V (typ.) ... +3 V (max.)
Data (In)	Input voltage range	0 V ... +60 V	± 25 V
	Supported interface type for digital inputs	Ground-based	Differential
	Threshold voltage range	0 V ... +12 V	± 125 mV (typ.)
	Hysteresis (for threshold)	600 mV (typ.)	250 mV (typ.)
	Resolution (for threshold)	8 bit	–
	Signal termination	–	<ul style="list-style-type: none"> ▪ 120 Ω with 5 nF in series (parallel termination) ▪ Switchable
Circuit diagrams		Digital In/Out 9 on page 1590	Flexible In/Out 1 on page 1608
Required channels		2	

¹⁾ The master APU functionality provides the angular position values to be used by other function blocks.

²⁾ Common mode voltage (V_{CM}) = $0.5 \cdot (V_{Signal} + V_{Inverted\ Signal})$ measured to the ground potential of the dSPACE real-time hardware.

General limitations

The channels for the clock and data signals must be located on the same I/O board.

More hardware data

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Digital Incremental Encoder Out

Where to go from here

Information in this section

Introduction (Digital Incremental Encoder Out).....	980
Overviews (Digital Incremental Encoder Out).....	984
Configuring the Function Block (Digital Incremental Encoder Out).....	988
Hardware Dependencies (Digital Incremental Encoder Out).....	992

Introduction (Digital Incremental Encoder Out)

Where to go from here

Information in this section

Introduction to the Function Block (Digital Incremental Encoder Out).....	980
Basics on the Generated Encoder Signals.....	981

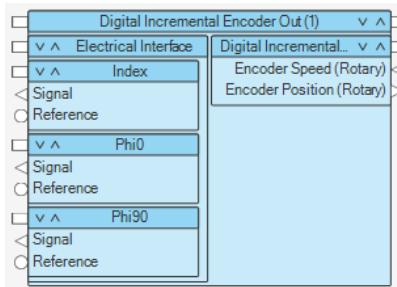
Introduction to the Function Block (Digital Incremental Encoder Out)

Function block purpose

The Digital Incremental Encoder Out function block simulates the signals provided by a rotary or linear incremental encoder. The function block supports the generation of three digital output signals (Phi0, Phi90 and Index).

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Simulating rotary and linear incremental encoders.
- Generating the encoder signals (Phi90, Phi0, Index) on the basis of speed values provided by the behavior model.
- Providing the position value of the simulated encoder to the behavior model.

Supported channel types

The Digital Incremental Encoder Out function block type supports the following channel types:

	SCALEXIO				MicroAutoBox III
Channel type	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	-
Hardware	DS2680	DS6101	DS6202	DS2621	-

Basics on the Generated Encoder Signals

Provided signals

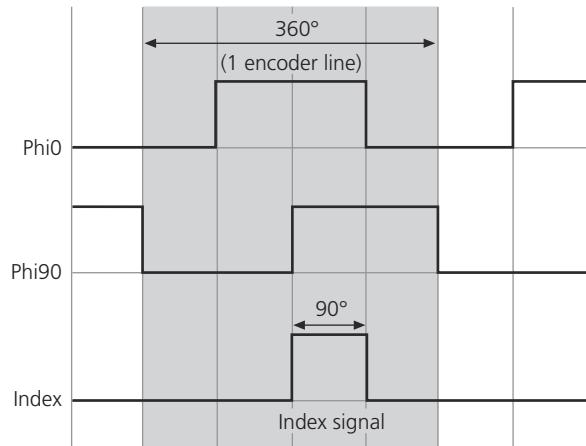
Incremental encoders are encoders which provide a series of pulses (increments), which are used to analyze position changes. Incremental encoders provide two sensor signals which are shifted by 90° against each other. This shift is used to detect the direction of movement. The outputs of a digital encoder are the square-wave signals Phi0 and Phi90.

Encoders usually provide a third signal. This is the index signal and can be used to calibrate the position control loop which analyzes the encoder signals.

Encoder lines

Incremental encoders provide the information for analysis in encoder lines. The number of lines per revolution (rotary encoder) or lines per m (linear encoder) determines the resolution of an encoder. The more encoder lines are available, the more precise is the result of the signal measurement.

The following illustration shows the shapes of the generated Phi0 and Phi90 digital signals together with the index signal. The gray-shaded area represents one encoder line. To describe the 90° phase shift of the Phi90 signal, one encoder line is represented by 360°.



The Phi0 and Phi90 signals together provide four edges in one encoder line. Each edge is considered for the analysis of the encoder position. One encoder line is

therefore represented by 4 pulse counts. The current value is stored in an internal position counter on the dSPACE real-time hardware. This method is called fourfold multiplication and is used to increase the accuracy.

The following table shows the relationship between the position values and the internal position counter for one encoder line:

Internal Position Counter	Position Value
0	0/4
1	1/4
2	2/4
3	3/4
4	4/4

Speed and direction updates

The simulated speed of the incremental encoder can be modified from within the behavior model during simulation run time. A new encoder speed value becomes valid immediately. It takes effect before the current encoder signal period has finished.

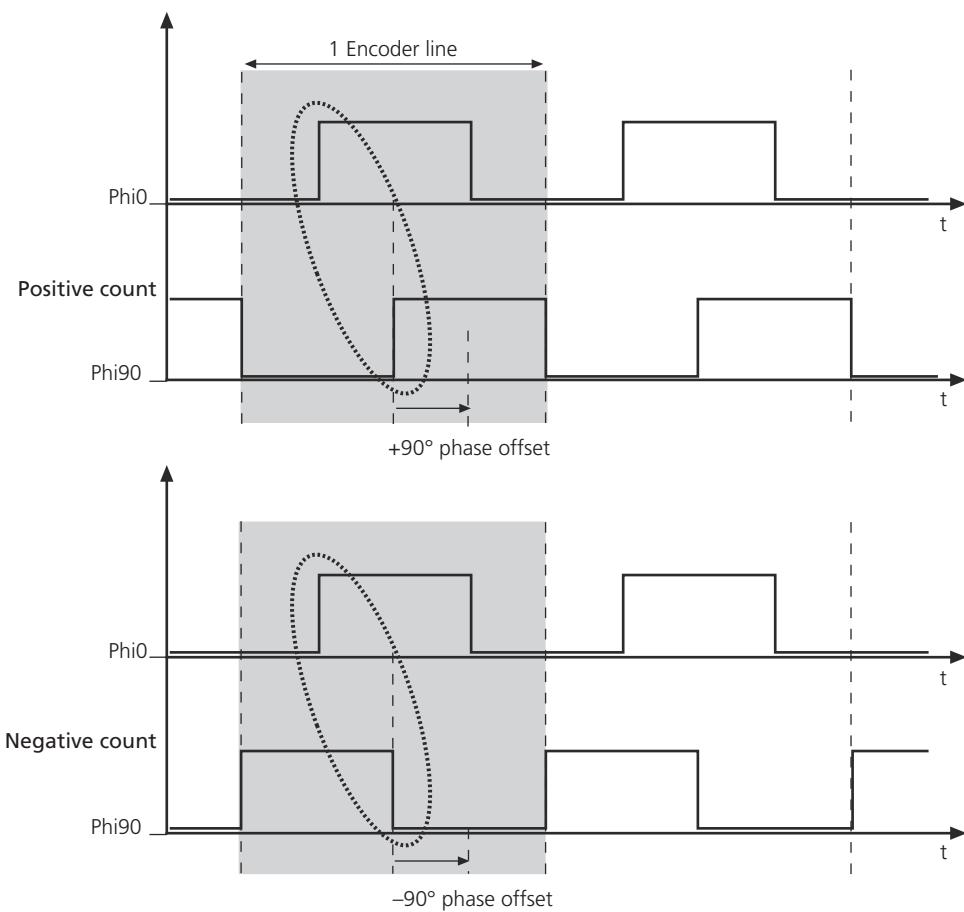
Direction updates are possible on fourfold positions. The direction of rotation (movement) depends on the sign (+ or -) of the speed value. To stop the movement, the speed value must be set to zero.

The possible speed range depends on the assigned real-time hardware. For concrete values, refer to [Hardware Dependencies \(Digital Incremental Encoder Out\)](#) on page 992.

Position measurement

The Digital Incremental Encoder Out function block determines the current position of the simulation.

The internal position counter (on the dSPACE real-time hardware) is incremented or decremented at every edge of the generated encoder signals. If two successive edges from the Phi0 and Phi90 signals have the same polarity, the counter is incremented by 1 (positive count). If two successive edges have a different polarity, the counter is decremented by 1 (negative count).



The counter increments and decrements can be interpreted as movement to the left or right. The simulated position is calculated and returned to the behavior model at every sample step, refer to the Encoder position function port.

Overviews (Digital Incremental Encoder Out)

Where to go from here

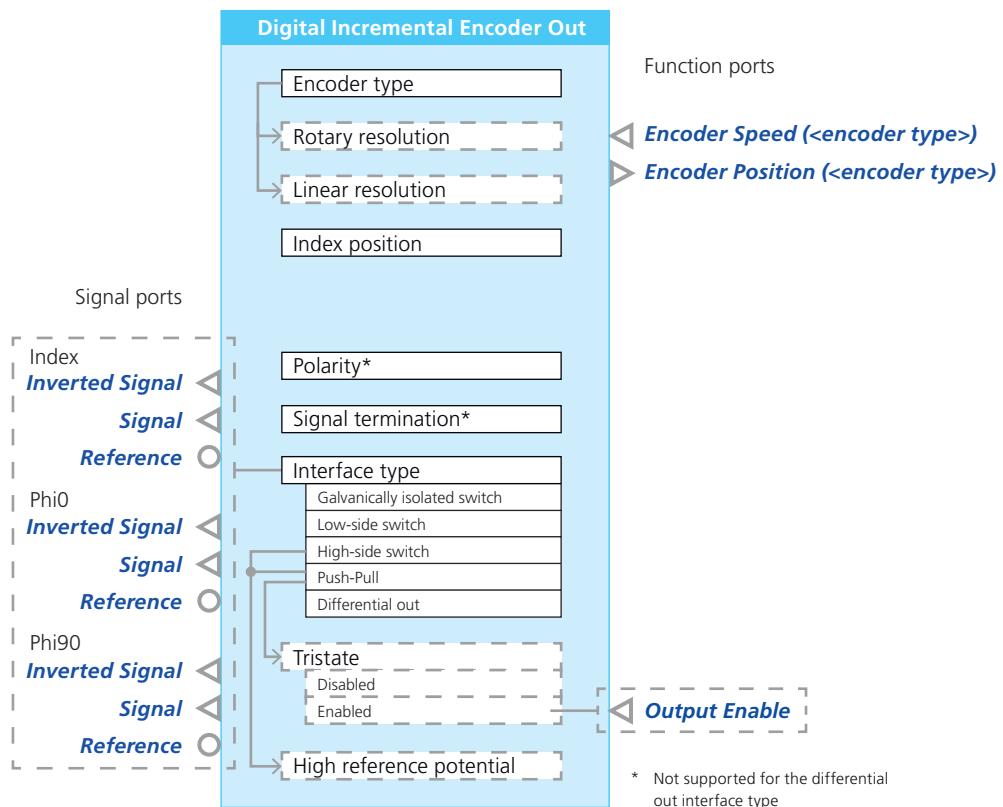
Information in this section

- | | |
|-----------------------------------------------------------------------------------------------|-----|
| Overview of Ports and Basic Properties (Digital Incremental Encoder Out)..... | 984 |
| Overview of Tunable Properties (Digital Incremental Encoder Out)..... | 987 |

Overview of Ports and Basic Properties (Digital Incremental Encoder Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Encoder Speed (<encoder type>)

This function import lets you define a speed value for the encoder from within the behavior model. The direction of rotation/movement depends on the sign (+ or -) of the speed value.

Value range	<ul style="list-style-type: none"> ▪ Depends on the settings of Encoder type property and the resolution (Rotary resolution or Linear resolution property): <ul style="list-style-type: none"> ▪ Rotary encoder: -161563,64 % ... +161563,64 % ▪ Linear encoder: -448,78 m/s ... +448,78 m/s ▪ 0: Stop of the rotation/movement. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the value range.
Dependencies	-

Encoder Position (<encoder type>)

This function outport writes the current position of the simulated encoder to the behavior model. The position value can be used for a position control loop within the model, for example, to start the closed-loop simulation with a certain encoder position.

Value range	<p>Depends on the settings of Encoder type property:</p> <ul style="list-style-type: none"> ▪ Rotary encoder: 0° ... 360°. Above 360° the position counter restarts from 0. ▪ Linear encoder: -Position_{max} ... +Position_{max} (in meter). The range depends on the value specified at the Linear resolution property. Positive and negative values are possible.
Dependencies	-

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Incremental Encoder Out) on page 992.
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Incremental Encoder Out) on page 992.
Dependencies	–

Inverted Signal

This signal port and the **Signal** signal port together represent the electrical connection points of a logical signal with two complementary potentials carrying the information in the voltage difference between these signals. Both signal ports (**Signal** and **Inverted Signal**) have identical electrical characteristics against the ground potential of the dSPACE real-time hardware.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Incremental Encoder Out) on page 992.
Dependencies	Available only if the Interface type property is set to Differential out .

Reference

This signal port is a reference port and provides the reference signal for the **Signal** signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Digital Incremental Encoder Out) on page 992.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Digital Incremental Encoder Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–	–
Function Ports			
Initial switch setting (Test Automation)	–	✓	
Initial substitute value (Test Automation)	–	✓	
Electrical Interface			
Polarity	✓	–	

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Digital Incremental Encoder Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (Digital Incremental Encoder Out).....	988
Configuring Standard Features (Digital Incremental Encoder Out).....	989

Configuring the Basic Functionality (Digital Incremental Encoder Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Choosing the encoder type (rotary or linear) and specifying encoder resolution
- Specifying the position of the index signal

Choosing the encoder type

You can choose to simulate a rotary or a linear incremental encoder.

The main difference between the two is the generation of the index signal:

- The index of the rotary encoder is generated periodically with each rotation.
- The linear encoder generates an index signal at a certain position only.

The properties for both encoder types are identical. However, the physical ranges and physical units of some properties and function ports are different. For example: The unit of resolution is lines/revolution for rotary encoders and μm for linear encoders.

Specifying encoder resolution

The resolution value defines:

- The amount of encoder lines which are generated per revolution (rotary encoder).
- The gap (in μm) between two generated encoder lines (linear encoder)

The more encoder lines are available (results in higher resolution), the more precise is the result for signal analysis.

Specifying position of index signal

You can generate an index signal, for example, to simulate a switching point. The position of the index signal can be configured with the Index position property. The index of the rotary encoder is generated periodically with each

rotation, while the linear encoder generates an index signal at a certain position only. When the rotary encoder is used, the value of the internal position counter (on the dSPACE real-time hardware) is reset with each new rotation. The possible range is therefore 0 ... 360°.

If you want to generate an index signal at a specific encoder line, you can use the following formula to calculate the angle (rotary encoder) or the position (linear encoder) which corresponds to the encoder line:

$$\text{Angle } [{}^\circ] = \text{Number of encoder line} \times \frac{360^\circ}{\text{Resolution [lines/revolution]}}$$

$$\text{Position [m]} = \text{Number of encoder line} \times \frac{\text{Resolution } [\mu\text{m}]}{10^6}$$

The specified position is internally rounded to an entire encoder line. However, the index signal is generated between two fourfold positions (between 180° and 270°) and not at the starting point of the encoder line.

Limitations

- Only one index position is provided per encoder.
- There is no generation of events or interrupts when an index signal is detected.

Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

Limitation A resistor connected in series to the output driver is supported only by the Digital In/Out 5 channel type. However, signal termination is not supported if you use the Differential out interface type. For a circuit diagram, refer to [Digital In/Out 5](#) on page 1586.

Configuring Standard Features (Digital Incremental Encoder Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and

important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the **Conflicts Viewer**. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull ▪ Differential out 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<p>Depending on the channel type, refer to:</p> <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109 ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112 ▪ Specifying Digital Output Signals (Flexible Out 1) on page 98
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.				–
Polarity of output signals	✓	✓	<ul style="list-style-type: none"> ▪ ✓ ▪ Not supported for the differential out interface type. 	✓	–
Channel multiplication	✓	–	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports					
Trigger level of electronic fuse	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 1 on page 1575](#)
- [Digital Out 3 on page 1577](#)

- [Digital In/Out 5](#) on page 1586
- [Flexible Out 1](#) on page 1604

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	Note that initialization behavior differs from the standard as follows: The specified initial value for the encoder position is internally rounded to an entire line position. Fourfold positions (1/4, 2/4 and 3/4 encoder lines) cannot be used as initial values.
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Digital Incremental Encoder Out)

SCALEXIO Hardware Dependencies (Digital Incremental Encoder Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	Digital Out 1	Flexible Out 1
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board
Output current range	<ul style="list-style-type: none"> ▪ 0 ... +80 mA_{RMS} for 1 channel ▪ 1.792 A_{RMS} for 28 channels (channel multiplication) 	<ul style="list-style-type: none"> ▪ 0 ... +40 mA_{RMS} for 3 channels ▪ 96 mA_{RMS} for 9 channels (channel multiplication)
High side reference voltage	+5 V ... +60 V	-60 V ... +60 V
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ Low-side switch ▪ High-side switch ▪ Push-pull
	In push-pull configuration the outputs can be set to high impedance (tristate) at run time.	
Configurable fuse	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Signal termination	–	–
Circuit diagram	Digital Out 1 on page 1575	Flexible Out 1 on page 1604
Required channels	3 (additional channels are required for current enhancement and operation in push/pull mode.)	

Channel type	Digital Out 3	Digital In/Out 5
Hardware	DS6101 Multi-I/O Board	DS6202 Digital I/O Board
Output current range	<ul style="list-style-type: none"> ▪ 0 ... +140 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	0 ... ±40 mA (each channel)
High side reference voltage	+5 V ... +60 V	<ul style="list-style-type: none"> ▪ +3.3 V and +5 V (switchable) ▪ +3.3 V (fix) for the differential out interface type
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull ▪ Differential out
	In push-pull configuration the outputs can be set to high impedance (tristate) at run time.	
Configurable fuse	–	–
Signal termination	–	▪ 68 Ω (serial termination)

Channel type	Digital Out 3	Digital In/Out 5
		<ul style="list-style-type: none"> ▪ Switchable ▪ Not supported for the differential out interface type
Circuit diagram	Digital Out 3 on page 1577	Digital In/Out 5 on page 1586
Required channels	3	3 (6 channels are required to operate the outputs using the differential out interface type.)

General limitations

Channel multiplication across several I/O boards is not supported.

DS2621 Signal Generation Board The following additional restrictions apply to channel multiplication and push/pull modes:

- Push/pull mode and channel multiplication (caused by current enhancement) are not possible at the same time because the maximum number of channels of the DS2621 board is not sufficient.
- When the push/pull mode is used, the following restrictions apply:
 - Every logical signal requests two channels (primary and auxiliary) on the real-time hardware to be assigned to the function block. These channels must be adjacent to one another on an I/O board. Example: primary channel = n, auxiliary channel = n + 1.
 - Channel 10 cannot be used as the primary channel because using channel 1 as the next adjacent channel is not supported.

DS6101 Multi-I/O Board Channel multiplication is not supported in general.

DS6202 Digital I/O Board The following limitations apply to the DS6202 Digital I/O Board:

- Channel multiplication is not supported in general.
- Only 6 Digital Incremental Encoder Out function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Wheelspeed Out

Where to go from here

Information in this section

Introduction (Wheelspeed Out).....	996
Overviews (Wheelspeed Out).....	1005
Configuring the Function Block (Wheelspeed Out).....	1012
Hardware Dependencies (Wheelspeed Out).....	1018

Information in other sections

Demo Model of a Wheelspeed Out Interface (Model Interface Package for Simulink - Modeling Guide 

Introduction (Wheelspeed Out)

Where to go from here

Information in this section

Introduction to the Function Block (Wheelspeed Out).....	996
Basics on Active Wheel Speed Sensors.....	997
Encoding the Data Protocol with the Wheelspeed Out Function Block.....	1000
Notes on the Data Protocol for Fast Rotating Wheels.....	1002
Notes on Simulating Non-Rotating Wheels.....	1003

Introduction to the Function Block (Wheelspeed Out)

Function block purpose

The Wheelspeed Out function block type simulates the signals provided by an active wheel speed sensor.

Unlike passive sensors, active wheel speed sensors have the advantage of measuring a wide range of rotational speeds, even very low speeds. The output signal of the function block is very flexible and covers different sensor types.

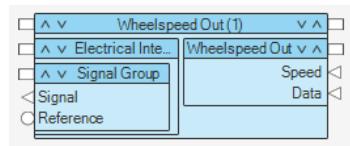
Typical applications are:

- Anti-lock braking systems
- Electronic stability control
- Navigation systems, tachometers, odometers, etc.

Passive sensors cannot be simulated with the Wheelspeed Out function block.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generating output signals based on speed pulses and/or data pulses.
- Specifying the characteristics of speed and data pulses with the function block and also from the behavior model.
- Supporting the simulation of intelligent wheel speed sensors that use data protocols.
- Supporting the simulation of non-rotating wheels by generating a modified output signal.
- Providing the value of the pitch position evaluated by an internal pitch counter to the behavior model.
- Adding noise to the output signal via a noise generator.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Wheelspeed Out function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III
Channel type	Analog Out 4	Analog Out 9	Flexible Out 1	-
Hardware	DS2680	DS6101	DS2621	-

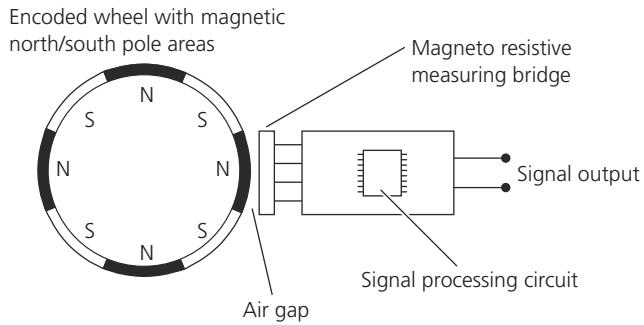
Basics on Active Wheel Speed Sensors

Principle of measuring rotational speed

There are two different types of active wheel speed sensors:

- The first type is based on a ferromagnetic wheel (e.g. cogwheel), which causes a change in the magnetic field at the sensor element. The sensor element contains a permanent magnet for this behavior.
- The second type is based on a magnetically encoded wheel, containing north- and south-pole areas.

Both versions use sensor elements which are based on the anisotropic magneto-resistive effect (AMR). The sensor element detects the changing of the magnetic field and a signal processing circuit generates a pulse at the output. A detected change in the magnetic field is called a pitch. In the illustration below, the magnetic field changes at every north/south and south/north pole transition.

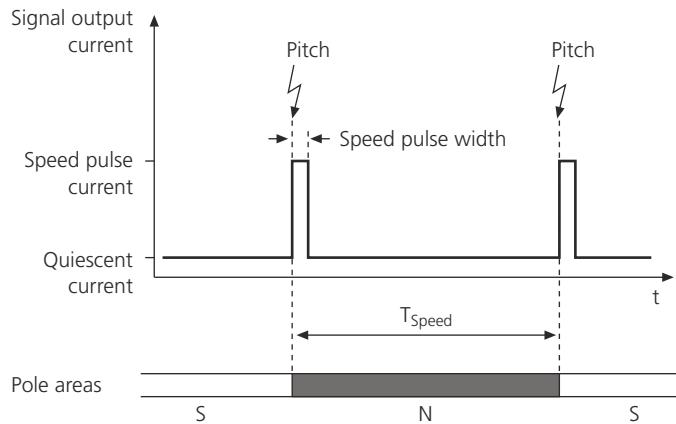


Characteristics of the output signal

The output signal of active wheel speed sensors provides a speed pulse signal, whose frequency is proportional to the pitches per second. Some active wheel speed sensors have auxiliary functionality. These sensors are called intelligent wheel speed sensors. They collect auxiliary data in addition to the speed and transmit it to the connected device (e.g. ECU).

Output signal of active wheel speed sensors without auxiliary functionality Active wheel speed sensors without auxiliary functionality do not have a data protocol.

The illustration below shows the relation of the signal output and an encoded wheel.



Term	Description
T_{Speed}	Time between two neighboring pitches.
Speed	Speed is the inverse of the time T_{Speed} (Speed = 1 / T_{Speed}).
Pitch	A detected change in the magnetic field.

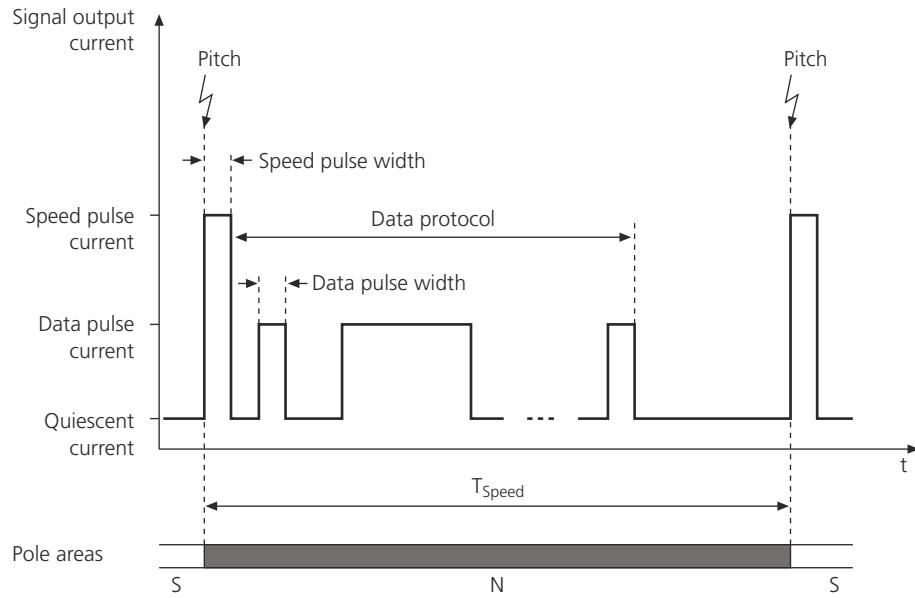
Output signal of intelligent wheel speed sensors Intelligent wheel speed sensors have a data protocol to transmit the auxiliary data.

Typical auxiliary data:

- Information on whether the wheel rotates
- Information on the direction of rotation

- Information on the air gap between the wheel and the magneto resistive measuring bridge
- Parity bits
- Status information

The illustration below shows the relation of the signal output and an encoded wheel. Auxiliary data is transmitted between two speed pulses via a data protocol.



Note

The first data pulse is output directly after the speed pulse.

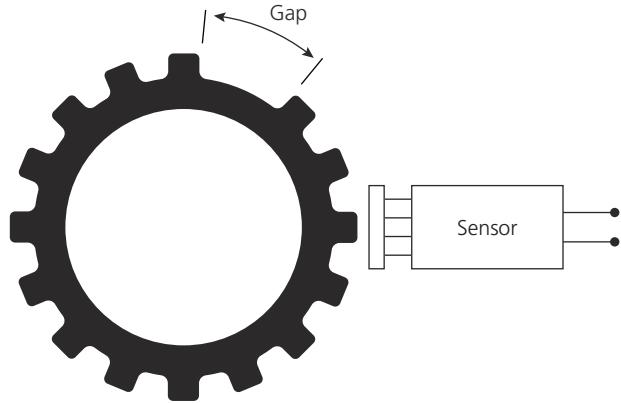
In the illustration above, the first data pulse is a low-level pulse and used as a gap to separate the speed and data output.

Term	Description
T_{Speed}	Time between two neighboring pitches.
Speed	Speed is the inverse of the time T_{Speed} (Speed = 1 / T_{Speed}).
Pitch	A detected change in the magnetic field.
Data protocol	The data protocol includes all data pulses. It is output directly after the speed pulse. For more information,

Term	Description
	refer to Encoding the Data Protocol with the Wheelspeed Out Function Block on page 1000.

Predefined gaps in the coding of the sensor wheel

Wheel speed sensors with predefined gaps in the coding of the sensor wheel are used to define an index position. For example, crankshaft sensors commonly use this method.



For specifying gaps in the coding, refer to [Configuring the Basic Functionality \(Wheelspeed Out\)](#) on page 1012.

Encoding the Data Protocol with the Wheelspeed Out Function Block

Context

The Wheelspeed Out function block has a configurable output signal that lets you simulate different kinds of data protocols. You can also simulate failures in the data protocol.

Encoding the data protocol

You have to encode the whole data protocol in the behavior model according to the sensor that is simulated. This has the advantage that you can easily use different coding methods of different sensor types.

The initial and stop values of a Wheelspeed Out function block are also specified according to the data protocol encoding.

The Wheelspeed Out function block type supports a maximum of 64 data pulses to encode the data protocol. The encoding includes:

- The number of data bits and their meanings (logical coding)
- How each data bit is transmitted to the external device via data pulses (physical coding)
- The specification of a gap to separate the speed and data output

Whether you need to specify a gap depends on the wheel speed sensor that is simulated.

Specifying the logical coding Data bits transmit the auxiliary data of an intelligent wheel speed sensor to the connected ECU.

The logical coding specifies:

- The number of data bits used
- How to read the data bits

For example, a logical coding might specify that a logical 1 for data bit 4 stands for wheels rotating clockwise.

You have to implement the logical coding of the sensor that is simulated in the behavior model.

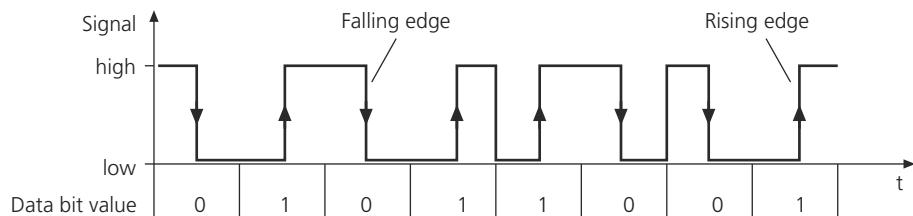
Specifying the physical coding The Wheelspeed Out function block type uses data pulses to physically code the data bits. The specification of the sensor that is simulated defines the physical coding, i.e., how each data bit is physically transmitted to the ECU.

The following binary coding methods can basically be represented by the data pulses:

- Physical codes which modulate the phase to represent the binary information. The data bits are coded by the edges of the data pulses. This is a common coding method for an intelligent wheel speed data protocol.

For example, a rising data pulse edge stands for a binary 1, a falling data pulse edge stands for a binary 0. This code is known as Manchester code. The illustration below shows you a sequence of data bits which is coded by the Manchester code.

Manchester code

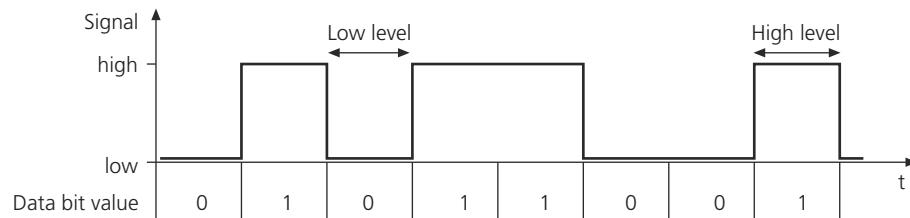


Rising edges are two data pulses, a low followed by a high data pulse amplitude. Falling edges are these two data pulses in reverse order. You need two data pulses for one data bit.

- Physical codes which modulate the amplitude to represent the binary information. The data bits are coded by the amplitude level of the data pulses. For example, a high level stands for a binary 1, a low level stands for a binary 0. This code is known as Non Return to Zero code (NRZ code). You need one data pulse for each data bit.

The illustration below shows you a sequence of data bits which is coded by the NRZ code.

NRZ code



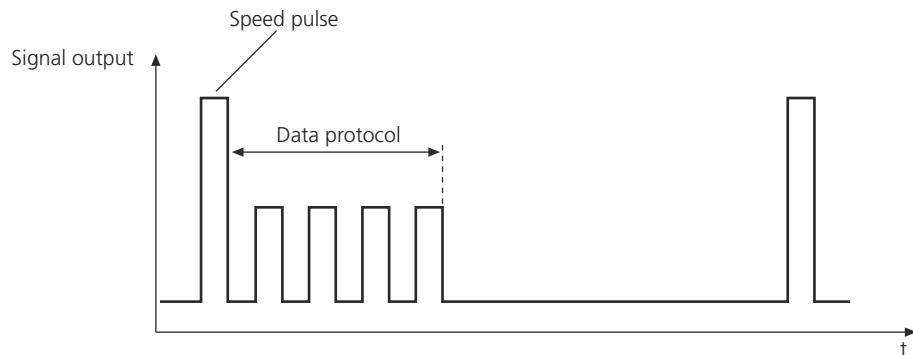
Specifying a gap to separate the speed and data output You must set the first/last data pulse(s) to 0 if you need a gap before and/or after the data output. The duration of the gap depends on the data pulse width and the number of pulses.

For a demo model, refer to [Demo Model of a Wheelspeed Out Interface \(Model Interface Package for Simulink - Modeling Guide](#)).

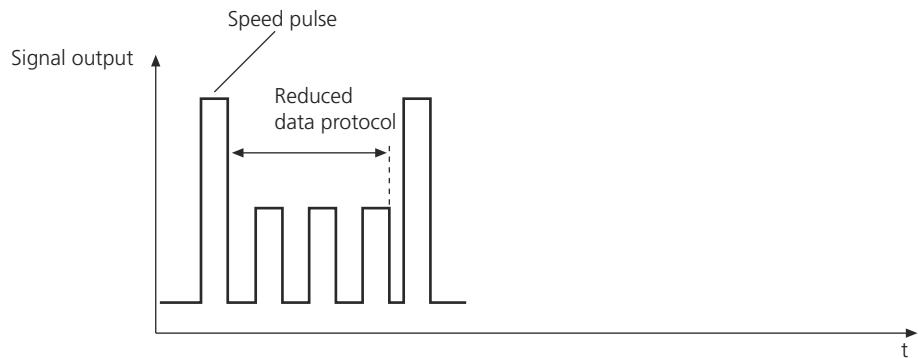
Notes on the Data Protocol for Fast Rotating Wheels

Effects of high speed values on the data protocol

Intelligent wheel speed sensors transmit the data protocol between two speed pulses.



The higher the speed value is, the shorter the time between two speed pulses is. At a specific speed, there is not enough time to transmit the data protocol. If the speed reaches this point, intelligent wheel speed sensors reduce the data protocol to extend the measurable speed range.



You have to implement this functionality in the behavior model for the following reasons:

- The data protocol might use the last data bit as a checksum and the sensor reduce some data bits in the middle.
- The wheel speed sensor might use a special data protocol during this operating condition.
- The wheel speed sensor might reduce more data bits than needed.

If the reduced data protocol is still too long, the data protocol will be cut automatically by the *Wheelspeed Out* function block.

For a demo model, refer to [Demo Model of a Wheelspeed Out Interface \(Model Interface Package for Simulink - Modeling Guide](#)

Notes on Simulating Non-Rotating Wheels

Detecting standstill mode

Active wheel speed sensors have a minimum measurable speed. If the measured speed is lower than the minimum speed, common sensors generate a modified output signal.

A modified output signal has the following advantages:

- It highlights a standstill wheel.
- It indicates that the sensor is in working order.
- It sends sensor status information to the ECU independent from the speed.

You can specify the minimum measurable speed with the *Standstill detection speed* property. If the function speed value is lower than the specified *Standstill detection speed*, the *Wheelspeed Out* function block switches to the standstill mode. The function block generates a modified output signal in this mode.

Modified output signal in the standstill mode

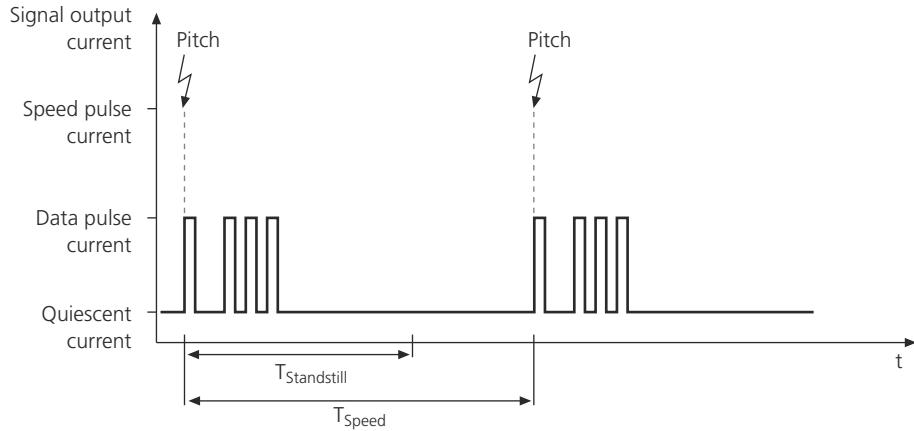
The *Wheelspeed Out* function block type modifies the output signal in the standstill mode as follows:

- The speed pulse amplitude goes from *Speed pulse current* level down to *Data pulse current* level to highlight the standstill mode.

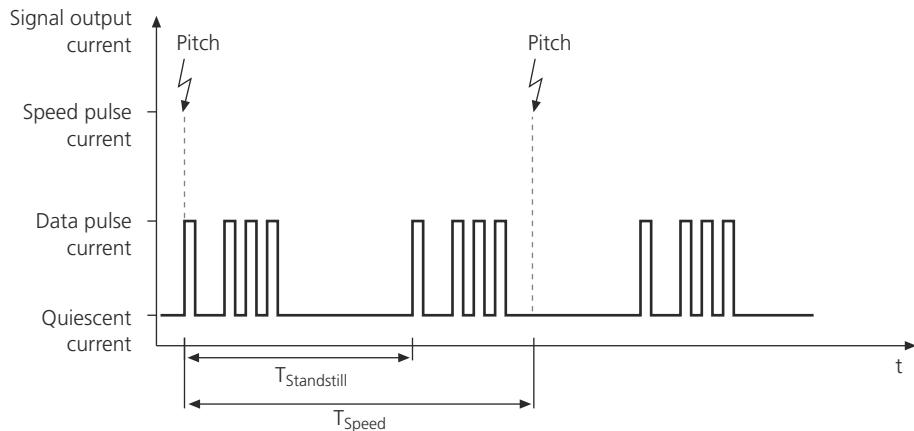
- The Standstill behavior property provides two options for the repetition rate:
 - Pitch synchronous transmission (default): The output signal is generated at the rate specified by the Speed function port value.
 - Periodic transmission: The output signal is generated periodically at the rate specified by the Standstill detection speed property. The signal always starts with a speed pulse, independently of the Speed function port value.
- If Standstill detection speed is configured to 0.0 pitches/s, the signal outport generates no data protocol while the Speed function port value is 0.0 pitches/s.

The following illustration shows you the difference between these options.

Pitch synchronous transmission



Periodic transmission



To detect standstill, the time between two neighboring pitches must be greater than the standstill detection time $T_{Standstill} = 1 / \text{Standstill detection speed}$.

Note

If the intelligent wheel speed sensor provides a special data protocol while standstill is detected, you have to specify it in the behavior model.

Overviews (Wheelspeed Out)

Where to go from here

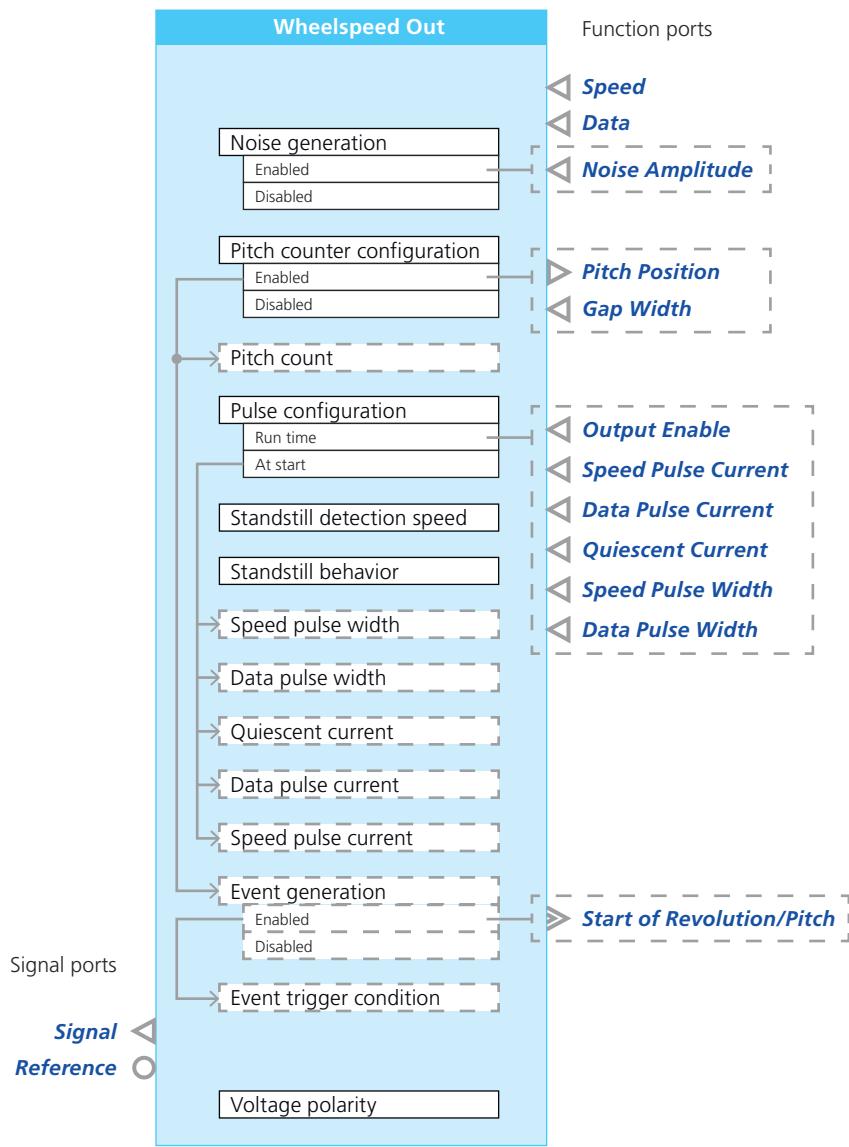
Information in this section

Overview of Ports and Basic Properties (Wheelspeed Out).....	1005
Overview of Tunable Properties (Wheelspeed Out).....	1010

Overview of Ports and Basic Properties (Wheelspeed Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Speed**

This function import defines the speed that the simulated sensor outputs.

Value range	<ul style="list-style-type: none"> 0 pitches/sec ... 50000 pitches/sec If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range.
Dependencies	–

Data

This function import defines a 64-bit serial data protocol from within the behavior model. The data protocol is sent between two speed pulses. If the sensor that is simulated does not provide a data protocol, the incoming values must be set to 0 in the behavior model.

Value range	{0; 0; 0; 0; 0; 0; 0} ... {255; 255; 255; 255; 255; 255; 255; 255}
Dependencies	-

The first data pulse is output directly after the speed pulse. You can specify a gap by low-level data pulses.

Data protocol notation:

- All 64 data pulses of the data protocol are written in a vector with 8 elements.
- The decimal number of an element is a binary number with 8 digits, each of which stands for a pulse. For example:
 $\{90;...\}_{\text{decimal}} = \{01011010;...\}_{\text{binary}}$.
A 1 represents the pulse level that is specified by the Data pulse current property. A 0 represents the pulse level that is specified by the Quiescent current property. The least significant digit represents the first output pulse, the most significant digit represents the last output pulse.
- The first element represents the first 8 output pulses, the second element the next 8 pulses, etc. in the decimal notation.
- There must be 8 vector elements, unused data pulses must be set to 0.

Noise Amplitude

This function import defines the amplitude of the noise generator from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... I_{\max} (in Ampere). ▪ The range depends on the follows: <ul style="list-style-type: none"> ▪ It depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the configurable range. ▪ 0 A: Noise signal is not added to the output signal.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Noise generation property is set to Enabled. ▪ Supported only for Flexible Out 1 channel type.

Pitch Position

This function outport provides the current value of the function block's internal pitch counter. Skipped pitches to simulate a gap in the coding of the sensor wheel are also counted.

Value range	<ul style="list-style-type: none"> ▪ 0 pitches ... 65,535 pitches ▪ The range depends on the Pitch count property. ▪ If the value of the pitch counter reaches the value of the Pitch count property, the pitch counter is reset to zero.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Pitch counter configuration property is set to Enabled.

Gap Width

This function import specifies the gap width from within the behavior model to simulate a sensor wheel with a gap in the coding. The gap width is defined as the number of pitches to be skipped. The resulting gap always begins with the first pitch when the pitch counter is zero.

Value range	<ul style="list-style-type: none"> ▪ 0 pitches ... 255 pitches ▪ The range depends on the Pitch count property. ▪ 0 pitches: A sensor wheel without a gap in the coding is simulated. ▪ If the value of the Gap Width port is greater than or equals the value of the Pitch count property, no pitches are generated.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Pitch counter configuration property is set to Enabled.

Output Enable

This function import enables the signal generation from within the behavior model. If the signal generation is disabled, no signals are generated and the output current is 0 A. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Pulse configuration property is set to Run time.

Speed Pulse Current

This function import specifies the speed pulse current of the signal outport from within the behavior model. The value is given by the sensor specification of the simulated wheel speed sensor type.

In standstill mode, speed pulses are generated with the data pulse current.

Value range	<ul style="list-style-type: none"> ▪ $I_{\min} \dots I_{\max}$ (in Ampere). ▪ The value range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018.
Dependencies	Available only if the Pulse configuration property is set to Run time.

Data Pulse Current

This function import specifies the data pulse current of the signal outport from within the behavior model. The value is given by the sensor specification of the simulated wheel speed sensor type.

Value range	<ul style="list-style-type: none"> $I_{min} \dots I_{max}$ (in Ampere). The value range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018.
Dependencies	Available only if the Pulse configuration property is set to Run time.

Quiescent Current

This function import specifies the quiescent current of the signal output from within the behavior model. The value is given by the sensor specification of the simulated wheel speed sensor type.

Value range	<ul style="list-style-type: none"> $I_{min} \dots I_{max}$ (in Ampere). The value range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018.
Dependencies	Available only if the Pulse configuration property is set to Run time.

Speed Pulse Width

This function import specifies the speed pulse width from within the behavior model. The value is given by the sensor specification of the simulated wheel speed sensor type.

Value range	<ul style="list-style-type: none"> $t_{min} \dots t_{max}$ (in seconds) The value range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018.
Dependencies	Available only if the Pulse configuration property is set to Run time.

Data Pulse Width

This function import specifies the data pulse width from within the behavior model. The value is given by the sensor specification of the simulated wheel speed sensor type.

Value range	<ul style="list-style-type: none"> $t_{min} \dots t_{max}$ (in seconds) The value range depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018.
Dependencies	Available only if the Pulse configuration property is set to Run time.

Start of Revolution/Pitch

This event port provides an I/O event each time a new revolution starts or a pitch is detected, according to the setting of the Event trigger condition property.

Value range	–
Dependencies	Available only if the Pitch counter configuration property and the Event generation property are set to Enabled.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Wheelspeed Out) on page 1018.
Dependencies	–

Overview of Tunable Properties (Wheelspeed Out)

Tunable properties

Tunable parameters can be accessed during run time, and the value of each parameter can be changed in the experiment software.

The following table shows all the tunable parameters available for the function block type if you set the Pulse configuration property to At start:

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Event trigger condition	✓	–
	Pitch count	✓	–
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Electrical Interface			
Speed pulse width	✓	–	
Data pulse width	✓	–	
Standstill detection speed	✓	–	
Standstill behavior	✓	–	
Quiescent current	✓	–	
Data pulse current	✓	–	
Speed pulse current	✓	–	
Voltage polarity	✓	–	

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

The following table shows all the tunable parameters available for the function block type if you set the Pulse configuration property to Run time:

	Parameter	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
Event trigger condition	✓	–	
Pitch count	✓	–	
Function Ports			
Initial switch setting (Test Automation)	–	✓	
Initial substitute value (Test Automation)	–	✓	
Electrical Interface			
Speed pulse width	–	✓	
Data pulse width	–	✓	
Standstill detection speed	✓	–	
Standstill behavior	✓	–	
Quiescent current	–	✓	
Data pulse current	–	✓	
Speed pulse current	–	✓	
Voltage polarity	✓	–	

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Wheelspeed Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (Wheelspeed Out).....	1012
Configuring Standard Features (Wheelspeed Out).....	1016

Configuring the Basic Functionality (Wheelspeed Out)

Overview

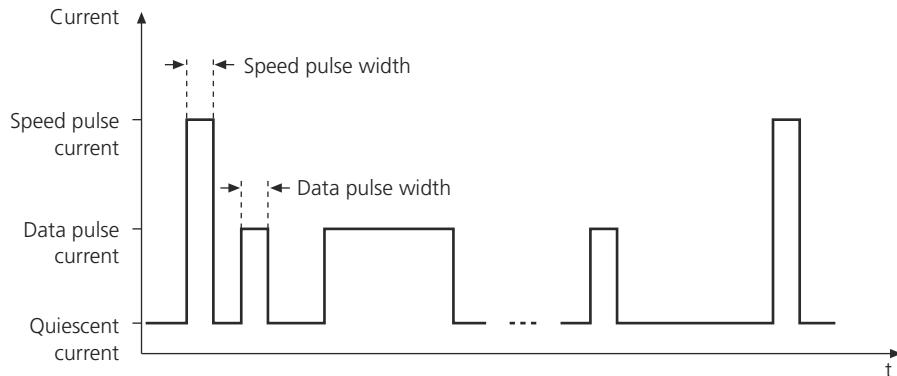
The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the characteristics of the output signal (speed pulse and data pulses).
- Specifying the standstill mode.
- Specifying a gap in the coding of the sensor wheel.
- Providing I/O events.
- Adding noise to the output signal via noise generator.
- Specifying the signal polarity.

Specifying the characteristics of the output signal

The Wheelspeed Out function block simulates active wheel speed sensors with two connecting pins. The speed and data information is serially transmitted on the same wire by modulating the current amplitude. The speed is defined as time interval T_{Speed} and the data information is transmitted via data pulses between two speed pulses.

All the electrical characteristics of the output signal in the illustration below can be specified.



Via the Pulse configuration property you can specify the characteristics as follows:

- **At Start:**

You configure the output signal via properties in the Properties Browser.

The configuration is used directly after the real-time application starts running on the hardware. You can change the configuration via the experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

- **Run time:**

The characteristics are specified via function ports from within the behavior model during run time.

Before you can output simulated sensor signals, you have to enable the signal ports via the Output Enable function port.

Specifying the standstill mode

The Wheelspeed Out function block detects the standstill mode, if the function speed value is lower than the specified value at **Standstill detection speed** property.

In the standstill mode, a modified output signal is generated. The speed pulse amplitude goes from **Speed pulse current** level down to **Data pulse current** level to highlight the standstill mode. The **Standstill behavior** property provides two options for the repetition rate: Pitch synchronous and periodic transmission. For details, refer to [Notes on Simulating Non-Rotating Wheels](#) on page 1003.

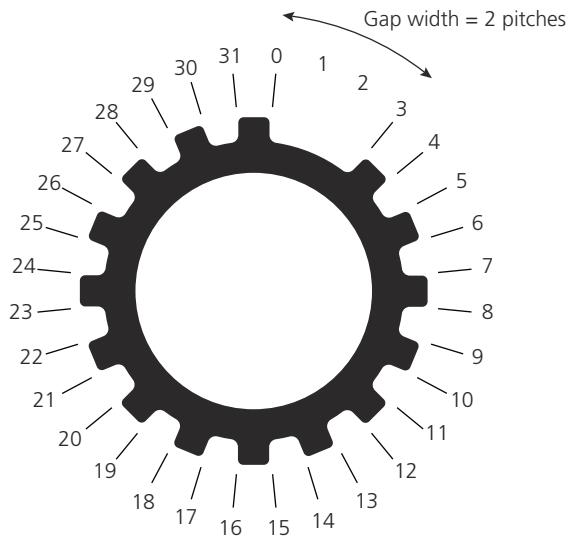
Specifying a gap in the coding of the sensor wheel

Wheel speed sensors with predefined gaps in the coding of the sensor wheel are used to define an index position. For example, crankshaft sensors commonly use this method. The Wheelspeed Out function block simulates these sensors by skipping certain pitches. Therefore, you have to specify the sensor wheel of the simulated sensor before you can specify the gap in the coding.

Note

Skipped pitches are generated with the quiescent current level for the given speed period (no speed and data pulses). Nevertheless, skipped pitches are regarded by the pitch counter and the I/O event generation.

Example of a sensor wheel with a gap in the coding The following illustration is an example for a sensor wheel with a predefined gap. Furthermore, the illustration shows you the pitch numbering used by the pitch counter of the Wheelspeed Out function block.



Specifying the sensor wheel You have to specify the sensor wheel of the simulated sensor as follows:

1. Enable the Pitch counter configuration property.
2. With the Pitch count property, specify the total number of pitches for one revolution including skipped pitches for the gap in the coding.

Keep in mind that each change in the magnetic field results in a pitch. In the example above, the value of the Pitch count property is 32.

Specifying the gap Within the behavior model, you specify the gap width by specifying the number of pitches to be skipped. The function block reads this value via the Gap Width function port and skips the pitches for the specified gap width.

The resulting gap always begins with the first pitch when the pitch counter is zero.

Tip

Defect cogs of a cogwheel can be simulated by changing the gap width during run time.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The Event generation property is available if you enable the Pitch counter configuration property.

Via the Event trigger condition property you can specify the triggering of the I/O event, either when the position of the first pitch is reached (Pitch Position function port changes to 0) or with every pitch. If you trigger the I/O event when the first pitch is reached, you have to specify the number of pitches per revolution including optionally skipped pitches with the Pitch count property.

To use the I/O events in the behavior model, you have to assign them to a task via the Start of Revolution/Pitch property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Adding noise to the output signal

The Wheelspeed Out function block provides an adjustable noise generator as an optional function. This lets you add a block-specific noise signal to the block's output signal. The noisy output signal is generated digitally and is not the result of an analog addition of two different signals. You can change the amplitude of the noise signal in the behavior model. To stop the noise signal at run time, its amplitude must be set to 0.

Limitation Noise generation is supported only for the Flexible Out 1 channel type.

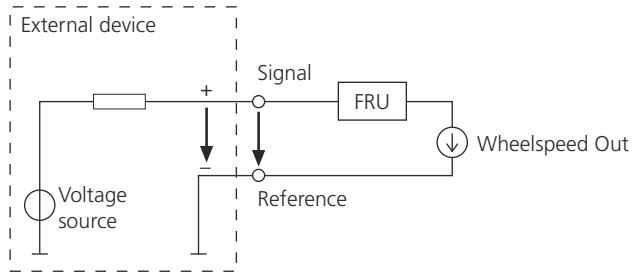
Specifying the signal polarity

The Wheelspeed Out function block is a current sink and must be fed by an external voltage source. Depending on the assigned hardware resource, the voltage polarity affects the correct behavior of the function block.

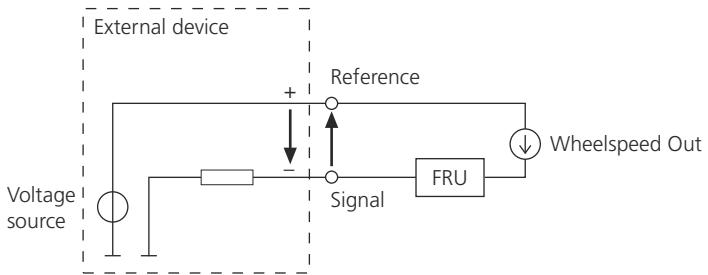
Flexible Out 1 channel type You must specify the voltage polarity of the signal ports with the Voltage polarity property to ensure the correct behavior of the function block.

Note that changing the Voltage polarity property also changes the signal path that the failure routing unit (FRU) is used on. This means that you can use the FRU on either the one or the other signal path by switching the polarity of the voltage. The following illustration shows a connection example with an FRU on the signal path.

Positive polarity



Negative polarity



For details on the FRU, refer to [Electrical Error Simulation Concept \(SCALEXIO Hardware Installation and Configuration\)](#).

Note

The Wheelspeed Out function block adjusts only the current. The feed voltage must be provided from an external source as shown in the illustrations above.

The Voltage polarity property is available in the variable description file. You can change it via experiment software. The changes do not take effect until the real-time application changes from stopped to running.

Analog Out 4, Analog Out 9 channel types The voltage polarity does not affect the functionality of the function block. The behavior is correct if the voltage polarity of the signal is negative as well as positive. Therefore the Voltage polarity property is not configurable for these channel types.

Configuring Standard Features (Wheelspeed Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Analog Out 4	Analog Out 9	Flexible Out 1	Further Information
Channel multiplication	✓	–	✓	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Analog Out 4](#) on page 1555
- [Analog Out 9](#) on page 1557
- [Flexible Out 1](#) on page 1604

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Wheelspeed Out)

SCALEXIO Hardware Dependencies (Wheelspeed Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Analog Out 4	Analog Out 9	Flexible Out 1
Hardware	DS2680 I/O Unit	DS6101 Multi-I/O Board	DS2621 Signal Generation Board
Event generation	✓	✓	✓
Output current range	<ul style="list-style-type: none"> ▪ +0.1 mA ... +30 mA for 1 channel ▪ 0.24 A for 8 channels (channel multiplication) 	+0.1 mA ... +30 mA	<ul style="list-style-type: none"> ▪ +20 µA ... +40 mA for 1 channel ▪ 0.4 A for 10 channels (channel multiplication)
Output voltage range	<p>The following conditions must be met:</p> <ul style="list-style-type: none"> ▪ Range for the lower potential of signal and reference: -2 V ... +18 V ▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V 	<p>The following conditions must be met:</p> <ul style="list-style-type: none"> ▪ Range for the lower potential of signal and reference: -2 V ... +18 V ▪ Voltage difference between signal and reference: ±3.5 V ... ±22 V 	-60 V ... +60 V
Noise generation	–	–	✓
Noise range	–	–	<ul style="list-style-type: none"> ▪ 0 ... +10 mA for 1 channel ▪ 0 ... +100 mA for 10 channels (channel multiplication)
Resolution	14 bit	14 bit	15 bit
Gain error	±0.2 % (typ.)	±0.2 % (typ.)	±0.1 % (typ.)
Offset error	±20 µA (typ.)	±30 µA (typ.)	±10 µA (typ.)
Speed range ¹⁾	0 ... 50,000 pitches/s	0 ... 50,000 pitches/s	0 ... 50,000 pitches/s
Pulse width range	0.01 ms ... 10 ms	0.01 ms ... 10 ms	0.01 ms ... 10 ms
Configurable fuse	–	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS}
Circuit diagram	Analog Out 4 on page 1555	Analog Out 9 on page 1557	Flexible Out 1 on page 1604
Required channels	1 (additional channels are required for current enhancements.)	1	1 (additional channels are required for current enhancements.)

¹⁾ The maximum speed depends on the specified speed pulse width. For the characteristics of the output signal, refer to [Basics on Active Wheel Speed Sensors](#) on page 997.

Synchronous signal update on DS2680 and DS6101	If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware. For details, refer to Optimal Selection of Analog Output Channels on DS2680 and DS6101 on page 1539.
General limitations	Channel multiplication across several I/O boards is not supported. DS6101 Multi-I/O Board Channel multiplication is not supported in general.
More hardware data	DS2680 I/O Unit For more board-specific data, refer to Data Sheet of the DS2680 I/O Unit (SCALEXIO Hardware Installation and Configuration) .
	DS2621 Signal Generation Board For more board-specific data, refer to Data Sheet of the DS2621 Signal Generation Board (SCALEXIO Hardware Installation and Configuration) .
	DS6101 Multi-I/O Board For more board-specific data, refer to Data Sheet of the DS6101 Multi-I/O Board (SCALEXIO Hardware Installation and Configuration) .

Block-Commuted PWM Out

Where to go from here

Information in this section

Introduction (Block-Commuted PWM Out).....	1022
Overviews (Block-Commuted PWM Out).....	1026
Configuring the Function Block (Block-Commuted PWM Out).....	1033
Hardware Dependencies (Block-Commuted PWM Out).....	1050

Introduction (Block-Commutated PWM Out)

Where to go from here

Information in this section

- [Introduction to the Function Block \(Block-Commutated PWM Out\)..... 1022](#)
- [Introduction to Block-Commutated PWM Signals..... 1023](#)

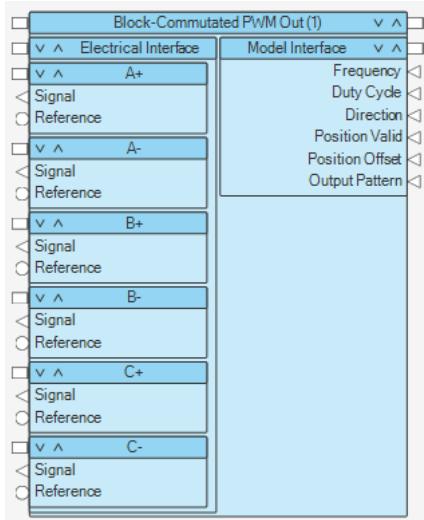
Introduction to the Function Block (Block-Commutated PWM Out)

Function block purpose

The Block-Commutated PWM Out function block generates block-commutated PWM signals to control three-phase brushless DC (BLDC) motors.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Generation of position-dependent signal patterns for controlling electric motors. The function block supports signal generation for three-phase motors (maximum of six output channels) that provide six motor sectors.
- Generation of individual position-independent signal patterns for controlling electric motors.

- Adjusting the connected motor to the function block (pole pair factor, sector offset).
- Receiving the duty cycle and the frequency from the behavior model to generate PWM signals.
- Receiving the current electrical angular position of the motor from a master APU provider and/or from the behavior model.
- Switching the motor direction from the behavior model.
- Providing I/O events and function triggers

Comparison to the Multi-Channel PWM Out function block

Compared to the Multi-Channel PWM Out function block, the Block-Commutated PWM Out function block offers a more convenient implementation in specific application scenarios. With the Block-Commutated PWM Out function block, you do not have to evaluate the angular position in the behavior model and calculate the duty cycle there, because this is done with the function block itself.

Available demo project

ConfigurationDesk provides the *EDrivesControlDemo* project. This project is an example for electric motor control using the DS6121 Multi-I/O Board. You can use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to [EDrivesControlDemo Project: Example of Electric Motor Control Using the DS6121 Multi-I/O Board \(ConfigurationDesk Demo Projects\)](#).

Supported channel types

The Block-Commutated PWM Out function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	Digital Out 8	–
Board	DS6121	–

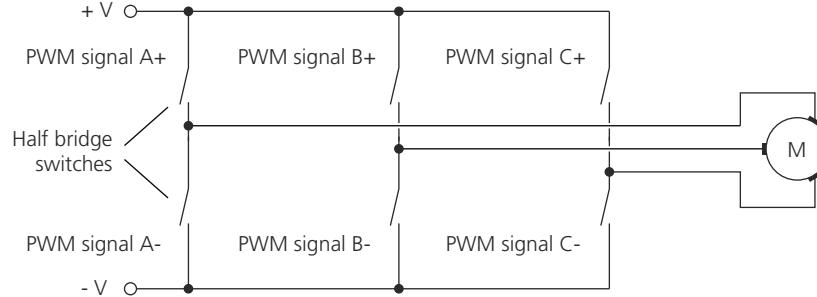
Introduction to Block-Commutated PWM Signals

Introduction

The signals referred to here as block-commutated PWM signals are mainly used to control brushless DC (BLDC) motors. Unlike brush motors, BLDC motors use external control signals to achieve commutation.

Controlling half bridges

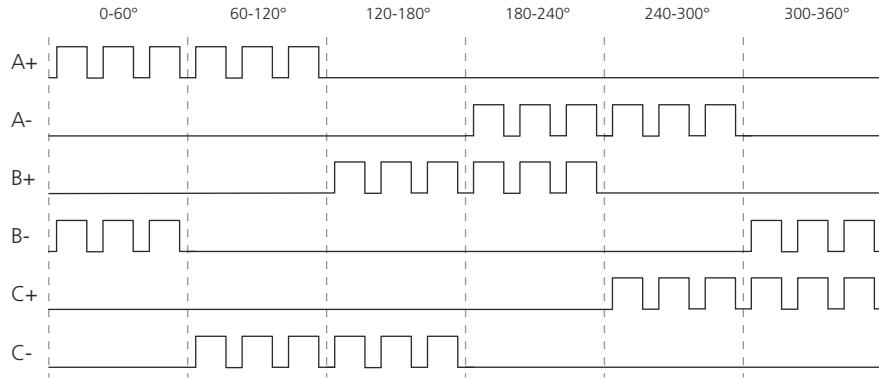
Block-commutated PWM signals are used to control half bridges that drive the connected electric motor. Each half bridge is controlled by different PWM signals. The following example shows a three-phase motor with three half bridges.



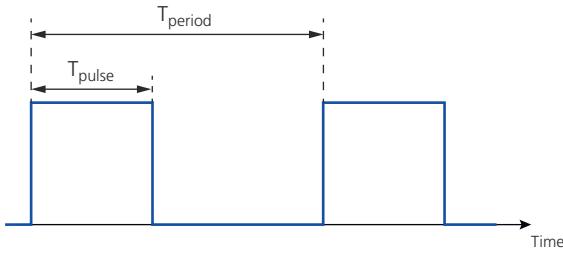
The half bridge switches are controlled in dependence on the position of the rotors in such a way that current flows through two of the three stator windings. This creates a magnetic field with a certain orientation. As the rotor continues to rotate, other switches are actuated so that the current flows through other windings, creating a magnetic field with a new orientation. The switchover causes the magnetic field vector in the stator to jump to the next sector.

PWM signal generation

Using the Block-Commuted PWM Out function block, you can specify different PWM signals, one for each half bridge switch for each sector of the motor. Refer to the following illustration.

**Basics on PWM signals**

PWM (pulse width modulation) signals are square-wave signals with a constant frequency. One use case of PWM signals is the power control of motors. A PWM signal is specified by the following characteristics:



Symbol	Definition
T_{pulse}	Pulse width
T_{period}	Period time

Modulation technique PWM signals are signals with a fixed frequency and a variable pulse width. The pulse width is expressed by the duty cycle, which is independent of the period time. The duty cycle is the ratio of the pulse duration to the period time:

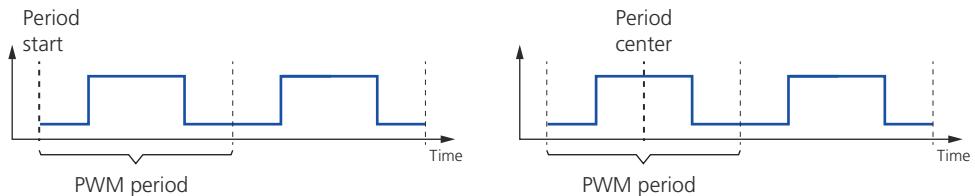
Symbol	Definition
D	$D = T_{pulse} / T_{period}$

The duty cycle of the PWM signal defines the torque of the motor.

PFM signals If the duty cycle is constant and the frequency used to control the signal information varies, the generated signals are called PFM (pulse frequency modulation) signals.

Characteristics of generated PWM/PFM signals Particular characteristics of the PWM/PFM signals generated with the Block Commutated PWM Out function block are:

- All PWM/PFM signals have the same period time/frequency.
- All PWM/PFM signals are updated at the same point in time (period start, period center).



Oversviews (Block-Commuted PWM Out)

Where to go from here

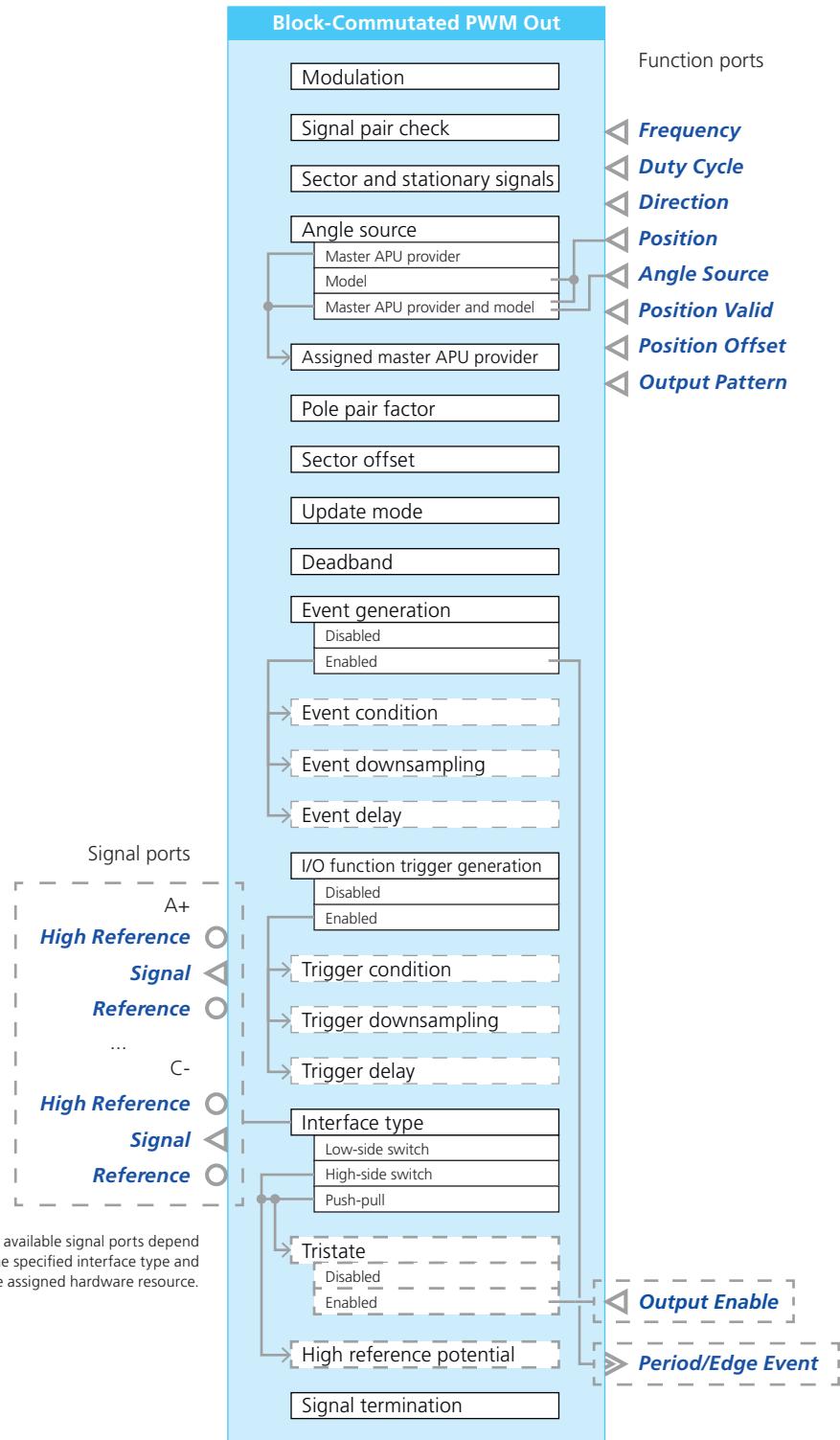
Information in this section

Overview of Ports and Basic Properties (Block-Commuted PWM Out).....	1026
Overview of Tunable Properties (Block-Commuted PWM Out).....	1032

Overview of Ports and Basic Properties (Block-Commuted PWM Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Frequency

This function import reads the common frequency value that is used for the generation of all PWM signals from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ $f_{\min} \dots f_{\max}$ (in Hertz). ▪ The range depends on the following: <ul style="list-style-type: none"> ▪ The assigned hardware resource. For specific values, refer to Hardware Dependencies (Block-Commuted PWM Out) on page 1050. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range. ▪ 0 Hz: The signal generation stops at the next update point, i.e., an update is not performed. The output signals keep the last levels. Note that the signal generation does not stop if the user saturation saturates the minimum frequency to a value greater than 0 Hz. <p>For the generated output signals, refer to the following table.</p>
Dependencies	–

The following table shows the output signals if the signal generation stops:

Update Mode	Generated Output Signals at 0 Hz
Period start	<p>After the current period ends, the signals are set to the following levels:</p> <ul style="list-style-type: none"> ▪ Duty cycle < 100%: The non-inverted signals are set to the inactive level and inverted signals to the active level. ▪ Duty cycle = 100%: The non-inverted signals are set to the active level and inverted signals to the inactive level.
Period center	<p>At the next period center, the signals are set to the following levels:</p> <ul style="list-style-type: none"> ▪ Duty cycle > 0%: The non-inverted signals are set to the active level and inverted signals to the inactive level. ▪ Duty cycle = 0%: The non-inverted signals are set to the inactive level and inverted signals to the active level.
Period start and center	Depends on the moment the next update of the signal generation takes place. Refer to <i>period start</i> and <i>period center</i> .

Duty Cycle

This function import reads a duty-cycle value that is used for generating PWM signals from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0% ... 100% ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Direction

This function imports reads the direction the connected motor should run from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0, 1: Forward The electric motor runs forward according to the signal patterns specified for the individual sectors of the motor. ▪ -1: Backward The electric motor runs backwards. When switching the port value to backward, the angle position is shifted internally by 180°. This means that the signals of the sector that is three blocks away are output.
Dependencies	-

Angle Source

This function import reads the angle source that is used to receive the current electrical motor position (= electrical angle value) from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1: Master APU provider The angle value is received from a master APU provider that is assigned to the function block. ▪ 2: Model The angle value is received from the behavior model via the Position function port.
Dependencies	Available only if the Angle source <i>property</i> is set to Master APU provider and model .

Position

This function import reads the electric angular position from the behavior model, for example, of a connected Hall encoder.

Value range	<ul style="list-style-type: none"> ▪ 0° ... +360° (effective range) ▪ If a received value is outside the effective range, the system converts the received value to an effective value. Examples: -400° is converted to +320. +380° is converted to +20°. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	Available only if the Angle source property is set to Model or to Master APU provider and model .

Position Valid

This function import reads from the behavior model, indicating whether the provided position value is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: Invalid The position provided at the Position function port is stated as invalid.
-------------	----------------------------------------------------------------------------------------------------------------------------------------------

	<p>The out-of-sync signal pattern is generated unless position-independent signals are activated via the Output pattern function port.</p> <ul style="list-style-type: none"> ▪ 1: Valid <p>The position provided at the Position function port is stated as valid.</p> <p>Depending on the value at the Output pattern function port, position-dependent (sector) or position-independent (stationary) signals are generated.</p>
Dependencies	–

Position Offset

This function import receives an offset angle to be added to the position value. The position offset can be used for field weakening of the connected motor.

Value range	<ul style="list-style-type: none"> ▪ -180° ... +180° (effective range) ▪ If a received value is outside the effective range, the system converts the received value to an effective value. Example: -400° is converted to -40°. +380° is converted to +20°. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	–

Output Pattern

This function import reads the pattern number for position-independent signal generation from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: Sector signals are generated for each motor sector according to the specified signal patterns. ▪ 1 ... 6: Position-independent (stationary) signals are generated. The signal pattern with the specified pattern number is generated.
Dependencies	–

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled .

Period Event

This event port provides an I/O event at the start and/or center of each PWM period.

You have to specify the trigger condition as follows:

- The trigger point for an I/O event (via Event trigger condition property).
- The downsampling of occurred trigger points (via Event downsampling property).
- The time delay between the occurrence of a trigger point and the generation of the related I/O event (via Event delay property).

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal port when the digital outputs are operating as high-side switches or are in push-pull mode.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Block-Commuted PWM Out) on page 1050 .
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Block-Commuted PWM Out) on page 1050 .
Dependencies	–

Reference

This signal port is a reference port and provides the low-side reference signal for the Signal signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Block-Commuted PWM Out) on page 1050 .
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Block-Commutated PWM Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Update mode	✓	–
	Event condition	✓	–
	Event downsampling	✓	–
	Event delay	✓	–
	Trigger condition	✓	–
	Trigger downsampling	✓	–
	Trigger delay	✓	–
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓
Electrical Interface			
	Deadband	✓	–
	Signal termination	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Block-Commuted PWM Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (Block-Commuted PWM Out).....	1033
Configuring Sector and Stationary Signals (Block-Commuted PWM Out).....	1039
Configuring Standard Features (Block-Commuted PWM Out).....	1048

Configuring the Basic Functionality (Block-Commuted PWM Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying signal patterns to control the power switches (for example, half bridges) of the connected electric motor. Refer to [Configuring Sector and Stationary Signals \(Block-Commuted PWM Out\)](#) on page 1039.
- Using the signal pair check. Refer to [Configuring Sector and Stationary Signals \(Block-Commuted PWM Out\)](#) on page 1039.
- Adjusting the function block to the connected motor (pole pair factor, sector offset).
- Specifying the angular source, that provides the electrical motor position.
- Specifying the duty cycle and/or frequency update for PWM signal generation.
- Specifying a deadband for PWM signals.
- Specifying the conditions for the generation of I/O events and function triggers (trigger point, downsampling value, time delay).
- Enabling signal termination.

Specifying a pole pair factor

Lets you specify a pole pair factor that is used as a conversion factor for the input position to calculate the electric angle position of the connected motor.

The angle values provided by the master APU provider and/or from the behavior model must be converted to the electrical angle value of the connected motor. To achieve this match, you have to specify a pole pair factor.

The setting of the pole pair factor depends on the characteristic of the provided input position (electrical or mechanical):

- If an electrical position is provided, for example, by a Hall encoder the following cases apply:
 - The pole pair number of the encoder (PPN_Encoder) and the pole pair number of the connected motor (PPN_Motor) match. In this case, set the pole pair factor to 1.
 - The pole pair number of the connected motor is higher than the pole pair number of the encoder. In this case, set the pole pair factor to the ratio of PPN_Motor/PPN_Encoder.

Note

- You cannot use an encoder with a higher pole pair number than the connected electric motor, if the angle values are provided by a master APU provider.
- The ratio of PPN_Motor/PPN_Encoder must be an integer number. This means, for example, that a motor with three pole pairs cannot be used in combination with an encoder with two pole pairs.

Examples for Setting the Pole Pair Factor

PPN ¹⁾ of Encoder (Electrical Position is Provided)	PPN ¹⁾ of Connected Motor	Required Pole Pair Factor Setting
1	2	2
2	2	1
2	4	2

¹⁾ PPN = pole pair number

- If a mechanical position is provided (for example, from an incremental encoder) a conversion to an electrical position is required to determine the current sector of the motor. In this case, the pole pair factor must be set to the pole pair number of the connected electric motor.

Examples for Setting the Pole Pair Factor

PPN ¹⁾ of Encoder (Mechanical Position is Provided)	PPN ¹⁾ of Connected Motor	Required Pole Pair Factor Setting
1	2	2
1	4	4

¹⁾ PPN = pole pair number

- If the position can be provided from a master APU provider or from the behavior model (switchable via Angle Source function port) the factor applies to both positions. In this case, you have to specify a pole pair factor that matches to the provided position of the master APU provider. In the behavior model, the angle position must be converted to fit this pole pair factor and then provided via the Position function port to the function block. Refer to the examples below.

Examples for Setting the Pole Pair Factor				
PPN¹⁾ of Encoder A (Position Provided via Master APU Provider)	PPN¹⁾ of Connected Motor	Required Pole Pair Factor Setting	PPN¹⁾ of Encoder B (Position Provided via Behavior Model)	Required Conversion in Behavior Model
1	2	2	2	The angle value of encoder B must be divided by 2.
4	4	1	2	The angle value of encoder B must be multiplied by 2.
4	4	1	1	The angle value of encoder B must be multiplied by 4.
2	4	2	1	The angle value of encoder B must be multiplied by 2.
1	3	3	1	No conversion required.

¹⁾ PPN = pole pair number

Specifying a sector offset

You can specify an offset angle to be added to the start angle of the first sector ($0^\circ - 60^\circ$). All sectors have the same angle range, so the start angle of all other sectors is automatically shifted with the specified value.

The sector offset can be used to adjust the signal pattern configuration to the connected motor, for example. This allows you to use a predefined signal pattern and then shift the signal pattern with the offset value, for example, by 120° (= two sectors) without changing the signal pattern used.

Specifying the source of angle position values

You have to specify the source from which the current position of the electric motor (= angle position) is received. You can select from the following options:

- Master APU provider:
The angle value is received from a master APU provider that is assigned to the function block (via Assigned master APU provider property).
- Model:
The angle value is received from the behavior model via the Position function port.
- Master APU provider and model:
The angle value is received from a master APU provider that is assigned to the function block or from the behavior model via the Position function port. You can switch the angle source at run time via the Angle Source function port from the behavior model.

Tip

The last setting is useful, if you use a combination of an absolute and a relative encoder, for example, a Hall encoder and an incremental encoder, to measure the angle position of an electric motor.

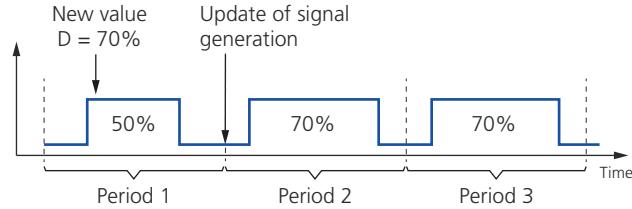
Use scenario: The first encoder (in this example, the Hall encoder) detects the position of a motor immediately but provides less accurate values. The second encoder (in this example, the incremental encoder) provides more accurate values than the first encoder, but requires up to one revolution of the motor to measure a valid (absolute) position. To get exact position values at *any* time, the behavior model provides the position from the Hall encoder until a valid position from the higher precision incremental encoder is available via the master APU provider (switchable via the Angle Source function port).

Specifying the update for PWM signal generation

At run time, you can specify new values for the duty cycle and/or for the frequency. The point in time of an update is essential for the generated PWM signals. With the Update mode property, you can specify the timing behavior of the update as follows:

- **Period start:**

In this mode, the PWM signals for all logical signals are updated at the period start. This means that new values do not take effect before the current PWM signal period is completed, as shown in the following illustration:

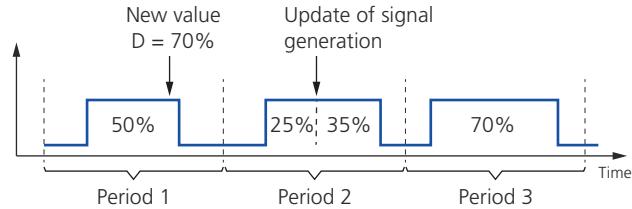


New values for the period and/or the duty cycle are updated at the start of a signal period (middle of the low pulses) of the PWM output signals.

New values for the period and/or the duty cycle are updated at the start of a signal period, i.e., signal periods represents new values from the beginning.

- **Period center:**

In this mode, the PWM signals for all logical signals are updated at the center of a PWM period. This means that new values do not take effect before the center of a PWM period is generated, as shown in the following illustration:



New values for the signal period and/or the duty cycle are updated at the center of a signal period, i.e., the first part of this signal period represents the half of the old values and the second part of this signal period represents the half of the new values.

- **Period start and center:**

In this mode, the PWM signals for all logical signals are updated at the period start and at the period center.

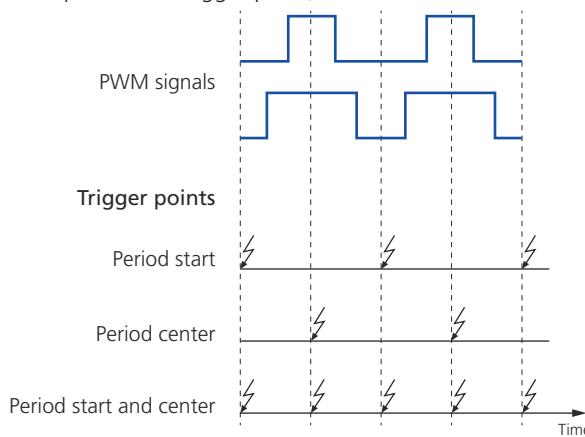
Providing I/O events and I/O function triggers

The function block can generate and provide:

- I/O events. They can be used as trigger sources for runnable functions in the behavior model. To use the I/O events in the behavior model, you have to assign them to a task via the Period Event property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).
- I/O Function triggers. These are trigger signals, that can be used as a trigger source for other function blocks.

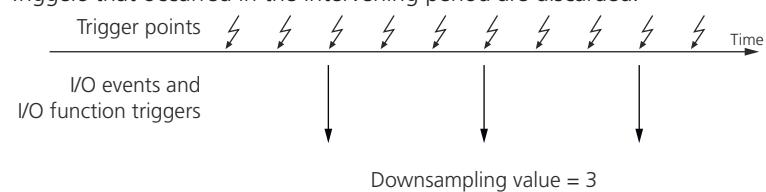
You have to specify the trigger conditions for generating I/O events and I/O function triggers. These are the trigger points, a downsampling value, and the delay for the generation.

Specifying the trigger point You can use period start, period center, or both points as a trigger point, as shown below.



Specifying a downsampling value The occurred triggers can be downsampled. The downsampling lets you specify whether to generate an I/O event or I/O function trigger on each trigger point or only on each n-th occurrence. The placeholder n is the specified downsampling value.

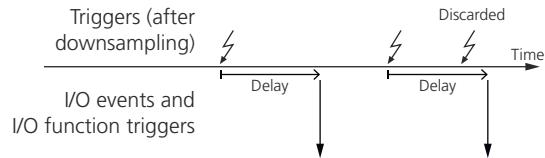
In the illustration below only every third trigger point is used for generation. Triggers that occurred in the intervening period are discarded.



The value range for the downsampling value depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Block-Commuted PWM Out\)](#) on page 1050.

Specifying a time delay for generation You can specify a time delay between the occurrence of a trigger point and the generation of the related I/O event or I/O function trigger.

If a (downsampled) trigger initiates the generation of an I/O event or I/O function trigger, the function block delays the generation until the specified time interval is elapsed. During the delay, all other downsampled triggers are discarded as shown below.



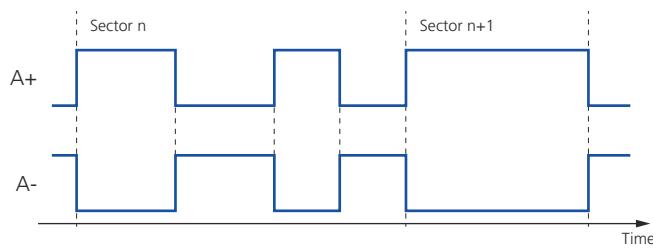
The value range for the delay time depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Block-Commuted PWM Out\)](#) on page 1050.

Specifying deadband to avoid shoot-through currents

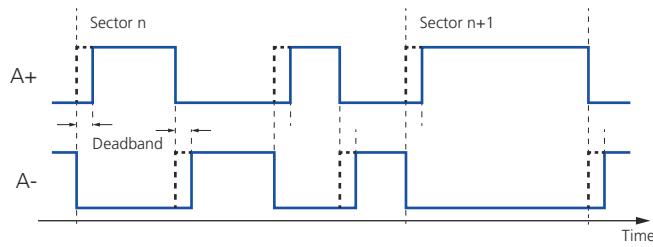
You can specify a deadband (time gap) between the rising and falling edges of the generated output signals (for example, A+) and their corresponding inverted output signals (for example, A-). The specified deadband is valid for all the channels of the related function block.

The deadband is used, for example, to prevent shoot-through currents on the half bridges. The high times of the inverted and non-inverted output signals are reduced according to the delay of their rising edges. The following illustration shows an example of a resulting signal shape. The deadband delays the edge generation of new pulses, as shown in the following illustration:

Without deadband



With deadband



If the resulting high time of the inverted or non-inverted output signal is less than the minimum pulse width, the signal remains at low level.

The value range of the deadband depends on the assigned hardware. For specific values, refer to [Hardware Dependencies \(Block-Commuted PWM Out\)](#) on page 1050.

Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

For a circuit diagram, refer to [Digital Out 8](#) on page 1581.

Configuring Sector and Stationary Signals (Block-Commuted PWM Out)

Introduction

To control the power switches (for example, half bridges) that drive the connected electric motor, you have to specify sector and stationary signals. Therefore, the function block provides predefined signal patterns and also the possibility to specify customer-specific patterns.

Sector signals

Sector signals are used for position-dependent signal generation. For each sector of the motor (for example, 0 ... 60°), you have to specify one signal pattern.

ConfigurationDesk provides a set of predefined signal patterns for sector signals that supports typical use scenarios. These predefined signal patterns are identified as modulation types.

Stationary signals

Stationary signals are independent from the motor sector and can therefore be used for position-independent signal generation. For example, they can be used to trigger emergency braking of the motor or to let the motor run down (to idle).

You can specify up to seven signal patterns with stationary signals, each contains one signal for each logical signal (A+, A-, B+, B-, C+, C-).

The last of these signal patterns is a specific out-of-sync signal pattern. This pattern is generated only if the position value is marked as invalid via the Position Valid function port and if the generation of position-dependent signals is activated via the Output Pattern function port (is set to 0).

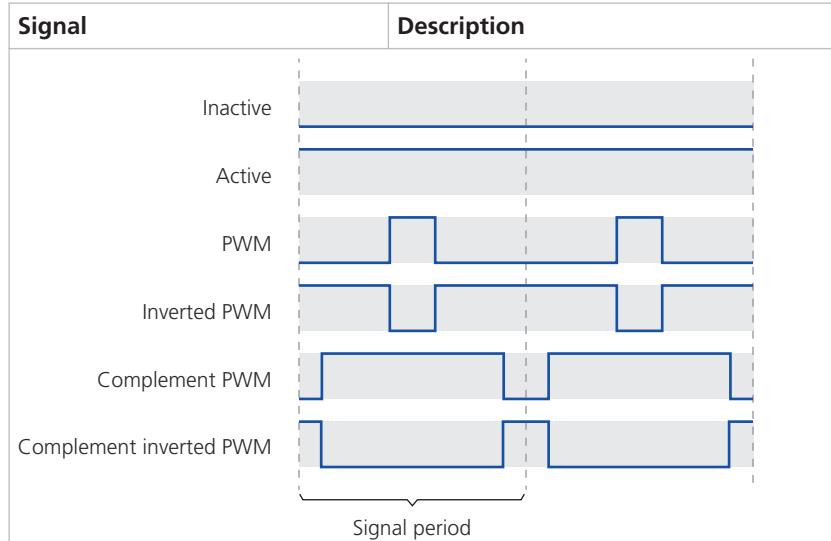
Switching between sector and stationary signals during runtime

At runtime you can switch from generating position-dependent sector signals to generating stationary signals from the behavior model via the **Output pattern** function port.

Supported signals

The following signals are supported and can be used for sector signals as well as for stationary signals:

Signal	Description
Inactive	A permanently inactive signal.
Active	A permanently active signal.
PWM	A PWM signal specified by the frequency and the duty cycle with a non-inverted output.
Inverted PWM	A PWM signal specified by the frequency and the duty cycle with an inverted output.
Complement PWM	A PWM signal specified by the frequency and the inverted duty cycle (100% - duty cycle) with a non-inverted output.
Complement inverted PWM	A PWM signal specified by the frequency and the inverted duty cycle (100% - duty cycle) with an inverted output.



The diagram illustrates the timing of six signal types over a signal period. The period is divided into four segments by vertical dashed lines.
 - **Inactive**: A horizontal blue line at the top.
 - **Active**: A horizontal blue line at the bottom.
 - **PWM**: A single blue square pulse centered in the first segment.
 - **Inverted PWM**: Two blue square pulses, one in the first segment and one in the third segment.
 - **Complement PWM**: Three blue square pulses, one in the first segment, one in the second segment, and one in the fourth segment.
 - **Complement inverted PWM**: Four blue square pulses, one in each segment.
 A double-headed arrow below the segments is labeled "Signal period".

Specifying signal patterns

ConfigurationDesk provides the **Sector and Stationary Signals Configuration** dialog to specify signal patterns for all output signals to be generated by the function block.



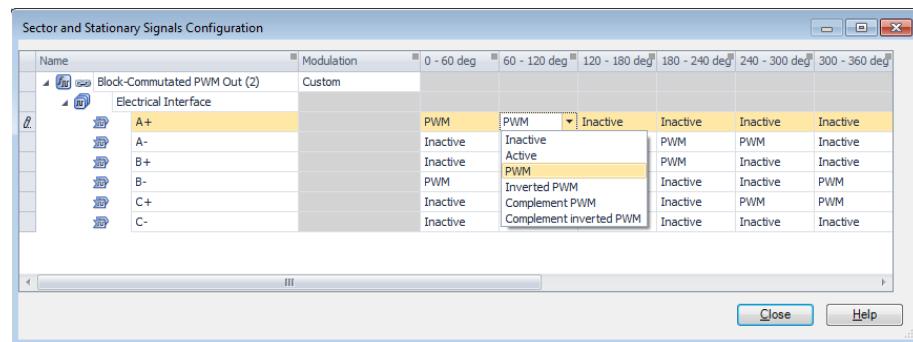
You can specify the following signal patterns:

- Signal patterns for each sector of the motor (for example, in the 0 - 60 deg column in the dialog). To do this, you have the following options:
 - You can select a set of predefined signal patterns via the Modulation column.

If you select one of these settings, all sector signals are fixed to a signal according to the selected modulation type. Therefore, the signal settings (Active, PWM, etc.) are read-only.

- Custom:

You can specify your own signal patterns for sector signals (one pattern for each sector). Refer to the illustration below.

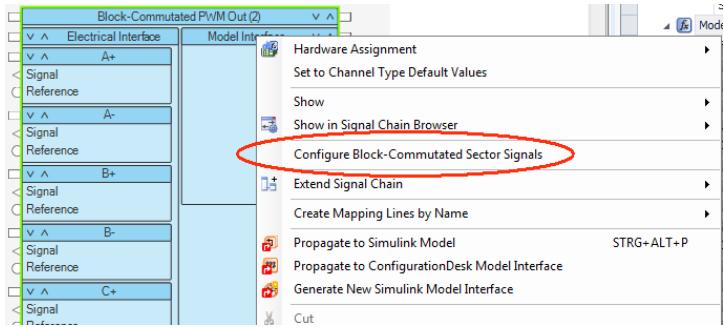


- All inactive:

All sector signals are fixed to Inactive. Therefore, the signal settings (Active, PWM, etc.) are read-only. If you use this setting, the position-dependent signal generation is disabled.

- Signal patterns for stationary signals in the Specific pattern <n> columns.
- Out-of-sync signal pattern in the Out-of-sync pattern column.

Accessing the configuration dialog You can access the dialog, via the Sector and stationary signals property in the Properties Browser or via the Configure Block-Commutated Sector Signals command (right-click on the function block) as shown below.

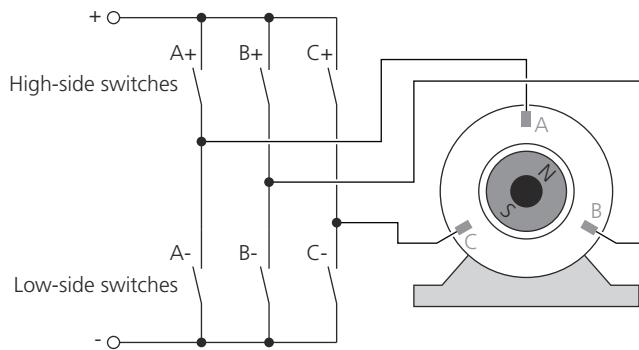


Tips for filling out the table There are useful tips on how to fill out the signal table:

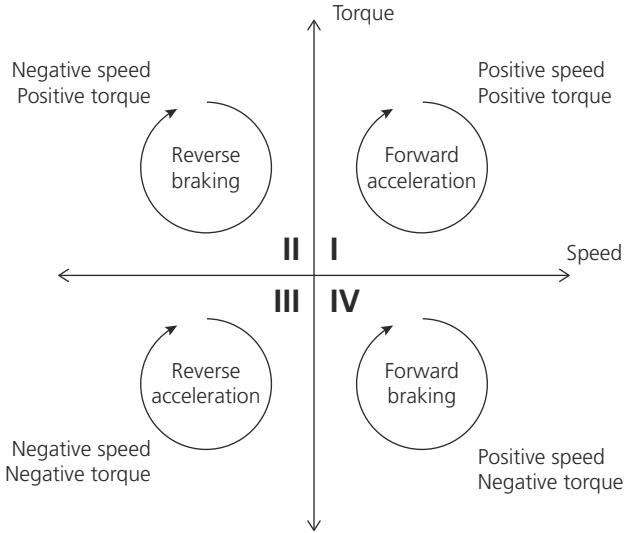
- If you want to start your configuration from a uniform setting, which can then be changed: Set the modulation to All inactive and then to Custom. Now you can change the settings from a unique basis.
- If you want to change a predefined setting of a specific modulation type: Select the modulation type you want to adapt and then select Custom. Now you can change the settings to your request.

Characteristics of predefined modulation types

The predefined modulation types are only intended for three-phase motors connected via commonly used half bridges. For more or less than three phases, the signal patterns must be adjusted accordingly. The illustration below is an abstract example and shows the half bridges of the power stages that control the motor.



The illustration below shows the 4-quadrant system that describes the operation of an electric motor.



2-quadrant bipolar The connected motor is controlled by a bipolar switching and a 2-quadrant operation. The high-side and the low-side switches of the half bridges are driven. The motor can run forwards and backwards, but it cannot be braked. It is therefore operated in the first and third quadrants. The supported duty cycle is in the range 0 to 100%.

You can change the direction of the motor at run time from the behavior model via the Direction function port.

2-quadrant high side The connected motor is controlled by a unipolar switching and a 2-quadrant operation. Only the high-side switches of the half bridges are driven. The motor can run forwards and backwards, but it cannot be braked. It is therefore operated in the first and third quadrants. The supported duty cycle is in the range 0 to 100%.

You can change the direction of the motor at run time from the behavior model via the Direction function port.

2-quadrant low side The connected motor is controlled by a unipolar switching and a 2-quadrant operation. Only the low-side switches of the half bridges are driven. The motor can run forwards and backwards, but it cannot be braked. It is therefore operated in the first and third quadrants. The supported duty cycle is in the range 0 to 100%.

You can change the direction of the motor at run time from the behavior model via the Direction function port.

2-quadrant symmetric The connected motor is controlled by a unipolar switching and a 2-quadrant operation. The high-side and low-side switches of the half bridges are driven alternately. The motor can run forwards and backwards, but it cannot be braked. It is therefore operated in the first and third quadrants. The supported duty cycle is in the range 0 to 100%.

You can change the direction of the motor at run time from the behavior model via the Direction function port.

The advantage of a symmetrical operation (compared to low-side and high-side switching) is that the heat losses are evenly distributed across all high-side and low-side switches, because all switches perform the same number of switching operations.

4-quadrant unipolar The connected motor is controlled by a unipolar switching and a 4-quadrant operation. The high-side and low-side switches of the half bridges are driven via complementary PWM signals. The two switches of a half bridge are operated with different duty cycles.

The motor can run forwards and backwards and can also be braked in both cases. It is therefore operated in all four quadrants.

A duty cycle value of 50% leads to an idle speed (no torque). A duty cycles of 50% to 100% leads to a forward acceleration of the motor and a duty cycle of 0% to 50% leads to a backward acceleration. You also can change the direction of the motor at run time from the behavior model via the Direction function port.

4-quadrant bipolar The connected motor is controlled by a bipolar switching and a 4-quadrant operation. The upper and lower switches of the half bridges are switched via inverted PWM signals. The two switches of a half bridge are operated with the same duty cycles.

The motor can run forwards and backwards and can also be braked in both cases. It is therefore operated in all four quadrants.

A duty cycle value of 50% leads to an idle speed (no torque). A duty cycles of 50% to 100% leads to a forward acceleration of the motor and a duty cycle of 0% to 50% leads to a backward acceleration. You also can change the direction of the motor at run time from the behavior model via the Direction function port.

Predefined settings of signals for modulation types

The following table shows the signal settings for the sectors and logical signals when selecting a specific modulation type.

The deadband between the two signals, (for example, A+ and A-) required mainly for the 4-quadrant modulation is not shown in the signal shapes below.

Modulation Type	Signal	0-60 deg	60-120 deg	120-180 deg	180-240 deg	240-300 deg	300-360 deg
2-quadrant bipolar	A+	PWM	PWM	Inactive	Inactive	Inactive	Inactive
	A-	Inactive	Inactive	Inactive	PWM	PWM	Inactive
	B+	Inactive	Inactive	PWM	PWM	Inactive	Inactive
	B-	PWM	Inactive	Inactive	Inactive	Inactive	PWM
	C+	Inactive	Inactive	Inactive	Inactive	PWM	PWM
	C-	Inactive	PWM	PWM	Inactive	Inactive	Inactive
	A+	PWM	PWM	Inactive	Inactive	Inactive	Inactive
	A-	Inactive	Inactive	Inactive	Active	Active	Inactive
	B+	Inactive	Inactive	PWM	PWM	Inactive	Inactive
	B-	Active	Inactive	Inactive	Inactive	Inactive	Active
	C+	Inactive	Inactive	Inactive	Inactive	PWM	PWM
	C-	Inactive	Active	Active	Inactive	Inactive	Inactive
2-quadrant high side	A+	PWM	PWM	Inactive	Inactive	Inactive	Inactive
	A-	Inactive	Inactive	Inactive	Active	Active	Inactive
	B+	Inactive	Inactive	PWM	PWM	Inactive	Inactive
	B-	Active	Inactive	Inactive	Inactive	Inactive	Active
	C+	Inactive	Inactive	Inactive	Inactive	PWM	PWM
	C-	Inactive	Active	Active	Inactive	Inactive	Inactive
	A+	PWM	PWM	Inactive	Inactive	Inactive	Inactive
	A-	Inactive	Inactive	Inactive	High	High	Inactive
	B+	Inactive	Inactive	PWM	PWM	Inactive	Inactive
	B-	High	Inactive	Inactive	Inactive	Inactive	High
	C+	Inactive	Inactive	Inactive	Inactive	PWM	PWM
	C-	Inactive	High	High	High	High	Inactive

Modulation Type	Signal	0-60 deg	60-120 deg	120-180 deg	180-240 deg	240-300 deg	300-360 deg
2-quadrant low side	A+	Active	Active	Inactive	Inactive	Inactive	Inactive
	A-	Inactive	Inactive	Inactive	PWM	PWM	Inactive
	B+	Inactive	Inactive	Active	Active	Inactive	Inactive
	B-	PWM	Inactive	Inactive	Inactive	Inactive	PWM
	C+	Inactive	Inactive	Inactive	Inactive	Active	Active
	C-	Inactive	PWM	PWM	Inactive	Inactive	Inactive
		0-60°	60-120°	120-180°	180-240°	240-300°	300-360°
	A+						
	A-						
	B+						
	B-						
	C+						
	C-						
2-quadrant symmetric	A+	Active	PWM	Inactive	Inactive	Inactive	Inactive
	A-	Inactive	Inactive	Inactive	Active	PWM	Inactive
	B+	Inactive	Inactive	Active	PWM	Inactive	Inactive
	B-	PWM	Inactive	Inactive	Inactive	Inactive	Active
	C+	Inactive	Inactive	Inactive	Inactive	Active	PWM
	C-	Inactive	Active	PWM	Inactive	Inactive	Inactive
		0-60°	60-120°	120-180°	180-240°	240-300°	300-360°
	A+						
	A-						
	B+						
	B-						
	C+						
	C-						

The figure displays two sets of timing diagrams for six signals (A+, A-, B+, B-, C+, C-) across six 60-degree phases. The top set of diagrams, labeled '2-quadrant low side', shows a sequence where signals transition between active, PWM, and inactive states. The bottom set, labeled '2-quadrant symmetric', shows a different sequence where signals transition between active, PWM, and inactive states. The x-axis for both sets is divided into six 60-degree segments by vertical dashed lines.

Modulation Type	Signal	0-60 deg	60-120 deg	120-180 deg	180-240 deg	240-300 deg	300-360 deg
4-quadrant unipolar	A+	PWM	PWM	Inactive	Complement PWM	Complement PWM	Inactive
	A-	Inverted PWM	Inverted PWM	Inactive	Complement inverted PWM	Complement inverted PWM	Inactive
	B+	Complement PWM	Inactive	PWM	PWM	Inactive	Complement PWM
	B-	Complement inverted PWM	Inactive	Inverted PWM	Inverted PWM	Inactive	Complement inverted PWM
	C+	Inactive	Complement PWM	Complement PWM	Inactive	PWM	PWM
	C-	Inactive	Complement inverted PWM	Complement inverted PWM	Inactive	Inverted PWM	Inverted PWM
4-quadrant bipolar	A+	PWM	PWM	Inactive	Inverted PWM	Inverted PWM	Inactive
	A-	Inverted PWM	Inverted PWM	Inactive	PWM	PWM	Inactive
	B+	Inverted PWM	Inactive	PWM	PWM	Inactive	Inverted PWM
	B-	PWM	Inactive	Inverted PWM	Inverted PWM	Inactive	PWM
	C+	Inactive	Inverted PWM	Inverted PWM	Inactive	PWM	PWM
	C-	Inactive	PWM	PWM	Inactive	Inverted PWM	Inverted PWM

Using the signal pair check

The Block-Commuted PWM function block provides a check to verify if the output signals for each signal pair (e.g., A+ and A-) are problematic. You can disable and enable the check via the Signal pair check property.

The check determines if the output levels for each signal pair can become active at the same time. This might cause a short circuit in unprotected half bridges.

The signal pair check includes the signal pattern specified for sector signals and for stationary signals.

Unproblematic signal settings in accordance with the signal pair check are:

- Inactive and all other signals
- PWM and Inverted PWM
- Complement PWM and Complement inverted PWM

If the check detects a problematic signal pair, a conflict is generated and displayed in the Conflicts Viewer.

NOTICE**Disabling the signal pair check in combination with problematic signal pairs.**

Risk of damage to the connected external device.

- Ensure, that disabling the signal pair check will not result in damage to the connected half bridges because a mismatching of the signals of a signal pair can cause short circuits in the half bridges.

Configuring Standard Features (Block-Commutated PWM Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 8	Further Information
Digital output signals	Interface type	<ul style="list-style-type: none"> ▪ High-side switch

Configuration Feature	Digital Out 8	Further Information
	<ul style="list-style-type: none"> ▪ Low-side switch ▪ Push-pull 	
Setting to high impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs by setting the channel to high impedance (tristate) from the behavior model during run time.	–

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 8](#) on page 1581

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Block-Commutated PWM Out)

SCALEXIO Hardware Dependencies (Block-Commutated PWM Out)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type.

Characteristics		Digital Out 8
Hardware		DS6121 Multi-I/O Board
Number of sectors for one motor revolution		6
Number of signals		6
Frequency	Range	<ul style="list-style-type: none"> ▪ 1 Hz ... 20 MHz (at 50% duty cycle) ▪ 1 Hz ... 400 kHz (at 1% or 99% duty cycle)
	Time resolution	8 ns
Minimum pulse duration		25 ns
Deadband		0 ... 400 µs
Delay time (for I/O events and function triggers)		0 ... 1.0 s
Downsampling value (for I/O events and function triggers)		1 ... 256
Supported interface types for digital outputs		<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull <p>In push-pull configuration, the outputs can be set to high impedance (tristate) during run time.</p>
High side reference voltage (high-side switch, push-pull)		+3.3 V and +5 V
Current range		0 mA ... ±40 mA (each channel)
Signal termination		<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable
Circuit diagram		Digital Out 8 on page 1581
Required channels		6 ¹⁾

¹⁾ One channel for each non-inverted PWM signal and one channel for each inverted PWM signal.

General limitations

- Channel multiplication is not supported in general.
- If the Block-Commutated PWM Out function block works as a trigger function provider: The hardware resources assigned to the Block-Commutated PWM Out function block and the hardware resources assigned to the function block using the Block-Commutated PWM Out function block as trigger source must be located on the same I/O board.

More hardware data

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet](#) of the [DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Field-Oriented Control In/Out

Where to go from here

Information in this section

Introduction (Field-Oriented Control In/Out).....	1054
Overviews (Field-Oriented Control In/Out).....	1061
Configuring the Function Block (Field-Oriented Control In/Out).....	1070
Hardware Dependencies (Field-Oriented Control In/Out).....	1085

Introduction (Field-Oriented Control In/Out)

Where to go from here

Information in this section

Introduction to the Function Block (Field-Oriented Control In/Out).....	1054
Basics on Field-Oriented Control.....	1056
Basics on Using the Function Block with Field-Oriented Control.....	1058
Basics on PWM Signal Generation.....	1059

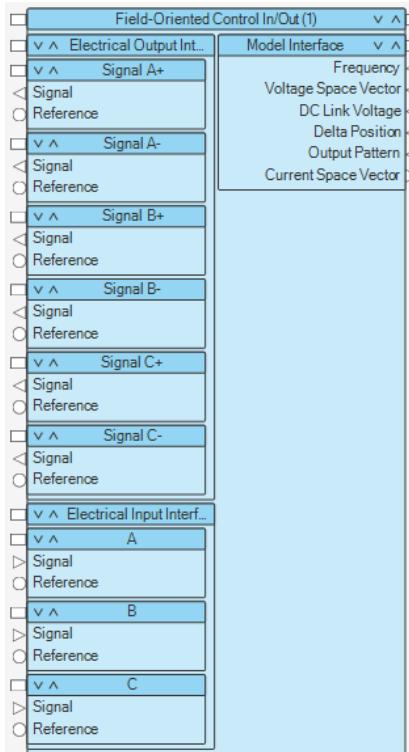
Introduction to the Function Block (Field-Oriented Control In/Out)

Function block purpose

The Field-Oriented Control In/Out function block generates PWM signals to drive, for example, permanent magnet synchronous motors. The function block provides all the processing required for controlling three-phase motors via the field-oriented control method, such as transforming voltage and current vectors to space vectors and vice versa.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Measuring three analog signals via external current transformers and converting them to motor phase current values.
- Performing DQZ and inverse DQZ transforms to adapt motor data to the behavior model and vice versa.
- Supporting pure sine commutation and sine commutation with third harmonic or min-max common mode injection for calculating the pole voltage vector.
- Generating PWM signals and their inverted signals from the pole voltage vector.
- Capturing motor angle positions from selectable angle sources and optional timestamps for each measurement.
- Providing I/O events and I/O function triggers synchronously to the PWM signal and providing them to the behavior model.

Field-oriented control compared to block-commutated PWM generation

In contrast to the Block-Commutated PWM Out function block, the Field-Oriented Control In/Out function block performs sine-commutation. In sine-commutation, the rotating field is rotated seamlessly with the rotor, while in block-commutation, the rotating field is rotated step-wise, sector by sector. The motor drive signals resulting from PWM signals generated by field-oriented control are sine-shaped signals with consistent torque.

Required channel types

The Field-Oriented Control In/Out function block type requires the following channel types and hardware:

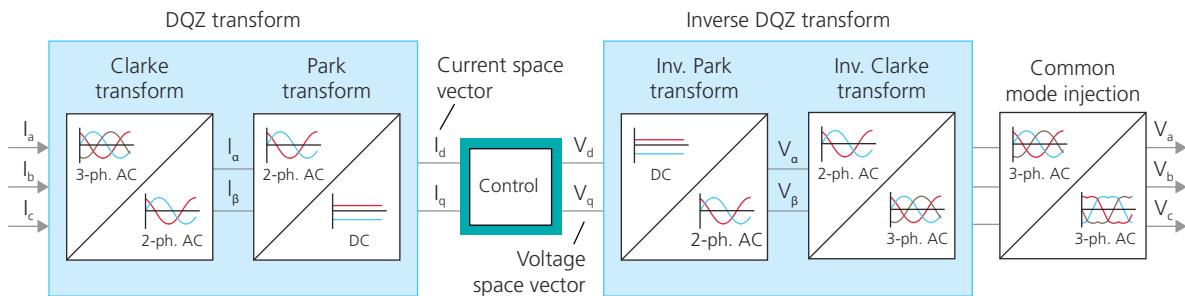
	SCALEXIO	MicroAutoBox III
Hardware	DS6121	—
Channel types	Channels of the following channel types are required: <ul style="list-style-type: none"> ▪ Digital Out 8 ▪ Analog In 16 	—

Basics on Field-Oriented Control

Introduction

Field-oriented control is a method for controlling electric motors that require sinusoidal drive signals, for example, permanent magnet synchronous motors. To enable controlling of such motors with simple controllers instead of controlling AC signals, field-oriented control uses two mathematical transforms called *DQZ* and *inverse DQZ* transform. DQZ and inverse DQZ transform can generally be used with currents and voltages. For field-oriented control, the DQZ transform transforms AC currents from the motor into DC values (the current space vector) for the controller, the inverse DQZ transform transforms DC control values (the voltage space vector) into AC voltage signals that are used for generating motor drive signals.

DQZ transforms consist of the Clarke and the Park transform, inverse DQZ transforms consist of the inverse Clarke and the inverse Park transform. The general principle of field-oriented control is shown in the following illustration.



Current and voltage space vectors have a third component: I_0 or V_0 . Both are not shown in the illustration, because if the controlled three-phase system is balanced, their value is 0 (zero).

Basics on Clarke and Park transforms

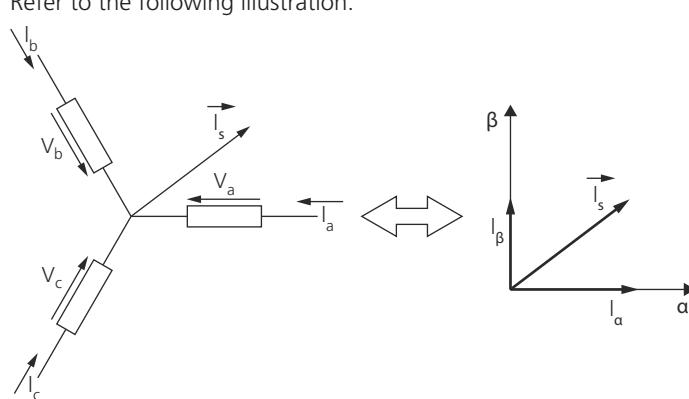
The Clarke and the Park transforms within the DQZ transform and their inversions operate as follows:

Clarke transform The Clarke transform transforms the three phase currents I_a , I_b , I_c of an electric motor that are offset by 120° from each other (for three-phase motors) into the currents I_α and I_β in an orthogonal reference frame. I_s is the space vector representing the currents.

For this transform, the following is assumed:

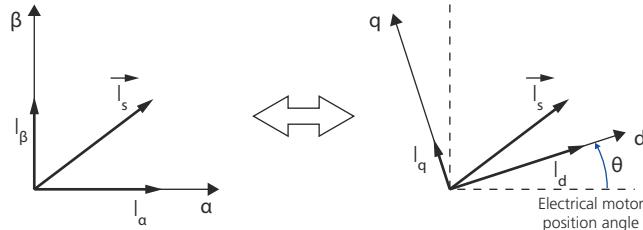
- The three-phase system is balanced, i.e., the sum of I_a , I_b , and I_c is zero. This enables the two resulting currents to include all the information about the three phase currents.
- The α -axis and the a -axis are aligned.

Refer to the following illustration:



Park transform For the Park transform, the stationary orthogonal reference frame with I_α and I_β is transformed into I_d and I_q in a reference frame that is rotated with the motor. This requires the current motor position angle θ to be available for each transform.

Refer to the following illustration:



In the rotating reference frame, I_d and I_q are constant values. I_d represents the field flux linkage and I_q represents the torque. I_s is the space vector representing the currents.

Matrices used for DQZ and inverse DQZ transforms

DQZ and inverse DQZ transforms can be performed as either amplitude-invariant or power-invariant. Amplitude-invariant transform is the commonly used transform method for electric drive control.

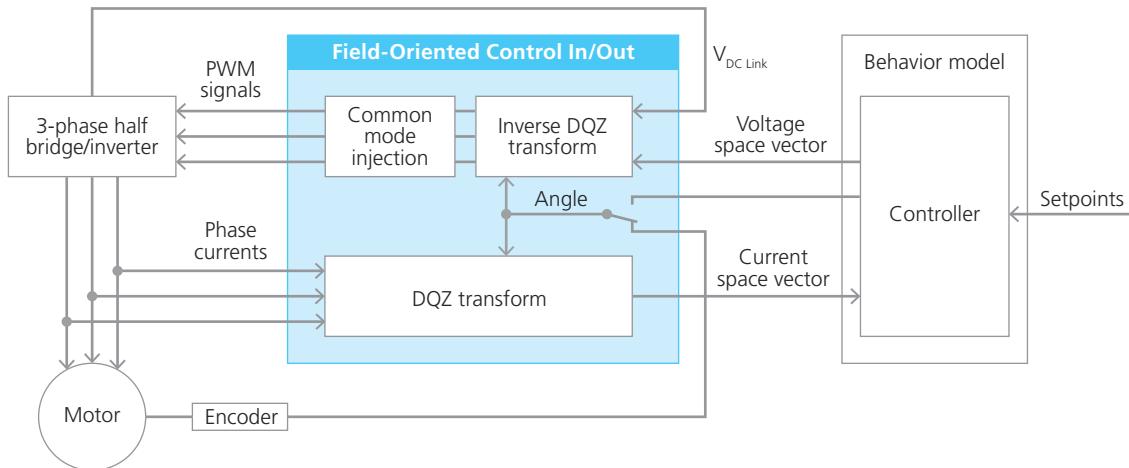
The following table shows the particular matrix for each of the possible transforms. For all four matrices, the a -axis is considered to be aligned with the d -axis. The angle value θ includes the motor position angle, the potential offset values, and a correction factor.

Amplitude-Invariant Transforms	Power-Invariant Transforms
Voltage and current magnitudes do not change during the transforms.	Power levels do not change during the transforms.
DQZ transform matrix	DQZ transform matrix
$\begin{bmatrix} I_d \\ I_q \\ I_z \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) \\ -\sin(\theta) & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix}$	$\begin{bmatrix} I_d \\ I_q \\ I_z \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) \\ -\sin(\theta) & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix}$
Inverse DQZ transform matrix	Inverse DQZ transform matrix
$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 1 \\ \cos\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) & 1 \\ \cos\left(\theta - \frac{4\pi}{3}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) & 1 \end{bmatrix} \begin{bmatrix} V_d \\ V_q \\ V_z \end{bmatrix}$	$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & \frac{1}{2} \\ \cos\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) & \frac{1}{2} \\ \cos\left(\theta - \frac{4\pi}{3}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) & \frac{1}{2} \end{bmatrix} \begin{bmatrix} V_d \\ V_q \\ V_z \end{bmatrix}$

Basics on Using the Function Block with Field-Oriented Control

Overview

The following illustration shows how the Field-Oriented Control In/Out function block and the transforms are integrated in a motor controller:



During run-time, the function block measures the motor phase currents and converts them to a current space vector $[I_d, I_q, I_0]$ via a DQZ transform after each measurement. The current space vectors are provided to the controller in the behavior model. Referencing the current space vectors and setpoints given by the user, the controller provides up-to-date voltage space vectors $[V_d, V_q, V_0]$. The function block transforms the voltage space vectors via inverse DQZ transforms and injection of common mode signals into PWM signals. The PWM signals control a three-phase half bridge/inverter that drives the motor.

Basics on PWM Signal Generation

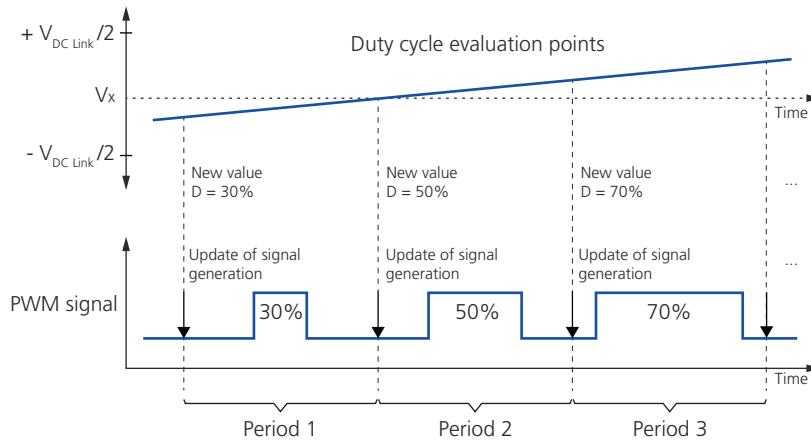
PWM signal generation

For PWM signal generation, the voltage space vector provided from the behavior model is transformed and modified by an added common mode signal to generate a pole voltage vector $[V_{a, b, c}]$. The vector entries are used to continuously calculate the duty cycles for the generated PWM signals as follows:

$$\text{Duty cycle} = \frac{V_{a, b, c}}{V_{DC \text{ Link}}} \cdot 100\% + 50\%$$

For $V_{a, b, c} = 0 \text{ V}$, the duty cycle is 50%.

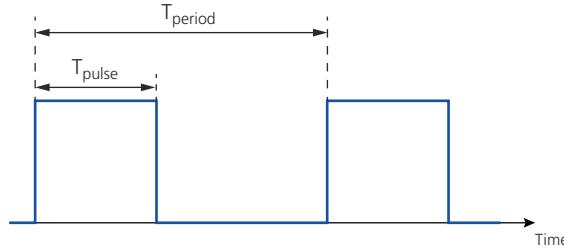
The process is shown for one PWM signal in the following illustration:



The duty cycle values are calculated at a high rate so that they are as accurate as possible for each update of the signal generation. At the update point, the latest calculated value becomes the new duty cycle. Update points can be specified for period start, period center, or period start and center.

Basics on PWM signals

PWM (pulse width modulation) signals are square-wave signals with a constant frequency. One use case of PWM signals is the power control of motors. A PWM signal is specified by the following characteristics:



Symbol	Definition
T_{pulse}	Pulse width
T_{period}	Period time

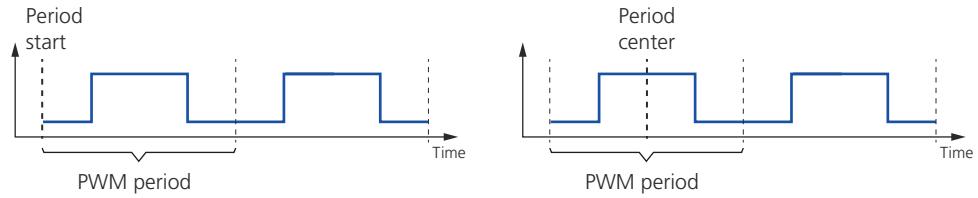
Modulation technique PWM signals are signals with a fixed frequency and a variable pulse width. The pulse width is expressed by the duty cycle, which is independent of the period time. The duty cycle is the ratio of the pulse duration to the period time:

Symbol	Definition
D	$D = T_{pulse} / T_{period}$

The mean values of the three PWM signals define the flux and the torque of the motor depending on the electric motor position angle.

Characteristics of generated PWM signals Particular characteristics of the PWM signals generated with the Field-Oriented Control In/Out function block are:

- All PWM signals have the same period time/frequency.
- All PWM signals are updated at the same point in time (period start, period center).



Overviews (Field-Oriented Control In/Out)

Where to go from here

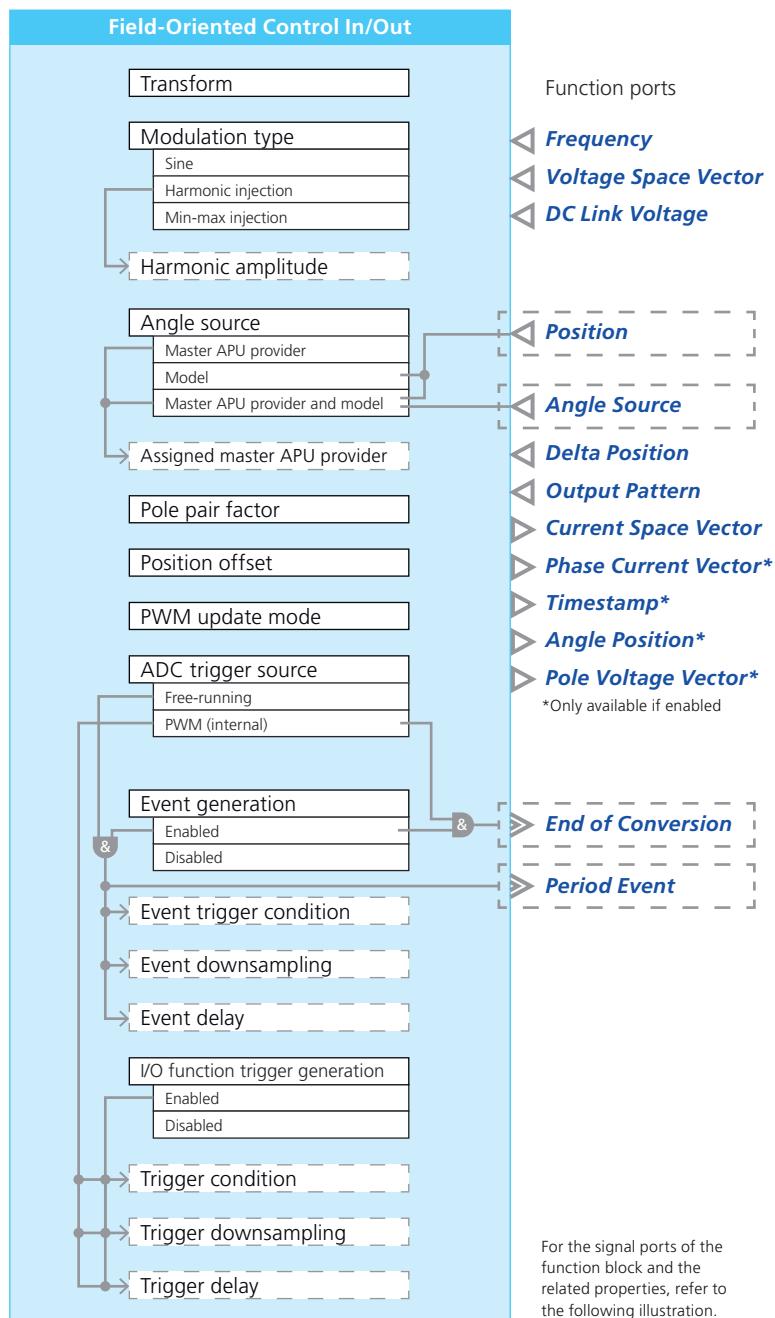
Information in this section

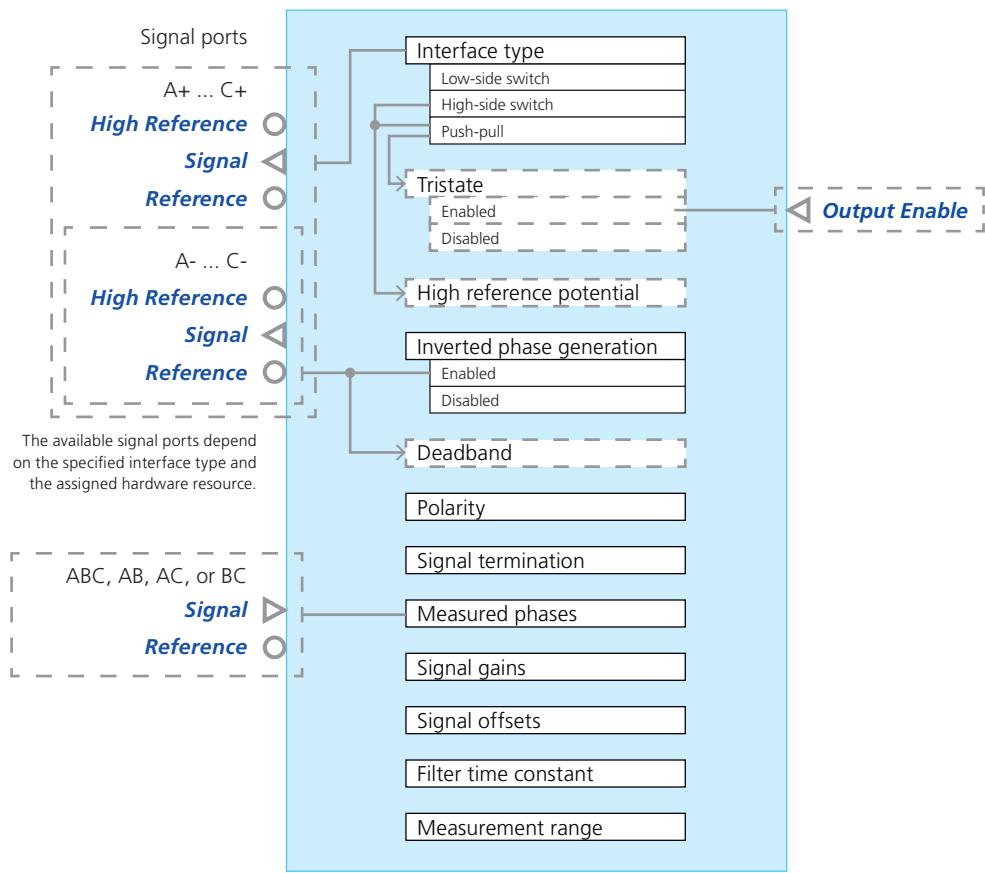
Overview of Ports and Basic Properties (Field-Oriented Control In/Out).....	1061
Overview of Tunable Properties (Field-Oriented Control In/Out).....	1068

Overview of Ports and Basic Properties (Field-Oriented Control In/Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



**Frequency**

This function import reads the common frequency value that is used for the generation of all PWM signals from the behavior model.

Value range	<ul style="list-style-type: none"> $f_{\min} \dots f_{\max}$ (in Hertz) The range depends on the following: <ul style="list-style-type: none"> The assigned hardware resource. For specific values, refer to Hardware Dependencies (Field-Oriented Control In/Out) on page 1085. If you use user saturation min/max values for saturation, the specified values can reduce the value range. 0 Hz: Signal generation stops at the next update point. The signal outports keep the last levels. Note that the signal generation does not stop if the user saturation saturates the minimum frequency to a value greater than 0 Hz.
Dependencies	–

Voltage Space Vector

This function import reads the voltage space vector $[V_d, V_q, V_0]$ from the behavior model. The voltage space vector is transformed via an inverse DQZ transform and

can additionally be modified by a common mode signal injection. The resulting pole voltages are used to generate the PWM signals.

Value range	<ul style="list-style-type: none"> ▪ -10^{+6} V ... $+10^{+6}$ V for each entry of the vector ▪ The vector size is three. ▪ If you use user saturation min/max values for saturation, the specified values might reduce the value range.
Dependencies	–

DC Link Voltage

This function import reads the DC link voltage from the behavior model. This voltage is the supply voltage of the three-phase half bridge/inverter and used for normalizing the voltage space vector.

Value range	<ul style="list-style-type: none"> ▪ 10^{-6} V ... 10^{+6} V ▪ If you use user saturation min/max values for saturation, the specified values might reduce the value range.
Dependencies	–

Angle Source

This function import reads the angle source that is used to receive the current electrical motor position (= electrical angle value) from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 1: Master APU provider The angle value is received from a master APU provider that is assigned to the function block. ▪ 2: Model The angle value is received from the behavior model via the Position function port.
Dependencies	Available only if the Angle source property is set to Master APU provider and model .

Position

This function import reads the current electrical motor position (= electrical angle value) from the behavior model, for example, of a connected Hall encoder.

Value range	<ul style="list-style-type: none"> ▪ 0° ... $+360^\circ$ (effective range) ▪ If a received value is outside the effective range, the system converts the received value to an effective value. Examples: -400° is converted to $+320^\circ$. $+380^\circ$ is converted to $+20^\circ$. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	Available only if the Angle source property is set to Model or to Master APU provider and model .

Delta Position

This function import reads an angle value from the behavior model, which determines an offset angle between DQZ transforms and inverse DQZ

transforms. For DQZ transforms, the delta position value is added to the electrical motor angle provided by the specified angle source and the offset angle specified at the Position offset property. The delta position value can be used, for example, to compensate or create an angle offset between DQZ transforms and inverse DQZ transforms.

Value range	<ul style="list-style-type: none"> ▪ 0° ... +360° (effective range) ▪ If a received value is outside the effective range, the system converts the received value to an effective value. Examples: -400° is converted to +320. +380° is converted to +20°. ▪ If you use user saturation min/max values for saturation, the specified values can reduce the value range.
Dependencies	-

Output Pattern

This function import reads from the behavior model which signals are output to the external three-phase half bridge/inverter.

Value range	<ul style="list-style-type: none"> ▪ 0: Normal The values of the pole voltage vector are used to generate the PWM signals that are output to the external three-phase half bridge/inverter. ▪ 1: Free-running All normal and inverted signal outputs are immediately set to inactive (all switches of the external three-phase half bridge/inverter are open). ▪ 2: Brake duty cycle 0 All normal signal outports are immediately set to inactive and all inverted signal outports are immediately set to active (0% duty cycle). ▪ 3: Brake duty cycle 100 All normal signal outports are immediately set to active and all inverted signal outports are immediately set to inactive (100% duty cycle).
Dependencies	-

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.

Current Space Vector

This function outport writes the current space vector [I_d , I_q , I_0] to the behavior model. The current space vector is the result of a DQZ transform that is performed on the measured phase currents.

Value range	-10^{+6} A ... 10^{+6} A for each entry in the vector.
Dependencies	–

Phase Current Vector

This function outport writes the phase current vector [I_a , I_b , I_c] to the behavior model. The entries of the phase current vector represent the motor phase currents that are evaluated from the measured voltages via signal offset and gain.

Value range	-10^{+6} A ... $+10^{+6}$ A for each entry in the vector.
Dependencies	Available only if the Capture phase current vector feature is enabled at the Feature activator property.

Timestamp

This function outport writes the time at which the motor phase currents are measured to the behavior model.

Value range	0 ... [Double.max] s
Dependencies	Available only if the Capture timestamps feature is enabled at the Feature activator property.

Angle Position

This function outport writes the angle value to the behavior model that is used for the DQZ transforms. This is the angle value provided by the specified angle source plus the offset angle specified at the Position offset property plus the angle value read from the Delta Position function port.

Value range	0 ... 360°
Dependencies	Available only if the Capture angle positions feature is enabled at the Feature activator property.

Pole Voltage Vector

This function outport writes the pole voltage vector [V_a , V_b , V_c] to the behavior model. The pole voltage vector is the transformed and modified voltage space vector [V_d , V_q , V_0]. The entries of the pole voltage vector are used to calculate the duty cycles for the generated PWM signals as follows:

$$\text{Duty cycle} = \frac{V_a, b, c}{V_{DC\ Link}} \cdot 100\% + 50\%$$

Value range	$-V_{DC\ Link}/2$... $+V_{DC\ Link}/2$ for each entry in the vector.
Dependencies	Available only if the Capture pole voltage vector feature is enabled at the Feature activator property.

Period Event

This event port provides an I/O event at the start and/or the center of each PWM period.

You have to specify the trigger condition as follows:

- The trigger point for an I/O event (via Event trigger condition property).
- The downsampling of occurred trigger points (via Event downsampling property).
- The time delay between the occurrence of a trigger point and the generation of the related I/O event (via Event delay property).

Value range	–
Dependencies	Available only if both of the following settings match: <ul style="list-style-type: none"> ▪ The ADC trigger source property is set to Free-running. ▪ The Event generation property is set to Enabled.

End of Conversion

This event port provides an I/O event each time a phase measurement is completed (end of A/D conversion).

Value range	–
Dependencies	Available only if both of the following settings match: <ul style="list-style-type: none"> ▪ The ADC trigger source property is set to PWM (internal). ▪ The Event generation property is set to Enabled.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal port when the digital outputs are operating as high-side switches or are in push-pull mode.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Field-Oriented Control In/Out) on page 1085.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Field-Oriented Control In/Out) on page 1085.
Dependencies	–

Reference

This signal port is a reference port and provides the low-side reference signal for the Signal signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Field-Oriented Control In/Out) on page 1085.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (Field-Oriented Control In/Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
PWM update mode	✓	-
Transform	✓	-
Harmonic amplitude	✓	-
Event trigger condition	✓	-
Event downsampling	✓	-
Event delay	✓	-
Trigger condition	✓	-
Trigger downsampling	✓	-
Trigger delay	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Input Interface		
Signal gains	✓	-
Signal offsets	✓	-
Filter time constant	✓	-
Electrical Output Interface		
Deadband	✓	-
Signal termination	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Field-Oriented Control In/Out)

Where to go from here	Information in this section
	<p>Adjusting the Function Block to the Connected Motor (Field-Oriented Control In/Out)..... 1070</p> <p>Configuring the Generation of PWM Signals (Field-Oriented Control In/Out)..... 1074</p> <p>Configuring the Trigger and I/O Event Generation Functionality (Field-Oriented Control In/Out)..... 1081</p> <p>Configuring Standard Features (Field-Oriented Control In/Out)..... 1084</p>

Adjusting the Function Block to the Connected Motor (Field-Oriented Control In/Out)

Overview	The function block type provides the following configuration features that adjust it to the connected motor: <ul style="list-style-type: none">▪ Specifying a pole pair factor.▪ Specifying the angular source that provides the electric motor position.▪ Specifying an offset angle for the electric motor position.▪ Specifying the analog signals to be measured and their processing (gain, offset, filtering).▪ Selecting the measurement range.
Specifying a pole pair factor	<p>Lets you specify a pole pair factor that is used as a conversion factor for the input position to calculate the electric angle position of the connected motor.</p> <p>The angle values provided by the master APU provider and/or from the behavior model must be converted to the electrical angle value of the connected motor. To achieve this match, you have to specify a pole pair factor.</p> <p>The setting of the pole pair factor depends on the characteristic of the provided input position (electrical or mechanical):</p> <ul style="list-style-type: none">▪ If an electrical position is provided, for example, by a Hall encoder the following cases apply:<ul style="list-style-type: none">▪ The pole pair number of the encoder (PPN_Encoder) and the pole pair number of the connected motor (PPN_Motor) match. In this case, set the pole pair factor to 1.

- The pole pair number of the connected motor is higher than the pole pair number of the encoder. In this case, set the pole pair factor to the ratio of PPN_Motor/PPN_Encoder.

Note

- You cannot use an encoder with a higher pole pair number than the connected electric motor, if the angle values are provided by a master APU provider.
- The ratio of PPN_Motor/PPN_Encoder must be an integer number. This means, for example, that a motor with three pole pairs cannot be used in combination with an encoder with two pole pairs.

Examples for Setting the Pole Pair Factor

PPN ¹⁾ of Encoder (Electrical Position is Provided)	PPN ¹⁾ of Connected Motor	Required Pole Pair Factor Setting
1	2	2
2	2	1
2	4	2

¹⁾ PPN = pole pair number

- If a mechanical position is provided (for example, from an incremental encoder) a conversion to an electrical position is required to determine the current sector of the motor. In this case, the pole pair factor must be set to the pole pair number of the connected electric motor.

Examples for Setting the Pole Pair Factor

PPN ¹⁾ of Encoder (Mechanical Position is Provided)	PPN ¹⁾ of Connected Motor	Required Pole Pair Factor Setting
1	2	2
1	4	4

¹⁾ PPN = pole pair number

- If the position can be provided from a master APU provider or from the behavior model (switchable via Angle Source function port) the factor applies to both positions. In this case, you have to specify a pole pair factor that matches to the provided position of the master APU provider. In the behavior model, the angle position must be converted to fit this pole pair factor and then provided via the Position function port to the function block. Refer to the examples below.

Examples for Setting the Pole Pair Factor

PPN ¹⁾ of Encoder A (Position Provided via Master APU Provider)	PPN ¹⁾ of Connected Motor	Required Pole Pair Factor Setting	PPN ¹⁾ of Encoder B (Position Provided via Behavior Model)	Required Conversion in Behavior Model
1	2	2	2	The angle value of encoder B must be divided by 2.
4	4	1	2	The angle value of encoder B must be multiplied by 2.
4	4	1	1	The angle value of encoder B must be multiplied by 4.

Examples for Setting the Pole Pair Factor				
PPN¹⁾ of Encoder A (Position Provided via Master APU Provider)	PPN¹⁾ of Connected Motor	Required Pole Pair Factor Setting	PPN¹⁾ of Encoder B (Position Provided via Behavior Model)	Required Conversion in Behavior Model
2	4	2	1	The angle value of encoder B must be multiplied by 2.
1	3	3	1	No conversion required.

¹⁾ PPN = pole pair number

Specifying the source of angle position values

You have to specify the source from which the current position of the electric motor (= angle position) is received. You can select from the following options:

- Master APU provider:
The angle value is received from a master APU provider that is assigned to the function block (via Assigned master APU provider property).
- Model:
The angle value is received from the behavior model via the Position function port.
- Master APU provider and model:
The angle value is received from a master APU provider that is assigned to the function block or from the behavior model via the Position function port. You can switch the angle source at run time via the Angle Source function port from the behavior model.

Tip

The last setting is useful, if you use a combination of an absolute and a relative encoder, for example, a Hall encoder and an incremental encoder, to measure the angle position of an electric motor.

Use scenario: The first encoder (in this example, the Hall encoder) detects the position of a motor immediately but provides less accurate values. The second encoder (in this example, the incremental encoder) provides more accurate values than the first encoder, but requires up to one revolution of the motor to measure a valid (absolute) position. To get exact position values at any time, the behavior model provides the position from the Hall encoder until a valid position from the higher precision incremental encoder is available via the master APU provider (switchable via the Angle Source function port).

Specifying an offset angle for the electric motor position

You can specify an offset angle at the Position offset property to adjust the angle value used for the transforms, for example, if the positions of the motor poles are different from 0°, 120°, and 240°. The specified value is added to the electric angle position of the motor so that it is applied for both DQZ and inverse DQZ transforms.

Note that you can define an additional offset angle in the behavior model, which determines an offset between DQZ and inverse DQZ transforms. For more information, refer to [Delta Position](#) on page 1064.

Specifying the analog signals to be measured and their processing (gain, offset, filtering)

Measured phases You have to specify the motor phases that are measured to evaluate the phase currents. You can select to measure all three phases (ABC) or a combination of two phases (AB, AC, or BC). In this case, the third phase current is automatically calculated by hardware. Note that this setting works only with balanced systems, where the sum of the phase currents is zero (0).

Tip

If your application requires field-oriented control for motors with more than three phases, contact dSPACE Support (www.dspace.com/go/supportrequest).

The phase currents are measured indirectly as voltages, for example, with current sensors. The measured voltages are converted into currents via signal gain values and signal offset values that you have to specify. Additionally, the measurement signals can be low-pass-filtered and you have to specify the filter time constant if filtering is required.

Signal gains You have to specify a gain factor (in Ampere/Volt) for each of the three phases.

Signal offsets You have to specify an offset value (in Volt) for each of the three phases. These values can be used, for example, to eliminate offset voltages of signals from sensors that have a unipolar supply voltage.

The specified signal gain values and offset values are used for calculating the phase currents from the measured voltages as follows:

$$I[A] = (\text{Meas. voltage}[V] - \text{Signal offset}[V]) \cdot \text{Signal gain} \left[\frac{A}{V} \right]$$

Note

You have to specify signal gain and offset values for all three phases even if only two phases are measured.

You can change the values of the signal gains and signal offsets via the experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Filter time constant You can activate a low-pass filter (PT1 filter) and specify its time constant on the basis of a specified number of samples (1 ... 256). The filter is active if the specified number of samples is greater than 1.

The following formulas apply for the effective time constant and the -3 dB cutoff frequency:

- Time constant [s] = $\frac{\text{Specified number of samples}}{\text{Sample rate}}$

- -3 dB cutoff frequency [Hz] = $\frac{\text{Sample rate}}{2\pi \cdot \text{Specified number of samples}}$

The value of the sample rate depends on the setting of the ADC trigger source property:

- For Free-running, the sample rate is the maximum possible sample rate of the assigned hardware resource. Refer to [Hardware Dependencies \(Field-Oriented Control In/Out\)](#) on page 1085.
- For PWM (internal), the sample rate depends on the signal frequency and the trigger condition for the A/D conversions:
 - If the trigger condition is Period start or Period center, the sample rate equals the PWM signal frequency.
 - If the trigger condition is Period start and center, the sample rate equals twice the PWM signal frequency.

The frequency value is defined in the behavior model and provided at the Frequency function port.

You can change the value of the filter time constant via the experiment software when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Selecting the measurement range

You have to select the measurement range for the analog input signals so that the range covers the expected input voltage level.

The voltage values for the low and high ranges depend on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Field-Oriented Control In/Out\)](#) on page 1085.

Configuring the Generation of PWM Signals (Field-Oriented Control In/Out)

Overview

The function block type provides the following configuration features to specify the generated PWM signals:

- Specifying the mode of DQZ and inverse DQZ transforms.
- Specifying the common mode injection signal for the pole voltage vector.
- Specifying the update mode for PWM signal generation.
- Providing inverted PWM signals.
- Enabling signal termination.
- Capturing additional data.

Specifying the mode of DQZ and inverse DQZ transforms

You have to specify which way DQZ and inverse DQZ transforms are performed.

- Amplitude-invariant transform:

For amplitude-invariant transforms, voltage and current magnitudes do not change during the transforms. This is the commonly used transform method for electric drive control.

- Power-invariant transform:

The power levels do not change during the transforms.

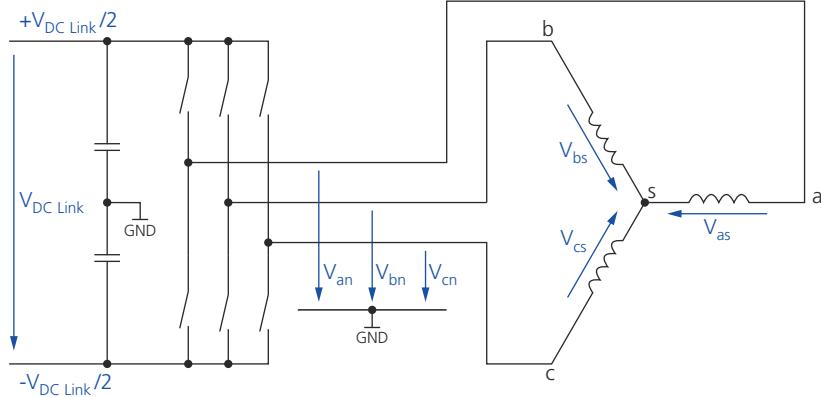
For information on the matrices used for all the transform variants, refer to [Basics on Field-Oriented Control on page 1056](#).

Specifying the common mode injection signal for the pole voltage vector

Depending on the use case, you can specify the generation of the pole voltages as pure sine waves or as sine waves modified by adding a common mode signal to the pole voltage vector. You have to select the common mode signal at the Modulation type property.

Wiring of the half bridge/inverter and the motor For clarification of the wiring and the related electric signals of the three-phase half bridge/inverter and the motor, refer to the following illustration first:

Three-phase half bridge/inverter



The voltages are specified as follows:

- $V_{DC\ Link}$:

▪ The DC Link voltage is the supply voltage of the three-phase half bridge/inverter. It is split into two halves $\pm V_{DC\ Link}/2$, which are referenced to ground.

- Pole voltages V_{an} , V_{bn} , V_{cn} :

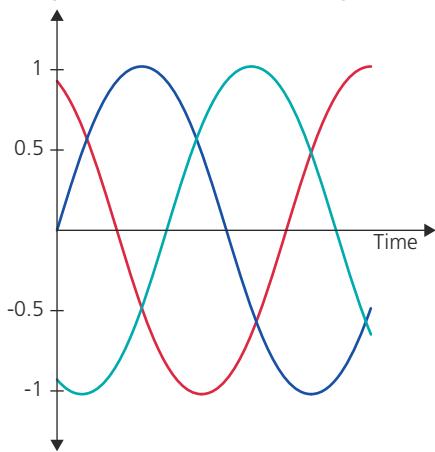
▪ The pole voltages are generated in the three-phase half bridge/inverter by the PWM signals. They are also referenced to ground. The pole voltages are either sine-shaped or modified by harmonic or min-max injection.

▪ The amplitudes of the pole voltages cannot exceed the voltage of the PWM signals, which is $\pm V_{DC\ Link}/2$, depending on the state of the half bridge/inverter switches.

- Phase voltages V_{as} , V_{bs} , V_{cs} :
- The phase voltages are the voltages across the motor windings.
- In contrast to the pole voltages, the phase voltages are not referenced to ground, their star point s is floating. Therefore, the phase voltages are always sine waves and not affected by common mode signals such as third harmonic or min-max injection.

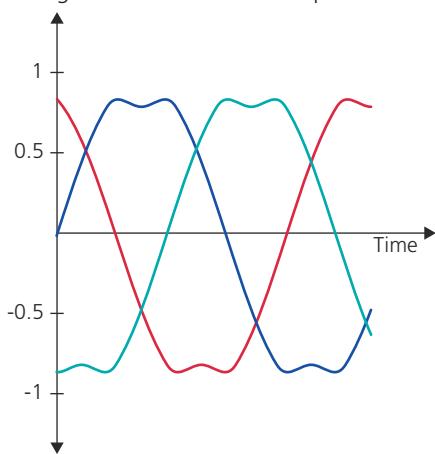
The selected common mode signal type (= modulation type) determines the shape and the amplitude of the pole voltages as follows:

Sine No common mode signal is added. The pole voltages resulting from the generated PWM signals are pure sine waves. Shape and amplitude of the pole voltage and the phase voltages are equal. Their maximum amplitude is $\pm V_{DC\ Link}/2$. Refer to the following illustration:



The amplitude scale is normalized to $\pm V_{DC\ Link}/2$.

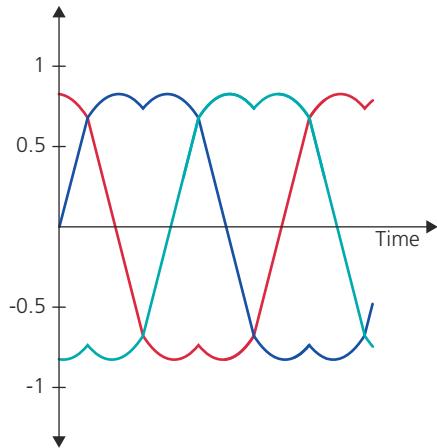
Harmonic injection A common third harmonic sine signal is added to the fundamental sine waves of the pole voltages. This changes the shape of the pole voltages and reduces their amplitude as shown in the following illustration:



The amplitude scale is normalized to $\pm V_{DC\ Link}/2$.

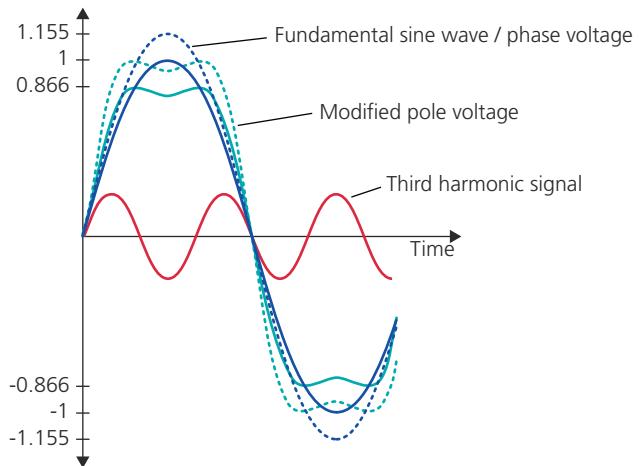
For harmonic injection, you can specify the amplitude of the added harmonic signal at the **Harmonic amplitude** property.

Min-max injection The common mode signal is calculated from momentary minimum and maximum values of the three fundamental sine waves. The resulting signal equals a triangle signal with three times the frequency of the fundamental and is added to the fundamental sine waves. This changes the shape and reduces the amplitude of the pole voltages as shown in the following illustration:



The amplitude scale is normalized to $\pm V_{DC\ Link}/2$.

Use case and effects of harmonic and min-max injection Third harmonic and min-max injection are used to increase the amplitude of the phase voltages beyond the DC link voltage as follows: The injections reduce the amplitude of the pole voltages by up to 15%. The amplitude of the fundamentals (= amplitude of the phase voltages) can then be increased until the amplitude of the flattened pole voltages reach the level of the DC link voltage. The following illustration shows the original signals (solid lines), the increased signals (dotted lines), and an exemplary third harmonic signal:



The amplitude scale is normalized to $\pm V_{DC\ Link}/2$.

Tip

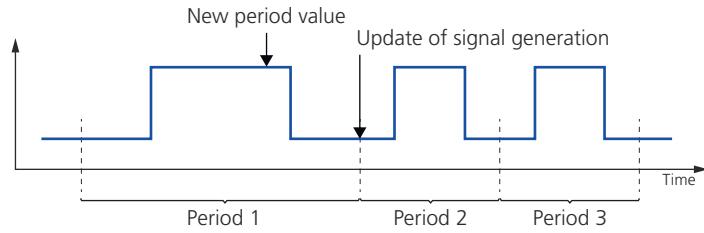
If your application requires signal modifications such as bus clamping, contact dSPACE Support (www.dspace.com/go/supportrequest).

Specifying the update mode for PWM signal generation

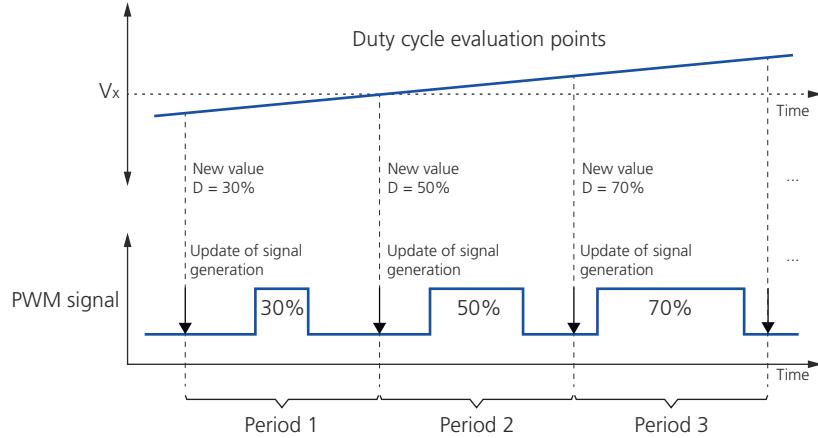
At run time, you can define new frequency values from within the behavior model. New duty cycle values are evaluated continuously by the function block if the value provided at the Output Pattern function port is Normal. For information on the behavior for other values, refer to [Output Pattern](#) on page 1065. During normal operation, the point in time of an update is essential for the generated PWM signals. With the PWM update mode property, you can specify the timing behavior of the update as follows:

- **Period start:**

In this mode, the PWM signals for all logical signals are updated at the period start. This means that new values do not take effect before the current PWM signal period is completed, as shown for period values in the following illustration:



New duty cycle values are evaluated continuously from the pole voltage vector. At the start of each signal period, the latest calculated value becomes the new duty cycle as shown in the following illustration:

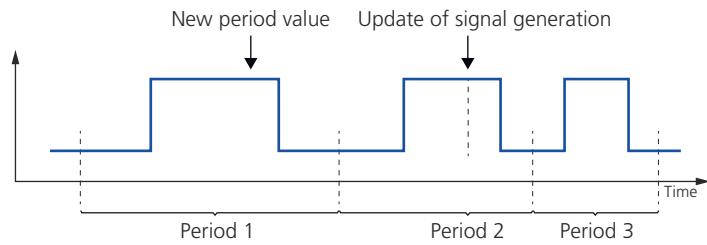


New values for the period and/or the duty cycle are updated at the start of a signal period, i.e., signal periods represent new values from the beginning.

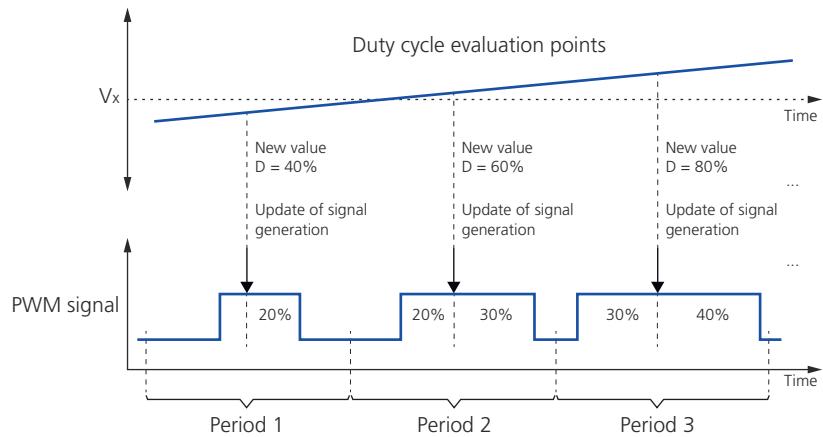
- **Period center:**

In this mode, the PWM signals for all logical signals are updated at the center of a PWM period. This means that new values do not take effect before the

center of a PWM period is generated, as shown for period values in the following illustration:



New duty cycle values are evaluated continuously from the pole voltage vector. At the center of each signal period, the latest calculated value becomes the new duty cycle as shown in the following illustration:



New values for the signal period and/or the duty cycle are updated at the center of a signal period, i.e., the first part of this signal period represents the half of the old values and the second part of this signal period represents the half of the new values.

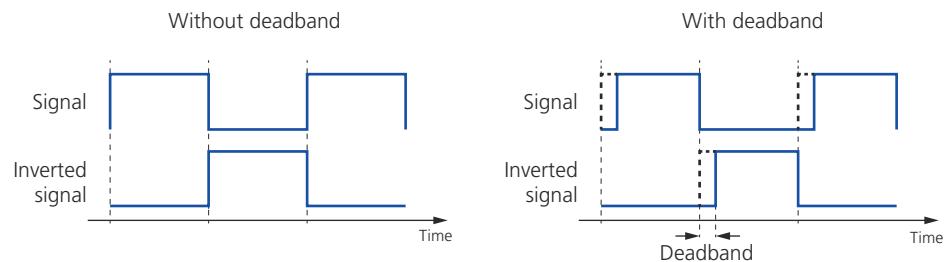
- **Period start and center:**

In this mode, the PWM signals for all logical signals are updated at the period start and at the period center.

Providing inverted signals

You can enable the generation of an additional inverted signal for each generated non-inverted signal with the **Inverted phase generation** property.

Avoiding shoot-through currents To avoid shoot-through currents on a connected driver, you can specify a deadband (time gap) between the pulses of the non-inverted signal and its inverted signal with the **Deadband** property. The deadband delays the edge generation of new pulses, as shown in the following illustration:



The value range of the deadband depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Field-Oriented Control In/Out\) on page 1085](#).

Enabling signal termination

To reduce signal wave reflections on the cable, a series resistor adapts the driver output impedance to the typical wave impedance of a connected cable.

Background: When the driver outputs a signal change, a wavefront is sent through the connected cable. The wavefront is reflected at the end of the connected cable if the cable is not terminated. The resulting voltage level is the sum of the voltage level of the forwarded wavefront and the voltage level of the reflected wavefront. However, signal termination at the external device eliminates voltage level overshoots caused by signal reflection, and you can drive the output signal without signal termination at the dSPACE hardware.

For a circuit diagram, refer to [Digital Out 8 on page 1581](#).

Capturing additional data

Depending on your requirements, you can enable capturing the following data and writing them to the behavior model via specific function ports:

- Timestamps
- Angle positions
- Pole voltage vector
- Phase current vector

The captured data can be helpful, for example, to perform plausibility checks in the behavior model. You are recommended to activate capturing only for required data to save computation time. For more information on the function ports that can be additionally activated and the data they provide, refer to [Overview of Ports and Basic Properties \(Field-Oriented Control In/Out\) on page 1061](#).

Configuring the Trigger and I/O Event Generation Functionality (Field-Oriented Control In/Out)

Overview

The function block type provides the following configuration features that influence the triggering of A/D conversions and the generation of I/O function triggers and I/O events:

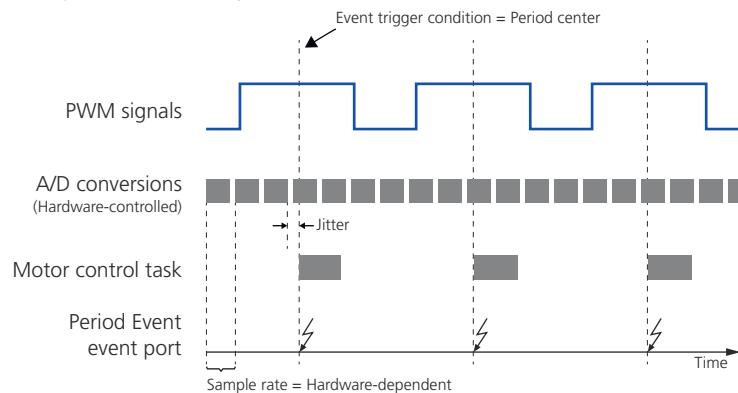
- Specifying the trigger source for starting A/D conversions.
- Specifying the conditions for the generation of I/O events and I/O function triggers (trigger point, downsampling value, time delay).

Specifying the ADC trigger source

You have to specify the trigger source that starts the A/D conversions for capturing the motor phase signals present at the signal imports.

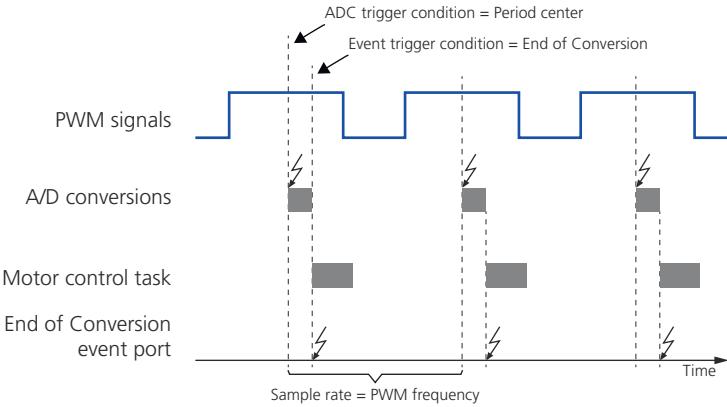
The following trigger sources are provided:

- Select Free-running for triggering A/D conversions independently from the behavior model. A/D conversions are then controlled by the assigned hardware resource and performed at the maximum available sample rate. I/O events and function trigger generation can be enabled and configured, for example, to trigger the motor control task. Note that A/D conversions and PWM signals are not synchronized and jitter occurs.



For specific sample rate values, refer to [Hardware Dependencies \(Field-Oriented Control In/Out\)](#) on page 1085.

- Select PWM (internal) for triggering A/D conversions by the PWM signal. You have to configure the trigger generation via the related properties. If enabled, I/O events are generated at the end of each A/D conversion. They can be used, for example, to start the motor control task. The I/O events are also provided at the End of Conversion event port.



The sample rate is different for the two trigger sources. Therefore, you might have to adjust the time constant of the low-pass filter at the **Filter time constant** property if you change the trigger source.

Providing I/O events and I/O function triggers

The function block can generate and provide I/O events and I/O function triggers.

I/O events I/O events can be used as trigger sources for runnable functions in the behavior model. To use the I/O events in the behavior model, you have to assign them to a task, for example, the motor control task. Depending on the setting of the ADC trigger source property, I/O events are generated as follows:

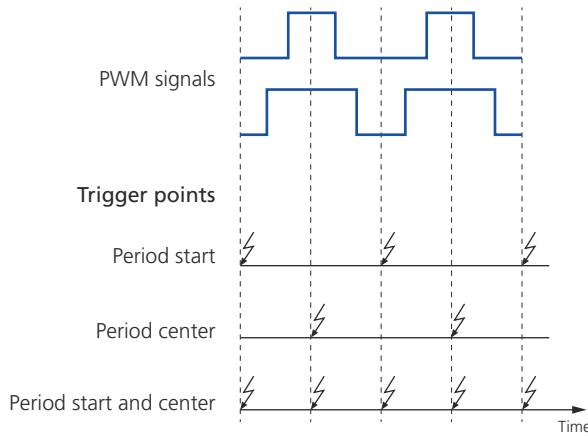
- If the ADC trigger source is **PWM (internal)**, an I/O event is generated each time an A/D conversion is completed. Event trigger conditions cannot be specified for this setting. You can assign the generated I/O events to a task via the **End of Conversion** property.
- If the ADC trigger source is **Free-running**, I/O events are generated according to the event trigger conditions, which you have to specify at the related properties. You can assign the generated I/O events to a task via the **Period Event** property.

For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#)

I/O Function triggers I/O function triggers can be used as a trigger source for other function blocks. If the ADC trigger source property is set to **PWM (internal)**, the generated I/O function triggers are also used to start the A/D conversions. In both cases, the specified trigger conditions apply.

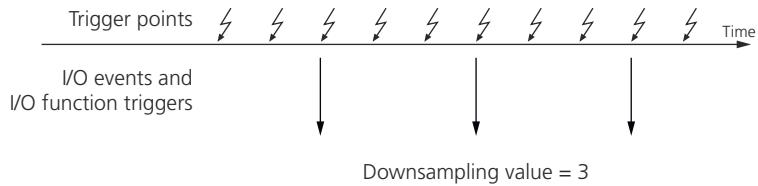
You have to specify the trigger conditions for generating I/O events and I/O function triggers (except for I/O events if the ADC trigger source is **PWM (internal)**). These are the trigger points, a downsampling value, and the delay for the generation.

Specifying the trigger point You can use period start, period center, or both points as a trigger point, as shown below.



Specifying a downsampling value The occurred triggers can be downsampled. The downsampling lets you specify whether to generate an I/O event or I/O function trigger on each trigger point or only on each n-th occurrence. The placeholder n is the specified downsampling value.

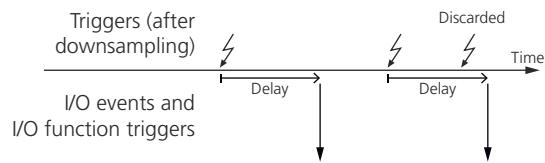
In the illustration below only every third trigger point is used for generation. Triggers that occurred in the intervening period are discarded.



The value range for the downsampling value depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Field-Oriented Control In/Out\)](#) on page 1085.

Specifying a time delay for generation You can specify a time delay between the occurrence of a trigger point and the generation of the related I/O event or I/O function trigger.

If a (downsampled) trigger initiates the generation of an I/O event or I/O function trigger, the function block delays the generation until the specified time interval is elapsed. During the delay, all other downsampled triggers are discarded as shown below.



The value range for the delay time depends on the assigned hardware resource. For specific values, refer to [Hardware Dependencies \(Field-Oriented Control In/Out\)](#) on page 1085.

Configuring Standard Features (Field-Oriented Control In/Out)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature		Channel Type Combination	More Information	
		Digital Out 8	Analog In 16	
Digital output signals	Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–	–
	High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs by setting the channel to high impedance (tristate) from the behavior model during run time.	–	–
	Polarity of output signals	✓	–	–
Analog input signals	Measurement range	–	✓	–

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 8](#) on page 1581
- [Analog In 16](#) on page 1552

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	More Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Field-Oriented Control In/Out)

SCALEXIO Hardware Dependencies (Field-Oriented Control In/Out)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 8/Analog In 16	
Hardware	DS6121 Multi-I/O Board	
Electrical Output Interface	Channel type	Digital Out 8
	Number of signals	6
	Frequency	Range 1 Hz ... 20 MHz
		Time resolution 8 ns
	Minimum pulse duration	25 ns
	Deadband	0 ... 400 µs
	Delay time (for I/O events and I/O function triggers)	0 ... 1.0 s
	Downsampling value (for I/O events and I/O function triggers)	1 ... 256
	Supported interface types for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull
		In push-pull configuration, the outputs can be set to high impedance (tristate) during run time.
	High-side reference voltage (high-side switch, push-pull)	+3.3 V and +5 V
	Current range	0 mA ... ±40 mA (each channel)
	Signal termination	<ul style="list-style-type: none"> ▪ 68 Ω (series termination) ▪ Switchable
	Circuit diagram	Digital Out 8 on page 1581
	Required channels	6 ¹⁾

Channel Type		Digital Out 8/Analog In 16
Electrical Input Interface	Channel type	Analog In 16
	Sample rate	0 ... 2 MS/s
	Measurement range	Low ²⁾
		±10 V (20 Vpp)
	Signal amplitude	High ²⁾
		±30 V (60 Vpp)
	Signal offset	Low ²⁾
		0 ... 20 Vpp
	Required channels	High ²⁾
Circuit diagram		Analog In 16 on page 1552

¹⁾ One channel for each non-inverted PWM signal and one channel for each inverted PWM signal.

²⁾ Setting of the Measurement range property.

³⁾ One channel for each measured motor phase current.

General limitations

- Channel multiplication is not supported in general.
- If the Field-Oriented Control In/Out function block works as a trigger function provider: The hardware resources assigned to the Field-Oriented Control In/Out function block and the hardware resources assigned to the function block using the Field-Oriented Control In/Out function block as trigger source must be located on the same I/O board.

More hardware data

DS6121 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration](#) (pdf)).

Bus Configuration

Introduction (Bus Configuration)

Where to go from here

Information in this section

Introduction to the Function Block (Bus Configuration).....	1088
Basics on Bus Configurations.....	1089

Introduction to the Function Block (Bus Configuration)

Function block purpose

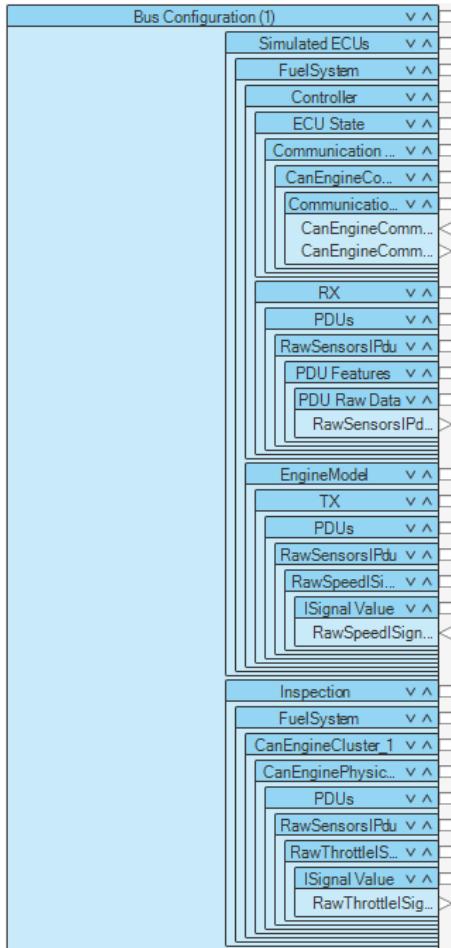
The Bus Configuration function block is the graphical representation of a bus configuration and provides access to its function ports and event ports. Bus configurations are the main element of the Bus Manager. They let you implement bus communication based on communication matrices in the signal chain.

For more information, refer to:

- Bus Manager in ConfigurationDesk: [Introduction to the Bus Manager \(ConfigurationDesk Bus Manager Implementation Guide](#) 
- Bus Manager (stand-alone): [Introduction to the Bus Manager \(Bus Manager Stand-Alone\) Implementation Guide](#) 

Block overview

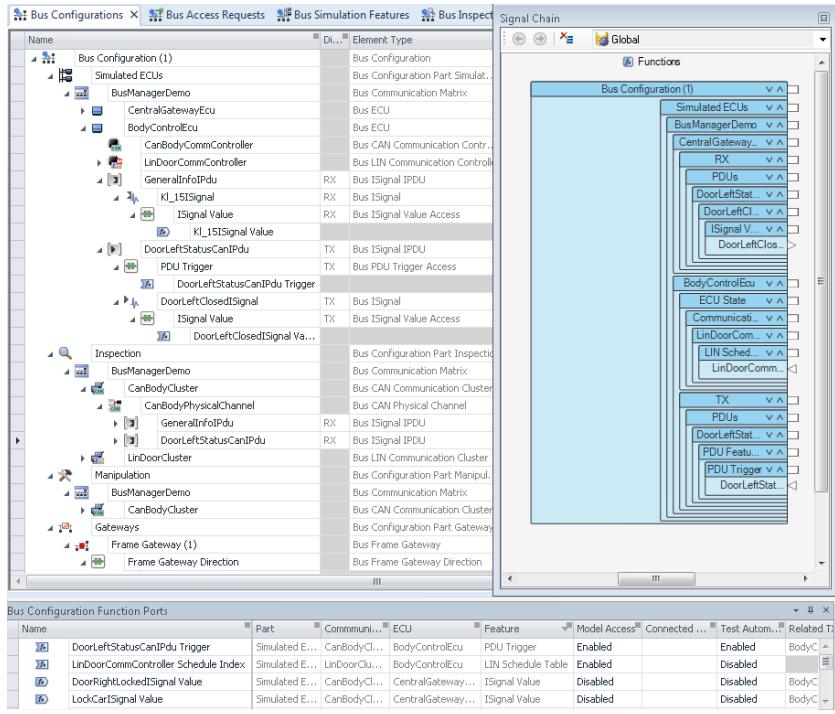
The following illustration shows an example of the function block.



Basics on Bus Configurations

Introduction

Bus configurations are the main element of the Bus Manager. They let you implement bus communication based on communication matrices in the signal chain. Each bus configuration can be accessed via specific tables and its Bus Configuration function block. The tables provide access to the complete structure of the bus configurations, e.g., to the ISignals you want to simulate, their function ports, the bus access requests, etc. The Bus Configuration function block can be accessed via working views and provides a structured overview of the function ports and event ports of a bus configuration.



For more information on working with bus configurations, refer to:

- Bus Manager in ConfigurationDesk: [ConfigurationDesk Bus Manager Implementation Guide](#)
- Bus Manager (stand-alone): [Bus Manager \(Stand-Alone\) Implementation Guide](#)

CAN

Where to go from here

Information in this section

Introduction (CAN).....	1092
Overviews (CAN).....	1094
Configuring the Function Block (CAN).....	1106
Hardware Dependencies (CAN).....	1115

Introduction (CAN)

Introduction to the Function Block (CAN)

Function block purpose

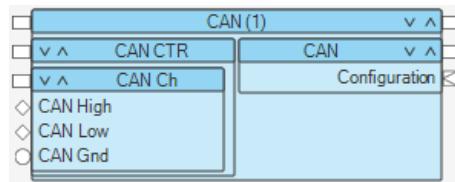
The CAN function block type is one part of implementing CAN communication in real-time applications. It lets you specify the hardware access for CAN communication. For example, you can configure transceiver settings for the assigned bus channel. The CAN communication itself must be modeled and supplied via one of the following providers:

- RTI CAN MultiMessage Blockset (refer to [Modeling a CAN Bus Interface \(Model Interface Package for Simulink - Modeling Guide\)](#))
- Bus Manager (refer to [Introduction to the Bus Manager \(ConfigurationDesk Bus Manager Implementation Guide\)](#))
- V-ECU implementations (refer to [Special Aspects of V-ECU Implementations Containing CAN Controllers \(ConfigurationDesk Real-Time Implementation Guide\)](#))
- Bus simulation containers (refer to [Working with Bus Simulation Containers \(ConfigurationDesk Real-Time Implementation Guide\)](#))
- ECU interface containers (refer to [ECU Interfacing with SCALEXIO or MicroAutoBox III Systems \(ConfigurationDesk Real-Time Implementation Guide\)](#))

In addition, you can use the CAN function block to initialize bus channels that are not involved in CAN communication. These channels can then be accessed during run time via ControlDesk or Real-Time Testing (RTT) (e.g., for monitoring purposes).

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Using CAN FD mode and specifying CAN FD settings.
- Using low-power mode.
- Enabling power wake-up.
- Using partial networking.

Supported channel types

The CAN function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III			
Channel type	Bus 1	CAN 1	CAN 2	CAN 3	CAN 4	CAN 5	CAN 6
Hardware	DS2671	DS2672	<ul style="list-style-type: none">▪ DS6301▪ DS6341▪ DS6342	<ul style="list-style-type: none">▪ DS1511▪ DS1511B1	DS1513	DS4342	DS1521

Oversviews (CAN)

Where to go from here

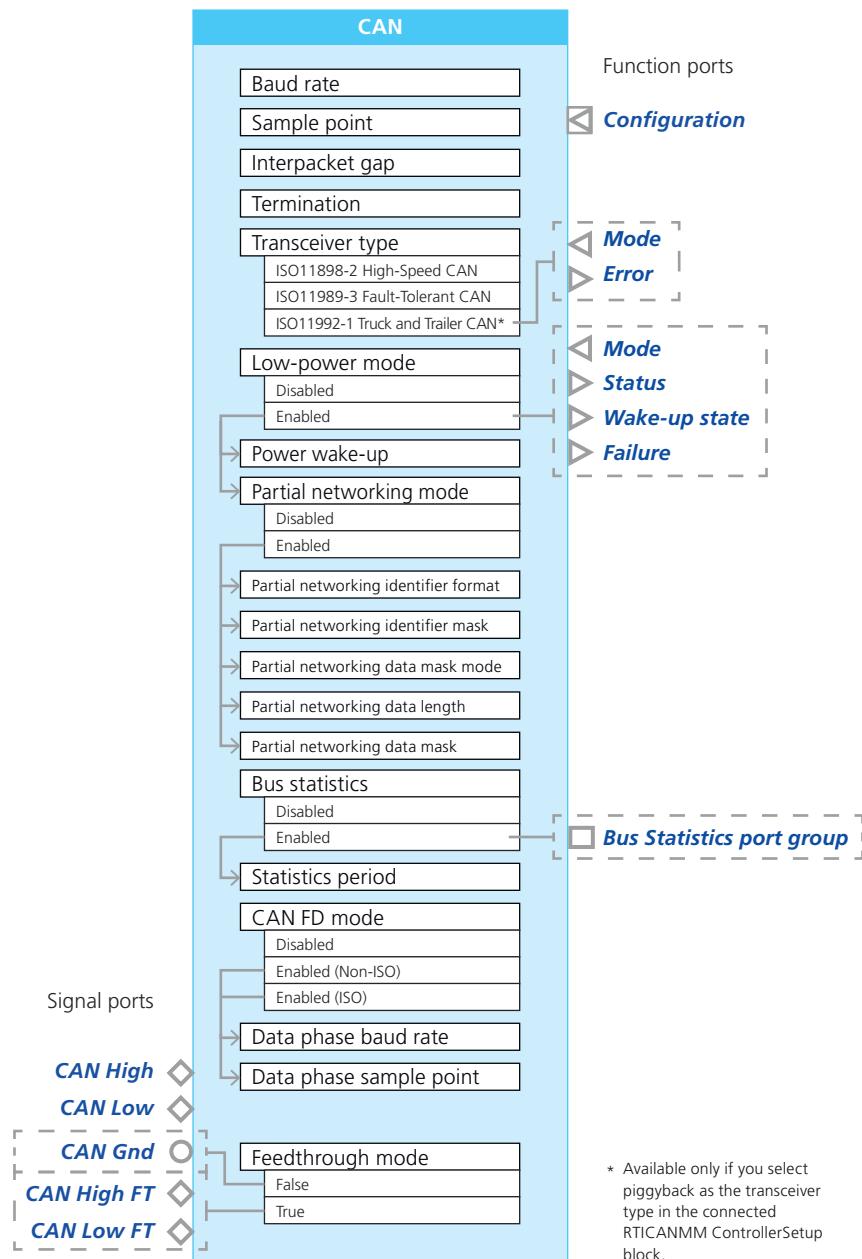
Information in this section

Overview of Ports and Basic Properties (CAN).....	1094
Overview of Tunable Properties (CAN).....	1104

Overview of Ports and Basic Properties (CAN)

Overview illustration

Function block The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



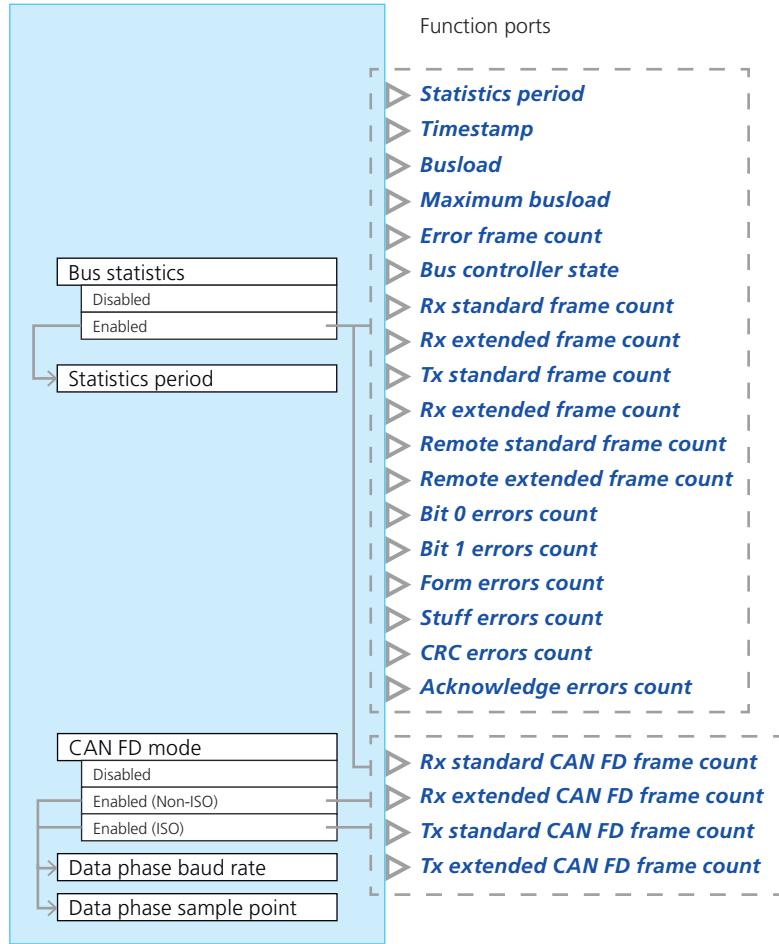
Function ports

Configuration**Mode****Error****Mode****Status****Wake-up state****Failure****Bus Statistics port group**

* Available only if you select piggyback as the transceiver type in the connected RTICANMM ControllerSetup block.

Bus Statistics port group This function port group is available only if the Bus statistics property is enabled.

The function ports of the Bus Statistics port group provide statistic information of the bus communication for the behavior model.



Configuration

This function import provides the interface for mapping one of the following blocks or controllers to the CAN function block:

- One RTICANMM ControllerSetup block of the RTI CAN MultiMessage Blockset
- One CAN controller of a V-ECU implementation
- One or more CAN controllers of one or more bus simulation containers (BSC files)

When mapped, this function port provides configuration data that configures the CAN function block and sets several function block properties to read-only.

Note

When you use the CAN function block with the Bus Manager or ECU Interface Configuration function blocks, you cannot use the block with RTICANMM ControllerSetup blocks or CAN controllers of V-ECU implementations or bus simulation containers. Do not map the Configuration function port in this case.

Mode (for the ISO 11992-1 Truck and Trailer CAN transceiver)

This function import receives the operation mode for the ISO 11992-1 Truck and Trailer CAN transceiver.

Tip

Depending on the configuration, there can be two Mode function ports for the CAN function block, one for the Control function and one for the Transceiver function. This Mode function port is available for the Control function.

Value range	<ul style="list-style-type: none"> ▪ 0: Off ▪ 1: Single-wire L ▪ 2: Single-wire H ▪ 3: Two-wire
Dependencies	Available only if the Transceiver type property is set to ISO11992-1 Truck and Trailer CAN.

Error

This function outport provides the error state of the ISO 11992-1 Truck and Trailer CAN transceiver.

Value range	<ul style="list-style-type: none"> ▪ 0: Error ▪ 1: No error
Dependencies	Available only if the Transceiver type property is set to ISO11992-1 Truck and Trailer CAN.

Mode (for low-power mode)

This function import receives the operation mode for the transceiver when low-power mode is enabled. For more information on the low-power mode, refer to [Using low-power mode](#) on page 1108.

Tip

Depending on the configuration, there can be two Mode function ports for the CAN function block, one for the Control function and one for the Transceiver function. This Mode function port is available for the Transceiver function.

Value range	<ul style="list-style-type: none"> ▪ 0: Standby mode The transceiver is inactive and unable to transmit or receive data. ▪ 1: Sleep mode The transceiver is inactive and unable to transmit or receive data. ▪ 2: Silent mode The transceiver is active but only listens on the bus, i.e., it receives data but does not transmit or acknowledge any data on the bus. ▪ 3: Normal mode The transceiver is active, i.e., it receives and transmits data on the bus.
Dependencies	Available only if the Low power mode property is set to Enabled.

Status

This function outport provides the operation mode of the transceiver when low-power mode is enabled. For more information on the low-power mode, refer to [Using low-power mode](#) on page 1108.

Value range	<ul style="list-style-type: none"> ▪ 0: Standby mode The transceiver is inactive and unable to transmit or receive data. ▪ 1: Sleep mode The transceiver is inactive and unable to transmit or receive data. ▪ 2: Silent mode The transceiver is active but only listens on the bus, i.e., it receives data but does not transmit or acknowledge any data on the bus. ▪ 3: Normal mode The transceiver is active, i.e., it receives and transmits data on the bus.
Dependencies	Available only if the Low power mode property is set to Enabled.

Wake-up state

This function outport provides the wake-up state of the transceiver when low-power mode is enabled. For more information on the low-power mode, refer to [Using low-power mode](#) on page 1108.

Value range	<ul style="list-style-type: none"> ▪ 0: <ul style="list-style-type: none"> ▪ No wake-up is detected. ▪ A wake-up is detected but the transceiver is in normal mode. ▪ 1: A wake-up is detected while the transceiver is in standby or sleep mode.
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	This value applies only to the current sampling step: The wake-up wakes up the transceiver, i.e., the transceiver operates in normal mode in the following sampling step.
Dependencies	Available only if the Low power mode property is set to Enabled.

Failure

This function outport indicates if the transceiver sets a failure flag while being in silent or normal mode. The transceiver sets a failure flag if it detects a failure in the transceiver control or on the lines of the bus channel (e.g., short circuit). The detectable failure types depend on the transceiver that is mounted on the bus channel of real-time hardware.

Value range	<ul style="list-style-type: none"> ▪ 0: No failure flag is set. ▪ 1: A failure flag is set.
Dependencies	Available only if the Low power mode property is set to Enabled.

Statistics period

This function outport provides the bus statistics period which is specified via the Statistics period property of the function block.

Value range	0.001 ... 30 s
Dependencies	Available only if the Bus statistics property is enabled.

Timestamp

This function outport provides the time stamp of the Bus Statistics function. The time stamp is set after the specified bus statistics period ends and the values of the Bus Statistics function are available at the related function ports.

Value range	0.0 ... [Double.max] s
Dependencies	Available only if the Bus statistics property is enabled.

Busload

This function outport provides the bus load that is measured on the channel.

Value range	0.0 ... 1.0
Dependencies	Available only if the Bus statistics property is enabled.

Maximum busload

This function outport provides the maximum bus load that is measured on the channel since the application was started.

Value range	0.0 ... 1.0
Dependencies	Available only if the Bus statistics property is enabled.

Error frame count

This function outport provides the number of error frames that occurred in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Bus controller state

This function outport provides the state of the bus channel's CAN controller according to the controller's Error Management Logic (EML). The EML operates with two counters, the Receive Error Counter and the Transmit Error Counter. When errors are detected on the bus (e.g., a faulty transmitted frame), the counter values are decremented and incremented according to the rules specified in the CAN specification.

Value range	<ul style="list-style-type: none"> ▪ 0: Undefined ▪ 1: The channel is not initialized. ▪ 2: Error active state The CAN controller is active and both counters are below the error warning limit. ▪ 3: Error warning state At least one counter is equal or exceeds the error warning limit. The CAN controller sets an error warning (EWRN) bit. ▪ 4: Error passive state At least one counter exceeds the error passive limit. The CAN controller is still active but sets a passive error flag. The controller can recover from this state itself. ▪ 5: Bus off state The Transmit Error Counter exceeds the bus off limit. The CAN controller disconnects itself from the bus. To recover, an external action is required (bus off recovery). ▪ 6: Sleep state The CAN controller is in sleep state, i.e., it does not transmit or receive data via the bus. This state is not part of the EML and it is therefore not determined by the counter values. The controller recovers from this state when it detects activity on the bus.
Dependencies	Available only if the Bus statistics property is enabled.

Rx standard frame count

This function outport provides the number of successfully received RX frames of standard identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Rx extended frame count This function outport provides the number of successfully received RX frames of extended identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Tx standard frame count This function outport provides the number of successfully received TX frames of standard identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Tx extended frame count This function outport provides the number of successfully received TX frames of extended identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Remote standard frame count This function outport provides the number of successfully received remote frames of standard identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Remote extended frame count This function outport provides the number of successfully received remote frames of extended identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Rx standard CAN FD frame count This function outport provides the number of successfully received RX CAN FD frames of standard identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the CAN FD mode property is set to Enabled (non-ISO) or Enabled (ISO) and the Bus statistics property is enabled.

Rx extended CAN FD frame count

This function outport provides the number of successfully received RX CAN FD frames of extended identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the CAN FD mode property is set to Enabled (non-ISO) or Enabled (ISO) and the Bus statistics property is enabled.

Tx standard CAN FD frame count

This function outport provides the number of successfully received TX CAN FD frames of standard identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the CAN FD mode property is set to Enabled (non-ISO) or Enabled (ISO) and the Bus statistics property is enabled.

Tx extended CAN FD frame count

This function outport provides the number of successfully received TX CAN FD frames of extended identifier format in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the CAN FD mode property is set to Enabled (non-ISO) or Enabled (ISO) and the Bus statistics property is enabled.

Bit 0 errors count

This function outport provides the number of bit 0 errors that occurred in the bus statistics period. The counter is incremented each time the CAN controller tries to send a dominant bus level but a recessive level is detected instead.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Bit 1 errors count

This function outport provides the number of bit 1 errors that occurred in the bus statistics period. The counter is incremented each time the CAN controller tries to send a recessive bus level but a dominant level is detected instead.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Form errors count

This function outport provides the number of format errors that occurred in the bus statistics period. The counter is incremented each time the format of a received frame deviates from the fixed format.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Stuff errors count

This function outport provides the number of stuff bit errors that occurred in the bus statistics period. The counter is incremented each time more than 5 consecutive equal bits occur in a part of a received frame where this is not allowed.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

CRC errors count

This function outport provides the number of checksum errors of a received frame that occurred in the bus statistics period.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

Acknowledge errors count

This function outport provides the number of acknowledge errors that occurred in the bus statistics period. The counter is incremented each time a frame sent by the CAN controller is not acknowledged.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Bus statistics property is enabled.

CAN High

This signal port is a bidirectional port and lets you transmit and receive CAN High signals.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (CAN) on page 1115.
Dependencies	–

CAN Low

This signal port is a bidirectional port and lets you transmit and receive CAN Low signals.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (CAN) on page 1115.
Dependencies	–

CAN Gnd

This signal port is a reference port and provides the reference signal for the CAN High and CAN Low signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (CAN) on page 1115.
Dependencies	Available only if the Feedthrough mode property is set to False.

CAN High FT

This signal port is a bidirectional port and lets you transmit and receive CAN High FT signals.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (CAN) on page 1115.
Dependencies	Available only if the Feedthrough mode property is set to True.

CAN Low FT

This signal port is a bidirectional port and lets you transmit and receive CAN Low FT signals.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (CAN) on page 1115.
Dependencies	Available only if the Feedthrough mode property is set to True.

Overview of Tunable Properties (CAN)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Partial networking identifier format	-	✓
	Partial networking identifier mask	-	✓
	Partial networking data mask mode	-	✓
	Partial networking data length	-	✓
	Partial networking data mask	-	✓
Function Ports			
	Initial switch setting (Test Automation)	-	✓
	Initial substitute value (Test Automation)	-	✓

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Electrical Interface	-	-	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (CAN)

Where to go from here

Information in this section

Configuring the Basic Functionality (CAN).....	1106
Configuring Standard Features (CAN).....	1113

Configuring the Basic Functionality (CAN)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Using CAN FD mode and specifying CAN FD settings.
- Using CAN bus termination.
- Using feedthrough mode.
- Using low-power mode.
- Using partial networking.
- Enabling power wake-up.
- Specifying the minimum time gap between two CAN frames.
- Using bus statistics.

Using CAN bus termination

The CAN function block lets you terminate the CAN bus.

Basics on CAN bus termination To avoid reflections on the bus, CAN bus lines must be terminated:

- High-speed CAN (according to ISO 11898-2)

The bus lines are terminated at each end, typically by termination resistors of $120\ \Omega$ each.

- Fault-tolerant CAN (according to ISO 11898-3)

Each bus member provides a termination. The required termination resistance of each bus member depends, for example, on the number of bus members, the bus topology, or the length of the bus lines. In total, the minimum termination resistance of the bus should at least be $100\ \Omega$.

Terminating a CAN bus via the CAN function block To terminate a CAN bus, you can use the termination resistors of dSPACE real-time hardware.

Depending on channel type and the transceiver type, the CAN function block lets you enable and disable termination or switch the termination resistance. For information on the available termination resistors, refer to [Hardware Dependencies \(CAN\)](#) on page 1115.

Note

- If you specify CAN communication via the RTI CAN MultiMessage Blockset, you must specify the termination in the **RTICANMM ControllerSetup** block. This setting is provided to the CAN function block when you map the configuration ports. You cannot change the specified termination via the CAN function block.
- If you enable Feedthrough mode for the CAN 5 or CAN 6 channel type, the termination of the channel is automatically disabled. For a circuit diagram, refer to [CAN 5](#) on page 1567 or [CAN 6](#) on page 1568, respectively.

Using feedthrough mode

The CAN function block lets you use feedthrough mode for the ISO 11898-2 High-Speed CAN transceiver type.

Basics on feedthrough mode Depending on the individual CAN bus topology, the SCALEXIO or MicroAutoBox III system might be connected at a stub of the CAN bus. Long stubs can reduce the signal quality and result in reflections on the bus. If the SCALEXIO or MicroAutoBox III system is connected at a stub, you can increase the signal quality and avoid reflections by enabling the feedthrough mode. When you do this, the feedthrough lines of the selected channel type are used for the bus communication. This reduces the stub length significantly.

For an overview of the channel types that support feedthrough mode, refer to [Hardware Dependencies \(CAN\)](#) on page 1115.

Enabling feedthrough mode It depends on your CAN bus topology if enabling feedthrough mode is useful:

- If the SCALEXIO or MicroAutoBox III system is connected at the end of the CAN bus, its bus lines must be terminated. Enabling feedthrough mode might reduce the signal quality in this case. Therefore, it is recommended not to enable feedthrough mode.
- If the SCALEXIO or MicroAutoBox III system is not connected at the end of the CAN bus, enabling feedthrough mode is often useful to reduce the stub length. When you do this, feedthrough signal ports are available for the CAN function block and the feedthrough pins of the SCALEXIO or MicroAutoBox III system are used for the bus signals.

NOTICE

Enabling the Feedthrough mode has the following effects on the Bus 1 channel type:

- The galvanic isolation is deactivated. The hardware can be destroyed if the connected CAN bus carries high voltages.
- The transceiver's VBAT pin is internally switched to +12 V supply voltage and the transceiver's GND pin is internally switched to system GND.

Refer to [Bus 1](#) on page 1560.

Note

If you enable Feedthrough mode for the CAN 5 or CAN 6 channel type, the termination of the channel is automatically disabled. For a circuit diagram, refer to [CAN 5](#) on page 1567 or [CAN 6](#) on page 1568, respectively.

Using low-power mode

The CAN function block supports low-power mode according to ISO 11898-2:2016 (formerly ISO 11898-5) for the ISO 11898-2 High-Speed CAN and ISO 11898-3 Fault-Tolerant CAN transceiver types.

Basics on low-power mode Low-power mode is used to reduce the power consumption (e.g., in a vehicle) by deactivating bus members that do not have to receive or transmit data. According to ISO 11898-2:2016, a wake-up that is sent via the bus wakes up all the deactivated bus members.

Enabling low-power mode When you enable low-power mode for the CAN function block, the transceiver of the assigned bus channel and real ECUs that are connected to the channel can change their operation mode, for example, depending on the bus communication during run time.

The channel's transceiver can operate in one of the following modes:

Mode	Purpose
Standby mode	The transceiver is inactive and unable to transmit or receive data.
Sleep mode	The transceiver is inactive and unable to transmit or receive data.
Silent mode	The transceiver is active but only listens on the bus, i.e., it receives data but does not transmit or acknowledge any data on the bus.
Normal mode	The transceiver is active, i.e., it receives and transmits data on the bus.

Tip

Sleep mode according to ISO 11898-2:2016 means that not only the transceiver but the entire ECU is deactivated.

If the transceiver is in standby or sleep mode and a wake-up is detected on the bus, the transceiver is woken up and set to normal mode. If the transceiver is in any other mode while a wake-up is detected on the bus, the wake-up has no effect on the transceiver's operation mode.

Note

When you enable low-power mode, the transceiver can only react on wake-ups. It cannot change its operation mode in any other case, e.g., go to sleep mode if there is no data to transmit or receive. Instead, you must set the transceiver's mode explicitly via the function ports that are available for the low-power mode.

Providing low-power mode settings via function ports Enabling low-power mode adds the following function ports to the CAN function block:

Function Port	Purpose
Mode	Lets you specify the operation mode of the transceiver, e.g., set the transceiver to sleep mode. If a wake-up is sent via the bus and the transceiver mode is set via the function port in the same sampling step, the last received command is executed. For example, the transceiver is in standby mode. Now, the transceiver is set to silent mode via the function port and a wake-up is sent in the same sampling step but the function port setting is received last by the real-time hardware. In this case, the transceiver is set to silent mode.
Status	Provides the operation mode of the transceiver, e.g., to a mapped behavior model.
Wake-up state	Provides the wake-up state of the transceiver, e.g., to a mapped behavior model.
Failure	Indicates if the transceiver has set a failure flag while being in silent or normal mode.

Using partial networking

The CAN function block supports partial networking according to ISO 11898-2:2016 (formerly ISO 11898-6) for the ISO 11898-2 High-Speed CAN transceiver type.

Basics on partial networking Partial networking is used to reduce the power consumption (e.g., in a vehicle) by deactivating bus members that do not have to receive or transmit data. According to ISO 11898-2:2016 and in contrast to low-power mode, partial networking supports selective wake-up, i.e., bus members are woken up when and for as long as required.

Enabling partial networking You can enable partial networking only if low-power mode is enabled for the CAN function block. When partial networking is enabled, the transceiver of the assigned bus channel operates in the same modes as with low-power mode but the wake-up behavior differs. The transceiver wakes up from sleep or standby mode only if it receives a valid wake-up frame. Other bus members (i.e., real ECUs connected to the channel) do not wake-up automatically but only if they receive valid wake-up frames themselves.

Filtering valid wake-up frames Wake-up frames must be defined outside of ConfigurationDesk (e.g., in a communication matrix). The CAN function block lets you filter wake-up frames by their identifier and/or frame data. Only wake-

up frames that pass the specified filters are used as wake-up frames for the transceiver.

For the filtering properties of the CAN function block, TRC file variables are generated. You can access these variables via experiment software and change the specified settings during run time.

Filtering wake-up frames by frame identifiers To filter wake-up frames by the frame identifier, you can specify an identifier mask and define the identifier format (standard or extended) that is masked. This allows you, for example, to mask only 11 identifier bits (standard format) while receiving wake-up frames with 29 identifier bits (extended format) or specifying more than 11 bit for the identifier mask.

If the masked identifier bits of a wake-up frame pass the identifier mask, the frame is used as valid wake-up frame for the transceiver. However, if you enabled the filtering of the frame data as well, the frame must pass the data filter to wake-up the transceiver.

Filtering wake-up frames by frame data The frame data of wake-up frames is used to wake-up specific network nodes. Each bit of the frame data can be related to one or more network nodes. A bit value of 1 wakes up the related network nodes.

To filter wake-up frames by the frame data, you must specify the number of data bytes and a data mask. Only wake-up frames with the specified number of data bytes are used as valid wake-up frames. Wake-up frames with more or less data bytes are ignored. By specifying a data mask, you can mask the frame bits to filter the network nodes that are woken up.

The following table is an example for a wake-up frame with a data length of 1 byte that is masked by a data mask. The wake-up frame would wake up all the network nodes that are related to bit 1, bit 3, and bit 5. Due to the specified data mask, only the network nodes related to bit 1 and bit 3 are woken up.

	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Wake-up frame	1	0	1	0	1	0	0	0
Data mask	1	1	1	0	0	0	1	0
Wake-up	✓	-	✓	-	-	-	-	-

Valid baud rates for partial networking The CAN function block lets you specify a baud rate for CAN communication. Partial networking is supported only for the following baud rates:

- 50 kBaud
- 100 kBaud
- 125 kBaud
- 250 kBaud
- 500 kBaud
- 1000 kBaud

If you specify another baud rate, normal CAN communication is still possible but no wake-up frames can be detected. However, in this case the transceiver's error

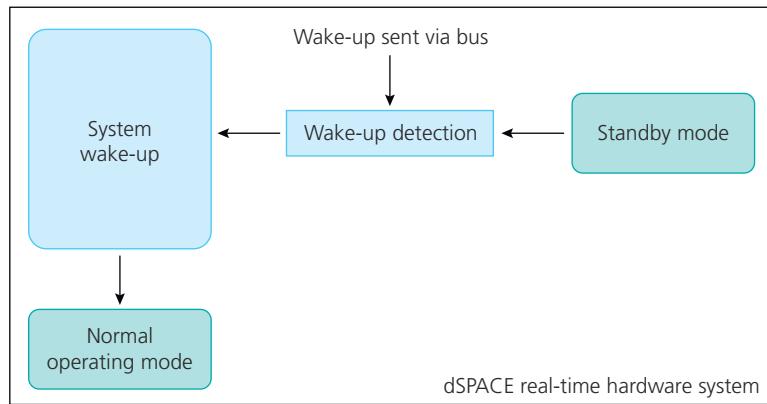
counter is incremented with each received frame. When the error counter's buffer is full, the transceiver wakes up.

Using partial networking for CAN communication modeled with the RTI CAN MultiMessage Blockset The RTI CAN MultiMessage Blockset lets you specify partial networking settings as well. These settings cannot be used with SCALEXIO systems. When you implement CAN communication modeled with the RTI CAN MultiMessage Blockset in a real-time application, the partial networking settings you specified via the RTI CAN MultiMessage Blockset are ignored and the settings specified in the CAN function block are used instead.

Enabling power wake-up

The CAN function block lets you enable power wake-up for the ISO 11898-2 High-Speed CAN and ISO 11898-3 Fault-Tolerant CAN transceiver types.

Basics on power wake-up When you enable power wake-up for the CAN function block, a CAN wake-up that is sent on the CAN bus can wake up a dSPACE real-time hardware system. In this case, the system can change its operation mode from standby to normal operating mode, as shown in the following example.



The following dSPACE real-time hardware systems support power wake-up via CAN:

- SCALEXIO AutoBox/LabBox systems based on a DS6001 Processor Board
- MicroAutoBox III systems

A typical use scenario for using power wake-up is to wake up the real-time hardware system via a CAN wake-up and to start the real-time application automatically.

Preconditions for waking up a real-time hardware system and starting the real-time application To wake up a real-time hardware system via a CAN wake-up and to start the real-time application automatically, the following preconditions must be fulfilled:

- The configuration of the real-time hardware system supports power wake-up. For more information on the required configurations, refer to:
 - SCALEXIO AutoBox/LabBox systems: [Waking up SCALEXIO AutoBoxes/LabBoxes via Bus Signals \(SCADEXIO – Hardware and Software Overview\)](#).

- MicroAutoBox III systems: [Powering Features \(MicroAutoBox III Hardware Installation and Configuration\)](#).
- A hardware resource that supports power wake-up is assigned to the CAN function block. Refer to [Hardware Dependencies \(CAN\)](#) on page 1115.
- The Power wake-up property of the CAN function block is set to Enabled.
- The real-time application is loaded to the flash memory of the real-time hardware system.

Tip

This is required only for starting the real-time application automatically when waking up the real-time hardware system. If the real-time application is not loaded to the flash memory, the system can be woken up but no application is executed.

Refer to [Downloading and Executing Real-Time Applications \(ConfigurationDesk Real-Time Implementation Guide\)](#).

- The transceiver of the used CAN channel is set to sleep mode *before* the real-time hardware system is set to standby mode.
You must explicitly set the transceiver to sleep mode via the Mode function port. Refer to [Using low-power mode](#) on page 1108.
- The real-time hardware system is set to standby mode *after* the transceiver is set to sleep mode.
It is recommended to set the system to standby mode by using the System Shutdown function block. Refer to [Basics on Using the System Shutdown Functionality](#) on page 1407.

If the preconditions are fulfilled and a wake-up is sent on the CAN bus, the real-time hardware system is woken up. The real-time application starts from the flash memory, and the transceiver of the CAN channel is set to normal mode.

Tip

If you work with a system of SCALEXIO AutoBoxes/LabBoxes that are connected via the power control bus, all the SCALEXIO AutoBoxes/LabBoxes wake up when at least one of them is woken up via a CAN wake-up.

Specifying the minimum time gap between two CAN frames

The CAN function block lets you specify the minimum time gap between two CAN frames.

Bus members often have different physical performances. For example, the processing resources of real ECUs might be lower than the resources of dSPACE real-time hardware. Therefore, CAN frames might be transmitted too fast for receiving bus members to decode consecutive CAN frames and frame data.

If this is the case, you can specify the minimum time gap between two CAN frames via the Frame transmission gap property. When you do this, you can ensure that all receiving bus members can decode consecutive CAN frames and no frame data is lost.

Configuring Standard Features (CAN)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO					
Configuration Feature	Bus 1	CAN 1	CAN 2	Further Information	
Measurement point	–	–	–	Specifying the Measurement Point for Input Signals on page 119	
Interface type	–	–	–	Specifying the Circuit Type for Voltage Input Signals on page 116	
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95	
Signal Ports					
Trigger level of electronic fuse	–	–	–	Specifying Fuse Settings on page 122	
Failure simulation support	✓	✓	–	Specifying Settings for Failure Simulation on page 123	
Usage of loads	–	–	–	Specifying Load Settings on page 121	
MicroAutoBox III					
Configuration Feature	CAN 3	CAN 4	CAN 5	CAN 6	Further Information
Measurement point	–	–	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	–	–	–	–	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports					
Trigger level of electronic fuse	–	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	–	–	–	–	Specifying Settings for Failure Simulation on page 123

MicroAutoBox III					
Configuration Feature	CAN 3	CAN 4	CAN 5	CAN 6	Further Information
Usage of loads	–	–	–	–	Specifying Load Settings on page 121

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Bus 1 on page 1560 ▪ CAN 1 on page 1563 ▪ CAN 2 on page 1564 	<ul style="list-style-type: none"> ▪ CAN 3 on page 1566 ▪ CAN 4 on page 1566 ▪ CAN 5 on page 1567 ▪ CAN 6 on page 1568

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (CAN)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (CAN).....	1115
MicroAutoBox III Hardware Dependencies (CAN).....	1117

SCALEXIO Hardware Dependencies (CAN)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	Bus 1	CAN 1	CAN 2
Hardware	DS2671 Bus Board	DS2672 Bus Module	<ul style="list-style-type: none"> ▪ DS6301 CAN/LIN Board ▪ DS6341 CAN Board ▪ DS6342 CAN Board
Transceiver type ¹⁾	<ul style="list-style-type: none"> ▪ ISO 11898-2 High-Speed CAN: TJA1043T ▪ ISO 11898-3 Fault-Tolerant CAN: TJA1054T ▪ ISO 11992-1 Truck and Trailer CAN²⁾ 	<ul style="list-style-type: none"> ▪ ISO 11898-2 High-Speed CAN: TJA1043T ▪ ISO 11898-3 Fault-Tolerant CAN: TJA1054T 	<ul style="list-style-type: none"> ▪ ISO 11898-2 High-Speed CAN: TJA1145/FD ▪ ISO 11898-3 Fault-Tolerant CAN: TJA1055T ▪ ISO 11992-1 Truck and Trailer CAN²⁾
Protected voltage range	-60 V ... +60 V ³⁾	-60 V ... +60 V	-58 V ... +58 V
Baud rate range	<ul style="list-style-type: none"> ▪ 40 kBd ... 1 MBd (ISO 11898-2 High-Speed CAN) ▪ 40 kBd ... 125 kBd (ISO 11898-3 Fault-Tolerant CAN) ▪ 125 kBd or 250 kBd (ISO 11992-1 Truck and Trailer CAN) 	<ul style="list-style-type: none"> ▪ 40 kBd ... 1 MBd (ISO 11898-2 High-Speed CAN) ▪ 40 kBd ... 125 kBd (ISO 11898-3 Fault-Tolerant CAN) ▪ 125 kBd or 250 kBd (ISO 11992-1 Truck and Trailer CAN) 	<ul style="list-style-type: none"> ▪ 15 kBd ... 1 MBd (ISO 11898-2 High-Speed CAN) ▪ 40 kBd ... 125 kBd (ISO 11898-3 Fault-Tolerant CAN) ▪ 125 kBd or 250 kBd (ISO 11992-1 Truck and Trailer CAN)
CAN FD mode	<ul style="list-style-type: none"> ▪ Supported only for the ISO 11898-2 High-Speed CAN transceiver ▪ ISO CAN FD ▪ Non-ISO CAN FD 	<ul style="list-style-type: none"> ▪ Supported only for the ISO 11898-2 High-Speed CAN transceiver ▪ ISO CAN FD ▪ Non-ISO CAN FD 	<ul style="list-style-type: none"> ▪ Supported only for the ISO 11898-2 High-Speed CAN transceiver ▪ ISO CAN FD ▪ Non-ISO CAN FD
Data phase baud rate range (supported only for CAN FD mode)	40 kBd ... 8 MBd ^{3), 4)}	40 kBd ... 8 MBd ⁴⁾	15 kBd ... 8 MBd

Channel type	Bus 1	CAN 1	CAN 2
Piggyback configuration	✓ (supported only for the ISO 11992-1 Truck and Trailer CAN transceiver)	–	✓ (supported only for the ISO 11992-1 Truck and Trailer CAN transceiver)
Termination	<ul style="list-style-type: none"> ▪ Switchable termination ▪ Parallel termination between CAN High and CAN Low 	<ul style="list-style-type: none"> ▪ Switchable termination ▪ Parallel termination between CAN High and CAN Low 	<ul style="list-style-type: none"> ▪ Switchable termination ▪ Parallel termination between CAN High and CAN Low
Termination resistance	<ul style="list-style-type: none"> ▪ 96 Ω or not terminated (ISO 11898-2 High-Speed CAN) ▪ 560 Ω or 5.6 kΩ (ISO 11898-3 Fault-Tolerant CAN) 	<ul style="list-style-type: none"> ▪ 120 Ω or not terminated (ISO 11898-2 High-Speed CAN) ▪ 560 Ω or 5.6 kΩ (ISO 11898-3 Fault-Tolerant CAN) 	<ul style="list-style-type: none"> ▪ 120 Ω or not terminated (ISO 11898-2 High-Speed CAN) ▪ 560 Ω or 5.6 kΩ (ISO 11898-3 Fault-Tolerant CAN)
Feedthrough mode	✓ (supported only for the ISO 11898-2 High-Speed CAN transceiver)	–	✓ (supported only for the ISO 11898-2 High-Speed CAN transceiver)
Power wake-up	–	–	✓ (supported only if the board is installed in a SCALEXIO LabBox system that is based on a DS6001 Processor Board)
Partial networking	–	–	✓ (supported only for the ISO 11898-2 High-Speed CAN transceiver)
Failure simulation	✓	✓	–
Galvanic isolation	✓ (only if feedthrough mode is disabled)	–	–
Circuit diagram	Bus 1 on page 1560	CAN 1 on page 1563	CAN 2 on page 1564
Required channels	1	1	1

¹⁾ The specifications of the mounted transceiver might be different. dSPACE might change the transceiver used without notice. This change does not affect the board revision number.

²⁾ Supported only by the RTI CAN MultiMessage Blockset. Available only if the function block is connected to a Configuration Port block, and if you selected piggyback as the transceiver type in the connected RTICANMM ControllerSetup block in the Simulink model.

³⁾ As of revision DS2671-09.

⁴⁾ The maximum data phase baud rate can be achieved with an optimized cable management under laboratory conditions, i.e., differential signal lines, 96 Ω wave impedance, no stubs.

More hardware data

DS2671 Bus Board For more board-specific data, refer to [DS2671 Bus Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2672 Bus Module For more board-specific data, refer to [DS2672 Bus Module \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6301 CAN/LIN Board For more board-specific data, refer to [DS6301 CAN/LIN Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6341 CAN Board For more board-specific data, refer to [DS6341 CAN Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6342 CAN Board For more board-specific data, refer to [DS6342 CAN Board \(SCALEIO Hardware Installation and Configuration](#) ().

MicroAutoBox III Hardware Dependencies (CAN)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	CAN 3	CAN 4	CAN 5	CAN 6
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board 	DS1513 Multi-I/O Board	DS4342 CAN FD Interface Module	DS1521 Bus Board
Transceiver type	ISO 11898-2 High-Speed CAN: MAXB13052ASA+	ISO 11898-2 High-Speed CAN: TJA1145T/FD	ISO 11898-2 High-Speed CAN: TJA1145T/FD	ISO 11898-2 High-Speed CAN: TJA1145T/FD
Protected voltage range	<ul style="list-style-type: none"> ▪ -80 V ... +80 V between a CAN line and GND ▪ -7.75 V ... +7.75 V between CAN High and CAN Low 	<ul style="list-style-type: none"> ▪ -58 V ... +58 V between a CAN line and GND ▪ -7.75 V ... +7.75 V between CAN High and CAN Low 	<ul style="list-style-type: none"> ▪ -58 V ... +58 V between a CAN line and GND ▪ -7.75 V ... +7.75 V between CAN High and CAN Low 	<ul style="list-style-type: none"> ▪ -58 V ... +58 V between a CAN line and GND ▪ -10.95 V ... +10.95 V between CAN High and CAN Low
Baud rate range	15 kBd ... 1 MBd	15 kBd ... 1 MBd	15 kBd ... 1 MBd	15 kBd ... 1 MBd
CAN FD mode	–	–	<ul style="list-style-type: none"> ▪ ISO CAN FD ▪ Non-ISO CAN FD 	<ul style="list-style-type: none"> ▪ ISO CAN FD ▪ Non-ISO CAN FD
Data phase baud rate range (supported only for CAN FD mode)	–	–	15 kBd ... 8 MBd	15 kBd ... 8 MBd
Termination	–	<ul style="list-style-type: none"> ▪ Switchable termination: Enabled or disabled ▪ Enabled: Parallel termination between CAN High and CAN Low 	<ul style="list-style-type: none"> ▪ Indirect switchable termination: ▪ Automatically enabled if feedthrough mode is disabled ▪ Automatically disabled if feedthrough mode is enabled ▪ Enabled: Parallel termination between CAN High and CAN Low 	<ul style="list-style-type: none"> ▪ Indirect switchable termination: ▪ Automatically enabled if feedthrough mode is disabled ▪ Automatically disabled if feedthrough mode is enabled ▪ Enabled: Parallel termination between CAN High and CAN Low
Termination resistance	–	120 Ω or not terminated	120 Ω or not terminated	120 Ω or not terminated

Channel type	CAN 3	CAN 4	CAN 5	CAN 6
Feedthrough mode	–	–	✓	✓
Power wake-up	–	✓	✓	✓
Partial networking	–	✓	✓	✓
Circuit diagram	CAN 3 on page 1566	CAN 4 on page 1566	CAN 5 on page 1567	CAN 6 on page 1568
Required channels	1	1	1	1

General limitations

CAN communication that is configured by using the RTI CAN MultiMessage Blockset is not supported by MicroAutoBox III hardware.

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS4342 CAN FD Interface Module For more module-specific data, refer to [DS4342 CAN FD Interface Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more board-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

FlexRay

Where to go from here

Information in this section

Introduction (FlexRay).....	1120
Overviews (FlexRay).....	1121
Configuring the Function Block (FlexRay).....	1142
Hardware Dependencies (FlexRay).....	1150

Introduction (FlexRay)

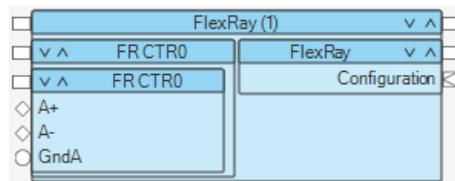
Introduction to the Function Block (FlexRay)

Function block purpose

The FlexRay function block type is one part of implementing FlexRay communication in real-time applications. It lets you specify the hardware access for FlexRay communication and control the communication of the FlexRay network separately for each FlexRay controller and FlexRay channel (A and/or B). The FlexRay communication itself must be modeled and supplied via the dSPACE FlexRay Configuration Package (refer to [Modeling a FlexRay Bus Interface \(Model Interface Package for Simulink - Modeling Guide](#) )).

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

- This are the main features:
- Configuring FlexRay controllers and channels.
 - Enabling common and controller-specific FlexRay features.
 - Configuring the initial state of the FlexRay communication.
 - Configuring the behavior of the real-time application regarding deadline violation.

Supported channel types

The FlexRay function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III	
Channel type	Bus 1	FlexRay 1	FlexRay 2	FlexRay 3	FlexRay 4
Hardware	DS2671	DS2672	DS6311	DS4340	DS1521

Overviews (FlexRay)

Where to go from here

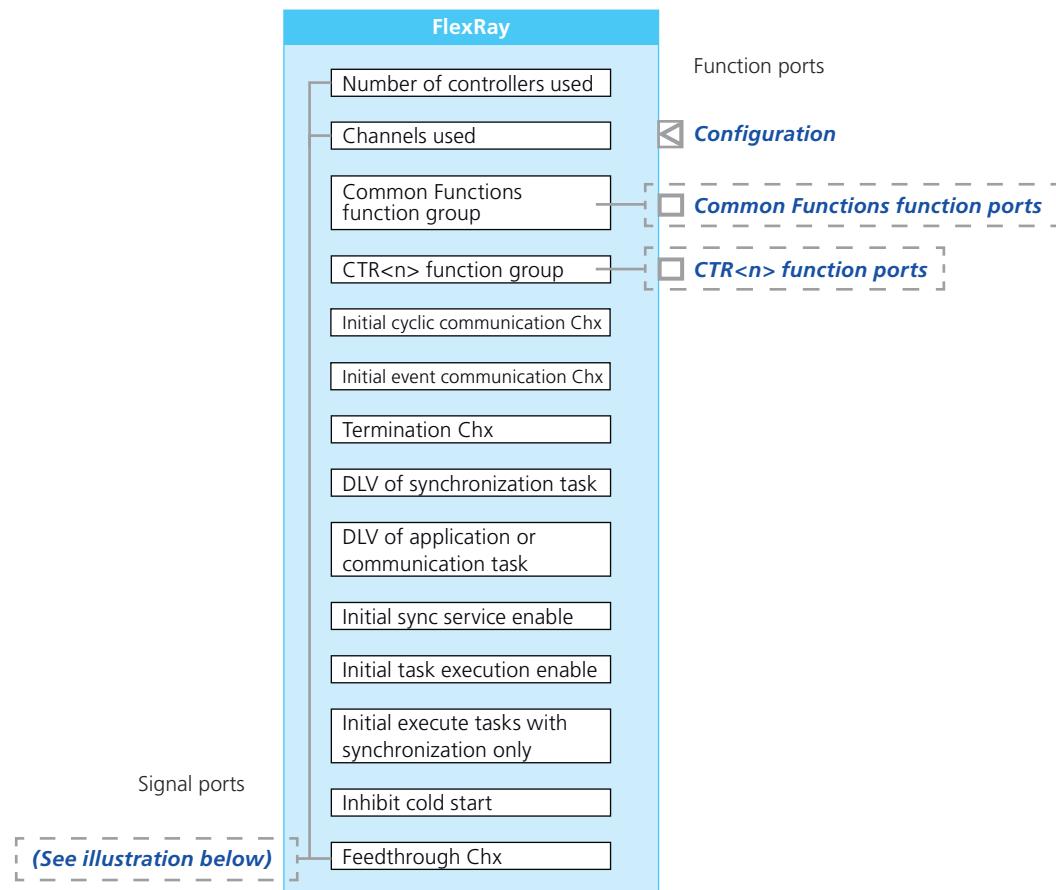
Information in this section

Overview of Ports and Basic Properties (FlexRay).....	1121
Ports and Properties of the Common Functions Function Group (FlexRay).....	1126
Ports and Properties of the Com Cyclic Control Chx Functions (FlexRay).....	1130
Ports and Properties of the Com Event Control Chx Port Functions (FlexRay).....	1132
Ports and Properties of the CTR< n > Function Groups (FlexRay).....	1135
Overview of Tunable Properties (FlexRay).....	1141

Overview of Ports and Basic Properties (FlexRay)

Overview illustration

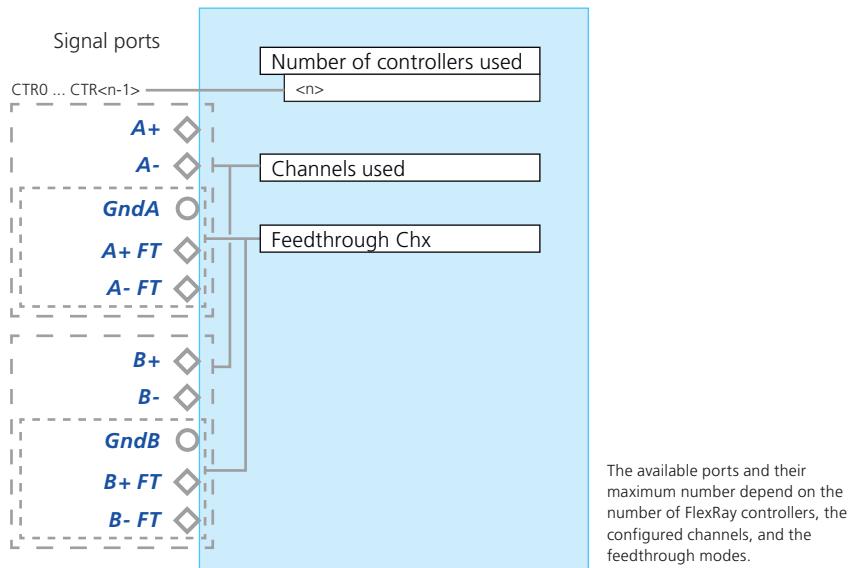
The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



For a detailed overview of the function groups, refer to:

- [Ports and Properties of the Common Functions Function Group \(FlexRay\) on page 1126](#)
- [Ports and Properties of the CTR< n > Function Groups \(FlexRay\) on page 1135](#)

Signal ports The following illustration shows the signal ports of the FlexRay function block.



Configuration

This function import provides the interface to connect a FLEXRAYCONFIG UPDATE block to the FlexRay function block and to use configuration data from the FLEXRAYCONFIG UPDATE block.

A+

This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel A via the bus line plus (BP).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to A or AB.

A-

This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel A via the bus line minus (BM).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to A or AB.

GndA This signal port is a reference port and provides the reference signal for the A+ and A- signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to A or AB and the Feedthrough ChA property is disabled.

A+ FT This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel A via the feedthrough bus line plus (BP).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to A or AB and the Feedthrough ChA property is enabled.

A- FT This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel A via the feedthrough bus line minus (BM).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to A or AB and the Feedthrough ChA property is enabled.

B+ This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel B via the bus line plus (BP).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to B or AB.

B- This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel B via the bus line minus (BM).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to B or AB.

GndB

This signal port is a reference port and provides the reference signal for the B+ and B- signal ports.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to B or AB and the Feedthrough ChB property is disabled.

B+ FT

This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel B via the feedthrough bus line plus (BP).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to B or AB and the Feedthrough ChB property is enabled.

B- FT

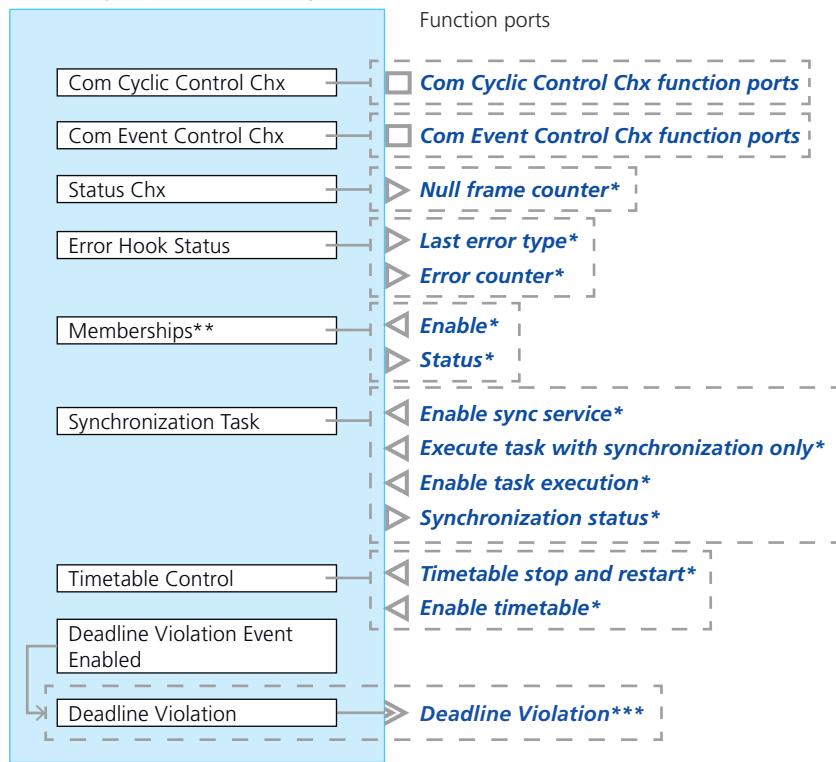
This signal port is a bidirectional port and lets you transmit and receive signals of FlexRay channel B via the feedthrough bus line minus (BM).

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (FlexRay) on page 1150.
Dependencies	Available only if the Channel used property is set to B or AB and the Feedthrough ChB property is enabled.

Ports and Properties of the Common Functions Function Group (FlexRay)

Overview illustration

The following illustration shows the function ports of the Common Functions function group of the FlexRay function block.



* Function port available if enabled

** This function is available only if you configured memberships in the FlexRay Configuration Tool and mapped the associated configuration port block to the FlexRay function block.

The maximum number of available ports depends on the number of configured memberships.

*** Property and event port available if enabled

For a detailed overview of the Com Cyclic Control Chx and Com Event Control Chx functions and their function ports, refer to:

- [Ports and Properties of the Com Cyclic Control Chx Functions \(FlexRay\) on page 1130](#)
- [Ports and Properties of the Com Event Control Chx Port Functions \(FlexRay\) on page 1132](#)

Null frame counter

This function output provides the number of received null frames that occurred during the communication for channel A or channel B. Received null frames are counted only if their ID is valid for the simulated FlexRay node. The value is incremented until the counter overflows. After an overflow, the counter starts with 0.

Tip

After a real-time application is loaded to dSPACE real-time hardware, the dSPACE FlexRay controllers might send null frames during some FlexRay communication cycles. This is because the controllers are synchronized before the fault-tolerant communication (FTCOM) code is executed. If these null frames have a negative impact on the application, you can reduce the time span between the controller synchronization and the start of FTCOM code execution. You can do this by specifying a hard start-up synchronization mode for the synchronization task in the FlexRay Configuration Tool.

Value range	0 ... $2^{32}-1$
Dependencies	Available for FlexRay channel A or B only if the Status ChA or Status ChB property is set to Null frame counter.

Last error type

This function import receives information on the type of the last FlexRay error that occurred.

Value range	<ul style="list-style-type: none"> ▪ 0: No error detected No errors occurred for the FlexRay communication. ▪ 1: Buffer locked error The communication code cannot be written to the buffer because it is still being used by the FlexRay controller. ▪ 2: Buffer full error The communication code cannot be written to the buffer because its current content has not been transmitted yet. ▪ 3: Local buffer corrupted The buffer contains invalid data (e.g., due to an invalid write access). ▪ 4: Invalid frame error An invalid frame was received (e.g., because two FlexRay nodes transmitted frames via the same slot).
Dependencies	Available only if the Error Hook Status property is set to Last error type.

Error counter

This function import receives the number of FlexRay errors that occurred.

Value range	0 ... $2^{32}-1$
Dependencies	Available only if the Error Hook Status property is set to Error counter.

Enable This function import receives the enable flag of the membership group with ID <n>.

Value range	<ul style="list-style-type: none"> ▪ 0: Communication of the ECUs assigned to the membership group is disabled. The controllers/buffers are not started. ▪ 1: Communication of the ECUs assigned to the membership group is enabled. The controllers/buffers are started.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Memberships property is set to Membership <n>/Enable. ▪ The maximum number of available ports depends on the number of memberships you configured in the FlexRay Configuration Tool.

Status This function outport provides status information on the membership group with ID <n>.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabling the membership group was successful. ▪ 1: Enabling the membership group was successful.
Dependencies	<ul style="list-style-type: none"> ▪ Available only if the Memberships property is set to Membership <n>/Status. ▪ The maximum number of available ports depends on the number of memberships you configured in the FlexRay Configuration Tool.

Enable sync service This function import receives the enable flag of the synchronization service.

Value range	<ul style="list-style-type: none"> ▪ 0: The synchronization service is disabled. ▪ 1: The synchronization service is enabled.
Dependencies	Available only if the Synchronization Task property is set to Enable sync service.

Execute tasks with synchronization only This function import receives the execution mode of tasks.

Value range	<ul style="list-style-type: none"> ▪ 0: Tasks are executed regardless of whether the host and the FlexRay cluster are synchronized. ▪ 1: Tasks are executed only if the host and the FlexRay cluster are synchronized.
Dependencies	Available only if the Synchronization Task property is set to Execute tasks with synchronization only.

Enable task execution

This function import receives the enable flag of task execution.

Value range	<ul style="list-style-type: none"> ▪ 0: FlexRay application tasks are stopped. <p>Note</p> <p>A task is stopped at the beginning of the next FlexRay cycle.</p> <ul style="list-style-type: none"> ▪ 1: FlexRay application tasks are started.
Dependencies	Available only if the Synchronization Task property is set to Enable task execution.

Synchronization status

This function outport provides the synchronization status of the real-time application.

Value range	<ul style="list-style-type: none"> ▪ 0: The real-time application is not synchronized to the FlexRay bus. ▪ 1: The real-time application is synchronized to the FlexRay bus.
Dependencies	Available only if the Synchronization Task property is set to Synchronization status.

Timetable stop and restart

This function import receives the stop flag for the execution of a timetable.

Value range	<ul style="list-style-type: none"> ▪ 0: The execution of a specific timetable for the current communication cycle is not stopped. ▪ 1: The execution of a specific timetable for the current communication cycle is stopped. The timetable is restarted automatically at the beginning of the next communication cycle. <p>Note</p> <p>The timetable stop and restart are triggered once when the port value changes from 0 to 1.</p>
Dependencies	Available only if the Timetable Control property is set to Timetable stop and restart.

Enable timetable

This function import receives the enable flag for the execution of a timetable.

Value range	<ul style="list-style-type: none"> ▪ 0: The execution of a specific timetable is disabled permanently. ▪ 1: The execution of a specific timetable is enabled permanently.
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Dependencies	Available only if the Timetable Control property is set to Enable timetable.
--------------	------------------------------------------------------------------------------

Deadline Violation

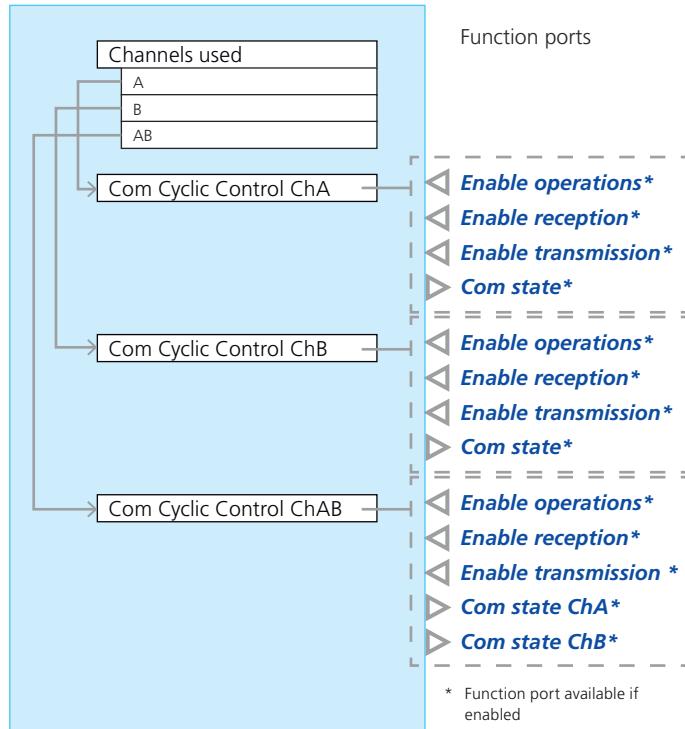
This event port provides an I/O event each time a deadline violation (DLV) occurs in a FlexRay synchronization task, application task, or communication task.

Value range	—
Dependencies	Available only if the Deadline Violation Event Enabled checkbox is selected.

Ports and Properties of the Com Cyclic Control Chx Functions (FlexRay)

Overview illustration

The following illustration shows the function ports of the Com Cyclic Control Chx functions.



The Com Cyclic Control Chx functions are available for the Common Functions function group. For an overview of the Common Functions function group, refer to [Ports and Properties of the Common Functions Function Group \(FlexRay\)](#) on page 1126.

Enable operations

This function import receives the enable flag for the cyclic communication of the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: Static and dynamic cyclic transmissions (TX) and receptions (RX) are disabled. ▪ 1: Static and dynamic cyclic transmissions (TX) and receptions (RX) are enabled.
Dependencies	Available for the related FlexRay channel only if the Com Cyclic Control ChA, Com Cyclic Control ChB, or Com Cyclic Control ChAB property is set to Enable operations.

Enable receptions

This function import receives the enable flag for the cyclic receptions (RX) of the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: Static and dynamic cyclic receptions (RX) are disabled. ▪ 1: Static and dynamic cyclic receptions (RX) are enabled.
Dependencies	Available for the related FlexRay channel only if the Com Cyclic Control ChA, Com Cyclic Control ChB, or Com Cyclic Control ChAB property is set to Enable receptions.

Enable transmission

This function import receives the enable flag for the cyclic transmissions (TX) of the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: Static and dynamic cyclic transmissions (TX) are disabled. ▪ 1: Static and dynamic cyclic transmissions (TX) are enabled.
Dependencies	Available for the related FlexRay channel only if the Com Cyclic Control ChA, Com Cyclic Control ChB, or Com Cyclic Control ChAB property is set to Enable transmission.

Com state

This function outport provides the state of the cyclic part of the communication layer for the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: The cyclic part of the communication layer is offline. ▪ 1: Receiving for the cyclic part is online, sending is offline. ▪ 2: Receiving for the cyclic part is offline, sending is online. ▪ 3: The cyclic part of the communication layer is online.
Dependencies	Available for FlexRay channel A or B only if the Com Cyclic Control ChA or Com Cyclic Control ChB property is set to Com state.

Com state ChA

This function outport provides the state of the cyclic part of the communication layer for FlexRay channel A.

Value range	<ul style="list-style-type: none"> ▪ 0: The cyclic part of the communication layer is offline.
-------------	---------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 1: Receiving for the cyclic part is online, sending is offline. ▪ 2: Receiving for the cyclic part is offline, sending is online. ▪ 3: The cyclic part of the communication layer is online.
Dependencies	Available only if the Com Cyclic Control ChAB property is set to Com state ChA.

Com state ChB

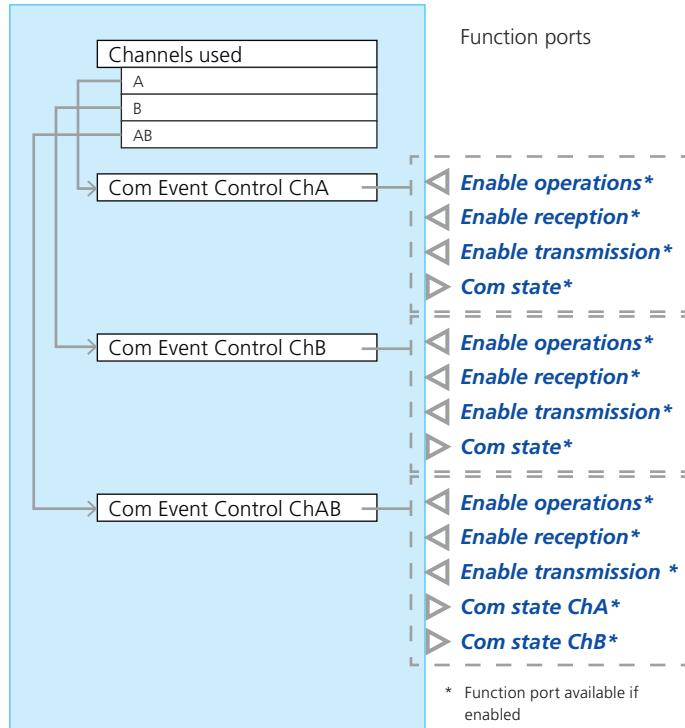
This function output provides the state of the cyclic part of the communication layer for FlexRay channel B.

Value range	<ul style="list-style-type: none"> ▪ 0: The cyclic part of the communication layer is offline. ▪ 1: Receiving for the cyclic part is online, sending is offline. ▪ 2: Receiving for the cyclic part is offline, sending is online. ▪ 3: The cyclic part of the communication layer is online.
Dependencies	Available only if the Com Cyclic Control ChAB property of the related channel is set to Com state ChB.

Ports and Properties of the Com Event Control Chx Port Functions (FlexRay)

Overview illustration

The following illustration shows the function ports of the Com Event Control Chx functions.



The Com Event Control Chx functions are available for the Common Functions function group. For an overview of the Common Functions function group, refer to [Ports and Properties of the Common Functions Function Group \(FlexRay\)](#) on page 1126.

Enable operations

This function import receives the enable flag for the event communication of the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: Static and dynamic event transmissions (TX) and receptions (RX) are disabled. ▪ 1: Static and dynamic event transmissions (TX) and receptions (RX) are enabled.
Dependencies	Available for the related FlexRay channel only if the Com Event Control ChA, Com Event Control ChB, or Com Event Control ChAB property is set to Enable operations.

Enable receptions

This function import receives the enable flag for the event receptions (RX) of the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: Static and dynamic event receptions (RX) are disabled. ▪ 1: Static and dynamic event receptions (RX) are enabled.
Dependencies	Available for the related FlexRay channel only if the Com Event Control ChA, Com Event Control ChB, or Com Event Control ChAB property is set to Enable receptions.

Enable transmission

This function import receives the enable flag for the event transmissions (TX) of the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: Static and dynamic event transmissions (TX) are disabled. ▪ 1: Static and dynamic event transmissions (TX) are enabled.
Dependencies	Available for the related FlexRay channel only if the Com Event Control ChA, Com Event Control ChB, or Com Event Control ChAB property is set to Enable transmission.

Com state

This function outport provides the state of the event-based part of the communication layer for the related FlexRay channel.

Value range	<ul style="list-style-type: none"> ▪ 0: The event-based part of the communication layer is offline. ▪ 1: Receiving for the event-based part is online, sending is offline. ▪ 2: Receiving for the event-based part is offline, sending is online. ▪ 3: The event-based part of the communication layer is online.
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Dependencies	Available for FlexRay channel A or B only if the Com Event Control ChA or Com Event Control ChB property is set to Com state.
--------------	-------------------------------------------------------------------------------------------------------------------------------

Com state ChA

This function outport provides the state of the event-based part of the communication layer for FlexRay channel A.

Value range	<ul style="list-style-type: none"> ▪ 0: The event-based part of the communication layer is offline. ▪ 1: Receiving for the event-based part is online, sending is offline. ▪ 2: Receiving for the event-based part is offline, sending is online. ▪ 3: The event-based part of the communication layer is online.
Dependencies	Available only if the Com Event Control ChAB property is set to Com state ChA.

Com state ChB

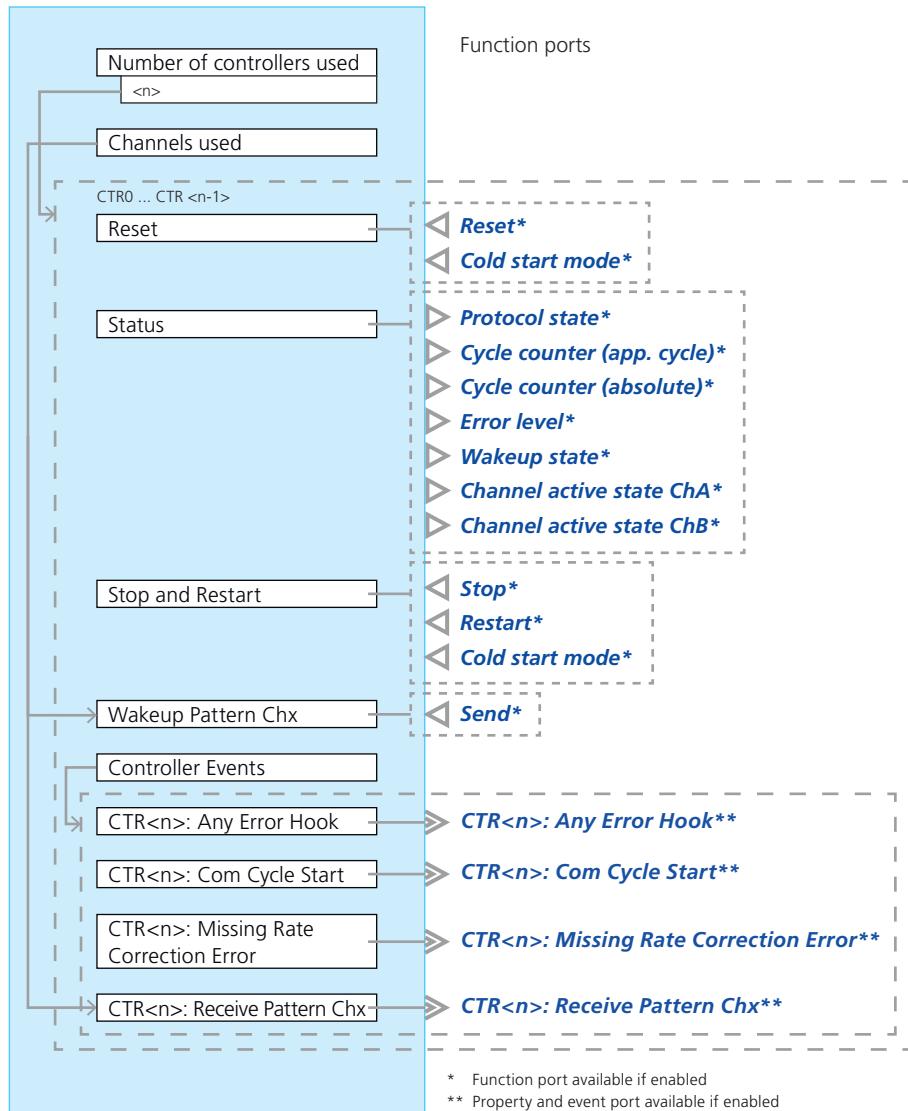
This function outport provides the state of the event-based part of the communication layer for FlexRay channel B.

Value range	<ul style="list-style-type: none"> ▪ 0: The event-based part of the communication layer is offline. ▪ 1: Receiving for the event-based part is online, sending is offline. ▪ 2: Receiving for the event-based part is offline, sending is online. ▪ 3: The event-based part of the communication layer is online.
Dependencies	Available only if the Com Event Control ChAB property of the related channel is set to Com state ChB.

Ports and Properties of the CTR< n > Function Groups (FlexRay)

Overview illustration

The following illustration shows the function ports of the CTR< n > function groups of the FlexRay function block.



Reset

This function import receives the reset status of the related FlexRay controller.

Value range	<ul style="list-style-type: none"> ▪ 0: The FlexRay controller is not reset to the configuration mode. ▪ 1: The FlexRay controller is set to the configuration mode and restarted afterwards. You can restart or reintegrate a FlexRay controller which already entered the configuration mode, or start or reintegrate a running FlexRay controller.
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Note</p> <p>The reset is triggered once when the port value changes from 0 to 1.</p>
Dependencies	Available only if the Reset property of the related FlexRay controller is set to Reset.

Cold start mode

This function import receives the cold start inhibit flag of the related FlexRay controller.

Value range	<ul style="list-style-type: none"> ▪ 0: The cold start inhibit flag of the FlexRay controller is not set. The controller can act as a leading coldstart node. This can impair the communication of a running FlexRay cluster. ▪ 1: The cold start inhibit flag of the FlexRay controller is set. The FlexRay controller can only act as a following coldstart node. This means that the node can either integrate itself into a running cluster or transmit startup frames after another coldstart node (the leading coldstart node) started the initialization of the cluster communication.
Dependencies	<ul style="list-style-type: none"> ▪ Available for the CTR<n> port group only if the Reset property of the related FlexRay controller is set to Cold start mode. ▪ Available for the Stop and Restart port group only if the Stop and Restart property of the related FlexRay controller is set to Cold start mode.

Protocol state

This function import receives information on the state of a FlexRay network.

Value range	<ul style="list-style-type: none"> ▪ 0: Configuration ▪ 1: Initialize schedule ▪ 2: Normal active operation ▪ 3: Normal passive operation ▪ 4: Integration consistency check ▪ 5: Integration listen ▪ 11: Wake up ▪ 21: Coldstart listen ▪ 22: Integration coldstart check ▪ 23: Join coldstart ▪ 24: Coldstart collision resolution ▪ 25: Coldstart consistency check ▪ 26: Coldstart gap ▪ 50: Default configuration ▪ 51: Ready ▪ 52: Halt ▪ 60: Wake-up standby ▪ 61: Wake-up listen
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 62: Wake-up send ▪ 63: Wake-up detect ▪ 70: Monitor mode ▪ 80: Startup prepare ▪ 81: Abort startup <p>For further information on the protocol states, refer to the FlexRay protocol specification.</p>
Dependencies	Available only if the Status property of the related FlexRay controller is set to Protocol state .

Cycle counter (app. cycle) This function import receives the value of the FlexRay communication counter according to the application cycle.

Value range	0 ... (application cycle duration / communication cycle duration -1)
Dependencies	Available only if the Status property of the related FlexRay controller is set to Cycle counter (app. cycle) .

Cycle counter (absolute) This function import receives the value of the FlexRay communication counter value according to the value of the CCCVR (current cycle counter value register) of the FlexRay controller. The communication counter starts with 0. It runs independently of the application cycle.

Value range	0 ... 63
Dependencies	Available only if the Status property of the related FlexRay controller is set to Cycle counter (absolute) .

Error level This function import receives information on the error level of the FlexRay controller.

Value range	<ul style="list-style-type: none"> ▪ 0: No problems have occurred. ▪ 1: Some minor problems have occurred. ▪ 2: There is a serious problem.
Dependencies	Available only if the Status property of the related FlexRay controller is set to Error level .

Wakeup state This function import receives information on the wake-up state of the FlexRay controller.

Value range	<ul style="list-style-type: none"> ▪ 0: Undefined ▪ 1: Received header <p>The communication controller has received a frame header without coding violation during the initial listen phase.</p>
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 2: Received wakeup The communication controller has received a valid wake-up pattern during the initial listen phase. ▪ 3: Collision header The communication controller has detected a collision during wake-up pattern transmission by receiving a valid header during the ensuing detection phase. ▪ 4: Collision wakeup The communication controller has detected a collision during wake-up pattern transmission by receiving a valid wake-up pattern during the ensuing detection phase. ▪ 5: Unknown The communication controller has detected a collision but did not detect a subsequent reception event that would allow the collision to be categorized as either 'Collision header' or 'Collision wake-up'. ▪ 6: Transmitted The wake-up pattern was completely transmitted.
Dependencies	Available only if the Status property of the related FlexRay controller is set to Wakeup state .

Channel active state ChA

This function import receives information on the state of FlexRay channel A.

Value range	<ul style="list-style-type: none"> ▪ 0: The channel is inactive, i.e., no valid frame/PDU was received during the current and the previous execution of the status block. ▪ 1: The channel is active.
Dependencies	Available only if the Status property of the related FlexRay controller is set to Channel active state ChA .

Channel active state ChB

This function import receives information on the state of FlexRay channel B.

Value range	<ul style="list-style-type: none"> ▪ 0: The channel is inactive, i.e., no valid frame/PDU was received during the current and the previous execution of the status block. ▪ 1: The channel is active.
Dependencies	Available only if the Status property of the related FlexRay controller is set to Channel active state ChB .

Stop

This function import receives information on the mode of the FlexRay controller.

Value range	<ul style="list-style-type: none"> ▪ 0: The FlexRay controller is not set to the configuration mode.
-------------	---------------------------------------------------------------------------------------------------------------------

- 1: The FlexRay controller is set to the configuration mode.

Note

The FlexRay controller stop is triggered once when the port value changes from 0 to 1.

Dependencies	Available only if the Stop and Restart property of the related FlexRay controller is set to Stop.
--------------	---------------------------------------------------------------------------------------------------

Restart

This function import receives the restart status of the FlexRay controller.

Value range	<ul style="list-style-type: none"> ▪ 0: The FlexRay controller is not restarted or reintegrated. ▪ 1: The FlexRay controller is restarted or reintegrated.
Dependencies	Available only if the Stop and Restart property of the related FlexRay controller is set to Restart.

Send

This function import receives information on the transmission of the wake-up pattern of a FlexRay channel. The port is edge-triggered. The wake-up pattern is sent on channel A or channel B when the port value changes from 0 to 1.

Value range	<ul style="list-style-type: none"> ▪ 0: The transmission of the wake-up pattern on channel A or channel B is not initiated. ▪ 1: The transmission of the wake-up pattern on channel A or channel B is initiated. That is, a wake-up pattern as defined by the wake-up count, idle time, and low time properties can be sent.
Dependencies	Available for FlexRay channel A or B only if the Wakeup Pattern ChA or Wakeup Pattern ChB property of the related FlexRay controller is set to Send.

CTR<n>: Any Error Hook

This event port provides an I/O event each time an error occurs during the FlexRay communication.

Value range	—
Dependencies	Available only if the Controller Events property is set to Any Error Hook.

CTR<n>: Com Cycle Start

This event port provides an I/O event each time a communication cycle starts.

Value range	—
Dependencies	Available only if the Controller Events property is set to Com Cycle Start.

CTR<n>: Missing Rate Correction Error

This event port provides an I/O event each time a controller loses synchronization and enters the passive state because the maximum odd cycles without rate correction value has been exceeded.

Value range	—
Dependencies	Available only if the Controller Events property is set to Missing Rate Correction Error.

CTR<n>: Receive Pattern ChA

This event port provides an I/O event each time a wake-up pattern is received on FlexRay channel A.

Value range	—
Dependencies	Available only if the Channel used property is set to A or AB and the Controller Events property is set to Receive Pattern ChA.

CTR<n>: Receive Pattern ChB

This event port provides an I/O event each time a wake-up pattern is received on FlexRay channel B.

Value range	—
Dependencies	Available only if the Channel used property is set to B or AB and the Controller Events property is set to Receive Pattern ChB.

Overview of Tunable Properties (FlexRay)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
	Initial cyclic communication Chx	✓	–
	Initial event communication Chx	✓	–
	Initial sync service enable	✓	–
	Initial task execution enable	✓	–
	Initial execute tasks with synchronization only	✓	–
Function Ports			
	Initial switch setting (Test Automation)	–	✓
	Initial substitute value (Test Automation)	–	✓
Electrical Interface			
	–	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (FlexRay)

Where to go from here

Information in this section

Configuring the Basic Functionality (FlexRay).....	1142
Configuring Standard Features (FlexRay).....	1147

Configuring the Basic Functionality (FlexRay)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Configuring FlexRay controllers
- Enabling FlexRay controller features, for example, to reset a FlexRay controller at run time, to provide status information to a mapped behavior model, or to stop and restart a FlexRay controller.
- Configuring FlexRay channels
- Enabling common FlexRay features, for example, to enable and disable FlexRay communication on a FlexRay channel at run time, to provide status information to a mapped behavior model, or to use synchronization tasks.
- Configuring the initial state of the FlexRay communication
- Configuring the behavior of the real-time application regarding deadline violation

Configuring FlexRay controllers

The FlexRay function block lets you specify the number of used FlexRay controllers and configure whether a used controller can act as a leading coldstart node or following coldstart node.

Number of used FlexRay controllers Each FlexRay function block lets you use up to 16 FlexRay controllers. Until the Configuration port of a FlexRay function block is not mapped to a Configuration Port block, you can specify the number of used FlexRay controllers in the Properties Browser. For example, this allows you to configure the controllers before a suitable Simulink model is available.

However, when you map the Configuration port of the FlexRay function block to a Configuration Port block, the number of used FlexRay controllers is derived from the related FLEXRAYCONFIG UPDATE block in the Simulink model. If

required, the number of used controllers is reduced according to the derived number and the settings of the affected controllers are lost.

Specifying the role of the coldstart node FlexRay controllers that are used by a FlexRay function block can act as leading coldstart nodes, i.e., they can actively start the initialization of a FlexRay cluster communication.

Tip

Whether a FlexRay controller can act as a leading coldstart node must be specified in the communication matrix that is used in the FlexRay Configuration Tool to configure the FlexRay communication.

Leading coldstart nodes might impair the communication of a running FlexRay cluster. You can therefore configure each controller as a following coldstart node by selecting the controller's **Inhibit cold start** checkbox. If you do this, the controller can either integrate itself into a running communication cluster or transmit startup frames after a leading coldstart node started the initialization of the cluster communication.

For more information on coldstart nodes, refer to the FlexRay protocol specification.

Enabling FlexRay controller features

The FlexRay function block provides various controller-specific features. You can enable these features separately for each FlexRay controller that is used by the function block. For each enabled feature, a function is added for the related controller to the FlexRay function block. Depending on the feature, you can enable one or more function ports or event ports. The ports let you exchange data with the mapped behavior model or provide controller events to the behavior model, for example.

The following table provides an overview of the available features, the related function names, and the available function ports and event ports.

Controller-Specific Feature	Function Name	Available Ports
Resetting the controller	Reset	Function ports: <ul style="list-style-type: none">▪ Reset▪ Cold start mode
Using status information of the controller	Status	Function ports: <ul style="list-style-type: none">▪ Protocol state▪ Cycle counter (app. cycle)▪ Cycle counter (absolute)▪ Error level▪ Wakeup state▪ Channel active state ChA▪ Channel active state ChB
Stopping and restarting the controller	Stop and Restart	Function ports: <ul style="list-style-type: none">▪ Stop▪ Restart▪ Cold start mode

Controller-Specific Feature	Function Name	Available Ports
Using wake-up patterns on FlexRay channel A and/or B	<ul style="list-style-type: none"> ▪ Wakeup Pattern ChA ▪ Wakeup Pattern ChB 	Function port: Send
Using events that are provided by the controller	Controller Task Interface	<p>Event ports:</p> <ul style="list-style-type: none"> ▪ Any Error Hook ▪ Com Cycle Start ▪ Missing Rate Correction Error ▪ Receive Pattern ChA ▪ Receive Pattern ChB

For more information on the individual functions and their ports, refer to [CTR<n> Controller Functions \(FlexRay\) \(ConfigurationDesk Function Block Properties\)](#).

Configuring FlexRay channels

The FlexRay function block lets you select the FlexRay channels, terminate the bus lines, and enable feedthrough mode.

Selecting FlexRay channels If the Configuration port of a FlexRay function block is not mapped to a configuration port of a Configuration Port block, you can select the FlexRay channels (A and/or B) that are used for the FlexRay communication. For example, this allows you to configure the channels before a suitable Simulink model is available.

However, when you map the Configuration port of the FlexRay function block to a Configuration Port block, the used FlexRay channels are derived from the related FLEXRAYCONFIG UPDATE block in the Simulink model. If you specified settings for a channel that is not used, these settings are lost.

Terminating bus lines To avoid reflections on the bus, FlexRay bus lines must be terminated at each end. If the SCALEXIO or MicroAutoBox III system is connected at the end of the FlexRay bus, you can use the termination resistors of dSPACE real-time hardware to terminate the FlexRay bus. For this purpose, the FlexRay function block lets you enable the termination separately for each FlexRay channel. If enabled, the related FlexRay channel is terminated with the termination resistor of the respective channel type. For channel-type-dependent resistor values, refer to [Hardware Dependencies \(FlexRay\)](#) on page 1150.

Note

If you enable feedthrough mode for the FlexRay 3 or FlexRay 4 channel type, the termination of the channel is automatically disabled. For a circuit diagram, refer to [FlexRay 3](#) on page 1611 or [FlexRay 4](#) on page 1613, respectively.

Using feedthrough mode Depending on the individual FlexRay bus topology, the SCALEXIO or MicroAutoBox III system might be connected at a stub of the FlexRay bus. Long stubs can reduce the signal quality and result in reflections on the bus.

To reduce the stub length and improve the signal quality in this case, the FlexRay function block lets you enable feedthrough mode separately for each used FlexRay channel. If you do this, feedthrough signal ports are available for the function block and the feedthrough lines of the selected channel type are used for the bus signals.

For an overview of the channel types that support feedthrough mode, refer to [Hardware Dependencies \(FlexRay\) on page 1150](#).

NOTICE

Enabling the feedthrough mode has the following effects on the Bus 1 channel type:

- The galvanic isolation is deactivated. The hardware can be destroyed if the connected FlexRay bus carries high voltages.
- The transceiver's VBAT pin is internally switched to +12 V supply voltage and the transceiver's GND pin is internally switched to system GND.

Refer to [Bus 1](#) on page 1560.

Note

- If the SCALEXIO or MicroAutoBox III system is connected at the end of the FlexRay bus, its bus lines must be terminated. Enabling feedthrough mode in this case might reduce the signal quality. Therefore, it is recommended not to enable feedthrough mode.
- If you enable feedthrough mode for the FlexRay 3 or FlexRay 4 channel type, the termination of the channel is automatically disabled. For a circuit diagram, refer to [FlexRay 3](#) on page 1611 or [FlexRay 4](#) on page 1613, respectively.

Enabling common FlexRay features

The FlexRay function block provides various common FlexRay features that you can enable for the function block. For each enabled feature, a function is added to the FlexRay function block. Depending on the feature, you can enable one or more function ports or event ports. The ports let you exchange data with the mapped behavior model or provide events to the behavior model, for example.

The following table provides an overview of the available features, the related function names, and the available function ports and event ports.

Common FlexRay Feature	Function Name	Available Ports
Controlling the cyclic communication of FlexRay channel A and/or B	<ul style="list-style-type: none"> ▪ Com Cyclic Control ChA ▪ Com Cyclic Control ChB ▪ Com Cyclic Control ChAB 	Function ports: <ul style="list-style-type: none"> ▪ Enable operations ▪ Enable reception ▪ Enable transmission ▪ Com state
Controlling the event communication of FlexRay channel A and/or B	<ul style="list-style-type: none"> ▪ Com Event Control ChA 	Function ports: <ul style="list-style-type: none"> ▪ Enable operations ▪ Enable reception ▪ Enable transmission

Common FlexRay Feature	Function Name	Available Ports
	<ul style="list-style-type: none"> ▪ Com Event Control ChB ▪ Com Event Control ChAB 	▪ Com state
Using status information of FlexRay channel A and/or B	<ul style="list-style-type: none"> ▪ Status ChA ▪ Status ChB 	Function port: Null frame counter
Using error hook status information	Error Hook Status	Function ports: <ul style="list-style-type: none"> ▪ Last error type ▪ Error counter
Enabling and disabling ECUs according to their membership	Membership <n>	Function ports: <ul style="list-style-type: none"> ▪ Membership <n>/Enable ▪ Membership <n>/Status
Specifying the behavior of the FlexRay synchronization task	Synchronization Task	Function ports: <ul style="list-style-type: none"> ▪ Execute tasks with synchronization only ▪ Enable task execution ▪ Synchronization status
Handling FlexRay timetables	Timetable Control	Function ports: <ul style="list-style-type: none"> ▪ Timetable stop and restart ▪ Enable timetable
Using deadline violation events	Common Task Interface	Event port: Deadline Violation

For more information on the individual functions and their ports, refer to [Common Functions Properties \(FlexRay\) \(ConfigurationDesk Function Block Properties\)](#).

Configuring the initial state of the FlexRay communication

The FlexRay function block lets you configure the initial state of the FlexRay communication. You can configure the following initial states:

- The initial state of the cyclic communication on FlexRay channel A and B, for example, enable reception only.
- The initial state of the event communication on FlexRay channel A and B, for example, enable transmission only.
- The initial state of the synchronization service, i.e., enabled or disabled.
- The initial state of FlexRay application tasks, i.e., started or stopped.
- The initial state of FlexRay tasks depending on the synchronization, for example, execute tasks only if the FlexRay configuration and the FlexRay cluster are synchronized.

For more information on the specific configuration settings, refer to [Common Function Block Properties \(FlexRay\) \(ConfigurationDesk Function Block Properties\)](#).

Configuring the behavior of the real-time application regarding deadline violation

During FlexRay communication, deadline violations can occur in application tasks, communication tasks, and synchronization tasks as follows:

Application or Communication Task	Synchronization Task
<p>A deadline violation occurs in the following case:</p> <ul style="list-style-type: none"> ▪ The application or communication task is executed directly before a synchronization task. ▪ The worst-case execution time (WCET) of the application or communication task is exceeded. 	A deadline violation occurs if the worst-case execution time (WCET) of the synchronization task is exceeded.

The FlexRay function block lets you configure the behavior of the real-time application if a deadline violation occurs. You can configure whether the real-time application terminates or keeps running but issues a warning message. You can configure a different behavior for deadline violations that occur in:

- FlexRay application tasks or communication tasks
- FlexRay synchronization tasks

Configuring the behavior of the real-time application regarding deadline violations applies only to FlexRay configurations with configuration ID '-'. For FlexRay configurations with a configuration ID 1, 2, or 3, the configured behavior is ignored.

Configuring Standard Features (FlexRay)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO				
Configuration Feature	Bus 1	FlexRay 1	FlexRay 2	Further Information
Measurement point	-	-	-	Specifying the Measurement Point for Input Signals on page 119

SCALEXIO				
Configuration Feature	Bus 1	FlexRay 1	FlexRay 2	Further Information
Interface type	–	–	–	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	–	–	Specifying Load Settings on page 121

MicroAutoBox III				
Configuration Feature	FlexRay 3	FlexRay 4	Further Information	
Measurement point	–	–	Specifying the Measurement Point for Input Signals on page 119	
Interface type	–	–	Specifying the Circuit Type for Voltage Input Signals on page 116	
Channel multiplication	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95	
Signal Ports				
Trigger level of electronic fuse	–	–	Specifying Fuse Settings on page 122	
Failure simulation support	–	–	Specifying Settings for Failure Simulation on page 123	
Usage of loads	–	–	Specifying Load Settings on page 121	

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"> ▪ Bus 1 on page 1560 ▪ FlexRay 1 on page 1609 ▪ FlexRay 2 on page 1610 	<ul style="list-style-type: none"> ▪ FlexRay 3 on page 1611 ▪ FlexRay 4 on page 1613

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (FlexRay)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (FlexRay).....	1150
MicroAutoBox III Hardware Dependencies (FlexRay).....	1151

SCALEXIO Hardware Dependencies (FlexRay)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	Bus 1	FlexRay 1	FlexRay 2
Hardware	DS2671 Bus Board	DS2672 Bus Module	DS6311 FlexRay Board
Transceiver type ¹⁾	TJA1080TS	TJA1080TS	TJA1081BTS
Voltage range	-60 V ... +60 V	-60 V ... +60 V	-60 V ... +60 V
Baud rate range	1 MBd ... 10 MBd	1 MBd ... 10 MBd	1 MBd ... 10 MBd
Supported FlexRay channels	<ul style="list-style-type: none"> ▪ FlexRay channel A ▪ FlexRay channel B 	<ul style="list-style-type: none"> ▪ FlexRay channel A ▪ FlexRay channel B 	<ul style="list-style-type: none"> ▪ FlexRay channel A ▪ FlexRay channel B
Termination	<ul style="list-style-type: none"> ▪ Switchable termination ▪ Parallel termination between bus line plus (BP) and bus line minus (BM) 	<ul style="list-style-type: none"> ▪ Switchable termination ▪ Parallel termination between bus line plus (BP) and bus line minus (BM) 	<ul style="list-style-type: none"> ▪ Switchable termination ▪ Parallel termination between bus line plus (BP) and bus line minus (BM)
Termination resistance	92 Ω or not terminated	92 Ω or not terminated	92 Ω or not terminated
Feedthrough mode	✓	–	✓
Galvanic isolation	✓ (only if feedthrough mode is disabled)	–	–
Circuit diagrams	Bus 1 on page 1560	FlexRay 1 on page 1609	FlexRay 2 on page 1610
Required channels	1 for each specified FlexRay controller and FlexRay channel (A and/or B)	1 for each specified FlexRay controller and FlexRay channel (A and/or B)	1 for each specified FlexRay controller and FlexRay channel (A and/or B)

¹⁾ The specifications of the mounted transceiver might be different. dSPACE might change the transceiver used without notice. This change does not affect the board revision number.

More hardware data

DS2671 Bus Board For more board-specific data, refer to [DS2671 Bus Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2672 Bus Module For more board-specific data, refer to [DS2672 Bus Module \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6311 FlexRay Board For more board-specific data, refer to [DS6311 FlexRay Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (FlexRay)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	FlexRay 3	FlexRay 4
Hardware	DS4340 FlexRay Interface Module	DS1521 Bus Board
Transceiver type	TJA1080TS	TJA1081GTS
Protected voltage range	<ul style="list-style-type: none"> ▪ -60 V ... +60 V between a bus line and GND ▪ -9.5 V ... +9.5 V between bus line plus (e.g., A+) and bus line minus (e.g., A-) 	<ul style="list-style-type: none"> ▪ -60 V ... +60 V between a bus line and GND ▪ -9.5 V ... +9.5 V between bus line plus (e.g., A+) and bus line minus (e.g., A-)
Baud rate range	1 MBd ... 10 MBd	1 MBd ... 10 MBd
Supported FlexRay channels	<ul style="list-style-type: none"> ▪ FlexRay channel A ▪ FlexRay channel B 	<ul style="list-style-type: none"> ▪ FlexRay channel A ▪ FlexRay channel B
Termination	<ul style="list-style-type: none"> ▪ Indirect switchable termination: <ul style="list-style-type: none"> ▪ Automatically enabled if feedthrough mode is disabled ▪ Automatically disabled if feedthrough mode is enabled ▪ Enabled: Parallel termination between bus line plus (A+ or B+) and bus line minus (A- or B-) 	<ul style="list-style-type: none"> ▪ Indirect switchable termination: <ul style="list-style-type: none"> ▪ Automatically enabled if feedthrough mode is disabled ▪ Automatically disabled if feedthrough mode is enabled ▪ Enabled: Parallel termination between bus line plus (A+ or B+) and bus line minus (A- or B-)
Termination resistance	91 Ω or not terminated	91 Ω or not terminated
Feedthrough mode	✓	✓
Circuit diagram	FlexRay 3 on page 1611	FlexRay 4 on page 1613
Required channels	1 for each specified FlexRay controller and FlexRay channel (A and/or B)	1 for each specified FlexRay controller and FlexRay channel (A and/or B)

More hardware data

DS4340 FlexRay Interface Module For more module-specific data, refer to [DS4340 FlexRay Interface Module Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more board-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

LIN

Where to go from here

Information in this section

Introduction (LIN).....	1154
Overviews (LIN).....	1156
Configuring the Function Block (LIN).....	1158
Hardware Dependencies (LIN).....	1162

Introduction (LIN)

Introduction to the Function Block (LIN)

Function block purpose

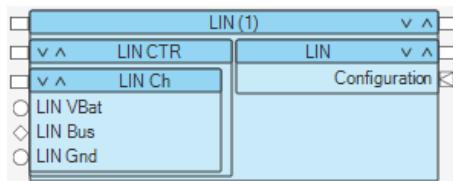
The LIN function block type is one part of implementing LIN communication in real-time applications. It lets you specify the hardware access for LIN communication. For example, you can configure transceiver settings for the assigned bus channel. The LIN communication itself must be modeled and supplied via one of the following providers:

- RTI LIN MultiMessage Blockset (refer to [Modeling a LIN Bus Interface \(Model Interface Package for Simulink - Modeling Guide](#) )
- Bus Manager (refer to [Introduction to the Bus Manager \(ConfigurationDesk Bus Manager Implementation Guide](#) )
- V-ECU implementations (refer to [Special Aspects of V-ECU Implementations Containing LIN Controllers \(ConfigurationDesk Real-Time Implementation Guide](#) )
- Bus simulation containers (refer to [Working with Bus Simulation Containers \(ConfigurationDesk Real-Time Implementation Guide](#) )

In addition, you can use the LIN function block to initialize bus channels that are not involved in LIN communication. These channels can then be accessed during run time via ControlDesk or Real-Time Testing (RTT) (e.g., for monitoring purposes).

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Specifying LIN bus settings.
- Enabling power wake-up.

Supported channel types

The LIN function block type supports the following channel types:

	SCALEXIO			MicroAutoBox III	
Channel type	Bus 1	LIN 1	LIN 2	LIN 3	LIN 4
Hardware	DS2671	DS2672	<ul style="list-style-type: none">▪ DS6301▪ DS6341	<ul style="list-style-type: none">▪ DS1511▪ DS1511B1▪ DS1513	DS1521

Overviews (LIN)

Overview of Ports and Basic Properties (LIN)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Configuration

This function import provides the interface for mapping one of the following blocks or controllers to the LIN function block:

- One RTILINMM ControllerSetup block of the RTI LIN MultiMessage Blockset
- One LIN controller of a V-ECU implementation
- One or more LIN controllers of one or more bus simulation containers (BSC files)

When mapped, this function port provides configuration data that configures the LIN function block and sets several function block properties to read-only.

Note

When you use the LIN function block with the Bus Manager, you cannot use the block with RTILINMM ControllerSetup blocks or LIN controllers of V-ECU implementations or bus simulation containers. Do not map the Configuration function port in this case.

LIN VBat

This signal port is a reference port and provides the power supply for the LIN transceiver.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (LIN) on page 1162.
Dependencies	–

LIN Bus

This signal port is a bidirectional port and lets you transmit and receive LIN bus signals.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (LIN) on page 1162.
Dependencies	–

LIN Gnd

This signal port is a reference port and provides the ground for the LIN VBat signal port and the reference signal for the LIN Bus signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (LIN) on page 1162.
Dependencies	–

Configuring the Function Block (LIN)

Where to go from here

Information in this section

Configuring the Basic Functionality (LIN).....	1158
Configuring Standard Features (LIN).....	1160

Configuring the Basic Functionality (LIN)

Overview

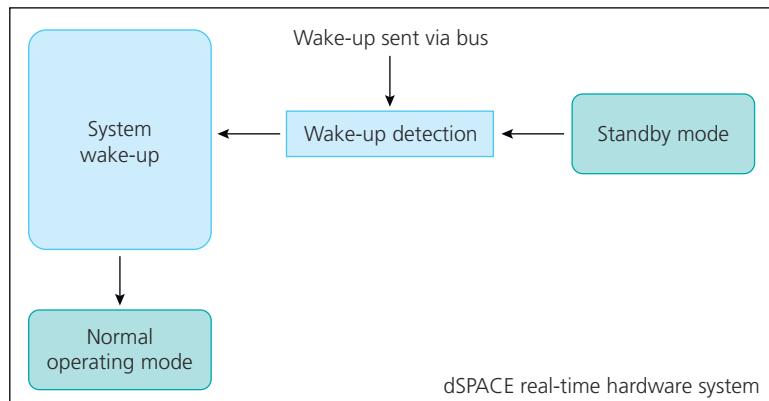
The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Specifying LIN bus settings according to the selected transceiver type
- Enabling power wake-up

Enabling power wake-up

The LIN function block lets you enable power wake-up.

Basics on power wake-up When you enable power wake-up for the LIN function block, a LIN wake-up that is sent on the LIN bus can wake up a dSPACE real-time hardware system. In this case, the system can change its operation mode from standby to normal operating mode, as shown in the following example.



The following dSPACE real-time hardware systems support power wake-up via LIN:

- SCALEXIO AutoBox/LabBox systems based on a DS6001 Processor Board
- MicroAutoBox III systems

A typical use scenario for using power wake-up is to wake up the real-time hardware system via a LIN wake-up and to start the real-time application automatically.

Preconditions for waking up a real-time hardware system and starting the real-time application To wake up a real-time hardware system via a LIN wake-up and to start the real-time application automatically, the following preconditions must be fulfilled:

- The configuration of the real-time hardware system supports power wake-up.
For more information on the required configurations, refer to:
 - SCALEXIO AutoBox/LabBox systems: [Waking up SCALEXIO AutoBoxes/LabBoxes via Bus Signals \(SCALEXIO – Hardware and Software Overview\)](#).
 - MicroAutoBox III systems: [Powering Features \(MicroAutoBox III Hardware Installation and Configuration\)](#).
- A hardware resource that supports power wake-up is assigned to the LIN function block. Refer to [Hardware Dependencies \(LIN\)](#) on page 1162.
- The Power wake-up property of the LIN function block is set to Enabled.
- The real-time application is loaded to the flash memory of the real-time hardware system.

Tip

This is required only for starting the real-time application automatically when waking up the real-time hardware system. If the real-time application is not loaded to the flash memory, the system can be woken up but no application is executed.

Refer to [Downloading and Executing Real-Time Applications \(ConfigurationDesk Real-Time Implementation Guide\)](#).

- The real-time hardware system is set to standby mode.
It is recommended to set the system to standby mode by using the System Shutdown function block. Refer to [Basics on Using the System Shutdown Functionality](#) on page 1407.

If the preconditions are fulfilled and a wake-up is sent on the LIN bus, the real-time hardware system and the transceiver of the LIN channel are woken up, and the real-time application starts from the flash memory.

Tip

If you work with a system of SCALEXIO AutoBoxes/LabBoxes that are connected via the power control bus, all the SCALEXIO AutoBoxes/LabBoxes wake up when at least one of them is woken up via a LIN wake-up.

Configuring Standard Features (LIN)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

SCALEXIO				
Configuration Feature	Bus 1	LIN 1	LIN 2	Further Information
Measurement point	–	–	–	Specifying the Measurement Point for Input Signals on page 119
Interface type	–	–	–	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	–	–	Specifying Load Settings on page 121
MicroAutoBox III				
Configuration Feature	LIN 3	LIN 4	Further Information	
Measurement point	–	–	Specifying the Measurement Point for Input Signals on page 119	
Interface type	–	–	Specifying the Circuit Type for Voltage Input Signals on page 116	
Channel multiplication	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95	
Signal Ports				
Trigger level of electronic fuse	–	–	Specifying Fuse Settings on page 122	
Failure simulation support	–	–	Specifying Settings for Failure Simulation on page 123	
Usage of loads	–	–	Specifying Load Settings on page 121	

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none">▪ Bus 1 on page 1560▪ LIN 1 on page 1615▪ LIN 2 on page 1615	<ul style="list-style-type: none">▪ LIN 3 on page 1616▪ LIN 4 on page 1616

Model interface

The model interface of the LIN function block does not provide standard configuration features.

Hardware Dependencies (LIN)

Where to go from here

Information in this section

SCALEXIO Hardware Dependencies (LIN).....	1162
MicroAutoBox III Hardware Dependencies (LIN).....	1163

SCALEXIO Hardware Dependencies (LIN)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Bus 1	LIN 1	LIN 2
Hardware	DS2671 Bus Board	DS2672 Bus Module	<ul style="list-style-type: none"> ▪ DS6301 CAN/LIN Board ▪ DS6351 LIN Board
Transceiver type ¹⁾	TLE6258-2G	TLE6258-2G	TLS7257SJ
Protected voltage range	-60 V ... +60 V (VBAT and bus lines ²⁾)	<ul style="list-style-type: none"> ▪ VBAT: 0 V ... +32 V ▪ Bus lines²⁾: -60 V ... +60 V 	-60 V ... +60 V (VBAT and bus lines ²⁾)
Working voltage range (VBAT and bus lines ²⁾)	+9 V ... +32 V	+9 V ... +32 V	+9 V ... +32 V
Baud rate range	1 kBd ... 20 kBd	1 kBd ... 20 kBd	1 kBd ... 20 kBd
Termination	Switchable master/slave termination	Switchable master/slave termination	Switchable master/slave termination
Termination resistance	<ul style="list-style-type: none"> ▪ 1 kΩ (master termination) ▪ 30 kΩ (slave termination) 	<ul style="list-style-type: none"> ▪ 1 kΩ (master termination) ▪ 30 kΩ (slave termination) 	<ul style="list-style-type: none"> ▪ 1 kΩ (master termination) ▪ 30 kΩ (slave termination)
Power wake-up	–	–	✓ (supported only if the board is installed in a SCALEXIO LabBox system that is based on a DS6001 Processor Board)
Failure simulation	✓	✓	–
Galvanic isolation	✓	–	–
Circuit diagrams	Bus 1 on page 1560	LIN 1 on page 1615	LIN 2 on page 1615
Required channels	1	1	1

¹⁾ The specifications of the mounted transceiver might be different. dSPACE might change the transceiver used without notice.

This change does not affect the board revision number.

²⁾ Bus lines: Pin to GND

More hardware data

DS2671 Bus Board For more board-specific data, refer to [DS2671 Bus Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2672 Bus Module For more board-specific data, refer to [DS2672 Bus Module \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6301 CAN/LIN Board For more board-specific data, refer to [DS6301 CAN/LIN Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6351 LIN Board For more board-specific data, refer to [DS6351 LIN Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (LIN)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel type	LIN 3	LIN 4
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	DS1521 Bus Board
Transceiver type	TLE6258-2G	TLE7257SJ
Protected voltage range	-27 V ... +32 V	-60 V ... +60 V
Working voltage range	+6 V ... +27 V	+5,5 V ... +40 V
Baud rate range	500 Bd ... 20 kBd	1 kBd ... 20 kBd
Termination	30 kΩ	Switchable master/slave termination: <ul style="list-style-type: none"> ▪ 1 kΩ (master termination) ▪ 30 kΩ (slave termination)
Power wake-up	-	✓
Circuit diagram	LIN 3 on page 1616	LIN 4 on page 1616
Required channels	1	1

General limitations

LIN communication that is configured by using the RTI LIN MultiMessage Blockset is not supported by MicroAutoBox III hardware.

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1521 Bus Board For more board-specific data, refer to [DS1521 Bus Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Ethernet

Where to go from here

Information in this section

Introduction to Implement Ethernet Communication.....	1166
Ethernet Setup.....	1168
Virtual Ethernet Setup.....	1179
UDP Receive.....	1188
UDP Transmit.....	1198
TCP.....	1207
PTP Master.....	1226
PTP Slave.....	1234
Ethernet Switch.....	1241

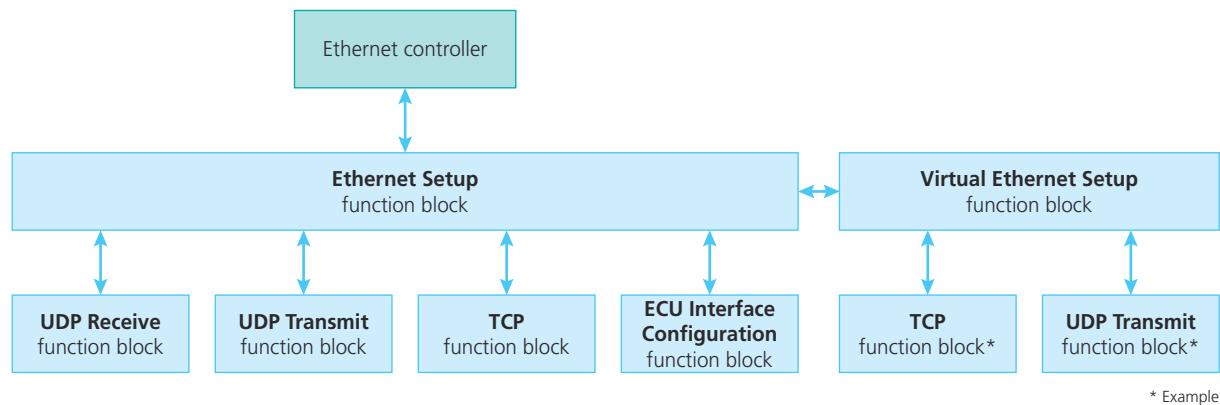
Introduction to Implement Ethernet Communication

Basics on Implementing Ethernet Communication in ConfigurationDesk

Basic structure

The Ethernet Setup function block provides access to an Ethernet controller. Several other function blocks can use the access at the same time to transport data values. Furthermore, you can add a virtual Ethernet controller to your network to provide additional IP configurations.

The following illustration gives you an overview of the Ethernet implementation in ConfigurationDesk.



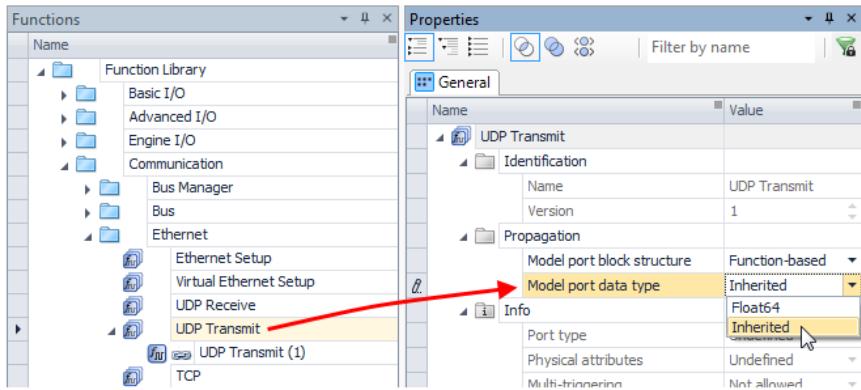
* Example

Optimizing the Ethernet data exchange

The floating-point data type is the default data type for generating or updating the model interface. The floating-point data type does not match the 8-bit unsigned integer (UInt8) data type of the Data Vector function ports of the TCP and UDP function blocks. Therefore, the data type is automatically converted when data values are exchanged with the behavior model. This conversion takes up processing power.

To avoid data type conversions, you can change the default data type settings for new or updated model port blocks to inherit the data type of the function ports they are derived from.

The following example shows the Model port data type property of the UDP Transmit function block type in the Properties Browser. The Model port data type property lets you change the setting to inherit the data type.



Processing the data exchange with different data types

The TCP and UDP protocol use the UInt8 data type to exchange data values. To exchange values of other data types via a TCP or UDP Ethernet connection, you can code the data values to several byte values and pack the coded values to a data vector and vice versa. For example: You can distribute the bits of a 32-bit floating point data value in 4 byte values that are packed to one data vector. The following illustration shows the example.

32-bit floating point value

Bit 1...8	Bit 9...16	Bit 17...24	Bit 25...32
-----------	------------	-------------	-------------



Data Vector with byte values

$$\left(\begin{array}{l} \text{Bit 1...8} \\ \text{Bit 9...16} \\ \text{Bit 17...24} \\ \text{Bit 25...32} \end{array} \right)$$

MATLAB Simulink, for example, provides blocks for byte packing/unpacking. For more information, refer to the MATLAB Simulink user documentation.

Related topics

Basics

ECU Interface Configuration.....	1361
Ethernet Setup.....	1168
TCP.....	1207
UDP Receive.....	1188
UDP Transmit.....	1198
Virtual Ethernet Setup.....	1179

Ethernet Setup

Where to go from here

Information in this section

Introduction (Ethernet Setup).....	1168
Overviews (Ethernet Setup).....	1169
Configuring the Function Block (Ethernet Setup).....	1170

Introduction (Ethernet Setup)

Introduction to the Function Block (Ethernet Setup)

Function block purpose

With the Ethernet Setup function block type, you can configure and initialize the access to an Ethernet controller of your dSPACE real-time hardware. The function block works as a provider: Other function blocks can use it to access the configured Ethernet controller.

Default display

The function block provides neither signal ports nor function ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Providing access to an Ethernet controller of the dSPACE real-time hardware.
- Configuring the network settings of the Ethernet controller.

Supported channel types

The Ethernet Setup function block type supports the following channel types:

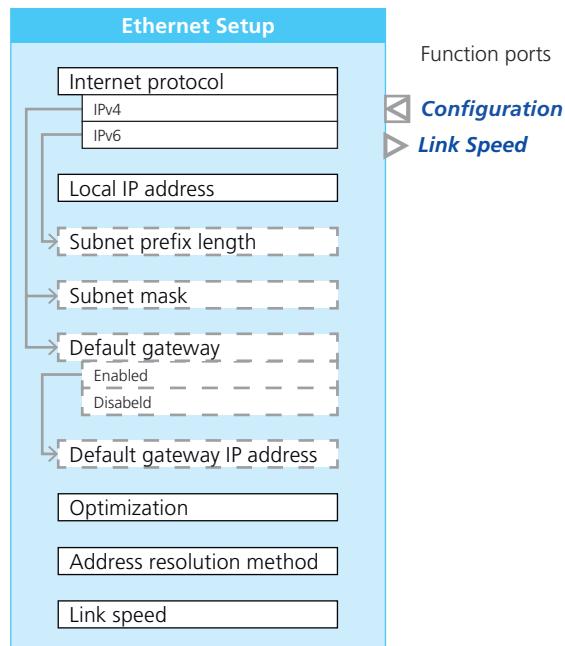
	SCALEXIO		MicroAutoBox III	
Channel type	Ethernet Adapter 1	Ethernet Adapter 2	Ethernet Adapter 2	Ethernet Adapter 3
Board	<ul style="list-style-type: none"> ▪ SCALEXIO Processing Unit ▪ DS6001 Processor Board ▪ DS6331-PE Ethernet Board ▪ DS6334-PE Ethernet Board ▪ DS6336-CS Ethernet Board ▪ DS6336-PE Ethernet Board 	<ul style="list-style-type: none"> ▪ DS6333-CS Automotive Ethernet Board ▪ DS6333-PE Automotive Ethernet Board ▪ DS6335-CS Ethernet Board 	DS1403 Processor Board	DS1521 Bus Board

Oversviews (Ethernet Setup)

Overview of Ports and Properties (Ethernet Setup)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Configuration	This function import is for future use. Do not map this port.
Link Speed	This function outport writes the current link speed of the assigned Ethernet controller to the behavior model.
Value range	0 MBit/s ... 10,000 MBit/s: <ul style="list-style-type: none">▪ 0 MBit/s: No Ethernet connection.▪ >0 MBit/s: A data link with the provided link speed is established.
Dependencies	–

| **Tunable properties** | The Ethernet Setup function block type does not provide tunable properties. Tunable properties can also be accessed and modified in the experiment software. |

Configuring the Function Block (Ethernet Setup)

Where to go from here	Information in this section
	Configuring the Basic Functionality (Ethernet Setup)..... 1170 Assigning an Ethernet Controller (Ethernet Setup)..... 1174 Configuring Standard Features (Ethernet Setup)..... 1178

Configuring the Basic Functionality (Ethernet Setup)

- Suppressing IGMP messages
- Specifying the link speed.

Specifying the local IP address

With the local IP address, you specify two parts for IP networking:

- You specify a unique IP address for the assigned Ethernet controller.
- You specify the subnetwork the Ethernet controller is connected to via the netmask (subnet mask/subnet prefix length).

The netmask determines the most significant bits of the local IP address as the subnetwork address, the least significant bits determine the host address.

The subnetwork address indicates the members of the same subnetwork. That means, all members must use the same Internet Protocol version, subnetwork address, and subnet mask or subnet prefix length. The members differ only in the host address.

Tip

If an external Ethernet devices is already configured, use the subnet mask/subnet prefix length of the configured device and the resulting subnetwork address.

For example: An external Ethernet device uses the IP address **192.168.0.1** and the subnet mask **255.255.255.0**.

The resulting IP address range for the local IP address is **192.168.0.2 ... 192.168.0.254** with the subnet mask **255.255.255.0** (**192.168.0.0** and **192.168.0.255** are reserved IP addresses).

Recommended IP address ranges The following table shows the recommended IP address ranges for specifying the local IP address:

IPv4	IPv6
10.0.0.1 ... 10.255.255.254	fc00::1 ... fdff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
172.16.0.1 ... 172.31.255.254	(fc00::/7)
192.168.0.1 ... 192.168.255.254	

Unsupported IP address ranges The following table shows the unsupported IP address ranges for a local IP address:

IPv4	IPv6
Addresses with 0 as the first octet such as the default IP address: 0.0.0.0 ... 0.255.255.255	The default IP address: 0:0:0:0:0:0:0::
Multicast IP addresses: 224.0.0.0 ... 239.255.255.255	Multicast IP addresses: ff00:: ... ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff (ff00::/8)
Loopback addresses: 127.0.0.0 ... 127.255.255.255	Loopback address: ::1

IPv4	IPv6
Addresses with 240 ... 255 as the first octet: 240.0.0.0 ... 255.255.255.255	Link-local addresses: fe80:: ... febf::ffff:ffff:ffff (fe80::/10)

Unfavorable IP address ranges The following IP address ranges are not recommended to be used as the local IP address:

- The first IP address of the subnetwork if all bits of the host part are set to 0.
For example: 192.168.0.0 with the subnet mask 255.255.255.0
- IPv4 Broadcast addresses: The last IP address of the subnetwork if all bits of the host part are set to 1.
For example: 192.168.0.255 with the subnet mask 255.255.255.0
- All IP addresses with a subnet mask whose first octet is less than 255.
For example: Subnet mask 0.0.0.0 or 254.0.0.0.

Workflow to specify the local IP address To specify the local IP address, you have to proceed in the following order:

1. Select the IPv4 or IPv6 Internet protocol version for IP network communication.
2. Specify a unique IP address for the Ethernet controller that is supported by ConfigurationDesk. Remember that the IP address includes the subnetwork address. The subnetwork address must fit to the connected Ethernet network.
For recommended IP addresses, refer to [Recommended IP address ranges](#) on page 1171.
3. Specify how many most significant bits of the IP address are used to specify the subnetwork address. This is done via the subnet mask for IPv4 addresses or the subnet prefix length for IPv6 addresses.

Using DHCP The Ethernet Setup function block does not support DHCP.

Specifying the default gateway

A default gateway lets you send IPv4 packets outside the local subnetwork that you specified via the subnet mask and the IP address. The default gateway to other networks can be the router of your network.

To make the default gateway known to the Ethernet controller, you have to enable the default gateway support and enter the IP address of the default gateway. The default gateway must be a member of the same subnetwork as the assigned Ethernet controller.

If you use a default gateway, IPv4 packets with IP addresses of other networks are sent to the default gateway. Then, the default gateway manages the next transport steps.

Note

The Ethernet Setup function block does not support default gateways for IPv6 Ethernet networks.

Optimizing the Ethernet controller

Depending on the use case, you can optimize the handling of data sets by selecting one of the following optimization modes:

- **Latency**

Select this mode if the reaction time is crucial for your application, e.g., for application with Ethernet triggered task. In this mode, the Ethernet controller is more responsive to packet handling. However, the Latency mode causes more processor load than the Throughput mode.

- **Throughput**

Select this mode to optimize the throughput of large data sets.

Note

There are function blocks that request a certain optimization mode or the mode is not configurable. For example, an ECU Interface Configuration function block usually requests a latency optimization. Depending on the function blocks that reference the Ethernet Setup function block, the Ethernet controller use the following optimization mode:

- If no referencing function block requests a certain optimization mode, the Ethernet controller uses the setting of the Ethernet Setup function block.
- If all referencing function blocks request the same optimization mode, but the mode is different to the setting of the Ethernet Setup function block, the Ethernet controller uses the requested optimization mode and ignores the setting of the Ethernet Setup function block.
- If the referencing function blocks request different optimization modes, the Ethernet controller uses the setting of the Ethernet Setup function block and ignores the requests of the referencing function blocks.

However, ConfigurationDesk generates a conflict if the specified optimization mode does not match the optimization mode that is requested by a referencing function block.

Specifying the address resolution method

You can specify the address resolution method that the Ethernet controller uses to map the IP address to a MAC address:

- **ARP/RFC826**

The Ethernet controller uses the address resolution protocol (ARP) according to IETF RFC 826 to resolve the MAC addresses of the network members. ARP is the common protocol for address resolution in Ethernet networks.

- **Learn MAC from multicast messages**

This method is intended for Ethernet networks that try to avoid ARP requests.

Suppressing IGMP messages

The basic task of the Internet group management protocol (IGMP) is to manage dynamic groups for IP multicast transmissions, whereby this management does not run via the assigned Ethernet controller, but via the routers of the Ethernet network. The routers accept requests for inclusion in a specific multicast group.

You can disable the sending of IGMP messages to reduce the bus load. However, this can affect the reception of multicast messages in Ethernet networks with routers and managed switches.

Note

The MicroAutoBox III and SCALEXIO with a Linux®-based operating system (introduced with firmware 5.0, dSPACE Release 2020-B) do not support the suppression of IGMP messages.

Specifying the link speed

You can specify the link speed of the assigned Ethernet controller.

If you use an Ethernet controller of the **Ethernet Adapter 2** channel type, the specified link speed affects the link between the assigned Ethernet controller and the internal Ethernet switch.

Note

The 10-Gbit Ethernet controller (Eth0_3) of the SCALEXIO Real-Time PC Version HPP 2.0 has the following limitations:

- The controller supports Ethernet connections only with a link speed of 1 Gbit/s and 10 Gbit/s.
- The link speed used is determined exclusively by means of autonegotiation. The setting of the **Link speed** property is ignored.

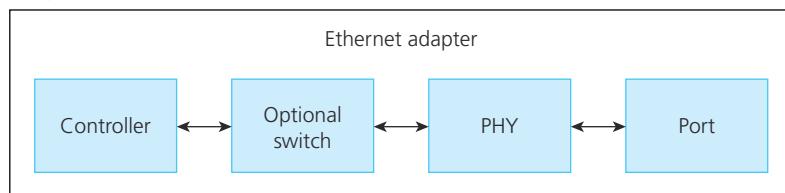
Assigning an Ethernet Controller (Ethernet Setup)

Introduction

An Ethernet controller is a part of an Ethernet adapter that you can assign to the **Ethernet Setup** function block.

Functional units of an Ethernet adapter

An Ethernet adapter provides the Ethernet controller for accessing the Ethernet network. The following illustration shows the functional units of an Ethernet adapter.



- **Controller:** Controls the access to the Ethernet network and is part of the data link layer.

- Optional switch: Switches the Ethernet connections between the controllers and ports. The switch is part of the data link layer.
- Only the Ethernet Adapter 2 channel type provides an internal Ethernet switch.
- PHY: Provides the electrical interface to the Ethernet network. The PHY is part of the physical layer.
- Port: Provides the mechanical interface to the Ethernet network, for example, an RJ45 connector. The port is part of the physical layer.

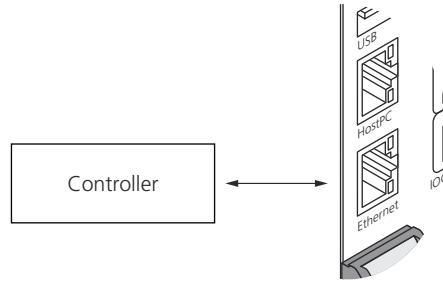
Ethernet Adapter 1 channel type

The Ethernet controllers of the Ethernet Adapter 1 channel type are directly mapped to one port.

The following illustrations shows the mapping between the Ethernet controllers and the ports (RJ45 connectors).

- Ethernet hardware with one controller:

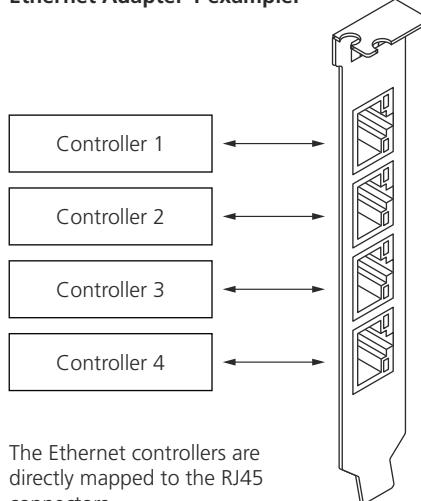
Ethernet Adapter 1 example:



The Ethernet controller is directly mapped to the RJ45 connector.

- Ethernet hardware with multiple controllers:

Ethernet Adapter 1 example:



The Ethernet controllers are directly mapped to the RJ45 connectors.

Note

SCALEXIO Real-Time PC HPP 2.0 only: If you create a hardware topology from scratch, only one Ethernet controller is available. To use all Ethernet controllers, import the predefined [dSPACE SCALEXIO Processing Unit with Intel 6208U processor HTFX file](#). Refer to [How to Import a Hardware Topology \(ConfigurationDesk Real-Time Implementation Guide\)](#).

The following tables show the dSPACE hardware that provides controllers of the Ethernet Adapter 1 channel type.

SCALEXIO				
Board	SCALEXIO Processing Unit			DS6001
	Product line HCP	Real-Time PC HE 1.0	Real-Time PC HPP 2.0	
Number of controllers	1	1	3 ¹⁾	1
More information	Variants of the SCALEXIO Processing Unit (SCALEXIO Hardware Installation and Configuration)			Overview of the DS6001 Processor Board (SCALEXIO Hardware Installation and Configuration)

¹⁾ If you create a hardware topology from scratch, import the predefined [dSPACE SCALEXIO Processing Unit with Intel 6208U processor HTFX file](#) to use all controllers.

SCALEXIO				
Board	DS6331-PE	DS6334-PE	DS6336-CS	DS6336-PE
Number of controllers	4	4	2	2
More information	Overview of the DS6331-PE Ethernet Board (SCALEXIO Hardware Installation and Configuration)	Overview of the DS6334-PE Ethernet Board (SCALEXIO Hardware Installation and Configuration)	Overview of the DS6336-CS Ethernet Board (SCALEXIO Hardware Installation and Configuration)	Overview of the DS6336-PE Ethernet Board (SCALEXIO Hardware Installation and Configuration)

Ethernet Adapter 2 channel type

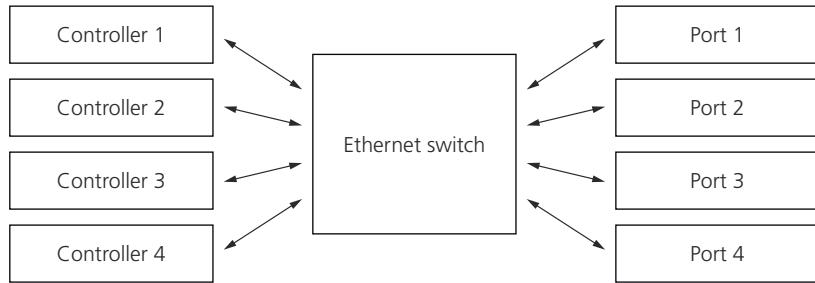
This channel type consists of Ethernet controllers that are connected to an internal Ethernet switch. The Ethernet switch is configurable via the web interface of the dSPACE hardware or via the [Ethernet Switch function block](#). For more information, refer to the following topics:

- [Ethernet Switch](#) on page 1241
- [Web Interface of the DS6001 Processor Board \(SCALEXIO Hardware Installation and Configuration\)](#)
- [Web Interface of the SCALEXIO Processing Unit \(SCALEXIO Hardware Installation and Configuration\)](#)

- [Configuring the I/O Ethernet Communication \(MicroAutoBox III Hardware Installation and Configuration\)](#)

The following illustrations shows the mapping between the Ethernet controllers and the ports (RJ45 connectors).

Ethernet Adapter 2 example:



The Ethernet controllers are mapped via an internal switch to the Ethernet ports.

The following table shows the dSPACE hardware that provide controllers of the Ethernet Adapter 2 channel type.

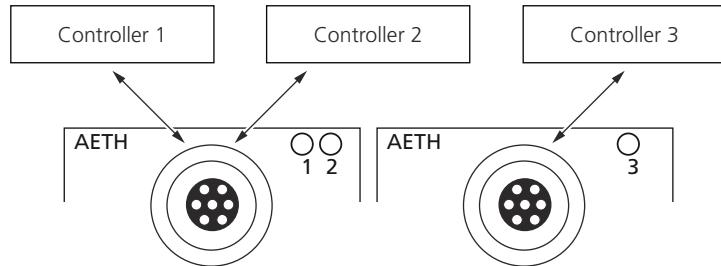
	SCALEXIO			MicroAutoBox III
Board	DS6333-CS ¹⁾	DS6333-PE ¹⁾	DS6335-CS ¹⁾	DS1403
Number of controllers	4	4	4	2 ²⁾
More information	Overview of the DS6333-CS Automotive Ethernet Board (SCALEXIO Hardware Installation and Configuration)	Overview of the DS6333-PE Automotive Ethernet Board (SCALEXIO Hardware Installation and Configuration)	Overview of the DS6335-CS Ethernet Board (SCALEXIO Hardware Installation and Configuration)	Ethernet Characteristics (MicroAutoBox III Hardware Installation and Configuration)

¹⁾ The physical layer transceivers (PHYs) of the ports 1 ... 4 are provided by Ethernet modules: The DS6330M1 module provides two automotive Ethernet PHYs, the DS6330M2 module provides two standard Ethernet PHYs. The Platform Manager lets you check which Ethernet modules are installed.

²⁾ A third controller can be activated via the web interface of the MicroAutoBox III. This controller is latency optimized and recommended only for fast bypassing. Refer to [How to Improve ECU Interfacing \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Ethernet Adapter 3 channel type

The Ethernet controllers of the Ethernet Adapter 3 channel type are directly mapped to one port as shown in the following illustration.



The following table shows the dSPACE hardware that provides controllers of the Ethernet Adapter 3 channel type.

	MicroAutoBox III
Board	DS1521
Number of controllers	3
More information	Ethernet Adapter 3 Characteristics (MicroAutoBox III Hardware Installation and Configuration)

Related topics**Basics**

[Limitations for Ethernet Communication \(ConfigurationDesk Real-Time Implementation Guide\)](#)

Configuring Standard Features (Ethernet Setup)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Test automation support	Configuring Test Automation Support on page 92

Virtual Ethernet Setup

Where to go from here

Information in this section

Introduction to the Functionality (Virtual Ethernet Setup).....	1179
Overviews (Virtual Ethernet Setup).....	1183
Configuring the Function Block (Virtual Ethernet Setup).....	1184

Introduction to the Functionality (Virtual Ethernet Setup)

Where to go from here

Information in this section

Introduction to the Function Block.....	1179
Basics on Virtual Ethernet.....	1180
Basics on the VLAN Tag.....	1182

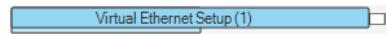
Introduction to the Function Block

Function block purpose

With the Virtual Ethernet Setup function block type, you can configure a virtual Ethernet controller to access an Ethernet network, including VLAN. The function block works as a provider: Other function blocks can use it to access the configured Ethernet controller.

Default display

The function block provides neither signal ports nor function ports. The illustration below shows the function block with its default settings.



Main features

These are the main features:

- Providing different IP addresses for different services on the same physical Ethernet controller.

- Accessing different IP subnetworks with one physical Ethernet controller.
- Accessing virtual local area networks (VLAN).

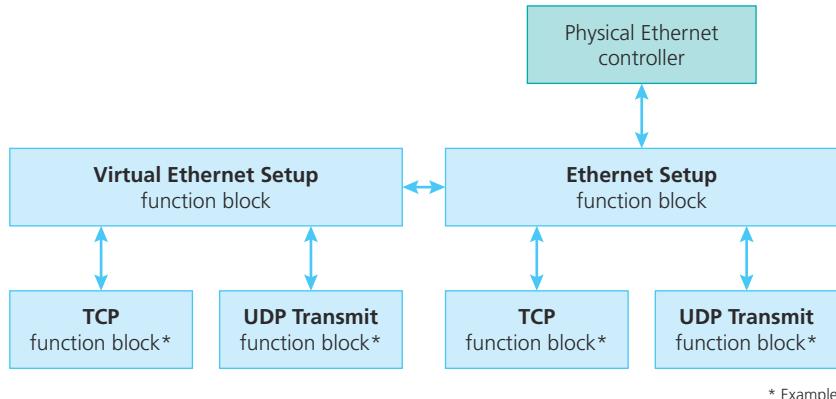
Basics on Virtual Ethernet

Introduction

Virtual Ethernet is the virtualization of network components to organize and manage the Ethernet network independent from the used hardware.

For example, you can add a virtual Ethernet controller to your Ethernet network to provide additional IP configurations for a physical Ethernet controller. Several virtual Ethernet controllers can use the same physical Ethernet controller.

The following illustration shows you the implementation and use of virtual Ethernet in ConfigurationDesk.



Furthermore, you can access a virtual local area network (VLAN). VLANs let you manage the LAN independently of the physical network.

Basics on virtual Ethernet controller

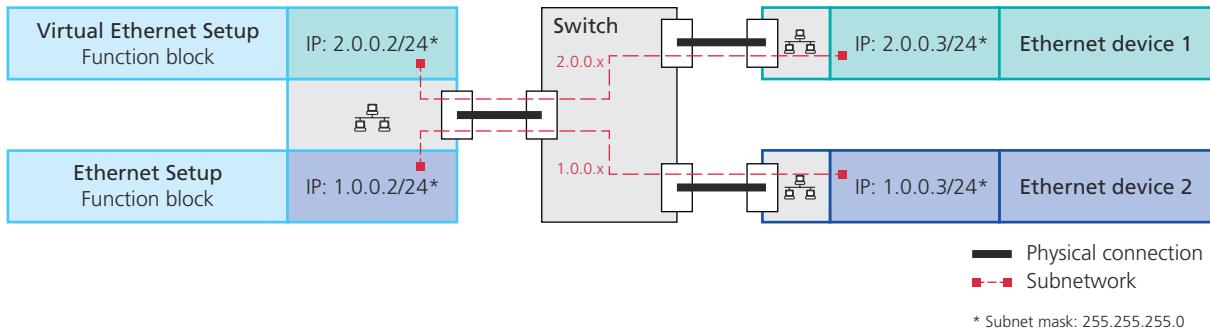
Virtual Ethernet controllers have their own local IP addresses and netmasks. For the electrical connection to the Ethernet network, they use physical Ethernet controllers.

The MAC addresses of the virtual Ethernet controllers can be specified as follows:

- You can use the MAC address of the used physical Ethernet controller. This configuration is like IP aliasing because several IP addresses belong to the same MAC address.
- You can specify custom MAC addresses. For other Ethernet devices, a virtual Ethernet controller with a custom MAC address does not differ from physical Ethernet controllers. You can also specify different IP addresses for one custom MAC address.

Benefits of virtual Ethernet controllers The virtualization of Ethernet controllers has the following benefits:

- You can provide different IP addresses for different services on the same physical Ethernet controller. For example, you can provide a virtual IP address to provide diagnostic information and a virtual IP address to control systems for driver safety.
- You can access different IP subnetworks with one physical Ethernet controller.



Basics on virtual local area networks (VLAN)

A VLAN based on IEEE 802.1Q adds a VLAN Tag to the Ethernet frame to organize the network.

The VLAN tag includes an identity number to identify the members of the same VLAN. Ethernet devices of the data link layer, such as Ethernet switches, read the VLAN tag and route the Ethernet frames between the members of the same VLAN:

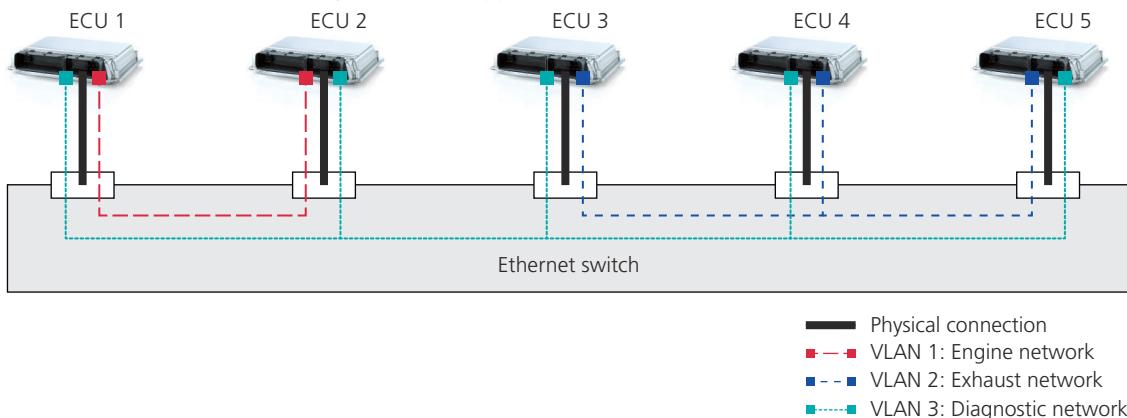
- Ethernet frames are forwarded only within the same VLAN. This means, even if an Ethernet device is a member of the same IP subnetwork as another Ethernet device, it cannot transmit or receive data if the Ethernet devices are in different VLANs.
- Multicast and broadcast messages are sent only to the members of the same VLAN.
- A VLAN is independent from IP networking. One VLAN can cover several IP subnetworks or only a part of an IP subnetwork.

For more information on the VLAN tag, refer to [Basics on the VLAN Tag](#) on page 1182.

Benefit of VLAN The virtualization of local area networks (VLAN) has the following benefits:

- You can organize the Ethernet network by the field of application without changing the IP network configuration.

The following illustration is an example for multiple VLANs that are organized by the field of application.



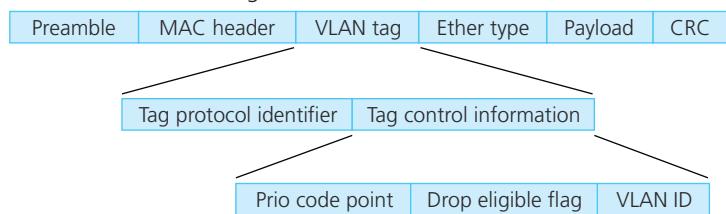
- VLANs let you prioritize the Ethernet communication to reduce the latency for prioritized Ethernet frames within the same VLAN. Higher prioritized Ethernet frames are routed before Ethernet frames with lower priority.

Using Ethernet switches that do not support VLAN such as unmanaged switches Ethernet switches that do not support VLAN generally route Ethernet frames without considering the VLAN tag. As a consequence, multicast and broadcast messages are sent to all connected Ethernet devices.

Basics on the VLAN Tag

Structure and content of VLAN tags

The VLAN tag is an optional 32-bit field that is added to the Ethernet frame. The following illustration shows you an Ethernet frame with a VLAN tag and the content of the VLAN tag.



The VLAN tag includes the following information:

- Tag protocol identifier
Indicates the used protocol for the VLAN. For VLANs based on IEEE 802.1Q, the value of the tag protocol identifier is 8100_{hex}. ConfigurationDesk automatically sets the tag protocol identifier.
- Tag control information:
 - Prio code point (PCP)
3-bit value that indicates the priority for the transmit control.

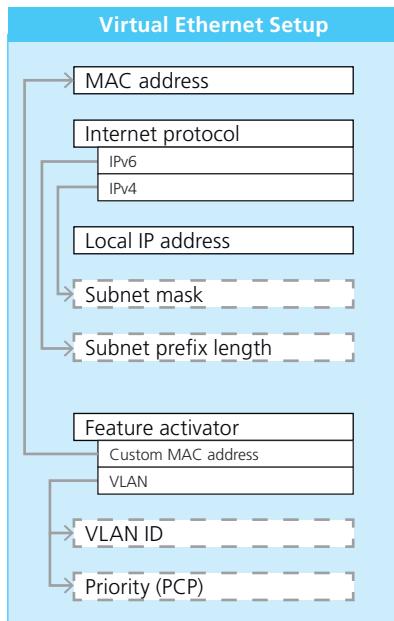
- Drop eligible flag (DEI)
Indicates whether Ethernet frames can be dropped in the presence of congestion. ConfigurationDesk automatically sets this flag to 0. The value 0 indicates that no Ethernet frame is eligible to be dropped.
- Identity number of the VLAN (VLAN ID)
12-bit value that indicates the membership in a VLAN.

Overviews (Virtual Ethernet Setup)

Overview of Ports and Properties (Virtual Ethernet Setup)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Ports

The Virtual Ethernet Setup function block does not provide signal ports or function ports.

Tunable parameters

The Virtual Ethernet Setup function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (Virtual Ethernet Setup)

Configuring the Basic Functionality (Virtual Ethernet Setup)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Referencing the Ethernet network.
- Specifying the local IP address.
- Customizing the MAC address.
- Specifying the access to a VLAN.

Referencing the Ethernet network

You reference an Ethernet network by assigning an Ethernet Setup function block. The Ethernet Setup function block provides the access to a physical Ethernet controller.

Observe the following points:

- If several function blocks reference the same Ethernet Setup function block, you must ensure that all function blocks belong to only one application process. This means, the model port blocks of all function blocks that are assigned to the Virtual Ethernet Setup function block must be mapped to the same behavior model as the function blocks that are assigned to the Ethernet Setup function block.
- If two Ethernet setup function blocks (Virtual Ethernet Setup or Ethernet Setup function blocks) are members of the same IP subnetwork, it is recommended that the function blocks use different physical Ethernet controllers. The use of different Ethernet controllers prevents an unintended behavior.

You specify the address of the IP subnetwork via the netmask and the local IP address. The netmask determines the most significant bits of the local IP address as subnetwork address.

Using a default gateway The virtual Ethernet controller cannot use a default gateway.

Specifying the local IP address

With the local IP address, you specify two parts for IP networking:

- You specify a unique IP address for the virtual Ethernet controller.
- You specify the subnetwork the virtual Ethernet controller is connected to via the netmask (subnet mask/subnet prefix length).

The netmask determines the most significant bits of the local IP address as the *subnetwork address*, the least significant bits determine the *host address*.

The subnetwork address indicates the members of the same subnetwork. That means, the members of a subnetwork differ only in the host address.

Tip

If an external Ethernet device is already configured, use the subnet mask/subnet prefix length of the configured device and the resulting subnetwork address.

For example: An external Ethernet device uses the IP address **192.168.0.1** and the subnet mask **255.255.255.0**.

The resulting IP address range for the local IP address is **192.168.0.2 ... 192.168.0.254** with the subnet mask **255.255.255.0** (**192.168.0.0** and **192.168.0.255** are reserved IP addresses).

Recommended IP address ranges The following table shows the recommended IP address ranges for specifying the local IP address:

IPv4	IPv6
10.0.0.1 ... 10.255.255.254	fc00::1 ... fdff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
172.16.0.1 ... 172.31.255.254	(fc00::/7)
192.168.0.1 ... 192.168.255.254	

Unsupported IP address ranges The following table shows the unsupported IP address ranges for a local IP address:

IPv4	IPv6
Addresses with 0 as the first octet such as the default IP address: 0.0.0.0 ... 0.255.255.255	The default IP address: 0:0:0:0:0:0:0:0 (::)
Multicast IP addresses: 224.0.0.0 ... 239.255.255.255	Multicast IP addresses: ff00:: ... ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff (ff00::/8)
Loopback addresses: 127.0.0.0 ... 127.255.255.255	Loopback address: ::1
Addresses with 240 ... 255 as the first octet: 240.0.0.0 ... 255.255.255.255	Link-local addresses: fe80:: ... febf::ffff:ffff:ffff:ffff (fe80::/10)

Unfavorable IP address ranges The following IP address ranges are not recommended to be used as the local IP address:

- The first IP address of the subnetwork if all bits of the host part are set to 0.
For example: 192.168.0.0 with the subnet mask 255.255.255.0
- IPv4 Broadcast addresses: The last IP address of the subnetwork if all bits of the host part are set to 1.
For example: 192.168.0.255 with the subnet mask 255.255.255.0
- All IP addresses with a subnet mask whose first octet is less than 255.
For example: Subnet mask 0.0.0.0 or 254.0.0.0.

Workflow to specify the local IP address To specify the local IP address, you have to proceed in the following order:

1. Select the IPv4 or IPv6 Internet protocol version for IP network communication.
2. Specify how many most significant bits of the IP address are used to specify the subnetwork address. This is done via the subnet mask for IPv4 addresses or the subnet prefix length for IPv6 addresses.
3. Specify a unique IP address for the virtual Ethernet controller that is supported by ConfigurationDesk. For recommended IP addresses, refer to [Recommended IP address ranges](#) on page 1185.

Note

Remember that the IP address includes the subnetwork address. The subnetwork address must fit to the connected Ethernet network. Furthermore, it is recommended that two Ethernet setup function blocks (Virtual Ethernet Setup or Ethernet Setup) function blocks do not use the same subnetwork address and the same physical Ethernet controller. Refer to [Referencing the Ethernet network](#) on page 1184.

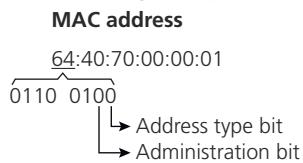
Customizing the MAC address

If the custom MAC address feature is enabled, you can customize the MAC address of the virtual Ethernet controller.

When you customize the MAC address, observe the following points:

- Flags within the MAC address indicates the administration type and address type. The address type must be a unicast address.

The following example of a MAC address shows you the position of the flags:



Flag	Value	Description
Address type bit	0	The MAC address is a unicast address.
	1	The MAC address is a multicast or broadcast address.

Flag	Value	Description
Administration bit	0	The MAC address is universally administrated by the manufacturer of the Ethernet device.
	1	The MAC address is locally administrated.

- Ethernet devices of the data link layer, such as Ethernet switches, use the MAC address for routing. Ethernet switches cannot clearly route the Ethernet frames to a physical Ethernet controller if the MAC address is not unique within the physical Ethernet network.

Make sure that the custom MAC address of the virtual Ethernet controllers are unique within the physical Ethernet network, for example, by setting the administration bit. If you specify the same custom MAC address for several virtual Ethernet controllers, reference the virtual Ethernet controllers to the same physical Ethernet controller.

Specifying the access to a VLAN

If you enable the VLAN feature, you can access a VLAN based on IEEE 802.1Q.

The VLAN ID indicates the membership to a specific VLAN. Within a VLAN, the priority to transmit Ethernet frames can be specified by setting a value for the priority code point (PCP) tag. The following table shows the different priority levels with their traffic classification:

Priority Level	Tag Value	Traffic Class
Low priority	1	Background
	0	Best effort
	2	Excellent effort
	3	Critical applications
	4	Video
	5	Voice
	6	Internetwork control
	7	Network control

UDP Receive

Where to go from here

Information in this section

Introduction (UDP Receive).....	1188
Overviews (UDP Receive).....	1190
Configuring the Function Block (UDP Receive).....	1193

Introduction (UDP Receive)

Where to go from here

Information in this section

Introduction to the Function Block (UDP Receive).....	1188
Basics on UDP/IP in the Signal Chain and Receiving Data Bytes.....	1189

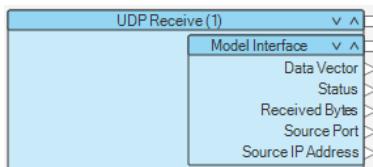
Introduction to the Function Block (UDP Receive)

Function block purpose

The UDP Receive function block type receives data bytes via UDP messages. The access to the Ethernet controller must be provided by an Ethernet Setup or Virtual Ethernet Setup function block.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Implementing a low-latency Ethernet communication.
- Receiving broadcast and multicast messages.

Basics on UDP/IP in the Signal Chain and Receiving Data Bytes

Introduction to the communication via UDP/IP

The UDP/IP communication is an end-to-end communication and provides a message-oriented transmission of data bytes.

The UDP/IP communication is suitable for the following applications:

- Time-sensitive applications that need a low-latency communication.
- Applications where error checking and correction are not necessary.

UDP/IP communication uses two protocols:

- The User Datagram Protocol (UDP)

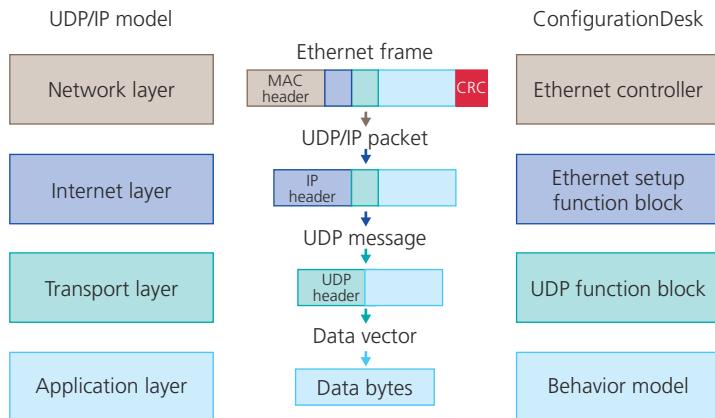
The UDP protocol is a connectionless protocol of the transport layer. The protocol does not check if the UDP messages that are received once are in the correct order.

- The Internet Protocol (IP)

The IP protocol is the protocol of the Internet layer and defines the routing of messages.

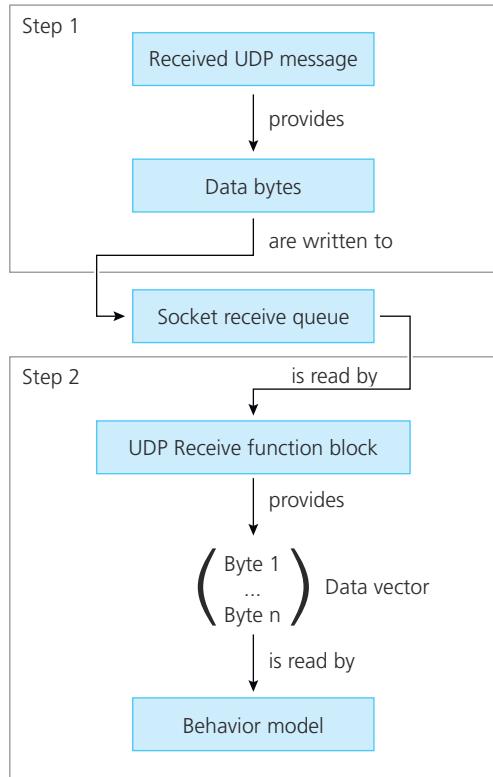
UDP/IP implementation in ConfigurationDesk

The UDP/IP model is a conceptual model on how the UDP/IP communication works. The illustration below shows the basic implementation of the UDP/IP model for receiving data bytes in ConfigurationDesk.



Receiving the UDP message

The UDP Receive function block uses a socket receive queue to receive the data bytes of a UDP message. The socket receive queue separates the process of receiving data bytes and forwarding them to the behavior model.



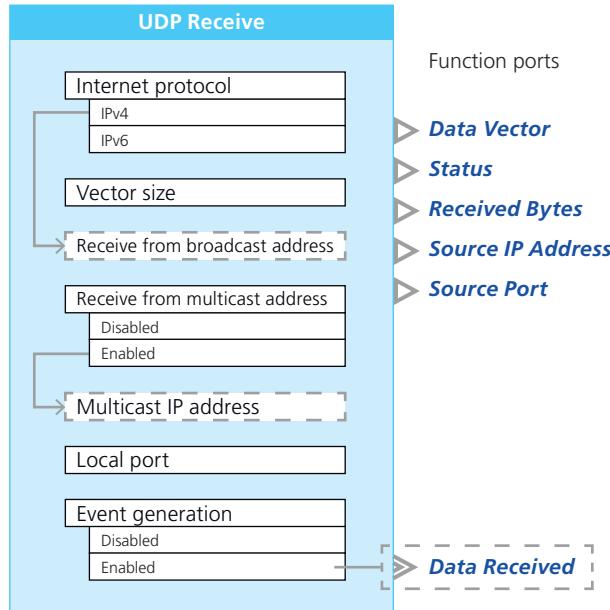
To read new data bytes of UDP messages, the behavior model must check the status of the function block to indicate a new data vector. A new data vector can be read from the behavior model until the vector is overwritten by newly received data. Because the size of the data vector does not change at runtime, the behavior model must also read the number of received data bytes to evaluate the data vector correctly.

Oversviews (UDP Receive)

Overview of Ports and Properties (UDP Receive)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Data Vector**

This function outport writes the data vector of the received UDP message to the behavior model.

If the received UDP message provides a data vector that exceeds the vector size of the function port, the exceeding data bytes are discarded.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry of the vector. ▪ The vector size depends on the Vector size property.
Dependencies	–

Status

This function outport writes status information on the Data Vector function port to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: No data read Since the last model step, no new data vector is available at the Data Vector function port. ▪ 1: Successful data read A new data vector is available at the Data Vector function outport.
Dependencies	–

Received Bytes

This function outport writes the number of valid data bytes that are provided by the Data Vector function port to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 65507 Byte
-------------	----------------------------------------------------------------------

	<ul style="list-style-type: none"> The range depends on the Vector size property.
Dependencies	–

Source IP Address

This function outport writes a vector to the behavior model. The vector provides the IP address of the Ethernet device that sent the UDP message.

Value range	<ul style="list-style-type: none"> 0 ... 255 for each entry of the vector. The Internet protocol property defines the number of entries: <ul style="list-style-type: none"> IPv4: Vector with 4 entries (32-bit IP address) IPv6: Vector with 16 entries (128-bit IP address) Each entry represents two hexadecimal digits of the IPv6 IP address in decimal notation. The common methods to abbreviate the IPv6 IP address are not supported.
Dependencies	–

Examples for vectors that represent an IP address:

- IPv4: 192;168;0;10 represents the IPv4 address **192.168.0.10**.
- IPv6: 32;1;13;184;116;178;250;80;0;1;203;62;3;117;35;69 represents the IPv6 address **2001:db8:74b2:fa50:1:cb3e:375:2345** ($32_{\text{Dec}} = 20_{\text{Hex}}$; $1_{\text{Dec}} = 01_{\text{Hex}}$; ...)

Source Port

This function outport writes the source port number to the behavior model. The source port is a part of the source socket. The source socket is the communication endpoint for the external Ethernet device to sent the UDP message.

Value range	0 ... 65535
Dependencies	–

Data Received

This event port provides an I/O event each time a new data vector is available at the Data Vector function port.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Tunable parameters

The UDP Receive function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (UDP Receive)

Where to go from here

Information in this section

- | | |
|--------------------------------------------------------|------|
| Configuring the Basic Functionality (UDP Receive)..... | 1193 |
| Configuring Standard Features (UDP Receive)..... | 1197 |

Configuring the Basic Functionality (UDP Receive)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Optimizing the data exchange with the behavior model.
- Referencing the Ethernet network.
- Specifying the communication endpoints.
- Specifying the receive functionality.
- Providing I/O events.
- Receiving broadcast and multicast messages.

Optimizing the data exchange with the behavior model

To optimize the data exchange between the behavior model and the function block, refer to [Basics on Implementing Ethernet Communication in ConfigurationDesk](#) on page 1166.

Referencing the Ethernet network

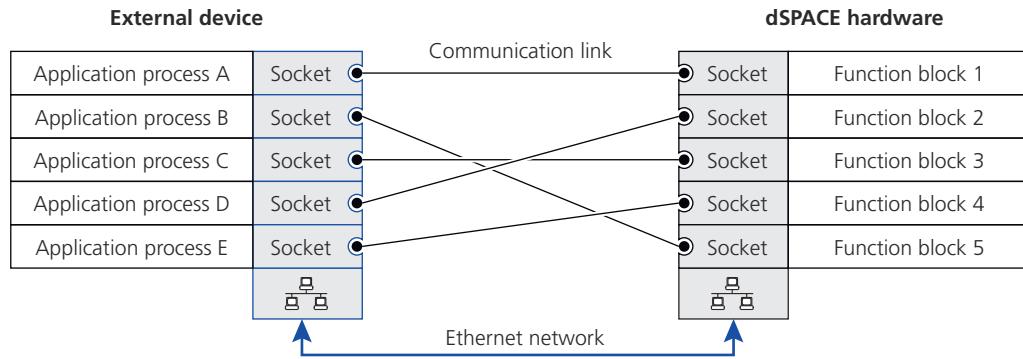
You reference an Ethernet network by assigning an Ethernet setup function block (Ethernet Setup or Virtual Ethernet Setup function block). The Ethernet setup function block provides the local IP address and the access to a configured Ethernet controller.

If several function blocks reference the same Ethernet setup function block, you must ensure that all function blocks belong to only one application process. This means, all their mapped model port blocks must be elements of the same behavior model. Otherwise, a conflict is generated and displayed in the [Conflicts Viewer](#).

Specifying the communication endpoints

Sockets clearly address the endpoints of a communication link between the sending application process and the UDP Receive function block. Therefore, UDP Receive function blocks must not use the same socket. A socket is the combination of the IP address, the transport protocol, and the port.

The following illustration is an example of communication links between sockets. The external device and the dSPACE hardware are physically connected via an Ethernet network.



Specifying the local socket address The local socket addresses the communication endpoint of a UDP Receive function block.

To specify the local socket address, you have to specify only a static port number for the local port.

The other parts of the local socket address are defined as follows:

- The IP address is defined by the assigned Ethernet setup function block.
- The transport protocol is UDP.

Tip

You can share the same port with the UDP Transmit function block, i.e., you can use the same port to receive and send UDP messages.

Reading the source socket address The source socket addresses the communication endpoint of the sending application process at the external Ethernet device.

The source socket address is provided by function ports of the UDP Receive function block.

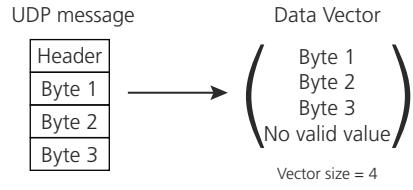
Specifying the used IP protocol The assigned Ethernet setup function block provides the IP address. Nevertheless, you have to set the IP protocol version to the version of the assigned Ethernet setup function block.

Specifying the receive functionality

The vector size of the Data Vector function port limits the maximum number of data bytes that can be received from a UDP message. Therefore, specify a vector size for the Data Vector function port that can provide the data bytes of the longest UDP message.

The following effects occur if the data vector size does not match the number of data bytes provided by the received UDP message.

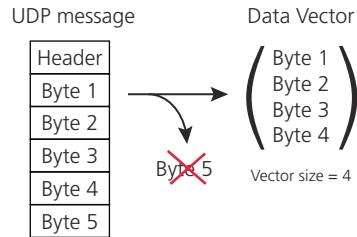
- Number of received data bytes < Data vector size:



Effect: All received data bytes are provided by the Data Vector function port, but not all elements of the data vector are valid.

The Received Bytes function port provides the number of valid data bytes.

- Number of received data bytes > Data vector size:



Effect: Not all received data bytes are provided by the Data Vector function port. Excess data bytes are discarded and only the first data bytes are forwarded to the Data Vector function port.

For more information on receiving data bytes, refer to [Basics on UDP/IP in the Signal Chain and Receiving Data Bytes](#) on page 1189.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

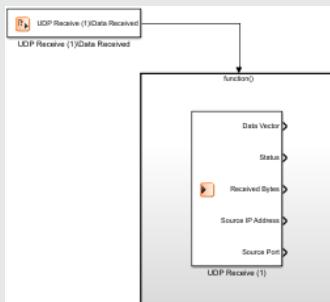
The UDP Receive function block can generate an I/O event each time a new UDP message is received and the data bytes can be read.

To use the I/O events in the behavior model, you have to assign them to a task via the Data Received property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

NOTICE**Unread data causes task overrun and data loss**

The UDP Receive function block generates I/O events until the data values of a received message are read. If the triggered runnable function does not read the received data bytes, the function block triggers the runnable functions multiple times. The multiple triggering prevents the execution of other tasks and causes a task overrun.

- Add the UDP Receive model port block to the runnable function that is triggered by the generated I/O events.

**Receiving broadcast messages**

Broadcast messages can be read by all members of a IPv4 network. IPv6 networks do not support broadcast messages.

To receive broadcast messages with the UDP Receive function block, you have to enable the receiving of broadcast messages. If you enable the receiving of broadcast messages, the UDP Receive function block writes the data bytes and source socket addresses of received broadcast messages to the behavior model.

Receiving multicast messages

Multicast messages can be read by participants of a multicast group.

To receive multicast messages with the UDP Receive function block, you have to enable the receiving of multicast IP addresses and to set the IP address of the multicast group.

The function block communicates the multicast group participation to the Ethernet network and writes the data bytes and source socket addresses of received multicast messages to the behavior model. The function block uses the IGMP protocol (IPv4) and ICMPv6 protocol (IPv6) to make the participation known to the Ethernet network.

Note

If you enable the reception of IPv4 multicast messages, the function block additionally receives broadcast messages although the reception of broadcast messages is disabled.

- To make sure that only unicast and multicast messages are taken into account, filter out potential broadcast messages in the behavior model, for example, by reading only messages with known source IP addresses and ports.

Configuring Standard Features (UDP Receive)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The UDP Receive function block does not provide an electrical interface.

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

UDP Transmit

Where to go from here

Information in this section

Introduction (UDP Transmit).....	1198
Overviews (UDP Transmit).....	1200
Configuring the Function Block (UDP Transmit).....	1203

Introduction (UDP Transmit)

Where to go from here

Information in this section

Introduction to the Function Block (UDP Transmit).....	1198
Basics on UDP/IP in the Signal Chain and the Transmitting of Data Bytes.....	1199

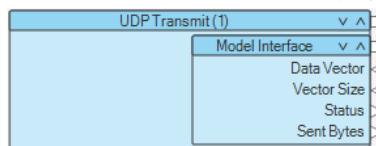
Introduction to the Function Block (UDP Transmit)

Function block purpose

The UDP Transmit function block type transmits data bytes via UDP messages. The access to the Ethernet controller must be provided by an Ethernet Setup or Virtual Ethernet Setup function block.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Implementing a low-latency Ethernet communication.
- Transmitting broadcast and multicast messages.

Basics on UDP/IP in the Signal Chain and the Transmitting of Data Bytes

Introduction to the communication via UDP/IP

The UDP/IP communication is an end-to-end communication and provides a message-oriented transmission of data bytes.

The UDP/IP communication is suitable for the following applications:

- Time-sensitive applications that need a low-latency communication.
- Applications where error checking and correction are not necessary.

UDP/IP communication uses two protocols:

- The User Datagram Protocol (UDP)

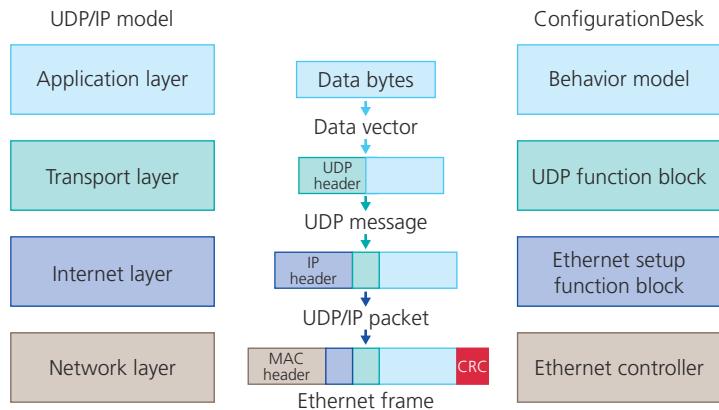
The UDP protocol is a connectionless protocol of the transport layer. The protocol does not check if the data bytes are sent correctly via the Ethernet network.

- The Internet Protocol (IP)

The IP protocol is the protocol of the Internet layer and defines the routing of messages.

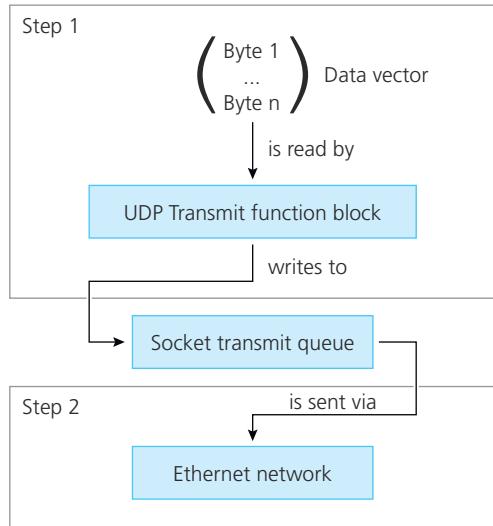
UDP/IP implementation in ConfigurationDesk

The UDP/IP model is a conceptual model on how the UDP/IP communication works. The illustration below shows the basic implementation of the UDP/IP model for transmitting data bytes in ConfigurationDesk.



Transmitting the UDP message

The UDP Transmit function block uses a socket transmit queue to transmit the data bytes via a UDP message. The socket transmit queue separates the process of reading the data bytes from the behavior model and sending them via the Ethernet network.



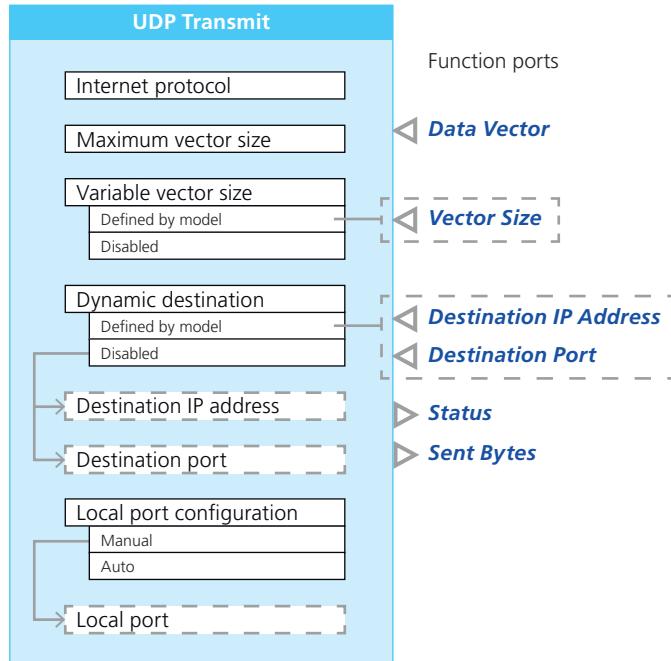
Function ports let you check whether the data bytes of the provided data vector are correctly sent to the socket transmit queue. The function block does not check if the data bytes are sent correctly via the Ethernet network.

Oversviews (UDP Transmit)

Overview of Ports and Properties (UDP Transmit)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Data Vector

This function import reads the data vectors to be transmitted from within the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... 255 for each entry of the vector. The maximum vector size depends on the Maximum vector size property.
Dependencies	–

Vector Size

This function import reads the data vector size from within the behavior model to specify the number of data bytes that are forwarded from the **Data Vector** function port to the socket transmit queue.

Value range	<ul style="list-style-type: none"> 0 ... 65507 Byte The range depends on the Maximum vector size property.
Dependencies	Available only if the Variable vector size property is set to Defined by model .

Destination IP Address

This function import reads the destination IP address from within the behavior model. The destination IP address is the IP address of the external Ethernet device that receives the UDP messages.

Value range	<ul style="list-style-type: none"> 0 ... 255 for each entry of the vector. The Internet protocol property defines the number of entries: <ul style="list-style-type: none"> IPv4: Vector with 4 entries (32-bit IP address)
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ IPv6: Vector with 16 entries (128-bit IP address) Each entry represents two hexadecimal digits of the IPv6 IP address in decimal notation. The common methods to abbreviate the IPv6 IP address are not supported.
Dependencies	Available only if the Dynamic destination property is set to Defined by model.

Examples for vectors that represent an IP address:

- IPv4: 192;168;0;10 represents the IPv4 address **192.168.0.10**.
- IPv6: 32;1;13;184;116;178;250;80;0;1;203;62;3;117;35;69 represents the IPv6 address **2001:db8:74b2:fa50:1:cb3e:375:2345** ($32_{\text{Dec}} = 20_{\text{Hex}}$; $1_{\text{Dec}} = 01_{\text{Hex}}$; ...)

Destination Port

This function import reads the destination port number from within the behavior model. The destination port number is a part of the destination socket. The destination socket is the communication endpoint at the external Ethernet device.

Value range	0 ... 65535
Dependencies	Available only if the Dynamic destination property is set to Defined by model.

Status

This function outport writes status information on the Data Vector function port to the behavior model. If a network fault occurs, the Message Viewer displays an error message with an error number.

Note

- The error message and the error number (errno) depends on the operating system that is executed on the real-time hardware. For example: If the host is down, QNX™ (SCALEXIO until Release 2020-A) outputs errno 264, whereas real-time Linux® (SCALEXIO as of Release 2020-B and MicroAutoBox III) outputs errno 112.
- The UDP protocol is a transport protocol for a low-latency communication, but it is not a reliable transport protocol. Therefore, the function block cannot check if the data bytes are sent correctly via the Ethernet network. Refer to [Basics on UDP/IP in the Signal Chain and the Transmitting of Data Bytes](#) on page 1199.

Value range	<ul style="list-style-type: none"> ▪ 0: No data sent No data bytes are forwarded to the socket transmit queue from the Data Vector function import. ▪ 1: Successful data sent The data bytes are successfully sent to the socket transmit queue from the Data Vector function import.
Dependencies	–

Sent Bytes

This function outport writes the number of data bytes that are forwarded from the Data Vector function port to the socket transmit queue.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 65507 Byte ▪ The range depends on the Maximum vector size property. The actual maximum value depends on the settings of the Variable vector size property and the provided value at the Vector Size function port.
Dependencies	–

Tunable parameters

The UDP Transmit function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (UDP Transmit)

Where to go from here**Information in this section**

- | | |
|--------------------------------------------------------------------------|------|
| Configuring the Basic Functionality (UDP Transmit) | 1203 |
| Configuring Standard Features (UDP Transmit) | 1206 |

Configuring the Basic Functionality (UDP Transmit)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Optimizing the data exchange with the behavior model.
- Referencing the Ethernet network.
- Specifying the communication endpoints.
- Specifying the transmit functionality.
- Transmitting broadcast and multicast messages.

Optimizing the data exchange with the behavior model

To optimize the data exchange between the behavior model and the function block, and to process data values other than 8-bit unsigned integer values, refer to [Basics on Implementing Ethernet Communication in ConfigurationDesk](#) on page 1166.

Referencing the Ethernet network

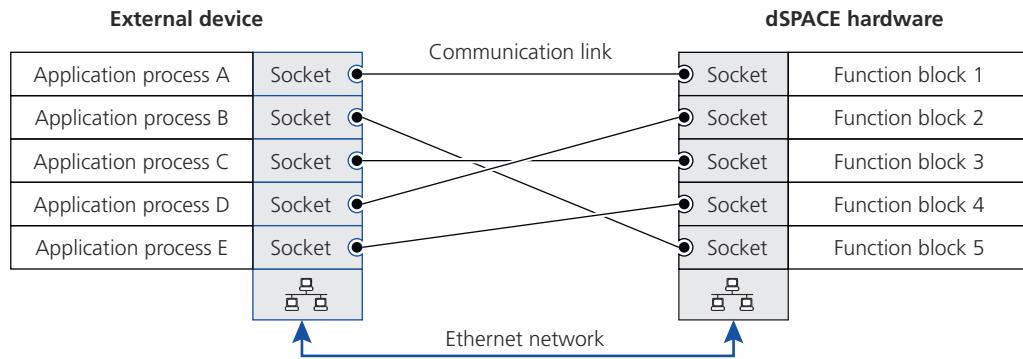
You reference an Ethernet network by assigning an Ethernet setup function block (Ethernet Setup or Virtual Ethernet Setup function block). The Ethernet setup function block provides the local IP address and the access to a configured Ethernet controller.

If several function blocks reference the same Ethernet setup function block, you must ensure that all function blocks belong to only one application process. This means, all their mapped model port blocks must be elements of the same behavior model. Otherwise, a conflict is generated and displayed in the Conflicts Viewer.

Specifying the communication endpoints

Sockets clearly address the endpoints of a communication link between the UDP Transmit function block and the receiving application process. Therefore, UDP Transmit function blocks must not use the same socket. A socket is the combination of the IP address, the transport protocol, and the port.

The following illustration is an example of communication links between sockets. The external device and the dSPACE hardware are physically connected via an Ethernet network.



Specifying the local socket address The local socket addresses the communication endpoint of a UDP Transmit function block.

To specify the local socket address, you have to specify only a port number for the local port. With the Local port configuration property, you can select whether to use an automatically set port with a random port number or to specify a static port number.

The other parts of the local socket address are defined as follows:

- The IP address is defined by the assigned Ethernet setup function block.
- The transport protocol is UDP.

Tip

You can share the same port with the UDP Receive function block, i.e., you can use the same port to receive and send UDP messages.

If you share ports, you must use static port numbers.

Specifying the destination socket address The destination socket addresses the communication endpoint of the receiving application process at the external Ethernet device.

With the Dynamic destination property, you can select whether to define the destination socket dynamically from within the behavior model or to specify a destination socket with a static address via properties.

Specifying the used IP protocol The assigned Ethernet setup function block provides the IP address. Nevertheless, you have to set the IP protocol version to the version of the assigned Ethernet setup function block.

Specifying the transmit functionality

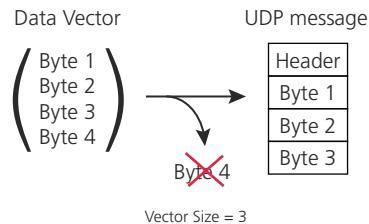
The vector size of the Data Vector function port limits the maximum number of data bytes that can be transmitted via a UDP message. Therefore, make sure that the value of the Maximum vector size property can forward the required amount of data bytes.

Note

If the Data Vector function port reads more data bytes per model step than the Ethernet network can transmit, a task overrun occurs. You have to reload the application to start the application again.

- Limit the number of data bytes that are provided to the function block via the Maximum vector size property.
- Increase the model step size.

Transmitting UDP messages with different data lengths With the Vector Size function port, you can specify the number of data bytes that are forwarded from the Data Vector function port to the socket transmit queue. This let you send UDP messages with different data lengths.



The Vector Size function port is available only if the Variable vector size property is set to Defined by model.

Transmitting broadcast and multicast messages

If you set the destination IP address to the address space for broadcast or multicast messages, the UDP Transmit function block transmits the broadcast/multicast messages. The table below shows you the IP address ranges.

IP Protocol	Broadcast Addresses	Multicast Addresses
IPv4	<ul style="list-style-type: none"> ▪ 255.255.255.255 ▪ All bits of the host address are set to 1, e.g., 192.168.0.255/24¹⁾. 	224.0.0.0 ... 239.255.255.255
IPv6	—	ff00:: ... ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff (ffff00::/8). ²⁾

¹⁾ IP address 192.168.0.255 with the netmask 255.255.255.0.

²⁾ IPv6 multicast addresses are organized in multicast groups based on the intended traffic. The multicast addresses ff01::/16, ff11::/16, ..., ffff1::/16 are interface-local scope multicast addresses. These addresses are useful only for loopback delivery of multicast messages within a node.

Keep in mind, that the IPv6 Internet protocol does not support broadcast messages.

Note

The transmission of data bytes via a UDP broadcast message is limited to 1472 Bytes, but you can transmit the maximum number of data bytes with multicast messages.

Configuring Standard Features (UDP Transmit)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The UDP Transmit function block does not provide an electrical interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

TCP

Where to go from here

Information in this section

Introduction to the Functionality (TCP).....	1207
Overviews (TCP).....	1213
Configuring the Function Block (TCP).....	1219

Introduction to the Functionality (TCP)

Where to go from here

Information in this section

Introduction to the Function Block.....	1207
Basics on TCP/IP in the Signal Chain.....	1208
Controlling the TCP/IP Connection.....	1209
Transmitting and Receiving Data Bytes.....	1211

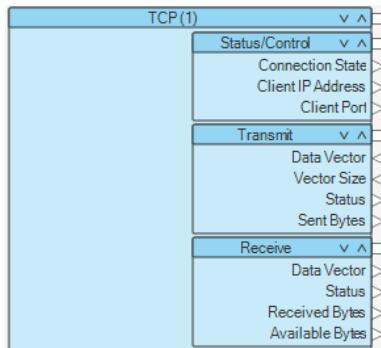
Introduction to the Function Block

Function block purpose

The TCP function block type can transmit and receive data bytes via TCP messages. The access to the Ethernet controller must be provided by an Ethernet Setup or Virtual Ethernet Setup function block.

Block overview

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.

**Main feature**

This function block lets you implement an Ethernet communication with a reliable transmission of data bytes.

Basics on TCP/IP in the Signal Chain

Introduction to the communication via TCP/IP

TCP/IP communication is an end-to-end communication and provides a reliable transmission of data bytes in a data stream. The reliable transmission includes the following features:

- Detecting of lost data bytes and automatic correction.
- Reconstruction of the order data bytes are sent.
- Control of the data flow to prevent an receiver or network overload.

TCP/IP communication uses two protocols:

- The Transmission Control Protocol (TCP)

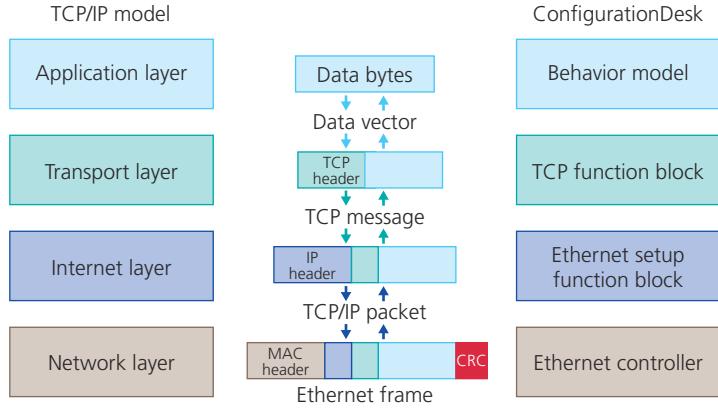
The TCP protocol is a connection-oriented protocol of the transport layer that works in both directions. Before data bytes can be transported over the network, a virtual connection must be established between the communication endpoints.

- The Internet Protocol (IP)

The IP protocol is the protocol of the Internet layer and defines the routing of messages.

TCP/IP model in ConfigurationDesk

The TCP/IP model is a conceptual model on how the TCP/IP communication works. The illustration below shows the basic implementation of the TCP/IP model in ConfigurationDesk.

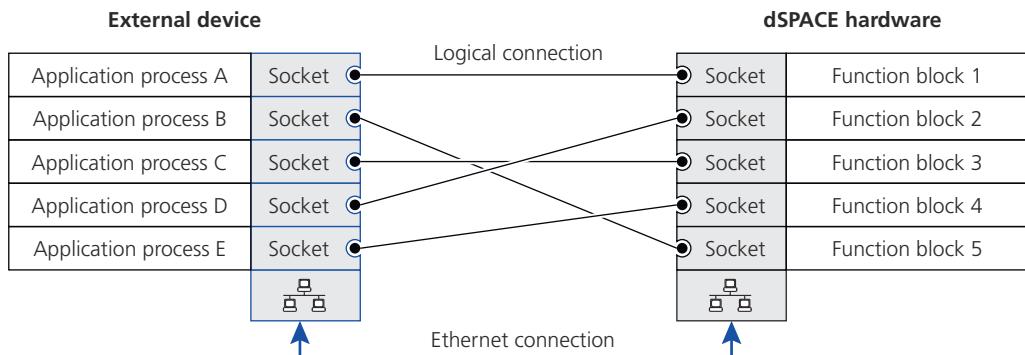


Controlling the TCP/IP Connection

Addressing the communication endpoints

Sockets clearly address the communication endpoints between the application process that runs on the external device and the TCP function block. A socket is the combination of the IP address, the transport protocol, and the port.

The following illustration is an example of logical connections between sockets. The external device and the dSPACE hardware are physically connected via an Ethernet connection.



Socket address of the application process that runs on the external device The function block provides the socket address when the logical connection is established. Only if you request a TCP connection, you must specify the IP address and the port number of the destination.

Socket address of the TCP function block (local socket address) The local socket address is specified as follows to address the function block:

- The IP address is set by the assigned Ethernet setup function block (Ethernet Setup or Virtual Ethernet Setup function block).

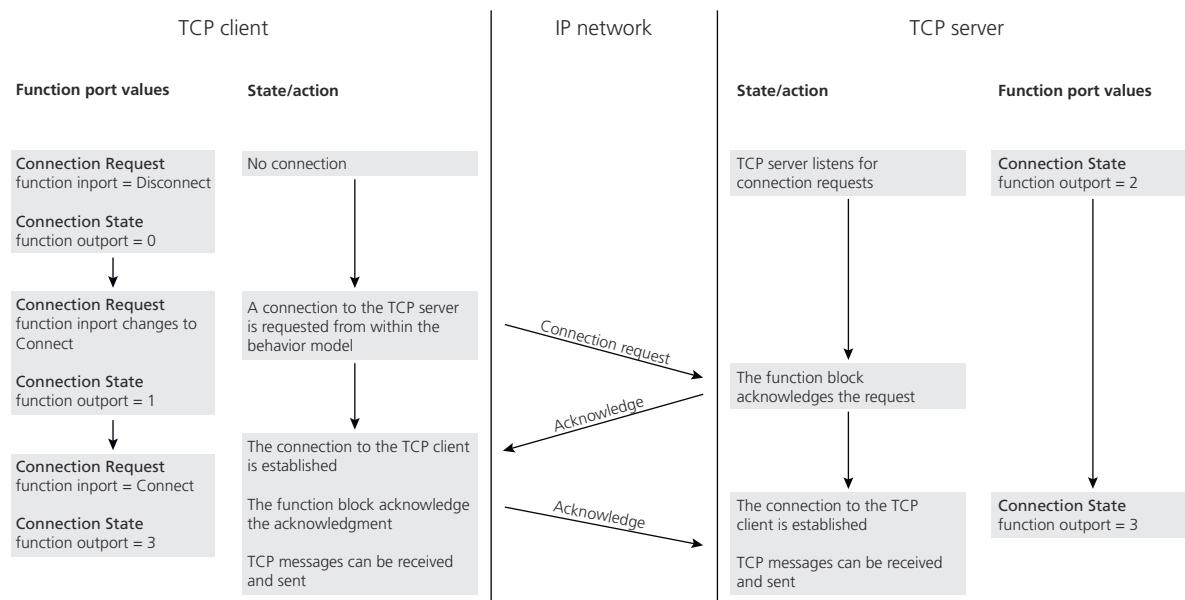
- TCP is the transport protocol.
- The local port is set via the TCP function block.

Establishing a TCP/IP connection

To establish a TCP/IP connection, the TCP client requests a connection and the TCP server listens for connection requests.

Depending on the TCP function block settings, a function block is either a TCP client or a TCP server. If you set the function block to run as a TCP client, the function block provides a function port to request a TCP connection.

The illustration below shows the connection process between two TCP function blocks and the values that are provided by the function ports to control the connection.

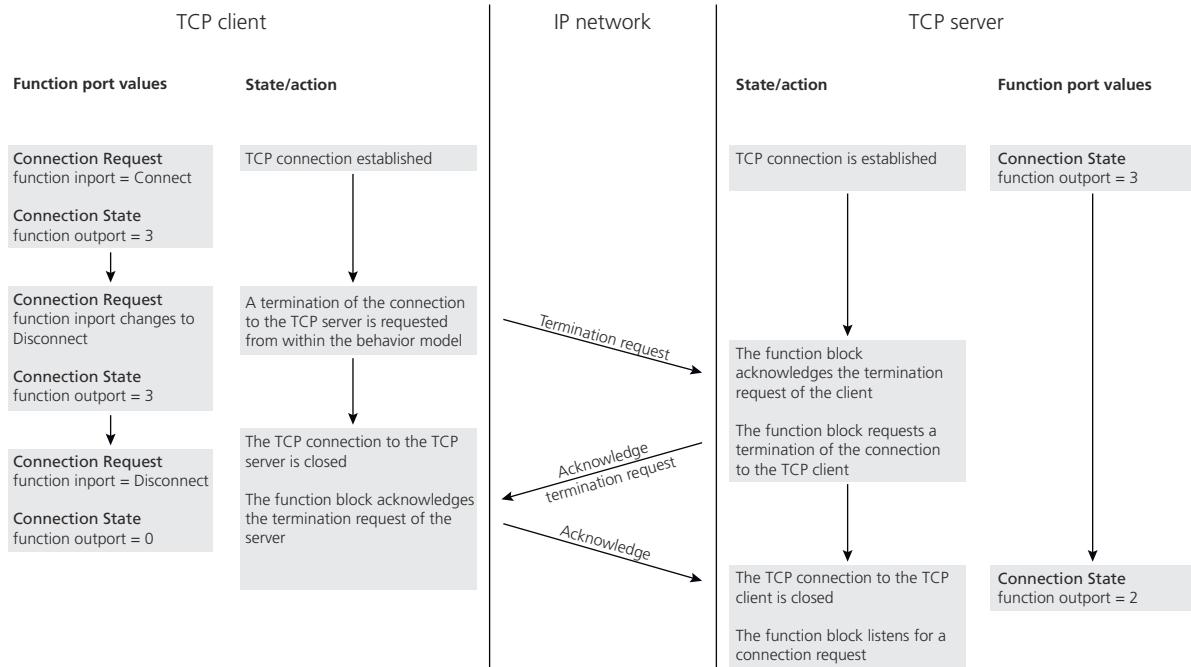


Note

A TCP function block that runs as a server can only be connected to one TCP client.

Terminating a TCP/IP connection

The illustration below shows the termination process between two connected TCP function blocks and the values that the function blocks provide to control the termination.

**Note**

A TCP function block that runs as a TCP server cannot terminate a TCP connection.

TCP server does not disconnect If the TCP client disconnects before the transmit or receive functionality of the TCP server is requested at least once by the real-time application, the TCP server remains in the connected state. A new connection to the TCP server cannot be established until you restart the real-time application.

Observe the following points to make a request of the transmit or receive functionality of the TCP server possible:

- At the TCP function block, enable at least one transfer functionality (transmit or the receive functionality).
- If you are modeling with enabled subsystems, make sure that the enabled subsystem is executed.

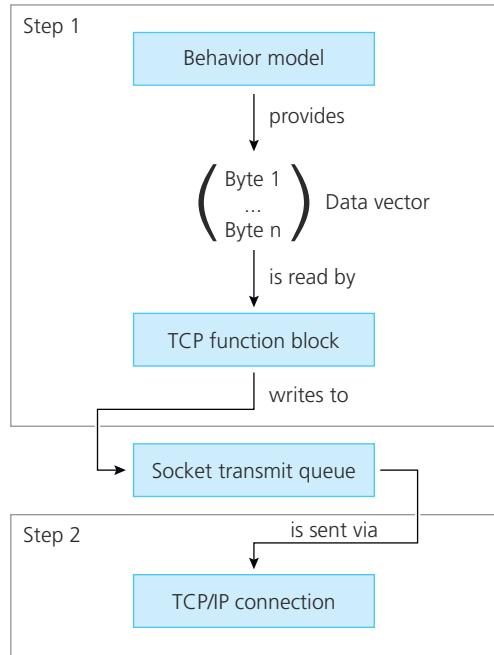
Transmitting and Receiving Data Bytes

Introduction

After a TCP/IP connection is established, the TCP function block can receive and transmit data bytes. Whether the TCP function block runs as server or client has no effect on the transmitting and receiving of data bytes.

Transmitting data bytes

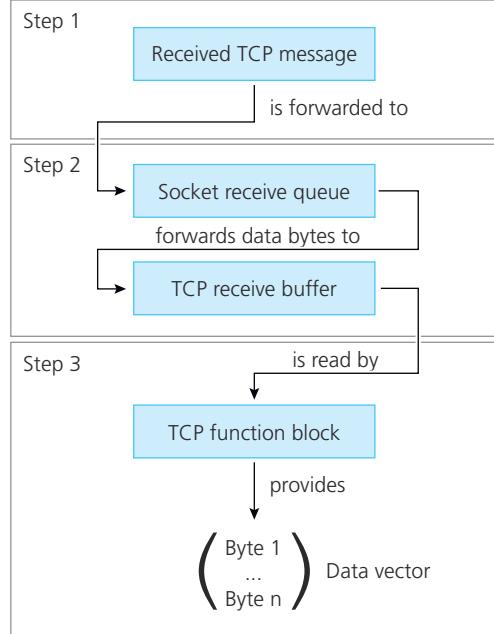
The TCP function block uses a socket transmit queue to transmit data bytes. The socket transmit queue separates the process of reading the data bytes from the behavior model and sending them via the TCP/IP connection.



Function ports let you check whether the provided data vector is properly written to the socket transmit queue. Properly sending of the data bytes from the socket transmit queue to the destination ensures the reliable TCP/IP connection.

Receiving data bytes

The data bytes are received in three steps as shown in the following illustration.



Step 2 separates the process of reading the data bytes from the received TCP message and writing the data bytes to the behavior model.

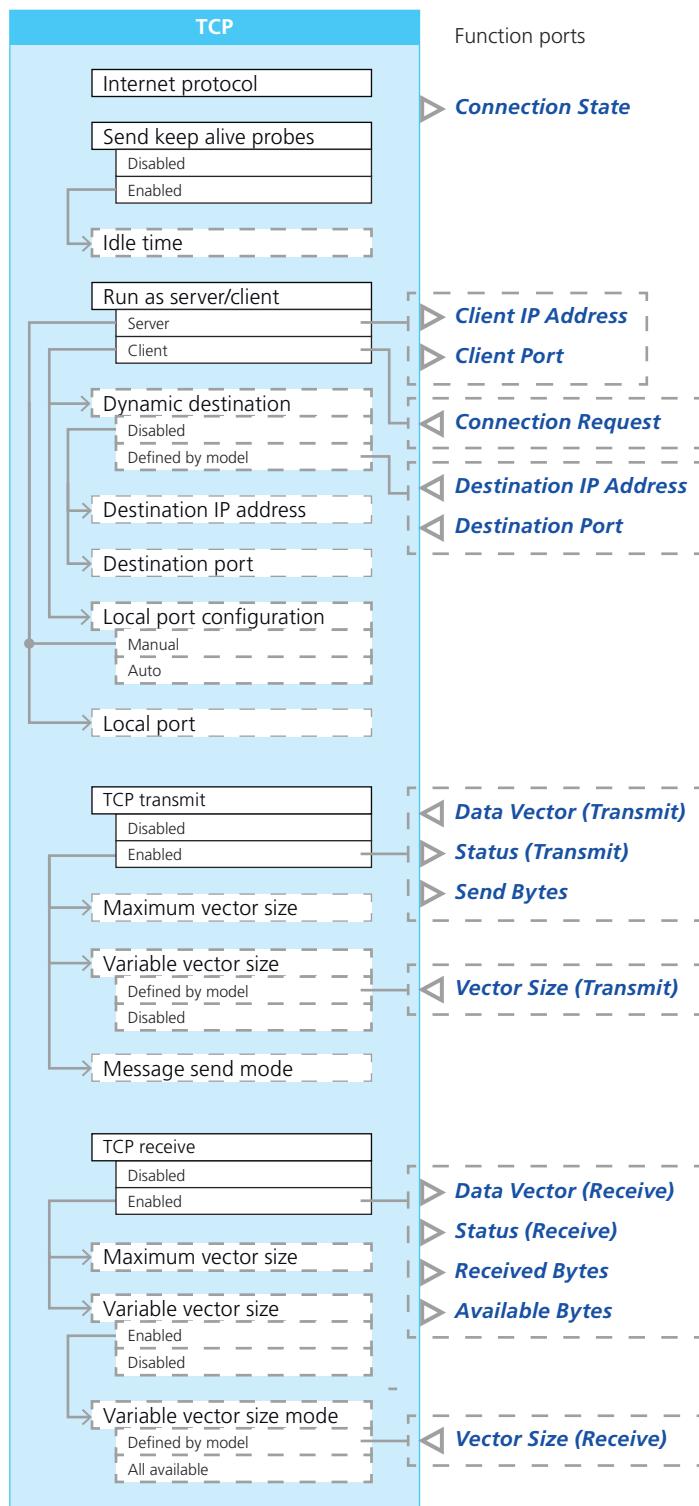
Function ports let you check whether the data bytes of the TCP receive buffer are properly written to the behavior model. Proper reception of the TCP message ensures the reliable TCP/IP connection.

Oversviews (TCP)

Overview of Ports and Properties (TCP)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Connection State

This function outport writes state information on the TCP/IP connection.

Value range	<ul style="list-style-type: none"> ▪ If the Run as server / client property is set to Server: <ul style="list-style-type: none"> ▪ 2: The function block listens for connection requests of a TCP client. ▪ 3: The function block is connected to a TCP client. ▪ 4: The connection to the TCP client has been broken. The function block restarts the listen process. ▪ If the Run as server / client property is set to Client: <ul style="list-style-type: none"> ▪ 0: The function block is not connected. ▪ 1: The function block tries to connect a TCP server. ▪ 3: The function block is connected to a TCP server. ▪ 4: The function block retries to connect a TCP server after the connection could not be established. ▪ 5: The function block cannot connect to the TCP server.
Dependencies	–

Client IP Address

This function outport writes a vector to the behavior model that provides the IP address of the connected TCP client.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry of the vector. ▪ The Internet protocol property defines the number of entries: <ul style="list-style-type: none"> ▪ IPv4: Vector with 4 entries (32-bit IP address) ▪ IPv6: Vector with 16 entries (128-bit IP address) <p>Each entry represents two hexadecimal digits of the IPv6 IP address in decimal notation. The common methods to abbreviate the IPv6 IP address are not supported.</p>
Dependencies	Available only if the Run as server / client property is set to Server.

Examples for vectors that represent an IP address:

- IPv4: 192;168;0;10 represents the IPv4 address **192.168.0.10**.
- IPv6: 32;1;13;184;116;178;250;80;0;1;203;62;3;117;35;69 represents the IPv6 address **2001:db8:74b2:fa50:1:cb3e:375:2345** ($32_{\text{Dec}} = 20_{\text{Hex}}$; $1_{\text{Dec}} = 01_{\text{Hex}}$; ...)

Client Port

This function outport writes the port number of the connected TCP client to the behavior model.

Value range	0 ... 65535
Dependencies	Available only if the Run as server / client property is set to Server.

Connection Request

This function import requests a TCP/IP connection from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ A change from Disconnect (0) to Connect (1) (rising edge): The function block requests a connection to the specified TCP server. ▪ A change from Connect (1) to Disconnect (0) (falling edge): The function block terminates the connection to the TCP server.
Dependencies	Available only if the Run as server / client property is set to Client.

Destination IP Address

This function import reads the destination IP address from within the behavior model. The destination IP address is a part of the destination socket. The destination socket is the connection endpoint at the external Ethernet device.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry of the vector. ▪ The Internet protocol property defines the number of entries: <ul style="list-style-type: none"> ▪ IPv4: Vector with 4 entries (32-bit IP address) ▪ IPv6: Vector with 16 entries (128-bit IP address) Each entry represents two hexadecimal digits of the IPv6 IP address in decimal notation. The common methods to abbreviate the IPv6 IP address are not supported.
Dependencies	Available only if the Run as server / client property is set to Client and the Dynamic destination property is set to Defined by model.

Examples for vectors that represent an IP address:

- IPv4: 192;168;0;10 represents the IPv4 address **192.168.0.10**.
- IPv6: 32;1;13;184;116;178;250;80;0;1;203;62;3;117;35;69 represents the IPv6 address **2001:db8:74b2:fa50:1:cb3e:375:2345** ($32_{\text{Dec}} = 20_{\text{Hex}}$; $1_{\text{Dec}} = 01_{\text{Hex}}$; ...)

Destination Port

This function import reads the destination port number from within the behavior model. The destination port number is a part of the destination socket. The destination socket is the connection endpoint at the external Ethernet device.

Value range	0 ... 65535
Dependencies	Available only if the Run as server / client property is set to Client and the Dynamic destination property is set to Defined by model.

Data Vector (Transmit)

This function import reads the data vectors to be transmitted from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry of the vector. ▪ The maximum vector size depends on the Maximum vector size property.
Dependencies	Available only if the TCP transmit property is set to Enabled.

Status (Transmit)

This function outport writes status information on the Data Vector (Transmit) function port to the behavior model. If a network fault occurs, the Message Viewer displays an error message with an error number.

Note

The error message and the error number (errno) depends on the operating system that is executed on the real-time hardware. For example: If the host is down, QNX™ (SCALEXIO until Release 2020-A) outputs errno 264, whereas real-time Linux® (SCALEXIO as of Release 2020-B and MicroAutoBox III) outputs errno 112.

Value range	<ul style="list-style-type: none"> ▪ 0: No data sent No data bytes are sent to the socket transmit queue from the Data Vector function import. ▪ 1: Successful data sent The data bytes are successfully sent to the socket transmit queue from the Data Vector function import.
Dependencies	Available only if the TCP transmit property is set to Enabled.

Sent Bytes

This function outport writes the number of data bytes that are sent to the socket transmit queue from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 200704 Byte ▪ The range depends on the Maximum vector size property. The actual maximum value depends on the settings of the Variable vector size property and the provided value at the Vector Size (Transmit) function port.
Dependencies	Available only if the TCP transmit property is set to Enabled.

Vector Size (Transmit)

This function import reads the data vector size from within the behavior model to specify the number of data bytes that are forwarded from the Data Vector (Transmit) function port to the socket transmit queue.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 200704 ▪ The range depends on the Maximum vector size property.
Dependencies	Available only if the TCP transmit property is set to Enabled and the Variable vector size property is set to Defined by model.

Data Vector (Receive)

This function outport writes the data vectors with the received data bytes to the behavior model.

Data bytes of received TCP messages are first written to the TCP receive buffer. If the required number of data bytes are available, the data bytes are forwarded from the TCP receive buffer to the Data Vector (Receive) function port. Exceeding data bytes remain in the TCP receive buffer and will be forwarded the next time.

You specify the required number of data bytes to be forwarded with the Variable vector size and Variable vector size mode properties, and the value provided at the Vector Size (Receive) function port.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry of the vector. ▪ The maximum vector size depends on the Maximum vector size property.
Dependencies	Available only if the TCP receive property is set to Enabled.

Status (Receive)

This function outport writes status information on the Data Vector (Receive) function port to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: No data read The Data Vector (Receive) function port is not updated with a new data vector from the TCP receive buffer. ▪ 1: Successful data read A new data vector is successfully read from the TCP receive buffer and forwarded to the Data Vector (Receive) function port.
Dependencies	Available only if the TCP receive property is set to Enabled.

Received Bytes

This function outport writes the number of data bytes that are forwarded to the Data Vector (Receive) function port to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 200704 Byte ▪ The range depends on the Maximum vector size property.
Dependencies	Available only if the TCP receive property is set to Enabled.

Available Bytes

This function outport writes the number of data bytes that are available in the TCP receive buffer to the behavior model.

Value range	0 ... 1000000 Byte
Dependencies	Available only if the TCP receive property is set to Enabled.

Vector Size (Receive)

This function import reads the vector size for the data bytes that are forwarded to the Data Vector (Receive) function port.

The function block forwards only the data bytes of the TCP receive buffer that match the read vector size. Excess data bytes remain in the TCP receive buffer and will be forwarded the next time.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 200704 ▪ The range depends on the Maximum vector size property.
Dependencies	Available only if the TCP receive property is set to Enabled and the Variable vector size mode property is set to Defined by model.

Signal ports

The TCP function block does not provide signal ports.

Tunable parameters

The TCP function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (TCP)

Where to go from here**Information in this section**

[Configuring the Basic Functionality \(TCP\)](#).....1220

[Configuring Standard Features \(TCP\)](#).....1224

Configuring the Basic Functionality (TCP)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Optimizing the data exchange with the behavior model.
- Referencing the Ethernet network.
- Running the function block as server or client.
- Specifying the communication endpoints.
- Specifying the transmit functionality.
- Specifying the receive functionality.
- Sending keep-alive probes to check the connection and/or prevent disconnections.

Optimizing the data exchange with the behavior model

To optimize the data exchange between the behavior model and the function block, and to process data values other than 8-bit unsigned integer values, refer to [Basics on Implementing Ethernet Communication in ConfigurationDesk](#) on page 1166.

Referencing the Ethernet network

You reference an Ethernet network by assigning an Ethernet setup function block (Ethernet Setup or Virtual Ethernet Setup function block). The Ethernet setup function block provides the local IP address and the access to a configured Ethernet controller.

If several function blocks reference the same Ethernet setup function block, you must ensure that all function blocks belong to only one application process. This means, all their mapped model port blocks must be elements of the same behavior model. Otherwise, a conflict is generated and displayed in the **Conflicts Viewer**.

Running the function block as server or client

You select whether the function block runs as a TCP server or TCP client. The selection impacts the ability to establish or terminate a TCP connection as follows:

- A TCP server only listens for a connection request. It cannot request or terminate a connection.
- A TCP client requests or terminates a connection. It cannot listen for a connection request.

Whether the TCP function block runs as server or client has no effect on the transmitting and receiving of data bytes.

Specifying the communication endpoints

Sockets clearly address the endpoints of a TCP connection between the function block and the application process. A socket is the combination of the IP address, the transport protocol, and the port.

To specify the socket address of the TCP function block, you only have to specify the port number for the local port. TCP function blocks that are assigned to the same Ethernet setup function block must not use the same local port, because this results in a socket address that cannot be mapped clearly to one TCP function block.

If the function block runs as a TCP client, you also have to specify the destination socket address to request a TCP connection to an external Ethernet device.

The table below shows how the socket addresses are specified for the different operation modes:

Operation Mode	Local Socket Address of the Function Block		Socket Address of the External Ethernet Device
	IP Address	Local Port	
TCP server	Defined by the assigned Ethernet setup function block.	You specify the local port with the Local port property.	Automatically provided when the TCP connection is established.
TCP client	Defined by the assigned Ethernet setup function block.	With the Local port configuration property, you can select whether to use an automatically set port with a random port number or to specify a static port.	With the Dynamic destination property, you can select whether to define the destination socket dynamically from within the behavior model or to specify a static socket via properties.

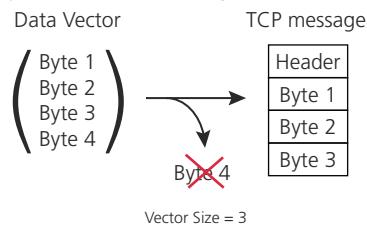
Specifying the used IP protocol The assigned Ethernet setup function block provides the IP address. Nevertheless, you have to set the IP protocol version to the version of the assigned Ethernet setup function block.

Specifying the transmit functionality

If you enable the transmit functionality, you can send TCP messages to the connected external Ethernet device.

The vector size of the Data Vector (Transmit) function port limits the maximum number of data bytes that can be transmitted via a TCP message. Therefore, make sure that the value of the Maximum vector size property of the Transmit function can forward the required number of data bytes.

Transmitting TCP messages with different data lengths With the Vector Size function port you reduce the number of data bytes that are forwarded from the Data Vector (Transmit) function port to the socket transmit queue. This lets you send TCP messages with different data length.



The Vector Size function port is available only if the Variable vector size property is set to Defined by model.

Providing the data bytes The Data Vector function port reads the data bytes that are provided by the behavior model. If the Data Vector function port reads more data bytes per model step than the Ethernet network can transmit, the socket transmit queue will overflow. In this case, the Message Viewer displays a warning message and the Status (Transmit) function port changes to 0.

If a task overrun occurs, take one of the following measures:

- Decrease the setting of the Maximum vector size property.
- Increase the model step size.
- Optimize the Ethernet data exchange. Refer to [Basics on Implementing Ethernet Communication in ConfigurationDesk](#) on page 1166.
- In the Executable Application table, increase the setting of the Number of Accepted Overruns property.

Selecting the message send mode You can select a delay when sending the data bytes from the socket transmit queue to increase data throughput (Nagle algorithm) or you can immediately send each data vector to reduce the latency.

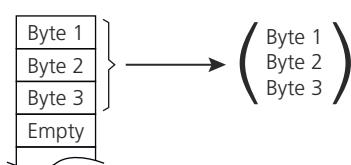
Specifying the receive functionality

If you enable the receive functionality, you can receive TCP messages from the connected external Ethernet device.

The vector size of the Data Vector (Receive) function port limits the maximum number of data bytes that can be forwarded from the TCP receive buffer. Therefore, you must specify a value for the Maximum vector size property of the Receive function that can forward the required number of data bytes.

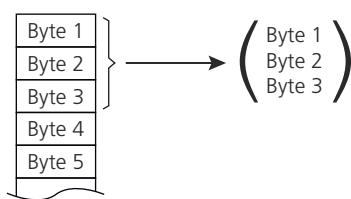
The data vector size has the following effects on forwarding data bytes to the Data Vector function port.

- Number of data bytes within the TCP receive buffer = data vector size:
TCP receive buffer Data Vector



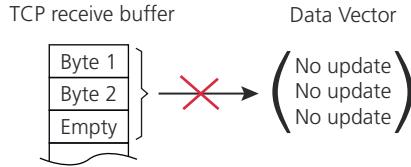
Effect: All data bytes are forwarded to the Data Vector function port.

- Number of data bytes within the TCP receive buffer > data vector size:
TCP receive buffer Data Vector



Effect: Excess data bytes remain in the TCP receive buffer and only the first data bytes are forwarded to the Data Vector function port.

- Number of data bytes within the TCP receive buffer < Data vector size:

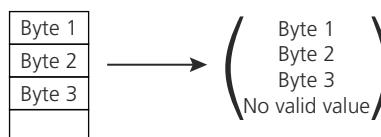


Effect: The data bytes remain in the TCP receive buffer and no data bytes are forwarded to the Data Vector function port.

Specifying the number of data bytes forwarded to the Data Vector function port

You can enable vector size modes that let you reduce the number of data bytes that are forwarded from the TCP receive buffer to the Data Vector (Receive) function port. This lets you read data bytes from the TCP buffer even if the number of data bytes in the TCP receive buffer is less than the vector size of the Data Vector function port.

The following table shows the different modes that you can select.

Mode	Forwarded Data Bytes	Resulting Data Vector
Defined by model	You define the number of data bytes to be forwarded from the TCP receive buffer with the Vector Size function port. The Available Bytes function port provides the number of data bytes that can be forwarded.	TCP receive buffer → Data Vector  <p>The number of forwarded data bytes is provided by the Received Bytes function port.</p>
All available	All available data bytes of the TCP receive buffer are forwarded if the number of data bytes does not exceed the setting of the Maximum Vector Size property.	

Note

A task overrun can occur if data vectors with many data bytes are read from the TCP receive buffer.

If a task overrun occurs, take one of the following measures:

- Decrease the setting of the Maximum vector size property.
- Increase the model step size.
- In the Executable Application table, increase the setting of the Number of Accepted Overruns property.
- Optimize the Ethernet data exchange. Refer to [Basics on Implementing Ethernet Communication in ConfigurationDesk](#) on page 1166.

Sending keep-alive probes

With keep-alive probes you can prevent a disconnection due to network inactivity or check whether a TCP connection is still up and running. Keep-alive probes are acknowledge messages without data.

Checking the TCP connection A TCP connection can be interrupted, for example, if the connected external Ethernet device is suddenly switched off. If the TCP function block only waits for incoming messages, it does not detect the interrupted connection. This means, if the external Ethernet device is available again, a new TCP connection cannot be established, because the TCP function block holds on the previous connection.

If keep-alive probes are enabled, the TCP function block sends an keep alive probe if the communication is idle for longer than specified. If the external Ethernet device does not acknowledge the keep alive probe, the TCP function block detects an interrupted connection and changes the connection state:

- A function block that runs as a server starts to listen for a new connection request.
- A function block that runs as a client changes to a disconnected state. The function block can establish a new connection if the behavior model requests a new one.

For more information on the Connection State port, refer to [Overview of Ports and Properties \(TCP\)](#) on page 1213.

Preventing disconnection The connection tracking functionality of the network can disconnect a TCP connection if it is not used for a longer time. For example, firewalls and proxy server can discard obsolete connections to keep track of only the current connections that pass through them. If a TCP connection is unused for a longer time, it might be detected as an obsolete connection.

If keep-alive probes are enabled, the TCP function block sends an keep alive probe if the communication is idle for longer than specified. These probes prevents the tracking functionality of the network from detecting the TCP connection as an obsolete connection.

Setting the idle time The maximum idle time depends on the use case, e.g., how fast a switched-off Ethernet device must be detected. Reducing the idle time increases the bus load, but the TCP function block detects disconnections earlier and the reduction makes a network disconnection less likely.

Configuring Standard Features (TCP)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The TCP function block does not provide an electrical interface.
-----------------------------	------------------------------------------------------------------

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

PTP Master

Where to go from here

Information in this section

Introduction (PTP Master).....	1226
Overviews (PTP Master).....	1229
Configuration (PTP Master).....	1230

Introduction (PTP Master)

Where to go from here

Information in this section

Introduction to the Function Block (PTP Master).....	1226
Basics on Using the PTP Master Function Block.....	1227

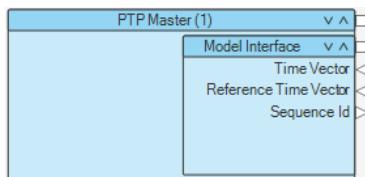
Introduction to the Function Block (PTP Master)

Function block purpose

The PTP Master function block provides a master time to which Ethernet devices can synchronize their clocks via a precision time protocol (PTP).

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Propagating a master time via synchronization frames over Ethernet.
- Supporting the IEEE 802.1AS and AUTOSAR protocols for time synchronization via Ethernet.

Supported hardware

The PTP Master function block supports the following hardware:

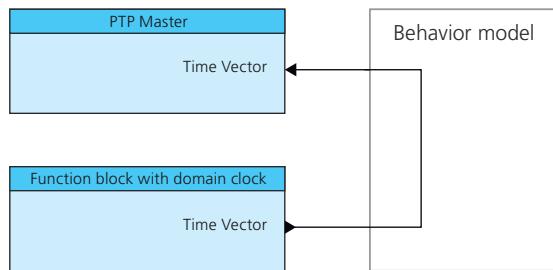
	SCALEXIO	MicroAutoBox III
Board	<ul style="list-style-type: none"> ▪ SCALEXIO Processing Unit as of Real-Time PC version 2.0¹⁾ and product line HPP¹⁾. ▪ DS6331-PE Ethernet Board ▪ DS6333-CS Automotive Ethernet Board ▪ DS6333-PE Automotive Ethernet Board ▪ DS6334-PE Ethernet Board ▪ DS6335-CS Ethernet Board 	<ul style="list-style-type: none"> ▪ DS1403 Processor Board ▪ DS1521 Bus Board

¹⁾ For identifying the version of the SCALEXIO Real-Time PC, refer to [Variants of the SCALEXIO Processing Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

Basics on Using the PTP Master Function Block

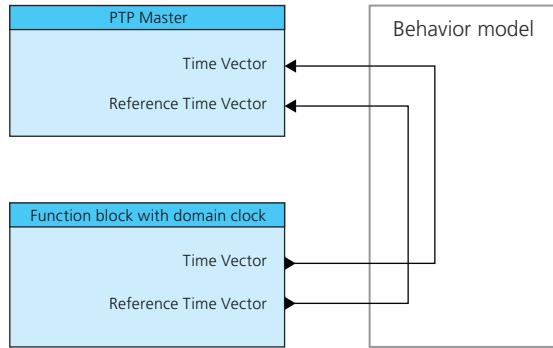
Basic principle

The PTP Master function block sends synchronization frames that are used by the PTP slaves to synchronize their clocks to the provided time values. The PTP Master function block gets the time values from a domain clock provided by another function block. The following illustration shows the basic principle.



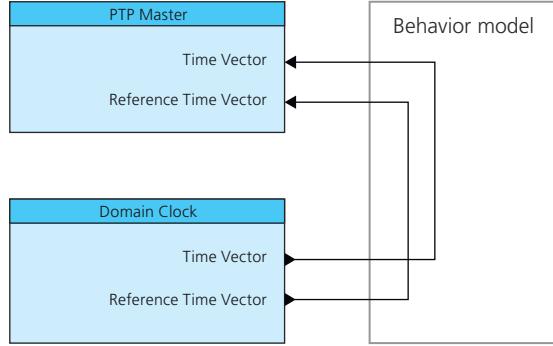
The sending of the time values through the behavior model takes time that results in a propagation of synchronization frames with out-of-date time values.

To compensate for the delay, the time sources provide reference time values. A reference time value represents the point in time when a time value was captured. With the reference time value, the PTP Master function block can calculate the delay to transmit time values to the PTP Master function block and compensates the delay. The following illustration shows the connection with reference time values.



Propagating the local domain clock

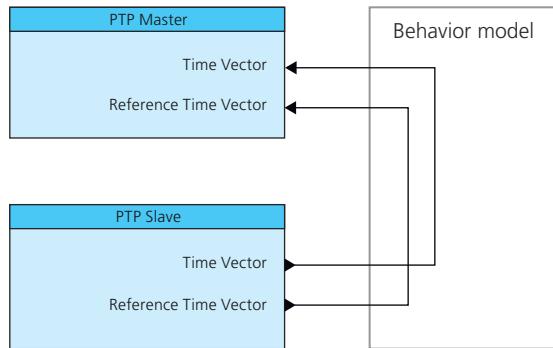
The Domain Clock function block provides a local domain clock. The following illustration shows the required connection.



For more information on the Domain Clock function block, refer to [Domain Clock](#) on page 1463.

Implementing a time gateway

A time gateway propagate the domain clock of one network to another network. The following illustration shows the required connection to implement the time gateway from one Ethernet network to another Ethernet network.



For more information on the PTP Slave function block, refer to [PTP Slave](#) on page 1234.

Presentation of time values

ConfigurationDesk uses a time vector to represent a time value with high accuracy. Two entries represent one time value:

- The first entry represents the seconds.
- The second entry represents the nanoseconds.

The following example shows a time vector and the represented time value.

$$\left(\begin{array}{c} 1\,234\,\text{s} \\ 567\,000\,000\,\text{ns} \end{array} \right) \rightarrow 1,234\,\text{s} + 567,000,000\,\text{ns} = 1,234.567\,\text{s}$$

Value range of the nanosecond value If you provide a time vector, you have to make sure that only time values <1 s are provided by the second entry of the time vector.

The function block uses only the last 9 digits of the nanosecond value and discards higher order digits as shown in the following illustration.

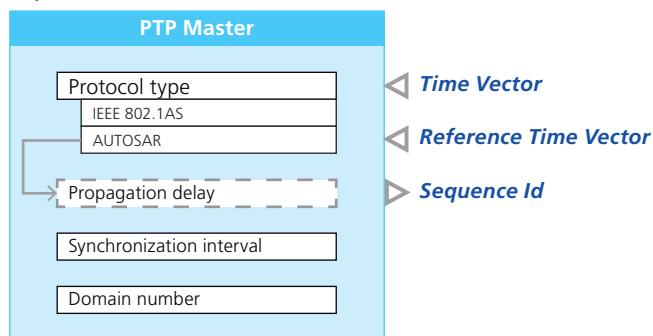
$$\left(\begin{array}{c} 5\,\text{s} \\ 1\,200\,000\,000\,\text{ns} \end{array} \right) \rightarrow \left(\begin{array}{c} 5\,\text{s} \\ \textcolor{red}{+}200\,000\,000\,\text{ns} \end{array} \right) \rightarrow 5.2\,\text{s}$$

Oversviews (PTP Master)

Overview of Ports and Basic Properties (PTP Master)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Time Vector**

This function import reads the time value to be transmitted with the synchronization frame via Ethernet.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 2^{48} for each entry of the vector. ▪ The vector size is 2. Both entries represent one time value: <ul style="list-style-type: none"> ▪ The first entry represents the seconds.
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> The second entry represents the nanoseconds. <p>Only the last 9 digits of the nanosecond value are used.</p>
Dependencies	–

Reference Time Vector

This function import reads the reference time value from the behavior model. The reference time value represents the point in time when the time value at the Time Vector function port was captured. The reference time value is used by the function block to compensate for time delays during the transmission of the time signal through the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... 2^{48} for each entry of the vector. The vector size is 2. Both entries represent one time value: <ul style="list-style-type: none"> The first entry represents the seconds. The second entry represents the nanoseconds. <p>Only the last 9 digits of the nanosecond value are used.</p>
Dependencies	–

Sequence Id

This function outport writes the sequence ID of the last transmitted synchronization frame to the behavior model. The sequence ID is a rolling counter and part of the synchronization frame. The function block increments the sequence ID with each sent synchronization frame. You can use the ID to check the transmission of the frames.

Value range	0 ... 65535
Dependencies	–

Tunable properties

The PTP Master function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuration (PTP Master)

Where to go from here**Information in this section**

- | | |
|-------------------------------------------------------|------|
| Configuring the Basic Functionality (PTP Master)..... | 1231 |
| Configuring Standard Features (PTP Master)..... | 1232 |

Configuring the Basic Functionality (PTP Master)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Selecting the protocol type for synchronization
- Specifying the propagation of the synchronization frames
- Referencing the Ethernet network

Selecting the protocol for synchronization

You can select one of the following protocol types for time synchronization:

- IEEE 802.1AS:
Standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks
- AUTOSAR:
Specification of time synchronization over Ethernet

Selecting the synchronization interval

You can select the time interval the PTP Master function block sends a synchronization frame. A shorter interval increases the precision of the synchronized clocks but increases the bus load.

Setting the domain number

A time domain is an interacting set of clocks that are synchronized to one another. Different time domains can be required for evaluating the sensor values, for example.

The domain number lets you specify the time domain for which the synchronization frames are valid. The following table shows the possible settings for the different protocols.

Protocol	Possible Setting
AUTOSAR	Up to 16 time domains within the same Ethernet network. The domain number must be in the range 0 ... 15. The dSPACE real-time hardware supports only one time domain per physical Ethernet controller. To use different time domains, you have to use different physical Ethernet controllers.
IEEE 802.1AS	One time domain, because this protocol does not support different time domains. The domain number must be set to 0.

Setting the propagation delay

The propagation of the synchronization frames over Ethernet takes time. The connected PTP slaves can compensate for this delay if the duration is known. Therefore, both protocols include a mechanism to measure the delay and to compensate for it.

Using the AUTOSAR protocol The Ethernet network topology of an automotive application is known and the propagation delay does not change. Therefore, the AUTOSAR protocol lets you set a fixed value for the propagation delay as an alternative to the path delay measurement. A fixed value for the propagation delay reduces the bus load.

The following table shows the possible settings.

Propagation Delay Setting	Description
>0 s	The PTP slaves use the set value for delay compensation.
0 s	As soon as a measured value for the propagation delay is available, the function block replaces 0 s with the measured value.

Using the IEEE 802.1AS protocol If you use the IEEE 802.1AS protocol, the propagation delay is always measured and automatically compensated.

Referencing the Ethernet network

You reference an Ethernet network by assigning an Ethernet setup function block (Ethernet Setup or Virtual Ethernet Setup function block). The Ethernet setup function block provides the local IP address and the access to a configured Ethernet controller.

If several function blocks reference the same Ethernet setup function block, you must ensure that all function blocks belong to only one application process. This means, all their mapped model port blocks must be elements of the same behavior model. Otherwise, a conflict is generated and displayed in the Conflicts Viewer.

Note

Time synchronization in a VLAN

- VLAN is supported only by the AUTOSAR protocol. Ethernet devices that use the IEEE 802.1AS protocol for time synchronization ignore the VLAN ID.
- PTP frames with a VLAN tag are not supported by the 10-Gbit Ethernet controller (Eth0_3) of the SCALEXIO Real-Time PC Version HPP 2.0.

Configuring Standard Features (PTP Master)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

This function block does not provide an electrical interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

PTP Slave

Where to go from here

Information in this section

Introduction (PTP Slave).....	1234
Overviews (PTP Slave).....	1236
Configuration (PTP Slave).....	1238

Introduction (PTP Slave)

Where to go from here

Information in this section

Introduction to the Function Block (PTP Slave).....	1234
Basics on Time Values.....	1235

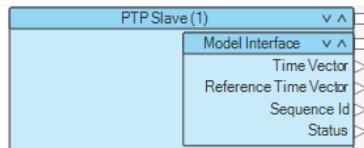
Introduction to the Function Block (PTP Slave)

Function block purpose

The PTP Slave function block provides time values to the behavior model that are synchronized via a precision time protocol (PTP) to an Ethernet time master.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Providing time values that are synchronous to an Ethernet time master.
- Providing reference time values to compensate for delays.

- Supporting the IEEE 802.1AS and AUTOSAR protocols for time synchronization.

Supported hardware

The PTP Slave function block type supports the following hardware:

	SCALEXIO	MicroAutoBox III
Board	<ul style="list-style-type: none"> ▪ SCALEXIO Processing Unit as of Real-Time PC version 2.0¹⁾ and product line HPP¹⁾. ▪ DS6331-PE Ethernet Board ▪ DS6333-CS Automotive Ethernet Board ▪ DS6333-PE Automotive Ethernet Board ▪ DS6334-PE Ethernet Board ▪ DS6335-CS Ethernet Board 	<ul style="list-style-type: none"> ▪ DS1403 Processor Board ▪ DS1521 Bus Board

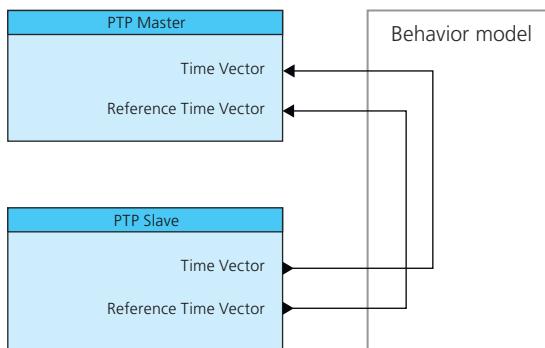
¹⁾ For identifying the version of the SCALEXIO Real-Time PC, refer to [Variants of the SCALEXIO Processing Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

Basics on Time Values

Use of reference time values

A reference time value represents the point in time when a time value is captured. The reference time is intended to compensate for time delays caused by the processing of the signal.

The following example shows a time gateway. The PTP Slave function block is used as the time source for a PTP Master function block.



With the reference time values, the PTP Master function block can calculate the time it takes to transmit the time values through the behavior model.

Evaluating measuring data of different Ethernet devices

If measurements are captured with timestamps derived from synchronized clocks, you can evaluate the values of different devices with time accuracy.

Function blocks that provide timestamps, such as the Voltage In function block, use the same base time as the PTP Slave function block.

By using the time difference between the reference time and the synchronized time you can migrate the timestamps from the function blocks to the synchronized time values. This lets you evaluate measurement data that is captured at the same time from different devices of the same time domain.

The following table gives you an example:

Voltage In Function Block	PTP Slave Function Block		Measured Voltages with Synchronized Timestamps
Measured Voltages with Timestamps	Reference Time Port	Time Port	
3.0 V @ 10 s	10 s	5.010 s	3.0 V @ 5.010 s
3.5 V @ 11 s	11 s	6.011 s	3.5 V @ 6.011 s
3.3 V @ 12 s	12 s	7.012 s	3.3 V @ 7.012 s

Presentation of time values

ConfigurationDesk uses a time vector to represent a time value with high accuracy. Two entries represent one time value:

- The first entry represents the seconds.
- The second entry represents the nanoseconds.

The following example shows a time vector and the represented time value.

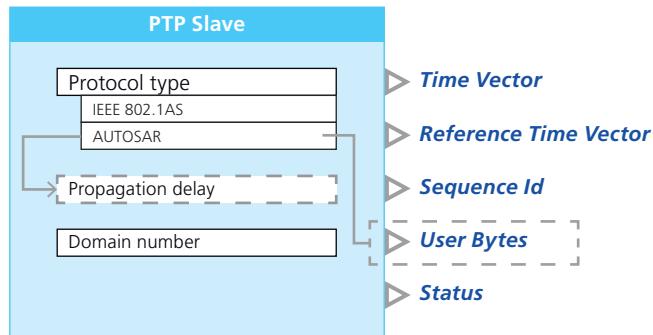
$$\begin{pmatrix} 1\,234\,\text{s} \\ 567\,000\,000\,\text{ns} \end{pmatrix} \rightarrow 1,234\,\text{s} + 567,000,000\,\text{ns} = 1,234.567\,\text{s}$$

Oversviews (PTP Slave)

Overview of Ports and Basic Properties (PTP Slave)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

**Time Vector**

This function outport writes the current time value to the behavior model.

Value range	<ul style="list-style-type: none"> 0 ... 2^{48} for each entry of the vector. The vector size is 2. Both entries represent one time value: <ul style="list-style-type: none"> The first entry represents the seconds. The second entry represents the nanoseconds. Only the last 9 digits of the nanosecond value are used.
Dependencies	–

Reference Time Vector

This function outport writes the reference time value to the behavior model. A reference time value represents the point in time when the time value of the Time Vector function port was captured. The reference time value can be used for compensating time delays when processing the signals.

Value range	<ul style="list-style-type: none"> 0 ... 2^{48} for each entry of the vector. The vector size is 2. Both entries represent one time value: <ul style="list-style-type: none"> The first entry represents the seconds. The second entry represents the nanoseconds. Only the last 9 digits of the nanosecond value are used.
Dependencies	–

Sequence Id

This function outport writes the sequence ID of the last received synchronization frame to the behavior model. The sequence ID is a rolling counter that lets you check whether new time synchronization frames are received. The time master increments the sequence ID with each sent synchronization frame.

Value range	0 ... 65535
Dependencies	–

User Bytes

This function outport writes the user bytes of the last received synchronization frame to the behavior model. User bytes are supported only by the AUTOSAR protocol and provide a customizable content.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry ▪ The vector size is 3.
Dependencies	Available only if the Protocol type property is set to AUTOSAR.

Status This function outport indicates whether the value of the Time Vector function port is valid.

Value range	<ul style="list-style-type: none"> ▪ 0: The provided time value is not valid. ▪ 1: The provided time value is valid.
Dependencies	–

Tunable properties The PTP Slave function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuration (PTP Slave)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------------------|------|
| Configuring the Basic Functionality (PTP Slave) | 1238 |
| Configuring Standard Features (PTP Slave) | 1240 |

Configuring the Basic Functionality (PTP Slave)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Selecting the protocol for synchronization
- Specifying the synchronization frame reception
- Referencing the Ethernet network

Selecting the protocol for synchronization

You can select one of the following protocol types for time synchronization:

- IEEE 802.1AS:
Standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks
- AUTOSAR:
Specification of time synchronization over Ethernet

Setting the domain number

A time domain is an interacting set of clocks that are synchronized to one another. Different time domains can be required for evaluating the sensor values, for example.

The domain number lets you specify whether the received synchronization frames are valid for the function block. The following table shows the possible settings for the different protocols.

Protocol	Possible Setting
AUTOSAR	Up to 16 time domains within the same Ethernet network. The domain number must be in the range 0 ... 15. The dSPACE real-time hardware supports only one time domain per physical Ethernet controller. To use different time domains, you have to use different physical Ethernet controllers.
IEEE 802.1AS	One time domain, because this protocol does not support different time domains. The domain number must be set to 0.

Setting the propagation delay

The propagation of the synchronization frames over Ethernet takes time. PTP slaves can compensate this delay if the duration is known. Therefore, both protocols include a mechanism to measure the delay and to compensate it.

Using the AUTOSAR protocol The Ethernet network topology of an automotive application is known and the propagation delay does not change. Therefore, the AUTOSAR protocol lets you set a fixed value for the propagation delay as an alternative to the path delay measurement. A fixed value for the propagation delay reduces the bus load.

The following table shows the possible settings.

Propagation Delay Setting	Description
>0 s	The function block uses the set value for delay compensation.
0 s	As soon as a value for the propagation delay is provided by the PTP master, the function block replaces 0 s with the provided value.

Using the IEEE 802.1AS protocol If you use the IEEE 802.1AS protocol, the propagation delay is always measured and automatically compensated.

Referencing the Ethernet network

You reference an Ethernet network by assigning an Ethernet setup function block (Ethernet Setup or Virtual Ethernet Setup function block). The Ethernet setup function block provides the local IP address and the access to a configured Ethernet controller.

If several function blocks reference the same Ethernet setup function block, you must ensure that all function blocks belong to only one application process. This means, all their mapped model port blocks must be elements of the same behavior model. Otherwise, a conflict is generated and displayed in the **Conflicts Viewer**.

Configuring Standard Features (PTP Slave)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

This function block does not provide an electrical interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Ethernet Switch

Where to go from here

Information in this section

Introduction (Ethernet Switch).....	1241
Overviews (Ethernet Switch).....	1245
Configuring the Function Block (Ethernet Switch).....	1247

Introduction (Ethernet Switch)

Where to go from here

Information in this section

Introduction to the Function Block (Ethernet Switch).....	1241
Basics on Ethernet Switches of the dSPACE Hardware.....	1242

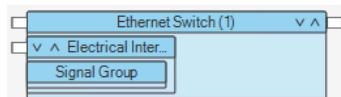
Introduction to the Function Block (Ethernet Switch)

Function block purpose

The Ethernet Switch function block lets you configure the switching of Ethernet traffic and the characteristics of the physical layer transceivers (PHYs).

Default display

The function block provides neither signal ports nor function ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Configuring the switching of Ethernet traffic based on the ports and/or the VLAN IDs of the Ethernet frames.
- Configuring the physical layer transceivers (PHYs) of the I/O Ethernet ports, for example, the link speed.
- Enabling and specifying the monitoring of Ethernet traffic.

Supported channel types

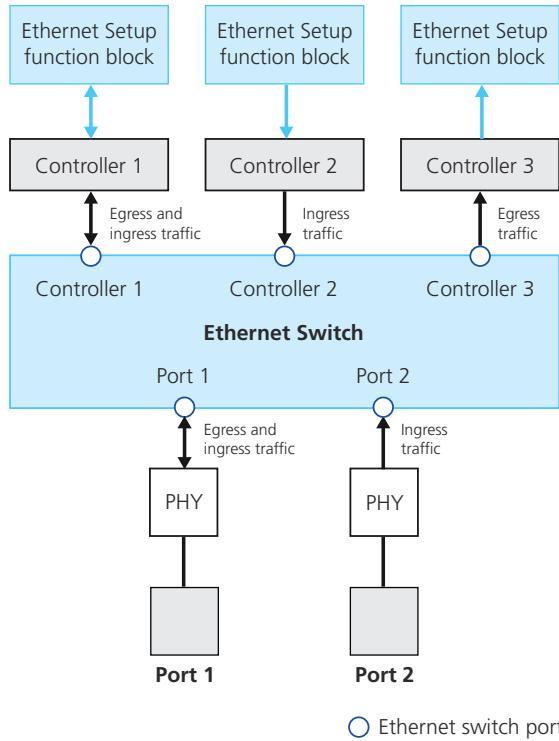
The Ethernet Switch function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	Ethernet Switch 1	Ethernet Switch 2
Board	<ul style="list-style-type: none"> ▪ DS6333 ▪ DS6335 	DS1403

Basics on Ethernet Switches of the dSPACE Hardware

Basic components and terms

An Ethernet switch switches Ethernet traffic between the controllers and the ports. The Ethernet switch is part of the data link layer. The following illustration shows basic terms of an Ethernet switch and the connected components.



The following table shows the description of the basic components and terms.

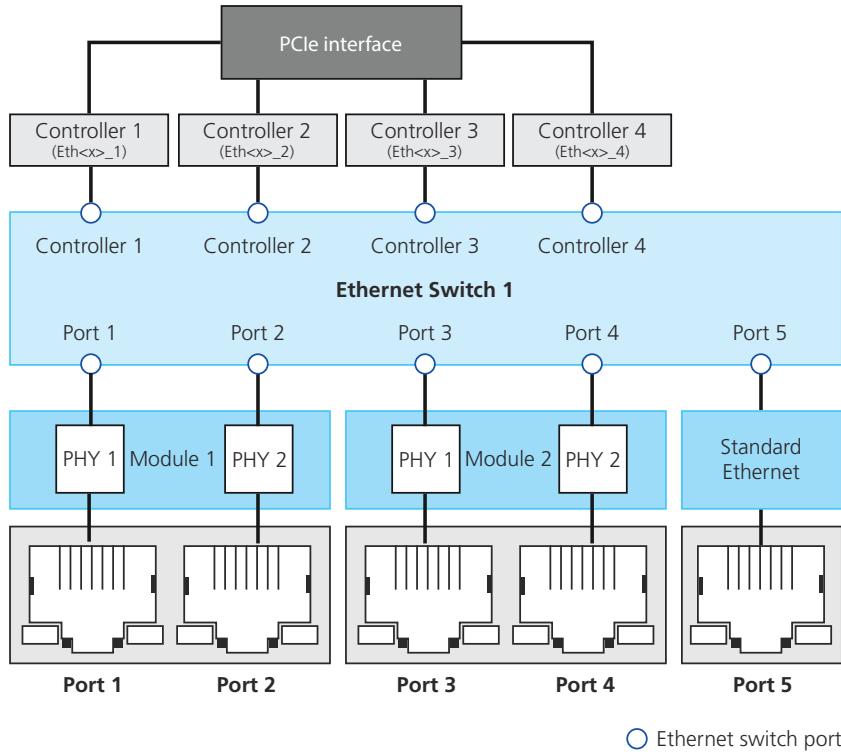
Component/Term	Description
Ethernet switch port	Electrical interface of the Ethernet switch. The name of the Ethernet switch port is usually the name of the component to which the port is connected. For example: The Controller 1 Ethernet switch port is connected to the Controller 1.

Component/Term	Description
Egress traffic	Outgoing Ethernet frames that are sent by an Ethernet switch port.
Ingress traffic	Incoming Ethernet frames that are received from an Ethernet switch port.
Controller	Controls the access to the Ethernet network and is part of the data link layer.
PHY	Provides the electrical interface to the Ethernet network. The PHY is part of the physical layer.
Port	Mechanical interface to the Ethernet network, for example, an RJ45 connector. The port is part of the physical layer.

Available Ethernet switches

The Ethernet switches of the dSPACE real-time hardware are grouped into different channel types depending on the number of Ethernet switch ports and the connected Ethernet components.

Ethernet Switch 1 The following illustration shows the Ethernet Switch 1 with the naming of the Ethernet switch ports and the components that are connected to the ports.



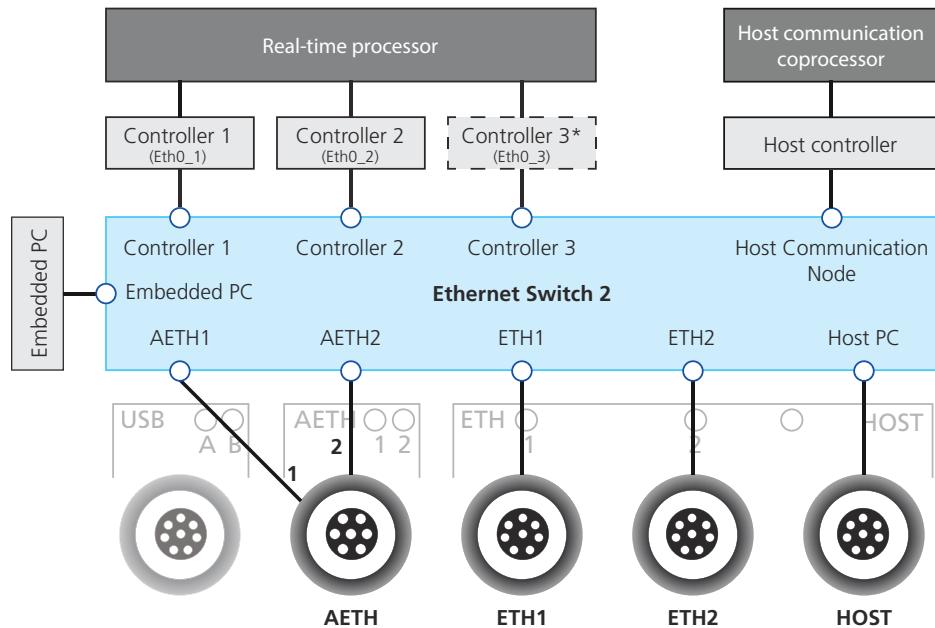
○ Ethernet switch port

The controller names are configurable via the Platform Manager. By default, the controllers are named as follows: Eth<slot number>_<controller number>.

For example: Eth1_3 is controller 3 of the Ethernet board that is installed to PCIe slot 1.

The naming of the controllers is independent from the used Ethernet board. For more information on the naming, refer to [DS6333-CS Automotive Ethernet Board Switch Properties \(ConfigurationDesk Hardware Resource Properties\)](#).

Ethernet Switch 2 The following illustration shows the Ethernet Switch 2 with the naming of the Ethernet switch ports and the components that are connected to the ports.



* Controller for fast bypassing. Enabled via web interface.

○ Ethernet switch port

The controller names are configurable via the Platform Manager. By default, the controllers of the DS1403 Processor Board are named as follows: Eth0_<controller number>.

Note

- The WLAN interface of a MicroAutoBox III (WLAN) is permanently switched to the host controller and cannot be configured.
- Controller 3 can be activated via the web interface of the MicroAutoBox III. This controller is latency optimized and recommended only for fast bypassing. For instructions on activating, refer to [How to Improve ECU Interfacing \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Methods to configure the switching of Ethernet traffic

You can configure the switching of Ethernet traffic independent from the physical connection to support virtual Ethernet networks (VLANs).

The Ethernet switch provides two methods to configure the switching of Ethernet traffic: The port-based VLAN configuration and the protocol-based VLAN configuration. Both methods can be used alone or in combination.

For more information on the port-based VLAN configuration, refer to [Configuring the Basic Functionality \(Ethernet Switch\) on page 1248](#).

For more information on the protocol-based VLAN configuration, refer to [Configuring VLANs Based on the Protocol \(Ethernet Switch\) on page 1251](#).

Tip

You can select predefined Ethernet switch configurations for specific use cases. The selected configuration takes effect if the real-time application does not provide an Ethernet switch configuration.

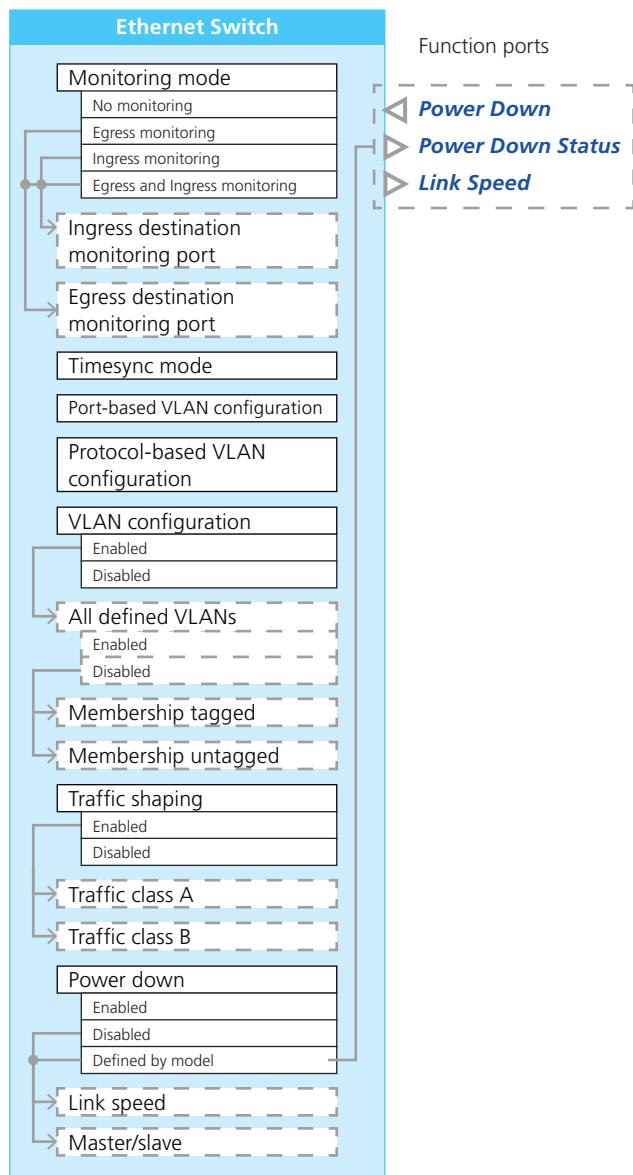
For more information, refer to [Configuring the I/O Ethernet Communication \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(717c9423604234786bbb49d8a97800d2_img.jpg\)](#)).

Oversviews (Ethernet Switch)

Overview of Ports and Basic Properties (Ethernet Switch)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Power Down

This function import powers down the PHY of the selected Ethernet switch port.

Value range	<ul style="list-style-type: none"> 0: Disabled The PHY of the Ethernet switch port is powered and can be used. 1: Enabled The PHY is not powered. The Ethernet switch port cannot communicate with an external Ethernet device.
Dependencies	Available only if the Power down property is set to Defined by model.

Power Down Status

This function outport indicates whether the PHY of the selected Ethernet switch port is currently powered down.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled The PHY of the Ethernet switch port is powered and can be used. ▪ 1: Enabled The PHY is not powered. The Ethernet switch port cannot communicate with an external Ethernet device.
Dependencies	Available only if the Power down property is set to Defined by model.

Link Speed

This function outport writes the current link speed of the PHY to the behavior model.

Value range	<ul style="list-style-type: none"> 0 Mbit/s ... 10,000 Mbit/s: ▪ 0 Mbit/s: No Ethernet connection. ▪ >0 Mbit/s: A data link with the provided link speed is established.
Dependencies	Available only if the Power down property is set to Defined by model.

Tunable properties

The Ethernet Switch function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (Ethernet Switch)

Where to go from here**Information in this section**

- | | |
|----------------------------------------------------------------|------|
| Configuring the Basic Functionality (Ethernet Switch)..... | 1248 |
| Configuring VLANs Based on the Protocol (Ethernet Switch)..... | 1251 |

Configuring the Basic Functionality (Ethernet Switch)

Overview

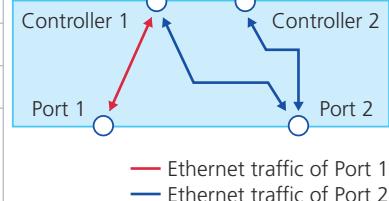
The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Configuring the switching of Ethernet traffic to support virtual Ethernet networks (VLANs):
 - Configuring VLANs based on the Ethernet switch ports.
 - Configuring VLANs based on the protocol. Refer to [Configuring VLANs Based on the Protocol \(Ethernet Switch\)](#) on page 1251.
- Configuring the PHYs of I/O Ethernet ports.
- Specifying the monitoring of Ethernet traffic.
- Specifying traffic shaping.
- Specifying the timesync mode.

Configuring VLANs based on the Ethernet switch ports

A port-based VLAN configuration defines the switching of Ethernet traffic by referencing Ethernet switch ports. Ethernet frames are forwarded only to Ethernet switch ports that are referenced.

The following example shows a simplified Ethernet switch with a port-based VLAN configuration.

Ethernet switch port	Referenced Ports	Resulting Ethernet Traffic
Controller 1	Port 1 and Port 2	
Controller 2	Port 2	
Port 1	Controller 1	
Port 2	Controller 1 and Controller 2	 — Ethernet traffic of Port 1 — Ethernet traffic of Port 2

Ethernet switch ports are always referenced to each other: The referenced Ethernet switch port automatically references the referencing port. You cannot differentiate between egress and ingress Ethernet traffic.

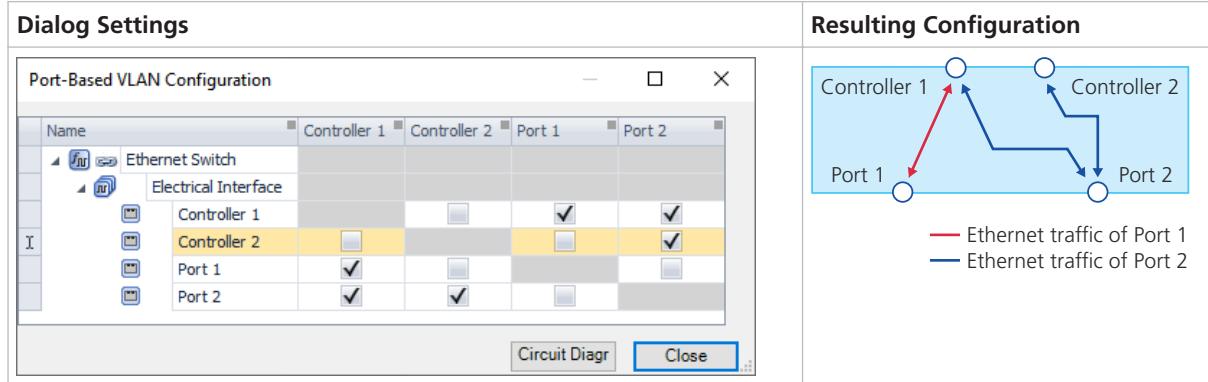
Before you can specify the port-based VLAN configuration, you must assign an Ethernet switch to the Ethernet switch function block. In ConfigurationDesk, you can specify the referenced ports for each Ethernet switch port via the [Port-Based VLAN Configuration](#) dialog.

Opening the Port-Based VLAN Configuration dialog Use one of the following methods to open the Port-Based VLAN Configuration dialog:

- Click  of the Port-based VLAN configuration property.
- Select [Configure VLAN - Port-based VLAN](#) from the context menu of the Ethernet Switch function block.

Example configuration The following illustration shows a simplified Ethernet switch with an example configuration.

The Port-Based VLAN Configuration dialog provides the Ethernet switch ports of the Ethernet switch that is assigned to the function block. For each Ethernet switch port you can select the Ethernet switch ports that are referenced to each other via checkboxes.



Tip

Click Circuit Diagr button in the dialog or select Show – Open Equivalent Circuit Diagram from the context menu of the function block to open the equivalent circuit diagram of the Ethernet switch that is assigned to the function block. The diagram shows the Ethernet switch ports and the components that are connected to the ports.

Configuring the PHYs of I/O Ethernet ports

A PHY provides the electrical interface to the Ethernet network. All PHYs of the mechanical ports to connect I/O Ethernet devices are configurable. You can configure the following characteristics:

- Powering

You can specify that a PHY is powered down permanently, or the behavior model can power down a PHY at run time, for example, for failure simulation.

- Link speed

You can specify the data rate. For standard Ethernet, you can also specify the transfer method (half-duplex or full-duplex) or you activate autonegotiation. To read the speed of the established data link at runtime, set the Power down property to Defined by model.

If you use traffic shaping, ensure that the sum of the specified bandwidths for traffic class A and B does not exceed the specified link speed.

- Master/slave (only automotive Ethernet)

You can set the automotive Ethernet port to master or slave. The PHY of the master port starts the training process to establish a link to the slave port.

Specifying the monitoring of Ethernet traffic

Ethernet traffic of each Ethernet switch port can be monitored. The following monitoring modes are selectable:

- No monitoring:
Deactivates the monitoring of Ethernet traffic.
- Egress monitoring:
Only Ethernet frames that are forwarded by an Ethernet switch port can be monitored.
- Ingress monitoring:
Only Ethernet frames that are received from an Ethernet switch port can be monitored.
- Egress and Ingress monitoring:
The complete Ethernet traffic can be monitored.

To output the monitored Ethernet traffic, you have to specify an Ethernet switch port as destination port: One destination port for egress monitoring, one destination port for ingress monitoring. However, you can use the same destination port for egress and ingress monitoring. A destination port itself cannot be monitored.

Note

A high data throughput at the Ethernet switch ports can lead to data loss on the monitoring channel. However, the data flow through the monitored Ethernet switch ports is not affected.

For example: The monitoring of the egress and ingress traffic in full duplex mode via the same destination port can exceed the maximum data rate of 1 GBit/s of the destination port.

Specifying traffic shaping

Traffic shaping can optimize or guarantee performance for forwarding Ethernet frames of a particular traffic class by delaying other Ethernet frames. The traffic class of an Ethernet frame depends on the PCP value of the VLAN tag. PCP is a 3-bit value that indicates the priority for the transmit control.

Traffic Class	Description
A	Ethernet frames with the PCP value 7.
B	Ethernet frames with the PCP value 6.
Best effort	Ethernet frames with the PCP priority other than 6 or 7 and all untagged Ethernet frames.

For the traffic classes A and B you can reserve bandwidth. Ethernet frames of the best effort class always use the bandwidth that is not used by the traffic classes A and B.

The sum of the traffic class A bandwidth and traffic class B bandwidth must not exceed the following data rates:

- The maximum data rate of the Ethernet switch port of 1000 Mbit/s.
- The specified value of the Link speed property if the selected Ethernet switch port is connected to an I/O port.

Specifying the timesync mode

The timesync mode according to IEEE 802.1AS is used to synchronize clocks throughout the Ethernet network. Precision time protocol (PTP) frames are used to synchronize the clocks.

You can specify to forward the PTP frames, or you can block all PTP frames.

The MicroAutoBox III and SCALEXIO with a Linux®-based operating system (introduced with firmware 5.0, dSPACE Release 2020-B) do not support the suppression of IGMP messages.

Configuring VLANs Based on the Protocol (Ethernet Switch)

Basics on the protocol-based VLAN configuration

A protocol-based VLAN configuration defines the switching of Ethernet traffic based on the VLAN ID of the Ethernet frames. The VLAN ID is a part of the VLAN tag. However, a VLAN tag is optional for Ethernet frames.

ConfigurationDesk provides two membership types for each Ethernet switch port:

- Membership tagged for Ethernet frames with a VLAN tag (tagged Ethernet frame).
- Membership untagged for Ethernet frames without a VLAN tag (untagged Ethernet frame).

Before you can specify the protocol-based VLAN configuration, you must assign an Ethernet switch to the Ethernet Switch function block.

In ConfigurationDesk, you can specify the membership for each Ethernet switch port via the properties or via the Protocol-Based VLAN Configuration dialog.

Opening the Protocol-Based VLAN Configuration dialog

Use one of the following methods to open the Protocol-Based VLAN Configuration dialog:

- Click  of the Protocol-based VLAN configuration property.
- Select Configure VLAN - Protocol-based VLAN from the context menu of the Ethernet Switch function block.

Defining the VLANs

You have to define the IDs of the VLANs that can be used to specify the membership of an Ethernet switch port:

1. Open the Protocol-Based VLAN Configuration dialog.
2. Add the desired number of VLANs by repeatedly clicking Add VLAN.
3. To change the VLAN IDs, close the Protocol-Based VLAN Configuration dialog.
4. In the Properties Browser, change the VLAN IDs to the desired IDs.

Configuring the forwarding of tagged Ethernet frames

Tagged Ethernet frames are Ethernet frames with a VLAN tag. By adding defined VLANs to the membership tagged, an Ethernet frame is forwarded only if the VLAN ID of the Ethernet frame matches the membership of the Ethernet switch port.

The following example shows a simplified Ethernet switch with a protocol-based VLAN configuration for tagged Ethernet frames.

Ethernet Switch Port	Membership Tagged of the Port	Resulting Ethernet Traffic
Controller 1	VLAN 1	
Controller 2	VLAN 2	
Port 1	VLAN 1 and VLAN 3	
Port 2	VLAN 2 and VLAN 3	<p>Legend: — VLAN 1 — VLAN 2 — VLAN 3 </p>

Configuring the forwarding of untagged Ethernet frames

Untagged Ethernet frames are Ethernet frames without a VLAN tag. By using the protocol-based VLAN configuration, the selected Ethernet switch port forwards untagged Ethernet frames as follows:

- Blocking all untagged Ethernet frames.
If no VLAN is specified for the Membership untagged property, only Ethernet frames with a VLAN ID of the Membership tagged property are forwarded. All other Ethernet frames are blocked.
- Using a VLAN for untagged Ethernet frames.
If a VLAN ID is specified for the Membership untagged property, the Ethernet switch port handles untagged Ethernet frames as follows:
 - Ingress traffic of Ethernet frames without VLAN ID are forwarded with the specified VLAN ID of the Membership untagged property.
 - Egress traffic of Ethernet frames with the specified VLAN ID of the Membership untagged property are forwarded without a VLAN ID.

Note

You can select only one VLAN ID for the Membership untagged property. The selected VLAN ID cannot be added to the membership tagged.

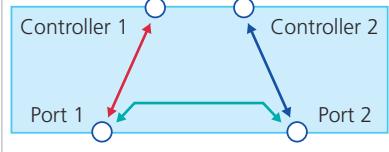
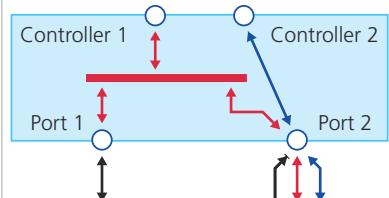
The following example shows a simplified Ethernet switch with a protocol-based VLAN configuration for untagged Ethernet frames.

Ethernet Switch Port	Membership of the Port		Resulting Ethernet Traffic
	Tagged	Untagged	
Controller 1	VLAN 1	-	
Controller 2	VLAN 2	-	
Port 1	-	VLAN 1	
Port 2	VLAN 1 and VLAN 2	-	<p>Legend:</p> <ul style="list-style-type: none"> VLAN 1 (Red) VLAN 2 (Blue) Untagged Ethernet frames (Black) <p>Behavior of Port 1 and Port 2 concerning untagged Ethernet frames:</p> <ul style="list-style-type: none"> Port 1 is configured to use a VLAN for untagged Ethernet frames. Before the Ethernet switch port forwards the Ethernet frames, it changes them: <ul style="list-style-type: none"> Ingress traffic: The port adds the VLAN 1 tag to untagged Ethernet frames. Egress traffic: The port removes the VLAN tag from Ethernet frames of the VLAN 1. Port 2 blocks all untagged Ethernet frames. Only Ethernet frames of VLAN 1 and VLAN 2 are forwarded.

Examples of dialog-based configurations

The following illustrations show the simplified Ethernet switch with example configurations.

The first row of the Protocol-Based VLAN Configuration dialog shows the Ethernet switch ports of the Ethernet switch that is assigned to the function block. For each Ethernet switch port you can select the VLAN membership by selecting Tagged for the membership tagged and Untagged for the membership untagged.

Dialog Setting	Resulting Configuration																									
<p>Protocol-Based VLAN Configuration</p> <table border="1" data-bbox="195 354 909 566"> <thead> <tr> <th>Name</th><th>Controller 1</th><th>Controller 2</th><th>Port 1</th><th>Port 2</th></tr> </thead> <tbody> <tr> <td>Ethernet Switch (1)</td><td></td><td></td><td></td><td></td></tr> <tr> <td>VLAN 1</td><td>Tagged</td><td>-</td><td>Tagged</td><td>-</td></tr> <tr> <td>VLAN 2</td><td>-</td><td>Tagged</td><td>-</td><td>Tagged</td></tr> <tr> <td>VLAN 3</td><td>-</td><td>-</td><td>Tagged</td><td>Tagged</td></tr> </tbody> </table> <p>Add VLAN Circuit Diagr Close</p>	Name	Controller 1	Controller 2	Port 1	Port 2	Ethernet Switch (1)					VLAN 1	Tagged	-	Tagged	-	VLAN 2	-	Tagged	-	Tagged	VLAN 3	-	-	Tagged	Tagged	 <p>Controller 1 Controller 2</p> <p>Port 1 Port 2</p> <ul style="list-style-type: none"> — VLAN 1 — VLAN 2 — VLAN 3
Name	Controller 1	Controller 2	Port 1	Port 2																						
Ethernet Switch (1)																										
VLAN 1	Tagged	-	Tagged	-																						
VLAN 2	-	Tagged	-	Tagged																						
VLAN 3	-	-	Tagged	Tagged																						
<p>Protocol-Based VLAN Configuration</p> <table border="1" data-bbox="195 686 909 897"> <thead> <tr> <th>Name</th><th>Controller 1</th><th>Controller 2</th><th>Port 1</th><th>Port 2</th></tr> </thead> <tbody> <tr> <td>Ethernet Switch (1)</td><td></td><td></td><td></td><td></td></tr> <tr> <td>VLAN 1</td><td>Tagged</td><td>-</td><td>Untagged</td><td>Tagged</td></tr> <tr> <td>VLAN 2</td><td>-</td><td>Tagged</td><td>-</td><td>Tagged</td></tr> <tr> <td>VLAN 3</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> </tbody> </table> <p>Add VLAN Circuit Diagr Close</p>	Name	Controller 1	Controller 2	Port 1	Port 2	Ethernet Switch (1)					VLAN 1	Tagged	-	Untagged	Tagged	VLAN 2	-	Tagged	-	Tagged	VLAN 3	-	-	-	-	 <p>Controller 1 Controller 2</p> <p>Port 1 Port 2</p> <ul style="list-style-type: none"> — VLAN 1 — VLAN 2 — Untagged Ethernet frames
Name	Controller 1	Controller 2	Port 1	Port 2																						
Ethernet Switch (1)																										
VLAN 1	Tagged	-	Untagged	Tagged																						
VLAN 2	-	Tagged	-	Tagged																						
VLAN 3	-	-	-	-																						

Tip

Click Circuit Diagr button in the dialog or select Show – Open Equivalent Circuit Diagram from the context menu of the function block to open the equivalent circuit diagram of the Ethernet switch that is assigned to the function block. The diagram shows the Ethernet switch ports and the components that are connected to the ports.

SENT Communication (SENT In, SENT Out)

Where to go from here

Information in this section

Introduction to SENT Communication.....	1256
SENT In.....	1276
SENT Out.....	1303

Introduction to SENT Communication

Where to go from here

Information in this section

Basics on the SENT Protocol.....	1256
Using Application-Specific Protocols.....	1260
Basics on User-Defined Protocol Files.....	1262
Basics on Multiplexing SENT Messages.....	1263
Creating User-Defined Protocol Files.....	1266

Basics on the SENT Protocol

Definition

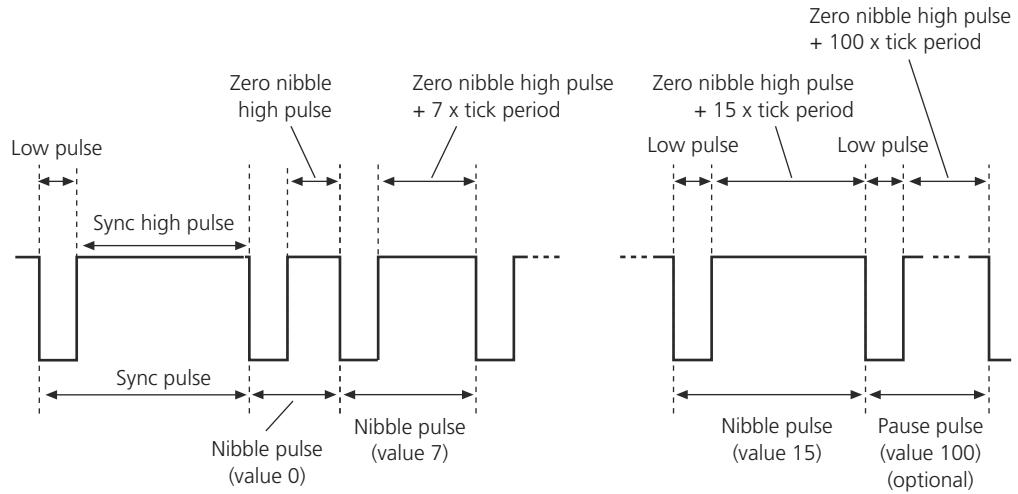
The SENT (Single Edge Nibble Transmission) protocol is defined in the SAE J2716 APR2016 SENT standard by the Society of Automotive Engineers (SAE). It is used to transmit data of high-resolution (10-bit or more) sensors as an alternative to an analog interface. The sensor signal is transmitted as a series of pulses.

Signal of a SENT message

SENT communication is based on the transmission of 4-bit data values, the so-called nibbles. Every SENT message consists of a sync (synchronization) pulse followed by one or more nibble pulses. The nibble values are encoded in the nibble pulse lengths. For regular SENT specification, the first nibble pulse is a status nibble and the last nibble pulse is a CRC nibble. All other nibble pulses contain data information. An optional pause pulse (transmitted at the end of the SENT message) can be used, for example, to create SENT messages with a constant length.

The number of nibble pulses used in every SENT message (including the status and CRC nibbles) can vary between different applications, but it must be constant for a specific SENT communication.

The following illustration shows a SENT message with relevant timing and nibble pulses transferring the nibble values 0, 7, and 15 and a pause pulse with a pause value of 100.



One SENT nibble pulse consists of one low pulse and one high pulse. The nibble value is encoded by the length of the high pulse. The low pulse has a fixed pulse length.

SENT pulse length	= Low pulse length + High pulse length
High pulse length	= Zero nibble high pulse + Nibble value * Tick period

Tick period The lengths of the pulses are based on a clock rate, the tick period. Every SENT pulse is a multiple of this tick period. The standard value is 3 μ s.

At the beginning of each message, the SENT transmitter sends a sync pulse of a defined number of tick periods. The SENT receiver measures the current length of the sync pulse and calculates the current length of the transmitter tick period.

Tick period tolerance A percentage that defines the maximum possible clock drift of a SENT transmitter. The standard value is $\pm 20\%$ if no pause pulse is used and $-20\% \dots +18\%$ if a Pause pulse is used.

Low pulse A pulse with a fixed length that marks the beginning of every SENT pulse. Its standard length is at least 4 tick periods.

Zero nibble high pulse The high part of a SENT nibble pulse that represent the nibble value of 0. Its length is constant and defined by the number of tick periods. The standard value is 7 tick periods.

Nibble value A 4-bit value that is transmitted via a nibble pulse.

Definition of SENT message pulses

A SENT message provides the following SENT pulses.

Sync pulse The beginning of every SENT message. It consists of a low pulse followed by a sync high pulse. The pulse length is measured between two consecutive falling edges and must be longer than the maximum possible

duration of a nibble pulse. The standard pulse length is 56 tick periods, 5 tick periods for the low pulse and 51 tick periods for the high pulse.

Nibble pulse A SENT pulse that encodes a nibble.

Nibble pulse is a generic term for the following pulses that encode a specific nibble:

- Status nibble

First nibble of a SENT message that provide status and data of serial messages. A serial message is a SENT message type, refer to [Message types](#) on page 1259.

- Data nibble

Nibbles to transmit data of fast messages. A fast message is a SENT message type, refer to [Message types](#) on page 1259.

- CRC nibble

Last nibble of a SENT message to provide a checksum.

- Counter nibble

Optional nibble to transmit the value of a rolling counter.

- Frame control nibble

Optional nibble to transmit multiplexed messages.

- Data consistency counter nibble

Optional nibble to transmit multiplexed messages.

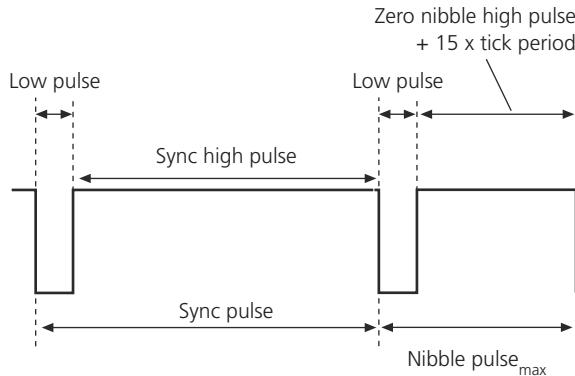
Pause pulse An optional fill pulse. It is transmitted at the end of a SENT message. The pause pulse can be used, for example, to create SENT messages with a constant number of tick periods.

Basic timing of SENT pulses

To transmit SENT message, the sync pulse must be longer than the maximum possible duration of a nibble pulse:

$$\text{Minimal sync pulse length} \geq \text{Maximal nibble pulse length}$$

The following illustration shows a sync pulse and a nibble pulse. The maximum duration of a nibble pulse is a pulse with the data value 15.



The following equations show the parameters that influence the pulse length.

Minimal sync pulse length The minimal sync pulse length is

$$\text{Minimal sync pulse length} = \text{Sync pulse ticks} \cdot \text{Tick period}_{\min}$$

with

$$\text{Sync pulse ticks} = \text{Low pulse ticks} + \text{Synchronization pulse high ticks}$$

and

$$\text{Tick period}_{\min} = \text{Nominal tick period}(1 - \text{Tick period tolerance}).$$

Maximal nibble pulse length The maximal nibble pulse length is

$$\text{Maximal nibble pulse length} = \text{Nibble pulse ticks}_{\max} \cdot \text{Tick period}_{\max}$$

with

$$\text{Nibble pulse ticks}_{\max} = \text{Zero nibble pulse highticks} + \text{Low pulse ticks} + 15$$

and

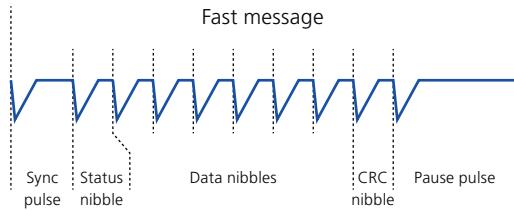
$$\text{Tick period}_{\max} = \text{Nominal tick period}(1 + \text{Tick period tolerance})$$

Message types

The SENT specification defines the following types of messages that can be transmitted via SENT messages:

- Fast messages

The data of fast messages is encoded in the data nibbles of one SENT message as shown in the following example.



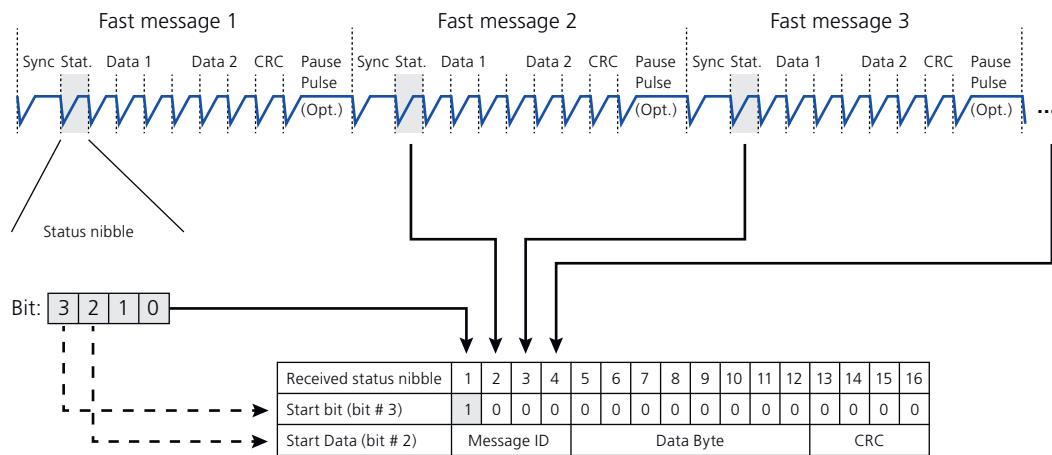
- Serial messages

Serial messages are used to transmit additional data via SENT messages. 2 bits of the first nibble are encoded to transmit data via a fixed number of consecutive SENT messages.

Serial message transmission A SENT message can transmit serial data via Bit 2 and Bit 3 of the first nibble, the status nibble.

The SAE J2716 APR2016 SENT standard specifies the *Short Serial message Format* and the *Enhanced Serial Message Format* for transmitting serial data.

In the *Short Serial Message Format* one serial message is composed of 16 consecutive SENT messages. All 16 SENT messages must be successfully received (without errors) for the serial value to be received. The 16-bit serial message consists of a 4 bit message ID, 8 bit data, and a CRC checksum as shown in the following illustration.



The *Enhanced Serial Message Formats* are composed of 18 consecutive SENT messages. There are two format types:

- Enhanced serial messages with 4-bit ID, 16-bit data, and a CRC checksum.
 - Enhanced serial messages with 8-bit ID and 12-bit data, and a CRC checksum.
- For more information on serial messages, refer to SAE J2716 APR2016 SENT standard.

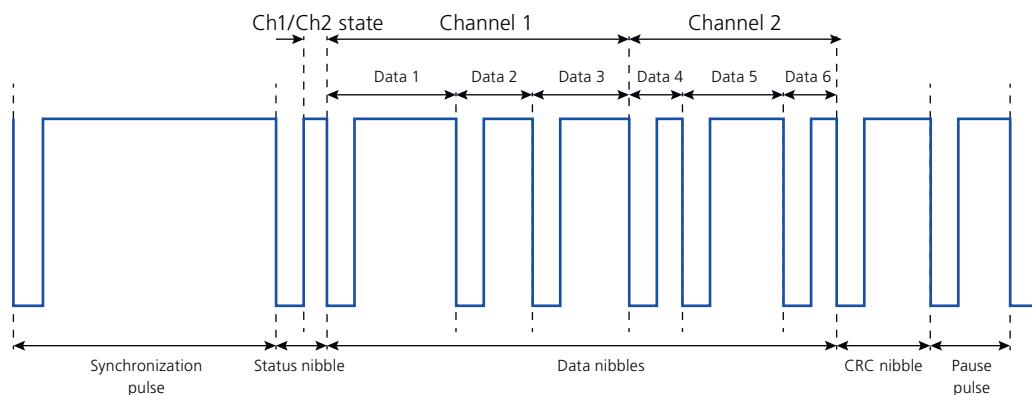
Using Application-Specific Protocols

Purpose

You can use an application-specific protocol to evaluate the values of the nibbles instead of evaluating their values separately in the behavior model.

Data of fast messages

If you use an application-specific protocol, the data nibbles can be grouped on two channels to transmit two data signals in one fast message. The following illustration is an example of a fast message with two data channels.



In ConfigurationDesk, the data is provided to the behavior model via one or two specific function ports (Channel 1 and Channel 2).

Using application specific protocols

The SENT In and SENT Out function blocks let you select protocol files providing application-specific protocols. The function blocks use the protocol to evaluate the nibble values of SENT messages.

You can use predefined or user-defined protocol files to support certain sensor applications.

Predefined protocol files Predefined protocol files are located in the `<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles` folder. The following table lists all predefined protocols and the order and contents of the data nibble pulses as specified in the SAE J2716 APR2016 SENT standard.

Protocol	Data Nibble Pulses						Value Range	
	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Channel 1	Channel 2
A_1.scpxml: Dual throttle position sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Ch2 LSN	Ch2 MidN	Ch2 MSN	0 ... 4095	0 ... 4095
A_2_1.scpxml: Mass air flow sensors 16 bit/8 bit	Ch1 MSN	Ch1 MidMSN	Ch1 MidLSN	Ch1 LSN	Ch2 LSN	Ch2 MSN	0 ... 65535	0 ... 255
A_2_2.scpxml: Mass air flow sensors 14 bit/10 bit	Ch1 MSN	Ch1 MidMSN	Ch1 MidLSN	Ch1/Ch2 LSN	Ch2 MidN	Ch2 MSN	0 ... 16383	0 ... 1023
A_3.scpxml: Single secure sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Counter MSN	Counter LSN	Inverted Copy Ch1 MSN	0 ... 4095	—
A_4_1.scpxml: Single sensors with zero nibble	Ch1 MSN	Ch1 MidN	Ch1 LSN	Counter MSN	Counter LSN	Zero	0 ... 4095	—
A_4_2.scpxml: Single secure with inverted MSB	Ch1 MSN	Ch1 MidN	Ch1 LSN	Counter MSN	Counter LSN	Inverted Copy Ch1 MSN	0 ... 4095	—
A_5_1.scpxml: Pressure sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Ch2 LSN	Ch2 MidN	Ch2 MSN	0 ... 4095	0 ... 4095
A_5_2.scpxml: Pressure secure sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Counter MSN	Counter LSN	Inverted Copy Ch1 MSN	0 ... 4095	—
A_6_1.scpxml: Temperature sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Ch2 LSN	Ch2 MidN	Ch2 MSN	0 ... 4095	0 ... 4095
A_6_2.scpxml: Temperature secure sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Counter MSN	Counter LSN	Inverted Copy Ch1 MSN	0 ... 4095	—
A_7_1.scpxml: Position/ratio sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Ch2 LSN	Ch2 MidN	Ch2 MSN	0 ... 4095	0 ... 4095
A_7_2.scpxml: Position/ratio secure sensors	Ch1 MSN	Ch1 MidN	Ch1 LSN	Counter MSN	Counter LSN	Inverted Copy Ch1 MSN	0 ... 4095	—

Protocol	Data Nibble Pulses						Value Range	
	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Channel 1	Channel 2
H_1.scp.xml: Two fast channels 12 bit/12 bit	Ch1 MSN	Ch1 MidN	Ch1 LSN	Ch2 LSN	Ch2 MidN	Ch2 MSN	0 ... 4095	0 ... 4095
H_2.scp.xml: One 12-bit fast channel	Ch1 MSN	Ch1 MidN	Ch1 LSN	—	—	—	0 ... 4095	—
H_3.scp.xml: One High-speed channel 12 bit	Most significant bits 11 ... 9	Bits 8 ... 6	Bits 5 ... 3	Least significant bits 2 ... 0	—	—	0 ... 4095	—
H_4.scp.xml: One secure sensor 12 bit	Ch1 MSN	Ch1 MidN	Ch1 LSN	Counter MSN	Counter LSN	Inverted Copy Ch1 MSN	0 ... 4095	—
H_5.scp.xml: One sensor 12 bit	Ch1 MSN	Ch1 MidN	Ch1 LSN	Zero	Zero	Zero	0 ... 4095	—
H_6.scp.xml: Two fast channels 14 bit/10 bit	Ch1 MSN	Ch1 MidMSN	Ch1 MidLSN	Ch1/Ch2 LSN	Ch2 MidN	Ch2 MSN	0 ... 16383	0 ... 1023
H_7.scp.xml: Two fast channels 16 bit/8 bit	Ch1 MSN	Ch1 MidMSN	Ch1 MidLSN	Ch1 LSN	Ch2 LSN	Ch2 MSN	0 ... 65535	0 ... 255
Definitions								
MSN	Most significant nibble							
LSN	Least significant nibble							
MiDN	Middle nibble							
MiDMSN	Middle most significant nibble							
MiDLSN	Middle least significant nibble							
Zero	Value of the data nibble pulse is zero							

User-defined protocol files You can create user-defined protocol files. Refer to [Basics on User-Defined Protocol Files](#) on page 1262.

Basics on User-Defined Protocol Files

Introduction

You can create a user-defined protocol file to define SENT communication for a specific application.

Possible applications

User-defined protocol files support the following applications:

- Communication with a SENT device to transmit data via fast messages and optional data via serial messages, for example, device identification information.

SENT communication supports up to two data channels and one rolling counter for fast messages and defined message formats for serial messages.

- Multiplexing of fast messages and/or serial messages that supports the following applications:
 - An application multiplexes fast messages, but not serial messages.
For example, a three-axis position sensor uses a separate fast message for each measured axis position and transmits a serial message for device identification information. The use of separate fast messages that are multiplexed to a data stream lets the sensor transmit high accuracy position values for each axis.
 - An application does not multiplex fast messages, but multiplexes serial messages.
For example, a linear position sensor transmits the measured values via fast messages and uses separate serial messages to transmit status and device identification information.
 - An application multiplexes the fast and serial messages.
For example, an application multiplexes the data and serial messages of different sensors to one data stream to reduce the cabling effort.

Protocol file type

The protocol files are XML files with the file name extension `.scp.xml`.

XML schema The `SentCommunicationProtocol.xsd` file is the XML schema for SENT protocol files. The file is located in the `<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles` folder.

Protocol consistency check

During the import of the protocol file, ConfigurationDesk checks the consistency of the protocol file. If the protocol file is not consistent with the XML schema, the Message Viewer displays an error message and ConfigurationDesk does not import the protocol file.

Furthermore, the Message Viewer displays a warning message if the protocol file does not define a status nibble or a CRC nibble, because these nibbles are required by the SAE J2716 APR2016 SENT standard.

Creating user-defined protocol files

For information on creating user-defined protocol files, refer to [Creating User-Defined Protocol Files](#) on page 1266.

Basics on Multiplexing SENT Messages

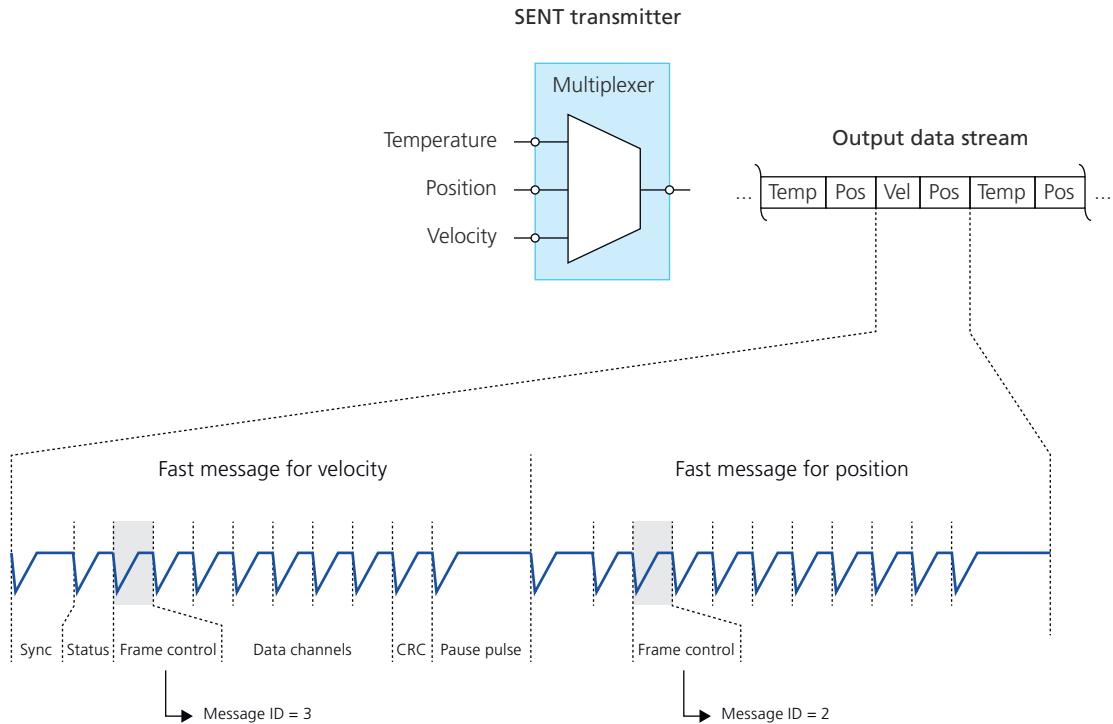
Introduction

The SENT specification includes the multiplexing of fast and/or serial messages to transmit different messages via a single SENT communication line.

Concept of SENT multiplexing

To multiplex SENT messages in ConfigurationDesk, you create and import a user-defined protocol file to specify the multiplexing of SENT messages.

Multiplexing of fast messages The following example shows a SENT transmitter that multiplexes three input signals to one data stream.



The SENT transmitter adds the data of each input signal in a fast message. The fast messages are output in the order defined in the user-defined protocol file. In the example, every second fast message is a fast message with the position sensor value (Pos). Up to 16 fast messages with the same number of nibbles can be multiplexed.

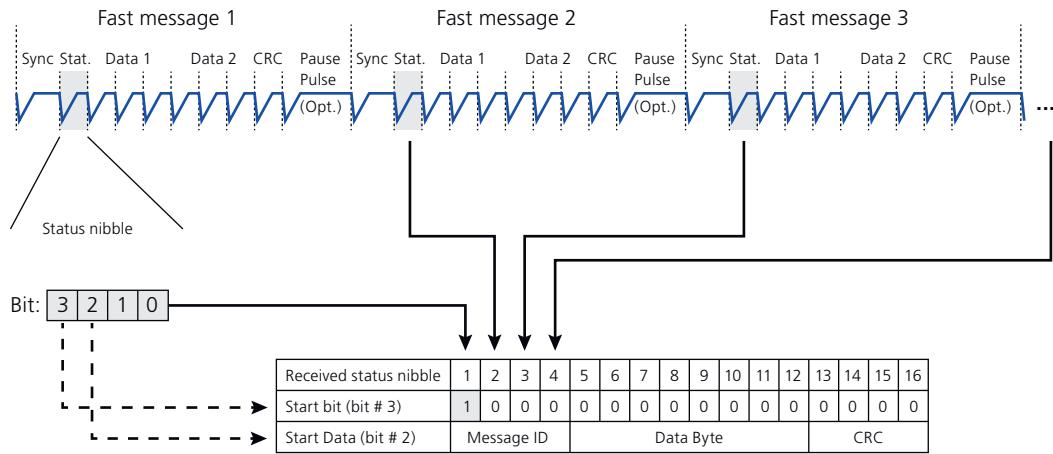
An additional frame control nibble enables the SENT receiver to demultiplex the data stream. The frame control nibble includes an identity number (message ID) to differ between the different fast messages. In the example, the temperature input has the message ID 1, the position input the message ID 2, and the velocity input the message ID 3.

Multiplexing of serial messages Up to 100 serial messages can be multiplexed, depending on the used serial message mode:

- Short serial message: 16 serial messages
- Enhanced serial message (4-bit ID): 16 serial messages
- Enhanced serial message (8-bit ID): 100 serial messages

The SENT transmitter outputs the serial messages in the order defined in the user-defined protocol file.

To demultiplex the data stream, the SENT receiver reads the message ID of the serial message. In contrast to fast messages, the message ID is always part of a serial message as shown in the following example of a short serial message.



Specifying the transmission order

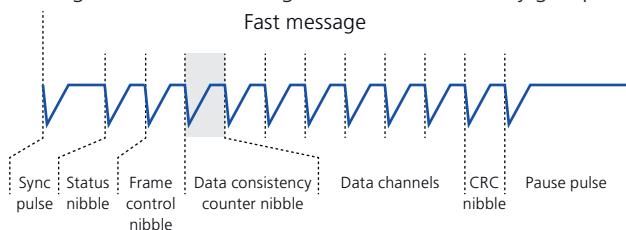
You define sequences in the user-defined protocol file to specify the transmission order of fast and/or serial messages:

- There are no restrictions on the order in which messages are sent.
- A sequence can consist all or a selection of the defined messages.
- A sequence can contain the same message several times. That means, some messages can be transmitted more frequently than other messages.
- Up to 10 sequences can be defined within a user-defined protocol file. The current sequence for multiplexing/demultiplexing can be selected via function ports.

Grouping fast messages

You can group fast messages to a data consistency group. This enables the receiver to provide information whether the fast messages belong together. For example: A three-axis position sensor transmits the measurement for each axis via a separate fast message. The fast messages can be grouped to ensure that the read values belong to the same measurement.

If a data consistency group is used, an additional data consistency counter nibble enables the SENT receiver to identify which fast messages belongs together and must be provided at the same time. The following example shows a SENT message with a fast message of a data consistency group.



Functionality of the data consistency counter The data consistency counter is a rolling counter that always increments before the first fast message of a data consistency group will be sent. The incremented counter value is sent with each fast message of the data consistency group. That means, all fast messages of the data consistency group have the same counter value.

At the SENT receiver, the data values of the received data consistency group are valid if all fast messages have the same counter value.

Multiplexing of data consistency groups A data consistency group is multiplexed as a single element. That means, fast messages that are not part of the data consistency group can be sent only before or after a data consistency group.

Note

All SENT messages must have the same number of nibbles if you multiplex fast messages. Keep in mind that fast messages of a data consistency group have an additional data consistency counter nibble in the SENT message.

Creating User-Defined Protocol Files

Purpose	You can create user-defined protocol files to define application-specific protocols.
Basics on SENT protocol files	For basics on SENT protocol files, refer to the following: <ul style="list-style-type: none">▪ Using Application-Specific Protocols on page 1260▪ Basics on User-Defined Protocol Files on page 1262▪ Basics on Multiplexing SENT Messages on page 1263
Rules for specifying user-defined protocols	In addition to the basic rules for defining user-defined protocols there are additional rules to support the transmitting of serial messages and the multiplexing of fast messages. Basic rules for user-defined protocols The protocol definition must comply with the following basic rules: <ul style="list-style-type: none">▪ The bits of a data channel or rolling counter must be mapped to a data nibble. You can transmit redundant data by mapping data bits twice. You must mark redundant data with the Redundant attribute. For each bit, only one redundant bit is allowed.▪ 4 bits must be mapped to each nibble.▪ The status nibble must be the first nibble of a fast message. The CRC checksum nibble must be the last nibble of a fast message. Both nibbles can be defined only once in a fast message.▪ The protocol must comply with the XML schema defined in the <code>SentCommunicationProtocol.xsd</code> file.

Additional rules to support serial messages The protocol definition must comply with the following additional rules to support serial messages:

- All serial message types must use the same format:
 - Short serial message format
 - Enhanced serial message (4-bit ID) format
 - Enhanced serial message (8-bit ID) format
- Each serial message type must have a unique message ID.
- At least one sequence in which serial messages are transmitted must be defined even if only one serial message type is defined.

Additional rules to support the multiplexing of fast messages The protocol definition must comply with the following additional rules to support the multiplexing of fast messages:

- All fast messages must have the same number of nibbles.
- A frame control nibble must be added to the fast message types. The position in the SENT message must be the same for all defined fast messages.
- The frame control nibbles of the fast message definitions must provide a unique message ID.
- If you group fast messages, a data consistency counter nibble must be added to the grouped fast message definitions. The position in the SENT message must be the same for all grouped fast messages.
- At least one sequence in which fast messages are transmitted must be defined.

Templates for user-defined protocol files

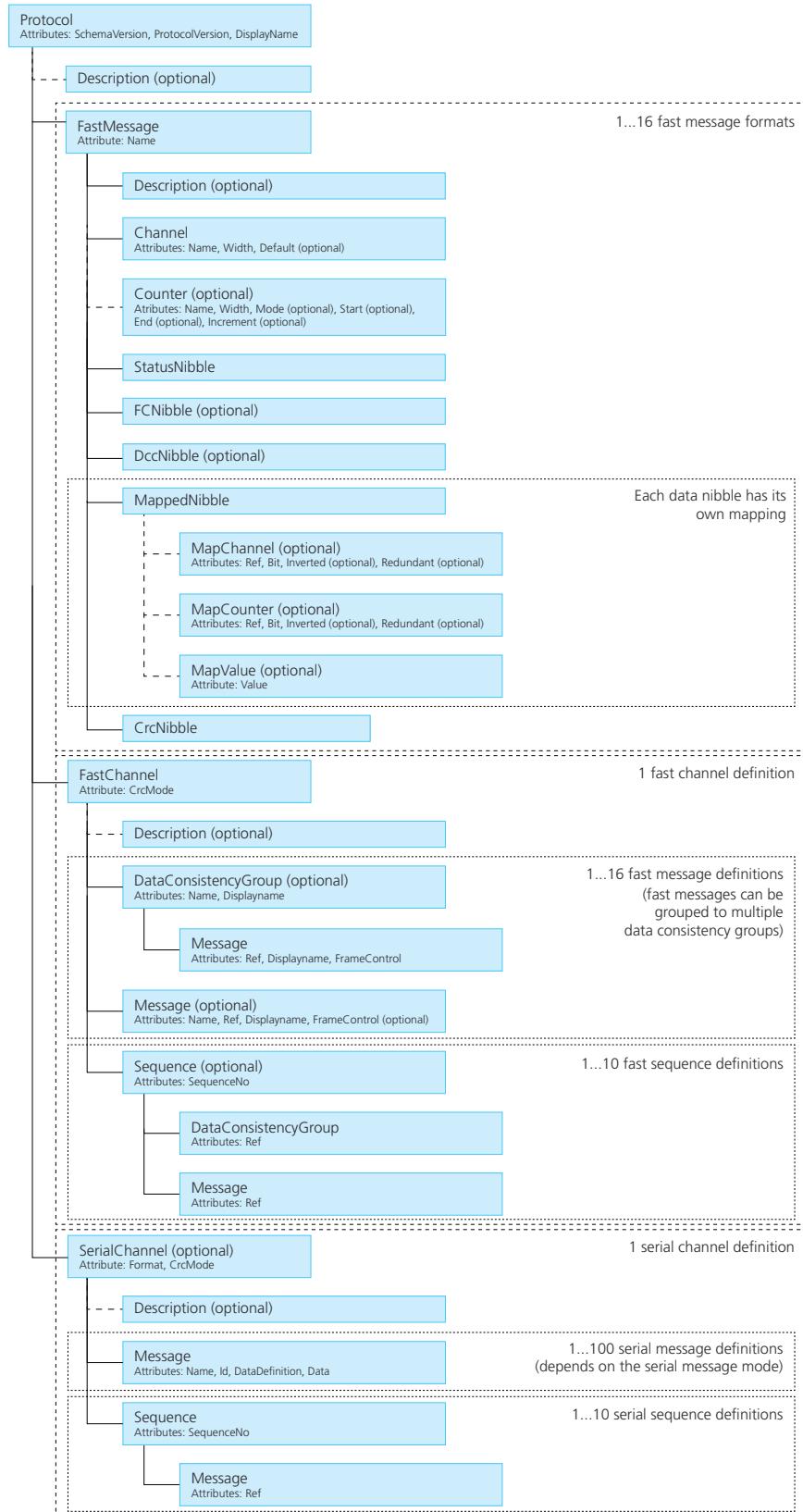
An application-specific template is located in the `<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles` folder:

- `3DPositionSensor.scp.xml`

You can use this template to create a user-defined protocol file.

XML schema

The following illustration shows the XML schema version 1.1 for defining SENT messages. For examples, refer to [Examples of user-defined protocols](#) on page 1273.



XML elements

The following table describes the elements that you can use to define the protocol in alphabetical order:

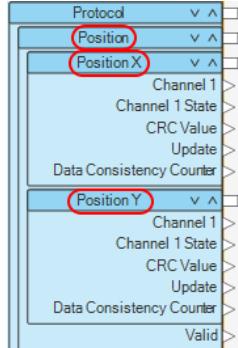
XML Elements	Description
Channel	Lets you define up to two data channels with predefined names. The number of bits for each channel can be defined with the Width attribute. The Name attribute of the data channel must be set to Channel 1 or Channel 2 . The following example defines the Channel 1 data channel with a data width of 12 bits: <Channel Name="Channel 1" Width="12"/>
Counter	Lets you define the rolling counter. You can define only one counter per fast message. The Name attribute of the counter must be set to Counter .
CrcNibble	Lets you add a 4-bit CRC checksum nibble to the message. This nibble must be the last nibble of the SENT message.
DataConsistencyGroup	Lets you define a group of fast messages that provide consistent data. Grouped fast messages share a common data consistency counter. Every referenced FastMessage element of this group must provide a DCCNibble element to provide a nibble with the counter value.
DccNibble	Lets you add the data consistency counter nibble to the fast message. The data consistency counter identifies whether the values of the DataConsistencyGroup element are consistent. The position in the SENT message must be the same for all grouped fast messages.
Description	Lets you enter a string as a description for the superior XML element. ConfigurationDesk does not import this description.
FastChannel1	Lets you define the fast messages and the possible sequences in which fast messages are transmitted. The fast messages are based on the format that is defined with the FastMessage element.
FastMessage	Lets you define the fast message format. Up to 16 different fast message formats can be defined. Each FastMessage element can define up to 32 nibbles. All FastMessage elements must contain the same number of nibbles.
FcNibble	Lets you add the frame control nibble to the fast message. The frame control nibble includes the identity number of the fast message. This nibble must be added for multiplexing fast messages and must have the same position in all defined fast messages.
MapChannel1	Lets you map up to 4 bits of a data channel to the nibble.
MapCounter	Lets you map up to 4 bits of the rolling counter to the nibble.
MappedNibble	Lets you define the value of one data nibble bit by bit. All 4 bits of a nibble must be defined, starting with the most significant bit. You can map bits from different sources to one nibble, i.e., you can use MapChannel1 , MapCounter , and MapValue elements in the same data nibble to define each individual bit. The following example uses the MapChannel1 element to map bit 8 and bit 9 of Channel 1 to the most significant bits

XML Elements	Description
	<p>and the <code>MapValue</code> element to map the values '1' and '0' to the least significant bits of the data nibble:</p> <pre><MappedNibble> <MapChannel Ref="Channel 1" Bit="8 9"/> <MapValue Value="1 0"/> </MappedNibble></pre>
<code>MapView</code>	Lets you map up to 4 bit values (0/1) to a nibble. The mapped values are constant and cannot be changed. The following example maps 4 bits to the nibble. The first value is the value of the most significant bit, the last value is the value of the least significant bit: <code><MapView Value="0 1 0 1"/></code>
<code>Message</code>	Lets you define the messages to be transmitted.
<code>Protocol</code>	Root element to define a user-defined protocol.
<code>Sequence</code>	Lets you define the order and count in which fast messages and serial messages are transmitted. Up to 10 different sequences can be defined.
<code>SerialChannel</code>	<p>Lets you define the serial messages and the possible sequences in which serial messages are transmitted. The serial messages are based on the format that is defined with the <code>Format</code> attribute.</p> <p>The number of serial messages is limited to 100.</p>
<code>SerialMessage</code>	<p>Lets you define one serial message.</p> <p>Depending on the serial message mode, you can define up to 100 <code>SerialMessage</code> messages with a unique serial message ID:</p> <ul style="list-style-type: none"> ▪ Short serial message: 16 serial messages ▪ Enhanced serial message (4-bit ID): 16 serial messages ▪ Enhanced serial message (8-bit ID): 100 serial messages
<code>StatusNibble</code>	Lets you add the status nibble to the message. This nibble must be the first nibble.

XML attributes

The following table describes the attributes that you can use to define the protocol in alphabetical order:

XML Attribute	Description
<code>Bit</code>	Lets you define up to 4 bits of a referenced channel or counter to be mapped to a nibble.
<code>CrcMode</code>	<p>Lets you define the algorithm for calculating the CRC checksum for a message. If you do not define the algorithm or if the specified value is invalid, the CRC checksum is calculated in the <code>Automatic</code> mode.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ▪ <code>Automatic</code>: The checksum is calculated according to the SAE J2716 Revision JAN2010 standard and later. ▪ <code>Automatic Legacy</code>: The checksum is calculated according to the SAE J2716 Revision FEB2008 standard and earlier. This mode is supported only for short serial messages. ▪ <code>Manual</code>: The checksum is provided by the behavior model. This mode is supported only for serial messages.
<code>Data</code>	<p>Lets you define a data value for the serial message.</p> <p>The value range depends on the serial message mode:</p> <ul style="list-style-type: none"> ▪ Short serial message: 0 ... 255

XML Attribute	Description
	<ul style="list-style-type: none"> ▪ Enhanced serial message (4-bit ID): 0 ... 65535 ▪ Enhanced serial message (8-bit ID): 0 ... 4095
DataDefinition	<p>Lets you define the type of the Data attribute.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ▪ Fixed: The value of the Data attribute is a fixed value, for example, to transmit the serial number of the sensor. Data function ports are not available for fixed data. ▪ Default: The value of the Data attribute is a default value that is used until a valid data value is available.
Default	<p>Lets you define a default value for a data channel. If you do not set Default to a value, this value is set to 0.</p>
DisplayName	<p>Lets you enter a name for the XML element that is displayed in ConfigurationDesk if you import the protocol file.</p> <ul style="list-style-type: none"> ▪ Protocol element: The Protocol name property displays the entered name in the Properties Browser. ▪ DataConsistencyGroup and Message (fast message) elements: The entered name is the name of the function port group that represents the XML element. If the DisplayName attribute is not defined, the Name attribute is used instead. <p>Supported characters: a ... z, A ... Z, 0 ... 9, underscore, and whitespace. The name must start with a letter.</p> <p>The following example shows how the DisplayName attribute is used in ConfigurationDesk.</p> <pre><DataConsistencyGroup Name="Pos" DisplayName="Position"> <Message Ref="DataPos" DisplayName="Position X" FrameControl="1" /> <Message Ref="DataPos" DisplayName="Position Y" FrameControl="2" /> </DataConsistencyGroup></pre> 
End	<p>Lets you specify the end value of the rolling counter. If the rolling counter reaches this value, it starts again with the value of the Start attribute.</p> <p>The value range depends on the defined number of counter bits: 0 ... $2^{\text{Width}} - 1$, whereas the value 0 has a special meaning. The value 0 represents the maximum value.</p> <p>The specified counter settings must be plausible and fulfill the following requirements:</p> <ul style="list-style-type: none"> ▪ The specified values for counter start, end, and increment must be inside the value range of the counter. The value range of the counter is defined by the Width attribute: 0 ... $2^{\text{Width}} - 1$. ▪ The direction of counting must match the start and end values of the counter. If the counter increments, the counter start value must be less than or equals the end value. If the counter decrement, the start value must be greater than or equals the end value. ▪ The direction of counting depends on the Increment attribute.

XML Attribute	Description
Format	<p>Lets you define the format of the serial message.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ▪ Short: A serial message frame comprises 16 fast messages. ▪ Enhanced with 4 Bit id: A serial message frame comprises 18 fast messages with the specified bit arrangement (4-bit ID, 16-bit data). ▪ Enhanced with 8 Bit id: A serial message frame comprises 18 fast messages with the specified bit arrangement (8-bit ID, 12-bit data).
FrameControl	Lets you define a unique message ID for a fast message. The value range is 0 ... 15.
Id	<p>Lets you define a unique message ID for a serial message.</p> <p>The value range depends on the serial message mode:</p> <ul style="list-style-type: none"> ▪ Short serial message: 0 ... 15 ▪ Enhanced serial message (4-bit ID): 0 ... 15 ▪ Enhanced serial message (8-bit ID): 0 ... 255
Increment	<p>Lets you specify the increment value for the rolling counter. Negative values let the counter decrement.</p> <p>The value range depends on the defined number of counter bits: $-2^{\text{Width}} \dots 2^{\text{Width}}$. If you do not specify the increment value, the value is 1.</p>
Inverted	Lets you invert the individual bits that are mapped to a nibble via the MapValue or MapCounter element by adding the Inverted="True" attribute.
Mode	<p>Lets you define how to calculate and provide the values of the rolling counter. If you do not define the mode, the values are calculated in the Automatic mode.</p> <p>Possible values</p> <ul style="list-style-type: none"> ▪ Automatic: The counter value is calculated by the function block. However, it can be modified by the behavior model via the Counter Modify function import. The resulting value can be written to the behavior model via the Counter Value function export. ▪ Manual: The counter value is provided by the behavior model via the Counter function port.
Name	<p>Lets you specify a unique name for the XML element. In the protocol, the name is used to reference the XML element.</p> <p>Supported characters: a ... z, A ... Z, 0 ... 9, underscore, and whitespace. The name must start with a letter.</p> <p>The Name attribute of serial messages is displayed in ConfigurationDesk if you import the protocol file. For fast messages, the Name attribute is displayed in Configuration only if no DisplayName attribute is defined.</p> <p>The following example shows how the Name attribute is used in ConfigurationDesk.</p> <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <pre><SerialChannel Format="Short" CrcMode="Automatic"> <Message Name="Message 1" Id="1" DataDefinition="Default" Data="1" /> <Message Name="Message 2" Id="2" DataDefinition="Default" Data="1" /> <Sequence SequenceNo="1"> (...)</Sequence> </SerialChannel></pre> </div> <div style="flex: 1;"> </div> </div>
ProtocolVersion	Lets you enter an unsigned integer number to differentiate between different protocol versions. Reserved for future use.
Redundant	Lets you add redundant data by adding the Redundant="True" attribute.

XML Attribute	Description
Ref	Lets you reference an XML element via the element's name.
SchemaVersion	Lets you define the version of the XML schema used to specify the protocol. Set the SchemaVersion to the current version 1.1 . The XML schema version 1 is a legacy version.
SequenceNo	Lets you enter an identifier for the sequence in the range 0 ... 15. The identifier is used to select a sequence from the behavior model via the Sequence Number function port.
Start	Lets you specify the minimum value of the rolling counter range. If the rolling counter reaches the value of the End attribute, it starts again with this value. The value range depends on the defined number of counter bits: 0 ... $2^{\text{Width}}-1$ The specified counter settings must be plausible and fulfill the following requirements: <ul style="list-style-type: none"> ▪ The specified values for counter start, end, and increment must be inside the value range of the counter. The value range of the counter is defined by the Width attribute: 0 ... $2^{\text{Width}}-1$. ▪ The direction of counting must match the start and end values of the counter. If the counter increments, the counter start value must be less than or equals the end value. If the counter decrement, the start value must be greater than or equals the end value. The direction of counting depends on the Increment attribute.
Value	Lets you specify bit values (0/1) for a nibble.
Width	Lets you specify the number of bits used for a channel or a counter. Up to 32 bits can be specified for a channel, up to 8 bits can be specified for a counter.

Examples of user-defined protocols

Use the following protocols as examples of how to use the XML attributes and elements according to the XML scheme.

The following protocol shows a protocol for an application with one fast message and one serial message.

```

<?xml version="1.0" encoding="Utf-8"?>
<Protocol xmlns="urn:dSPACE:SentCommunicationProtocol"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xi="http://www.w3.org/2003/XInclude"
           SchemaVersion="1.1"
           ProtocolVersion="1"
           DisplayName="Name displayed in the Properties Browser">
  <Description>
    Example for SENT communication without multiplexing
  </Description>

  <FastMessage Name="Temperature">
    <Channel Name="Channel 1" Width="8"/>
    <Counter Name="Counter" Width="4"/>
    <StatusNibble/>
    <MappedNibble>
      <MapChannel Ref="Channel 1" Bit="4 5 6 7"/>
    </MappedNibble>
    <MappedNibble>
      <MapChannel Ref="Channel 1" Bit="0 1 2 3"/>
    </MappedNibble>
    <MappedNibble>
      <MapCounter Ref="Counter" Bit="0 1 2 3"/>
    </MappedNibble>
    <MappedNibble>
      <MapChannel Ref="Channel 1" Bit="4 5 6 7"
                  Inverted="True" Redundant="True" />
    </MappedNibble>
    <CrcNibble/>
  </FastMessage>

  <FastChannel>
    <Message Ref="Temperature"
             Name="Temp" DisplayName="Temperature"/>
  </FastChannel>

  <SerialChannel Format="Enhanced with 4 Bit id" CrcMode="Automatic">
    <Message Name="Status" Id="1" DataDefinition = "Default" Data = "0" />
    <Sequence SequenceNo="0">
      <Message Ref="Status" />
    </Sequence>
  </SerialChannel>
</Protocol>

```

The following protocol shows a protocol for an application with multiplexed fast messages and multiplexed serial messages.

```

<?xml version="1.0" encoding="Utf-8"?>
<Protocol (...)>
  <Description>
    Example for multiplexed SENT communication
  </Description>

```

```

<FastMessage Name="Pos">
    <Channel Name="Channel 1" Width="8"/>
    <StatusNibble />
    <FcNibble />
    <DccNibble />
    <MappedNibble>
        <MapChannel Ref="Channel 1" Bit="0 1 2 3" />
    </MappedNibble>
    ...
    <CrcNibble/>
</FastMessage>
<FastMessage Name="Temp">
    <Channel Name="Channel 1" Width="8"/>
    <Counter Width="4" Mode="Automatic" Name="Counter"/>
    <StatusNibble />
    <FcNibble />
    <MappedNibble>
        <MapChannel Ref="Channel 1" Bit="4 5 6 7" />
    </MappedNibble>
    ...
    <CrcNibble></CrcNibble>
</FastMessage>
<FastChannel CrcMode="Automatic">
    <DataConsistencyGroup Name="Pos" DisplayName="Position">
        <Message Name = "Position X" Ref="Pos" DisplayName="Position X"
            FrameControl="1" />
        <Message Name = "Position Y" Ref="Pos" DisplayName="Position Y"
            FrameControl="2" />
    </DataConsistencyGroup>
    <Message Name="Temp" Ref="Temp" DisplayName="Temperature"
        FrameControl="3" />
    <Sequence SequenceNo="1">
        <DataConsistencyGroup Ref="Pos"/>
        <Message Ref="Temp" />
    </Sequence>
    <Sequence SequenceNo="2">
        <DataConsistencyGroup Ref="Pos" />
        <Message Ref="Temp" />
        <DataConsistencyGroup Ref="Pos" />
    </Sequence>
</FastChannel>
<SerialChannel Format="Enhanced with 4 Bit id" CrcMode="Automatic">
    <Message Name="Message 1" Id="1" DataDefinition = "Fixed" Data = "1" />
    <Message Name="Message 2" Id="2" DataDefinition = "Default" Data = "2" />
    <Message Name="Message 3" Id="3" DataDefinition = "Default" Data = "3" />
    <Sequence SequenceNo="1">
        <Message Ref="Message 1" />
        <Message Ref="Message 2" />
        <Message Ref="Message 3" />
    </Sequence>
    <Sequence SequenceNo="2">
        <Message Ref="Message 1" />
        <Message Ref="Message 2" />
        <Message Ref="Message 1" />
        <Message Ref="Message 3" />
    </Sequence>
</SerialChannel>
</Protocol>

```

SENT In

Where to go from here

Information in this section

Introduction (SENT In).....	1276
Overviews (SENT In).....	1277
Configuring the Function Block (SENT In).....	1290
Hardware Dependencies (SENT In).....	1299

Introduction (SENT In)

Introduction to the Function Block (SENT In)

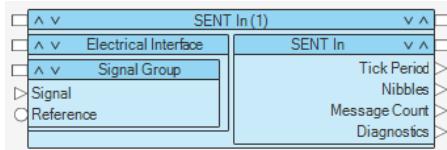
Function block purpose

The SENT In function block receives SENT messages. SENT is a protocol used between sensors and ECUs to transmit data of high-resolution sensors as an alternative to an analog interface. The sensors are typically throttle position sensors or mass air flow sensors.

The SENT In function block supports the SAE J2716 APR2016 SENT standard.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

The main features are:

- Receiving and evaluating SENT messages
- Receiving and evaluating serial messages in short and enhanced serial message format.
- Supports the multiplexing of messages.

- Supporting application specific protocols according to the SAE J2716 APR2016 standard.
- Specifying the timing of the expected SENT messages.
- Calculating and providing a CRC checksum to the behavior model.
- Evaluating and providing diagnostic information to the behavior model.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The Sent In function block supports the following channel types:

	SCALEXIO							MicroAutoBox III
Channel type	Digital In 1	Flexible In 2	Digital In 3	Flexible In 3	Flexible In 1	Digital In/Out 5	Digital In 4	
Hardware	DS2680		DS6101		DS2601	DS6202	<ul style="list-style-type: none"> ▪ DS1511 ▪ DS1511B1 ▪ DS1513 	

Oversviews (SENT In)

Where to go from here**Information in this section**

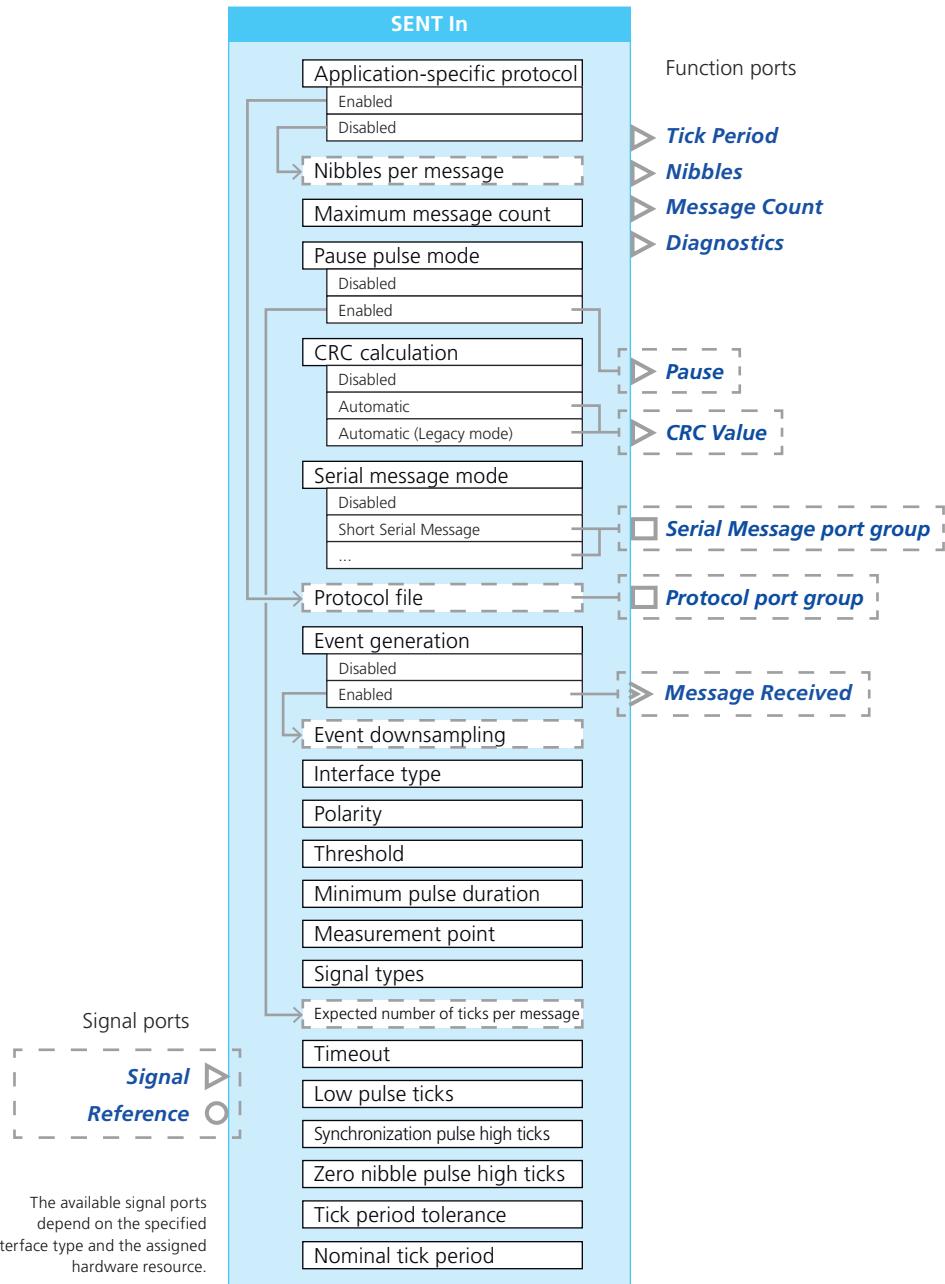
[Overview of Ports and Basic Properties \(SENT In\).....](#) 1277

[Overview of Tunable Properties \(SENT In\).....](#) 1289

Overview of Ports and Basic Properties (SENT In)

Overview illustration

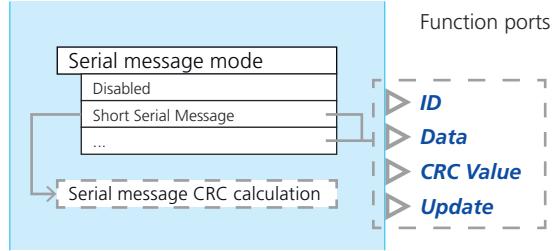
The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



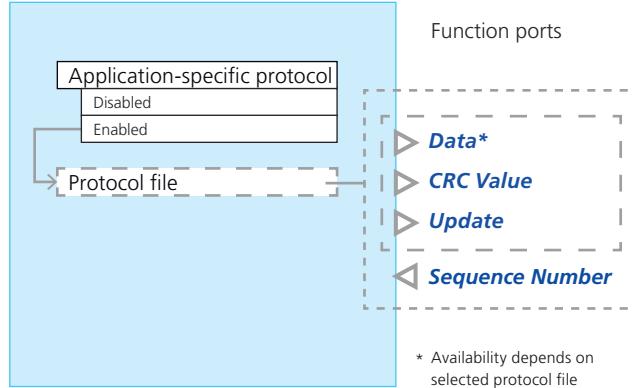
Serial Message port group The function ports of the Serial Message port group support the reception and evaluation of a serial message via a defined number of SENT messages. They are available only if the Serial Message Mode

property is set to a message format or the application-specific protocol supports serial messages.

The following illustration shows the ports for serial messages if the Serial Message Mode property is set to a message format.



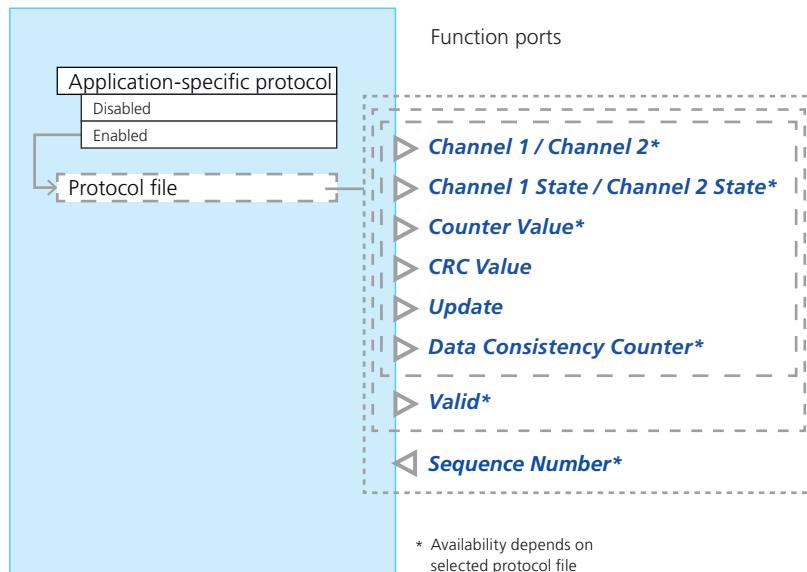
The following illustration shows the ports for serial messages if the application-specific protocol supports serial messages.



Protocol port group The function ports of the Protocol port group support the reception and evaluation of application-specific protocols (according to the SAE J2716 APR2016 SENT standard) for specific sensor applications. The Protocol port group is only available if the Application-specific protocol

property is set to Enabled and a protocol file is selected via the Protocol file property.

The following illustration shows the ports for protocol files.



Tick Period

This function outport provides the current tick period (in seconds) which is measured from the last received valid synchronization pulse.

Value range	65 ns ... 1 s
Dependencies	—

Nibbles

This function outport provides the nibble values vector to the behavior model which are received by the function block.

The nibble values are calculated as follows:

$$\text{Nibble value} = \text{Int8}(\text{Nibble pulse length}/\text{Tick period} + 0.5) - \text{Zero nibble pulse ticks}$$

with

$$\text{Zero nibble pulse ticks} = \text{Low pulse ticks} + \text{Zero nibble high pulse ticks}$$

Only the last complete received messages are returned. The Message Count function port provides the number of new messages.

Value range	<ul style="list-style-type: none"> ▪ - (Low pulse ticks + Zero nibble high pulse ticks) ... + 127 ▪ 0 ... 15: Allowed value range for nibbles. <p>When a nibble value outside the allowed range of 0 ... 15 is received, the nibble is returned with the message anyway. In addition the Diagnostics outport reports a nibble which is</p>
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>outside the valid range. The nibble value -128 defines a missing nibble.</p> <ul style="list-style-type: none"> The vector size is the product of the Nibbles per message property value and the Maximum message count property value. The vector begins with the nibble of the first received SENT message. For example, the vector {1; 2; 3; 4; 5; 6} with the property settings Nibbles per message = 3 and Maximum message count = 2 includes the nibble values of two SENT messages: Message 1 = {1; 2; 3} and message 2 = {4; 5; 6}.
Dependencies	Available only if the Application-specific protocol properties are set to Disabled.

Message Count

This function outport provides the number of messages that were received since the last read operation.

This information can be used to detect data loss, for example, caused by a model step size (sample time) that is too long. If messages are lost, the Message Count outport provides a value higher than the value of the Maximum message count property.

Value range	0 ... 65535 messages
Dependencies	-

Diagnostics

This function outport provides diagnostic information for the received SENT message via flags in a diagnostic word.

Value range	<ul style="list-style-type: none"> 0: No error. >0: See table below.
Dependencies	-

The following table shows the values and descriptions of the diagnostic flags:

Bit (Flag)	Value	Description
0	1	Too many nibbles in message When too many nibbles are received in a message, the excess nibbles are ignored and the diagnostic flag reports too many nibbles.
1	2	Too few nibbles in message When a message with too few nibbles is received, the missing nibbles are marked with the value "-128" and the diagnostic flag reports missing nibbles.
2	4	Nibble value out of range [0 ... 15] When a nibble with a value of <0 or >15 is received, this nibble is saved to the message anyway, and the diagnostic flag for a nibble out of the valid range is set.

Bit (Flag)	Value	Description
3	8	Synchronization pulse too long. When a synchronization pulse exceeds the valid range (upper limit) of the expected tick period specified by the Nominal tick period and Tick period tolerance properties, this is reported by the diagnostic flag. The nibble values are evaluated anyway.
4	16	Synchronization pulse too short. When a synchronization pulse falls below the valid range (lower limit) of the expected tick period specified by the Nominal tick period and Tick period tolerance properties, this is reported by the diagnostic flag. The nibble values are evaluated anyway.
5	32	When two consecutive synchronization pulses differ by more than 1/64 of the current synchronization pulse, this is reported by the diagnostic flag.
6	64	Fixed message length is different. When the tick periods in a received message differ from the value specified by Expected number of ticks per message, this is reported by the diagnostic flag. The message length includes the low and high ticks of all pulses of a message, including the sync pulse, the pause pulse and all nibble pulses. As preconditions for evaluation, the Pause pulse mode property must be enabled and the value specified by Expected number of ticks per message property must be set unequal to 0. When very long pause pulses are transmitted (pause pulse >> synchronization pulse), this diagnostic flag can be set erroneously, because normalization does not take effect. In this case, ignore this diagnostic flag and use only diagnostic flag 7 (see below) for diagnostic purposes.
7	128	The ratio of sync pulse length and complete message length is different to the expected value. When the ratio of sync pulse length and complete message length differs by more than 1/64 (approx. 1.5 %) between the expected value (= specified via relevant timing properties) and the current measured value, this is reported by the diagnostic flag. This lets you observe the clock drift of a SENT transmitter while it is transmitting a SENT message. As preconditions for evaluation, the Pause pulse mode property must be enabled and the value specified by the Expected number of ticks per message property must be set to nonzero.
8	256	Synchronization error When a sync pulse could not be detected unambiguously, this diagnostic flag is set. A synchronization error occurs, for example, when there are more potential sync pulses in a SENT message than specified in the SAE J2716 APR2016 SENT standard. The receiver will resynchronize, when the next valid sync pulse is detected.
9	512	Timeout If no complete SENT pulse (low and high pulse) is received within the time specified by the Timeout property, this diagnostic flag is set.
15	32,768	CRC error in serial message transmission. An error is detected for the transmission of a serial message by the CRC checksum value.

Bit (Flag)	Value	Description
16	65,536	CRC error in SENT message transmission. An error is detected for the transmission of a SENT message by the CRC checksum value.
17	131,072	Error detected for inverted data nibble pulse. Valid only for application-specific protocols containing a counter.
19	524,288	Wrong format of received serial message. The format of the received serial message does not match the format of the selected Enhanced Serial Message (via Serial Message Mode property). Example: You have specified the Enhanced Serial Message mode with 4-bit ID and 16-bit Data format, but a serial message in Enhanced Serial Message mode with 8-bit ID and 12-bit Data format has been received.
20	1,048,576	Incorrect sequence of received fast messages. The sequence of the received fast messages does not match the sequence that is defined by the application-specific protocol. Example: The received sequence does not match the expected sequence that is selected via the Sequence port.
21	2,097,152	Incorrect sequence of received serial messages. The sequence of the received serial messages does not match the sequence that is defined by the application-specific protocol. Example: The received sequence does not match the expected sequence that is selected via the Sequence port.
22	4,194,304	Mismatching constant values. The application-specific protocol specifies constant values for specific data bits. The received data does not match the constant bit values.
29	536,870,912	I/O buffer overflow. More pulses are captured than supported by the hardware resource. All data values are discarded.
30	1,073,741,824	SENT messages lost. More SENT messages are received than specified at the Maximum message count property. Only the last received messages are provided to the behavior model.
31	2,147,483,648	Internal communication error. Some data values might be lost.

For example, if a message has a nibble out of range (flag 2, value 4) and the synchronization pulse is too short (flag 4, value 16), the diagnostic word value is 20 (0x14). To get the information, you must evaluate the returned diagnostic word.

Pause

This function outport provides a pause value which is evaluated from the last received message. The pause value is calculated (like the nibble values) as follows:

$$\text{Pause value} = \text{Int16}(\text{Pause pulse length}/\text{Tick period} + 0.5) - \text{Zero nibble pulse ticks}$$

with

Zero nibble pulse ticks = Low pulse ticks + Zero nibble high pulse ticks

Note

The SENT In function block always interprets the last pulse of a SENT message as a pause pulse, if the Pause pulse mode is enabled.

Value range	<ul style="list-style-type: none"> ▪ - (<i>Low pulse ticks + Zero nibble high pulse ticks</i>) ... + 32767 ▪ -32768: Missing pause pulse. This value is output until the first message has been received.
Dependencies	Available only if the Pause pulse mode property is set to Enabled.

CRC Value

This function outport provides the CRC checksum value (calculated at run time by ConfigurationDesk) for every received SENT message to the behavior model.

Value range	0 ... 15
Dependencies	Available only if the CRC calculation property is set to Automatic or to Automatic (Legacy).

Message Received

This event port provides an I/O event each time a specific number of SENT messages has been received completely. At the Event downsampling property, you have to specify the number of received messages required to trigger an I/O event.

Value range	-
Dependencies	Available only if the Event generation property is set to Enabled.

ID

This function outport provides the received value for the message ID of the serial message to the behavior model.

Value range	Depends on the selected serial message mode as follows: <ul style="list-style-type: none"> ▪ Short Serial Message: 0 ... 15 ▪ Enhanced Serial Message (4-bit ID, 16-bit data): 0 ... 15 ▪ Enhanced Serial Message (8-bit ID, 12-bit data): 0 ... 255
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats and no protocol file is selected.

Data This function outport provides the received value for the data field of the serial message to the behavior model.

Value range	Depends on the selected serial message mode as follows: <ul style="list-style-type: none">▪ Short Serial Message: 0 ... 255▪ Enhanced Serial Message (4-bit ID, 16-bit data): 0 ... 65535▪ Enhanced Serial Message (8-bit ID, 12-bit data): 0 ... 4095
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats or a protocol file is selected that specifies a serial message without constant data.

CRC Value (of Serial Message port group) This function outport provides the CRC checksum value for the serial message (calculated at run time by ConfigurationDesk) to the behavior model.

Value range	Depends on the selected serial message mode as follows: <ul style="list-style-type: none">▪ Short Serial Message: 0 ... 15▪ Enhanced Serial Message: 0 ... 63
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats or a protocol file is selected that specifies serial messages.

Update (of Serial Message port group) This function outport provides an update flag to the behavior model every time enough SENT messages have been received to complete a serial message according to the SAE J2716 APR2016 SENT standard. For details, refer to [Configuring the Basic Functionality \(SENT In\)](#) on page 1290.

Value range	<ul style="list-style-type: none">▪ 1: Update. A complete serial message that matches the CRC checksum has been received. The values available at the function outports of the selected port group can be used by the behavior model.▪ 0: No update. No complete serial message has been received.
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats or a protocol file is selected that specifies serial messages.

Sequence Number (of Serial Message port group) This function import selects a sequence definition from the application-specific protocol file. A sequence defines the transmission order of serial messages in the data stream.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 15 ▪ The valid values depends on the sequences that are defined by the protocol file. Invalid values stops the receiving of SENT messages.
Dependencies	Available only if a protocol file is selected that specifies serial messages.

Channel 1 / Channel 2

This function outport provides the value for the data field of the related channel to the behavior model.

Value range	Depends on the selected application-specific protocol. For specific values, refer to Configuring the Basic Functionality (SENT In) on page 1290.
Dependencies	Available only if a protocol file is selected. The Channel 2 function port is available for protocol files that specify two channels.

Channel 1 State / Channel 2 State

This function outport provides a diagnostic flag to the behavior model.

The value of Channel 1 State function port is mapped to bit 0 of the status nibble, the value of Channel 2 State function port to bit 1 of this nibble.

Value range	<ul style="list-style-type: none"> ▪ 0: OK. Channel x is OK. ▪ 1: Error. Channel x is in error state.
Dependencies	Available only if a protocol file is selected. The Channel 2 function port is available for protocol files that specify two channels.

Counter Value

This function outport provides a counter value (calculated at run time by ConfigurationDesk) to the behavior model.

Value range	0 ... 255
Dependencies	Available only if a protocol file is selected that specifies a rolling counter.

CRC Value (of Protocol port group)

This function outport provides the CRC checksum value (calculated at run time by the function block) for the received SENT message that is provided by the port group.

Value range	0 ... 15
Dependencies	Available only if a protocol file is selected.

Update (of Protocol port group)

This function outport provides an update flag to the behavior model every time a fast message with the message ID of the selected port group and a matching CRC checksum has been received.

Value range	<ul style="list-style-type: none"> ▪ 1: Update. The function outports of the port group provide new data. ▪ 0: No update. No new fast message with a matching message ID has been received.
Dependencies	Available only if a protocol file is selected.

Data Consistency Counter

This function outport provides the value of the data consistency nibble that is received with the fast messages of the selected port group. Fast messages of a data consistency group are sent with the same counter value to identify which values of fast messages belong together.

Value range	0 ... 15
Dependencies	Available only if a protocol file is selected that specifies data consistency groups.

Valid

This function outport provides a flag to the behavior model to indicate if the data consistency counter of each grouped fast message is the same.

Value range	<ul style="list-style-type: none"> ▪ 1: Valid. The function outports of the data consistency group provide values that belong together. ▪ 0: Invalid. The function outports of the data consistency group provide values that do not belong together.
Dependencies	Available only if a protocol file is selected that supports data consistency groups.

Sequence Number (of Protocol port group)

This function import selects a sequence definition from the application-specific protocol file. A sequence defines the transmission order of fast messages in the data stream.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 15 ▪ The valid values depends on the sequences that are defined by the protocol file. Invalid values stops the receiving of SENT messages.
Dependencies	Available only if a protocol file is selected that multiplexes fast messages.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SENT In) on page 1299.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SENT In) on page 1299.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Load Signal

This signal port represents the electrical connection point to connect external loads.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Available only if the Connected load property is set to Use external load. ▪ Not supported for Digital In 3, Flexible In 3 and Digital In/Out 5 channel types. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported.

Load Reference

This signal port is a reference port and provides the reference signal for the Load signal signal port.

The port is not shown in the overview illustration.

Value range	Depends on the assigned hardware resource.
Dependencies	<ul style="list-style-type: none"> ▪ SCALEXIO hardware: <ul style="list-style-type: none"> ▪ Supported only for the Flexible In 1 channel type. ▪ Available only if the Connected load property is set to Use external load. ▪ MicroAutoBox III hardware: <ul style="list-style-type: none"> ▪ Not supported.

Overview of Tunable Properties (SENT In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Properties	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Event downsampling	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Low pulse ticks	✓	-
Synchronization pulse high ticks	✓	-
Zero nibble pulse high ticks	✓	-
Tick period tolerance	✓	-
Nominal tick period	✓	-
Timeout	✓	-
Expected ticks per message	✓	-
Polarity	✓	-
Threshold	✓	-
Measurement point	✓	-
Minimum pulse duration	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Note

Changing the timing of the SENT pulses can result in an unfavorable pulse configuration. The real-time application does not output a message if a nibble pulse is longer than or equal to the minimum pulse length of the synchronization pulse. For more information on the timing of SENT pulses, refer to [Basics on the SENT Protocol](#) on page 1256.

Configuring the Function Block (SENT In)

Where to go from here

Information in this section

Configuring the Basic Functionality (SENT In).....	1290
Using Application-Specific Protocols to Decode SENT Messages.....	1293
Reading Nibble Values to Decode SENT Messages.....	1295
Configuring Standard Features (SENT In).....	1296

Configuring the Basic Functionality (SENT In)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the timing of the expected SENT messages.
- Decoding the SENT messages with one of the following methods:
 - [Using Application-Specific Protocols to Decode SENT Messages](#) on page 1293
 - [Reading Nibble Values to Decode SENT Messages](#) on page 1295
- Reading serial messages with the function block.
- Reading pause pulses.
- Providing I/O events, for example, to read SENT messages asynchronously.
- Conditioning the input signal (polarity, threshold, pulse filter).

Specifying the timing of a SENT message

To receive a SENT message, the timing of the receiver must comply with the timing of the transmitter. Therefore you have to specify properties of the function block according to the signal, that is expected from the SENT transmitter.

Tick period The tick period is the base clock every SENT pulse is generated with. You can specify the expected pulse length of the tick period and the tick period tolerance. As the real-time hardware measures the actual tick period

when a message is received, the function block outputs a diagnostic flag if the measured length of the tick period is outside the specified tolerance.

Low pulse, zero nibble high pulse, synchronization high pulse These SENT-specific configurable properties are defined by a number of tick periods. For basic information on the timing of a SENT message, refer to [Basics on the SENT Protocol](#) on page 1256.

Timeout value You can specify a timeout value in seconds. If no SENT pulse is received within the specified time, the function block outputs a diagnostic flag and the receiver begins to resynchronize.

Expected tick periods per message When you receive SENT messages which provide pause pulses, you can use the measured pause pulse values for diagnostic purposes. The **Expected tick periods per message** property is available only in pause pulse mode to specify the expected length of the SENT messages.

The specified expected length is used to evaluate additional receiver diagnostics used for SENT messages with a fixed message length.

Reading serial messages

With the SENT In function block you can receive a serial message via a defined number of consecutive SENT messages from a SENT transmitter. According to the SAE J2716 APR2016 SENT standard, the function block supports the following formats:

- *Short Serial Message*: A serial message frame comprises 16 fast messages.
- *Enhanced Serial Message (4-bit ID, 16-bit data)*: A serial message frame comprises 18 SENT messages with the specified bit arrangement.
- *Enhanced Serial Message (8-bit ID, 12-bit data)*: A serial message frame comprises 18 SENT messages with the specified bit arrangement.

The following methods can be used to evaluate the SENT messages:

- The behavior model reads raw nibble values and evaluates the serial data from the first nibble of a SENT message. For more information on reading raw nibble values, refer to [Reading Nibble Values to Decode SENT Messages](#) on page 1295.
- The function block evaluates the serial messages. This can be activated as follows:
 - The user-defined protocol file specifies the evaluation of serial messages. For more information, refer to [Using Application-Specific Protocols to Decode SENT Messages](#) on page 1293.
 - You activate the evaluation of serial messages by setting the **Serial message mode** property to the used serial message format.

For basic information on the formats, refer to [Basics on the SENT Protocol](#) on page 1256.

Update functionality To ensure that only complete serial messages are received from the SENT transmitter, the function block counts the received SENT messages. If the required number of SENT messages has been received, the function block provides an update flag via the **Update** function port and updates the port values of the serial message port group.

Reading pause pulses for diagnostic purposes

Every SENT message can contain a pause pulse transmitted at the end of the SENT message (after the CRC nibble). This optional pause pulse is generated from the transmitter to create SENT messages with a constant number of tick periods.

You have to enable the Pause pulse mode property of the SENT In function block to configure the receiver for receiving pause pulses:

- You can use the measured pause pulse value in the behavior model for special advanced use cases. The value is available at the Pause function outport.
- If your transmitter sends a pause pulse to implement a constant message length, the receiver can process additional receiver diagnostics. This diagnostic information (provided via the Diagnostics outport) can be used in the behavior model. When you specify a nonzero value for the Expected number of ticks per message property, the receiver uses this expected message length for evaluation and can set diagnostic flag 6 and/or flag 7. For details, refer to [Basics on the SENT Protocol](#) on page 1256.

Note

The SENT In function block always interprets the last pulse of a SENT message as a pause pulse, if the Pause pulse mode is enabled.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The SENT In function block provides an I/O event after every n-th SENT message that has been received completely. N is the value that you have to specify for event downsampling. For example, if this value is 3, an I/O event is provided after every third received SENT message.

To use the I/O events in the behavior model, you have to assign them to a task via the Message Received property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Limitation

- SCALEXIO hardware: Event downsampling is not supported.

Conditioning the input signal

The SENT In function block provides the following features to condition the input signal to your requirements.

Specifying polarity of the input signal To adapt the SENT receiver to the polarity of the signal generated by the SENT transmitter, you can switch the Polarity of the signal available at the function port:

- Active high (= default setting specified in SENT standard)

The higher value (voltage/current) of the signal represents a binary 1, and the lower value represents a binary 0.

- Active low

The lower value (voltage/current) of the signal represents a binary 1, and the higher value represents a binary 0.

Specifying threshold value The adjustable threshold value lets you compensate voltage offsets on the input signal. You should set the threshold value so that the system can correctly distinguish between high and low pulses. The hysteresis value is fixed.

A rising edge of the input signal must exceed the threshold plus the half hysteresis value to be detected as a high pulse, and a falling edge of the input signal must fall below the threshold minus the half hysteresis value to be detected as a low pulse. For specific values, refer to [Hardware Dependencies \(SENT In\)](#) on page 1299.

Specifying pulse duration You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the Minimum pulse duration property are ignored.

For specific values, refer to [Hardware Dependencies \(SENT In\)](#) on page 1299.

Using Application-Specific Protocols to Decode SENT Messages

Introduction

Every SENT message provides a sequence of nibbles to transmit data. The function block can use an application-specific protocol to evaluate the values of the nibbles to provide the received data values at specific function ports. The data of fast messages are provided at the Channel 1 and Channel 2 function ports. The data of serial messages are provided at the Data function ports.

Protocol types

The following protocol types are supported:

- Predefined protocol files that can be used to support specific sensor applications. For more information, refer to [Using Application-Specific Protocols](#) on page 1260.
- User-defined protocol files that you created. For more information, refer to [Basics on User-Defined Protocol Files](#) on page 1262.

Importing an application-specific protocol

The Protocol file property lets you import a protocol file. To import a protocol file, you can enter its file path or you can open the standard Open dialog to browse for and select a protocol file.

There are no restrictions on the location of SENT protocol files. Predefined protocol files are located in the `<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles` folder.

Changing user-defined protocols Each time you change a protocol file, you must reimport the changed file so that the changes become effective.

After a reimport, you must map and configure protocol-based function ports again.

Reading the data of received messages

To read the data of the received SENT messages, the behavior model can read the data periodically with each model step or it can read the data asynchronously via I/O events. The Update function ports indicate which function ports provide new valid data.

Reading SENT messages periodically At run time, the timer task periodically calls the SENT In function block with each model step and reads the most recent data values.

To evaluate the data of serial messages and the multiplexing of fast messages, the received SENT messages are buffered in a message buffer. You can set the maximum number of buffered messages by using the **Maximum message count** property. The recommended setting can be calculated as follows:

$$\text{Maximum message count} > \frac{1.5 \cdot \text{Model step size}}{\text{Minimum message length}}$$

To calculate the minimum message length, add the ticks of all sync, nibble and pause pulses and multiply the value by the minimum tick period. All nibble pulses (status, data, CRC, etc.) represent the value 0 in this calculation unless a pause pulse ensures a constant message length. For messages with a constant length, use the constant length as minimum message length.

However, data loss can occur if a function port is updated more often than the behavior model reads the data. To read all data of received SENT messages, you can read the SENT messages asynchronously.

Reading SENT messages asynchronously Reading SENT messages asynchronously to the model steps requires a self-triggered system: The I/O events of the SENT In function block trigger a runnable function, and the task of the runnable function triggers the model port block that reads the provided data from the function block. To use this method, you have to enable event generation for the function block.

The I/O event of the SENT In function block is generated after each SENT message that was received completely. Depending on the real-time hardware, you can downsample the event generation to read more than one SENT message. The function ports provide the data of the last completely received messages.

If the I/O events are not downsampled, set the **Maximum message count** property to 1, because more SENT messages cannot be received since the last reading of the behavior model.

If the I/O events are downsampled, set the **Maximum message count** property to the same value as the **Event downsampling** property. This enables the function block to evaluate serial messages and multiplexed fast messages.

For information on providing I/O events, refer to [Configuring the Basic Functionality \(SENT In\)](#) on page 1290.

Reading Nibble Values to Decode SENT Messages

Introduction

Reading raw nibble values is the most flexible method for decoding data of SENT messages. In the behavior model, the value of each nibble must be evaluated to decode a received fast message and/or serial message.

Specifying number of nibbles

The number of nibbles is constant for all the SENT messages of one specific SENT application, but can vary between different applications. The number of nibbles includes all nibbles, such as data nibbles, the status nibble, and the CRC nibble.

To adapt the SENT In function block to the connected SENT transmitter, you have to use the same number of nibbles as provided by the transmitter.

Reading the nibble values

To read the nibble values of the received SENT messages, the behavior model can read the nibble values of several SENT messages periodically or it reads the nibble values of each SENT message asynchronously via I/O events.

Reading SENT messages periodically At run time, the timer task periodically calls the SENT In function block with each model step and reads the nibble values of the received SENT messages that are buffered in a message buffer.

To receive the nibble values without data loss due to a buffer overflow, the model step size (sample step) of the task in the behavior model must be shorter than the length of all SENT message that can be buffered. The buffer size can be set via the Maximum message count property. The recommended setting can be calculated as follows:

$$\text{Maximum message count} > \frac{1.5 \cdot \text{Model step size}}{\text{Minimum message length}}$$

To calculate the minimum message length, add the ticks of all sync, nibble and pause pulses and multiply the value by the minimum tick period. All nibble pulses (status, data, CRC, etc.) represent the value 0 in this calculation unless a pause pulse ensures a constant message length. For messages with a constant length, use the constant length as minimum message length.

The Message Count function port provides the number of SENT messages that were received since the last read operation to evaluate the valid nibble values.

Reading SENT messages asynchronously The reading of SENT messages asynchronously to the model steps requires a self-triggered system: The I/O events of the SENT In function block trigger a runnable function, and the task of the runnable function triggers the model port block that reads the new nibble values from the function block. To use this method, you have to enable event generation for the function block.

The I/O event of the SENT In function block is generated after each SENT message that was received completely. Depending on the real-time hardware, you can downsample the event generation to read the nibble values of more than one SENT message.

If the I/O events are not downsampled, set the **Maximum message count** property to 1, because the function block can only provide the nibble values of one SENT message.

If the I/O events are downsampled, set the **Maximum message count** property to the same value as the **Event downsampling** property. This setting buffers all received SENT messages until a new I/O event triggers the reading of the received messages.

For information on providing I/O events, refer to [Configuring the Basic Functionality \(SENT In\)](#) on page 1290.

Configuring Standard Features (SENT In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

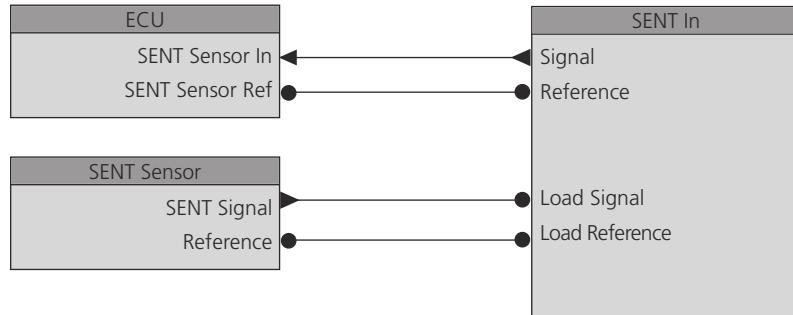
Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer . You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SCALEXIO				
Configuration Feature	Digital In 1	Digital In 3	Digital In/Out 5	Further Information
Measurement point	Load side	–	–	See below.
Interface type	Ground-based	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Note: The SENT In function block type does not support channel multiplication. Therefore the voltage and current ranges of a single hardware channel apply.				
Signal Ports				
Trigger level of electronic fuse	–	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	–	–	Specifying Load Settings on page 121

SCALEXIO				
Configuration Feature	Flexible In 1	Flexible In 2	Flexible In 3	Further Information
Measurement point	<ul style="list-style-type: none"> ▪ Load side ▪ ECU side 	Load side	–	See below.
Interface type	<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Ground-based 	Ground-based	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports				
Trigger level of electronic fuse	✓	–	–	Specifying Fuse Settings on page 122
Failure simulation support	✓	✓	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	✓	✓	–	Specifying Load Settings on page 121

MicroAutoBox III		
Configuration Feature	Digital In 4	Further Information
Measurement point	–	–
Interface type	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Channel multiplication	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports		
Trigger level of electronic fuse	–	Specifying Fuse Settings on page 122
Failure simulation support	–	Specifying Settings for Failure Simulation on page 123
Usage of loads	–	Specifying Load Settings on page 121

Mapping signal ports To use the SENT In function block type to test an ECU, you should implement the signal chain as shown below (Flexible In 1 channel type).



The SENT sensor is treated and implemented as an external load connected to the SENT In function block. The SCALEXIO system loops the signals to the connected ECU via this function block

To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

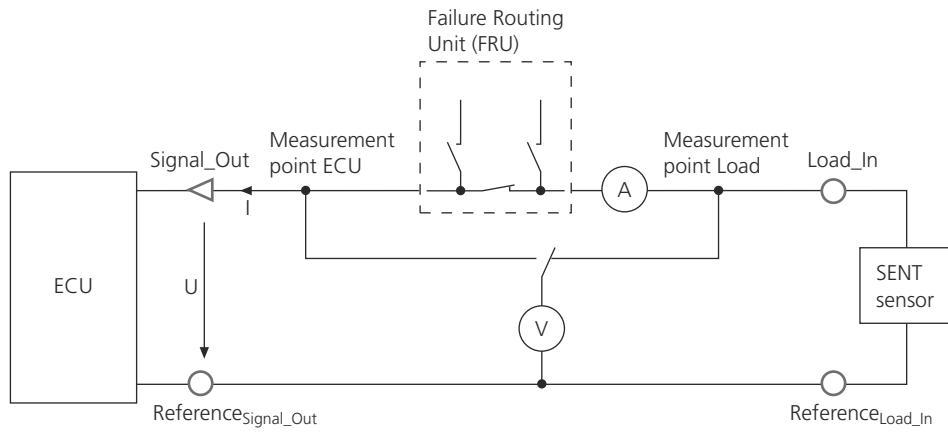
SCALEXIO	MicroAutoBox III
<ul style="list-style-type: none"><li data-bbox="649 910 981 918">▪ Digital In 1 on page 1569<li data-bbox="649 918 981 925">▪ Digital In 3 on page 1572<li data-bbox="649 925 981 931">▪ Digital In/Out 5 on page 1586<li data-bbox="649 931 981 937">▪ Flexible In 1 on page 1593<li data-bbox="649 937 981 944">▪ Flexible In 2 on page 1595<li data-bbox="649 944 981 952">▪ Flexible In 3 on page 1603	<ul style="list-style-type: none"><li data-bbox="981 910 1308 918">▪ Digital In 4 on page 1572

Measurement point

Note

The measurement point setting is not relevant and therefore ignored for the Digital In 3, Digital In 4, Digital In/Out 5, and Flexible In 3 channel types.

If you select the Flexible In 1 channel type, you can define whether to perform voltage measurement on the ECU side or on the load side (SENT sensor). The signal can be interrupted or short-circuited by an electrical failure routing unit (FRU). If the measurement is performed on the load side and load rejection is enabled, the signal can not be measured when the failure is activated.



Limitations If you select the Digital In 1 or Flexible In 2 channel type: The measurement point is on the load side and cannot be changed.

For further information, refer to [Basics of Electrical Errors Simulation in a SCALEXIO System \(SCALEXIO – Hardware and Software Overview\)](#).

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (SENT In)

Where to go from here

Information in this section

- [SCALEXIO Hardware Dependencies \(SENT In\)..... 1300](#)
- [MicroAutoBox III Hardware Dependencies \(SENT In\)..... 1301](#)

SCALEXIO Hardware Dependencies (SENT In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital In 1	Flexible In 2	Flexible In 1	Digital In 3	Flexible In 3	Digital In/Out 5
Hardware		DS2680 I/O Unit		DS2601 Signal Measurement Board	DS6101 Multi-I/O Board		DS6202 Digital I/O Board
Event generation		✓		✓	✓		✓
Nibbles per message		1 ... 32		1 ... 32	1 ... 32		1 ... 32
Maximum message count		1 ... 8		1 ... 8	1 ... 8		1 ... 8
Input current range		0 ... +6 A _{RMS}		0 ... +10 A _{RMS}	–		–
Input voltage range		0 ... +60 V		-60 V ... +60 V	0 ... +60 V		0 ... +30 V
Threshold	Input threshold voltage	0 ... +24 V		-60 V ... +60 V	0 ... +24 V	-10 V ... +10 V	0 ... +12 V
	Input hysteresis voltage	200 mV (typ.)	210 mV (typ.)	250 mV (typ.)	200 mV (typ.)		600 mV (typ.)
	DAC resolution	8 bit		14 bit	8 bit		8 bit
Supported interface type		Ground-based		<ul style="list-style-type: none"> ▪ Galvanically isolated ▪ Ground-based 	Ground-based		Ground-based
Measurement point		Load side		<ul style="list-style-type: none"> ▪ Load side ▪ ECU side 	–		–
Pulse filter	Minimum pulse duration	0.264 µs ... 131 µs		0.264 µs ... 131 µs	0.264 µs ... 131 µs		8 ns ... 10 ms
Configurable fuse		–		<ul style="list-style-type: none"> ▪ Load side ▪ 0.5 A_{RMS} ... 10 A_{RMS} 	–		–
Circuit diagrams		Digital In 1 on page 1569	Flexible In 2 on page 1595	Flexible In 1 on page 1593	Digital In 3 on page 1572	Flexible In 3 on page 1603	Digital In/Out 5 on page 1586
Required channels		1		1	1		1

General limitations

The SENT In function block does not support channel multiplication.

DS6202 Digital I/O Board Only 8 SENT In function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2601 Signal Measurement Board For more board-specific data, refer to [Data Sheet of the DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

MicroAutoBox III Hardware Dependencies (SENT In)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type		Digital In 4
Hardware		<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board
Event generation		✓
Nibbles per message		1 ... 64
Maximum message count		1 ... 48
Input voltage range		0 ... +40 V
Threshold	Input threshold voltage	+2 V (typ.)
	Input hysteresis voltage	1 V (typ.)
Pulse filter	Minimum pulse duration	0 ... 12.75 µs
Supported interface type		Ground-based
Circuit diagrams		Digital In 4 on page 1572
Required channels		1

General limitations

The SENT In function block does not support channel multiplication.

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to [DS1511 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

DS1513 Multi-I/O Board For more board-specific data, refer to [DS1513 Multi-I/O Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration\)](#).

SENT Out

Where to go from here

Information in this section

Introduction (SENT Out).....	1303
Overviews (SENT Out).....	1308
Configuring the Function Block (SENT Out).....	1321
Hardware Dependencies (SENT Out).....	1329

Introduction (SENT Out)

Where to go from here

Information in this section

Introduction to the Function Block (SENT Out).....	1303
Basics on the Message Buffer.....	1304
Basics on Simulating Faulty SENT Messages.....	1306

Introduction to the Function Block (SENT Out)

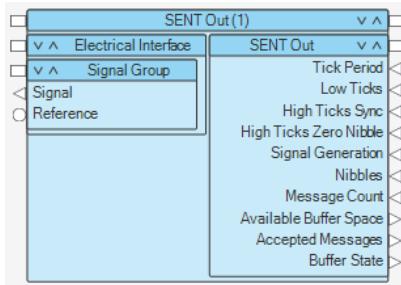
Function block purpose

The SENT Out function block type transmits SENT messages. SENT is a protocol used between sensors and ECUs to transmit data of high-resolution sensors as an alternative to an analog interface. The sensors are typically throttle position sensors or mass air flow sensors.

The SENT Out function block supports the SAE J2716 APR2016 SENT standard.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.

**Main features**

The main features are:

- Transmitting SENT messages.
- Transmitting serial messages in short and enhanced serial message format.
- Supports the multiplexing of messages.
- Supporting application specific protocols according to the SAE J2716 APR2016 standard.
- Specifying the timing of SENT messages.
- Providing diagnostic information to the behavior model.
- Generating I/O events and providing them to the behavior model.

Supported channel types

The SENT Out function block type supports the following channel types:

	SCALEXIO					MicroAutoBox III
Channel type	Digital Out 1	Digital Out 3	Digital In/Out 5	Flexible Out 1	-	MicroAutoBox III
Hardware	DS2680	DS6101	DS6202	DS2621	-	MicroAutoBox III

Basics on the Message Buffer

Basic principle

The SENT Out function block uses a message buffer to separate the transmission of SENT messages and the reading of the messages that are provided by the behavior model.

Function ports provide the following information on the message buffer and let you provide a suitable amount of messages to the message buffer during run time:

- Available buffer space for new messages that can be written from the behavior model to the message buffer.
You must specify the number of messages that can be provided to the message buffer to ensure that enough messages are buffered for one model step.
- Number of messages of the last provided messages that are accepted to be transmitted.

- Information on the buffer state. This information can be used to control whether a buffer underflow or overflow occurs.

Note

If you use an application-specific protocol, only the information on the buffer state is provided.

The following example shows the provided information and the generated output signal if 3 messages are provided from the behavior model per model step.

Model step	1	2	3	4
Available buffer space	3	3	3	3
Accepted messages	0	3	3	3
Buffer state	Ok	Ok	Ok	Ok
Provided messages	3 (1;2;3)	3 (4;5;6)	3 (7;8;9)	3 (10;11;12)
Generated messages	1	2	3	4

To ensure a continuous transmission of new messages, more messages are provided to the message buffer than the function block can generate per model step. This avoids a buffer underflow.

To avoid a buffer overflow and a loss of messages, function ports provide information on whether there is enough buffer space to provide new messages. In the example, the behavior model does not provide new messages when no buffer space is available. The following illustration shows what happens when a model step begins before the function block begins to provide the last messages (model steps 22 and 23).

Model step	20	21	22	23	24	+
Available buffer space	3	3	3	0	3	3
Accepted messages	0	3	3	3	0	3
Buffer state	Ok	Ok	Ok	Ok	Ok	Ok
Provided messages	3 (4;5;6)	3 (7;8;9)	3 (10;11;12)	0 ()	3 (13;14;15)	3
Generated messages	2	3	4	5	6	7

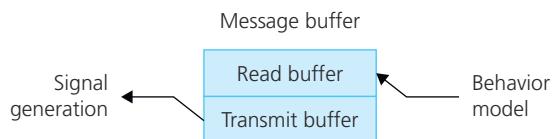
As an alternative, you can provide messages that were not accepted again to avoid a loss of messages (model step 24). The following illustration uses the example above to show this.

Model step	20	21	22	23	24	+
Available buffer space	3	3	3	0	3	3
Accepted messages	0	3	3	3	0	3
Buffer state	Ok	Ok	Ok	Ok	Overflow	Ok
Provided messages	3 (4;5;6)	3 (7;8;9)	3 (10;11;12)	3 (13;14;15)	3 (13;14;15)	3
Generated messages	2	3	4	5	6	7

Using application-specific protocols If you use an application-specific protocol, the available buffer space and the number of accepted messages are not provided by function ports. However, the information is internally used to provide a suitable amount of SENT messages to the message buffer.

Details on the message buffer

The message buffer is implemented as a swinging buffer with two internal buffers: The transmit buffer provides the messages that the signal generation outputs and the read buffer stores the messages that are read from the behavior model. To swing between the two internal buffers, only the pointers to the internal buffers have to be changed. When the message buffer swings, the transmit buffer becomes the read buffer and vice versa.



The message buffer swings if the signal generation of the function block reads all messages of the transmit buffer and the read buffer provides new messages.

The SENT specification requires the continuous transmission of messages. To ensure a continuous transmission if a buffer underflow occurs, the function block repeats the output of the complete transmit buffer until new messages are available. The following example shows the generated messages with a buffer underflow (model steps 1 and 4).

Model step	1	2	3	4
Available buffer space	2	2	2	2
Accepted messages	0	2	2	2
Buffer state	Ok	Underflow	Ok	Ok
Provided messages	2 (1;2)	2 (3;4)	2 (5;6)	2 (7;8)
Generated messages	1	2	1	2

The last row shows the sequence of generated messages: 1, 2, 1, 2, 3, 4, 5, 6, 7, 8, 7. A brace at the end of the sequence indicates a repetition of message 7.

Related topics

Basics

Encoding SENT Messages in the Behavior Model..... 1326

Basics on Simulating Faulty SENT Messages

Introduction

The SENT protocol defines the length of a pulse with the tick period. Pulse length discrepancies are specified with pulse values outside the valid range.

To simulate faulty SENT messages, you can specify nibble pulse and pause pulse values outside the valid range.

Possible faults

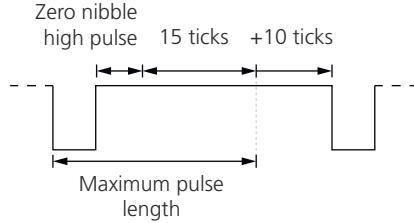
The following faults are specified with nibble values outside the possible range:

- High pulses or nibble pulses are too long.
- *Zero nibble high pulses* of nibble or pause pulses are too short.
- Nibble or pause pulses are missing.

Values of high pulses that are too long

A positive pulse value specifies a low pulse with the duration *low ticks* and a high pulse with duration *high ticks zero nibble + pulse value*. Nibble values higher than 15 specify high pulses that are too long.

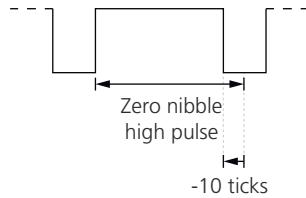
For example: The nibble value 25 specifies a high pulse that is 10 ticks longer than the maximum length of a nibble pulse. The following illustration shows the example:



Values of zero nibble high pulses that are too short

A negative pulse value (nibble value or pause value) defines a high pulse with fewer ticks than specified for the *high ticks zero nibble*.

For example: The valid range for nibble values is 0 ... 15. The nibble value -10 specifies a high pulse that is 10 ticks shorter than the zero nibble high pulse. The following illustration shows the example:



Values of missing nibbles

The nibble value -128 and the pause value -32768 specify a missing pulse. The output of a pulse is also omitted if the reduction of the *zero nibble high pulse* results in a high pulse length of 0 ticks or less.

Oversights (SENT Out)

Where to go from here

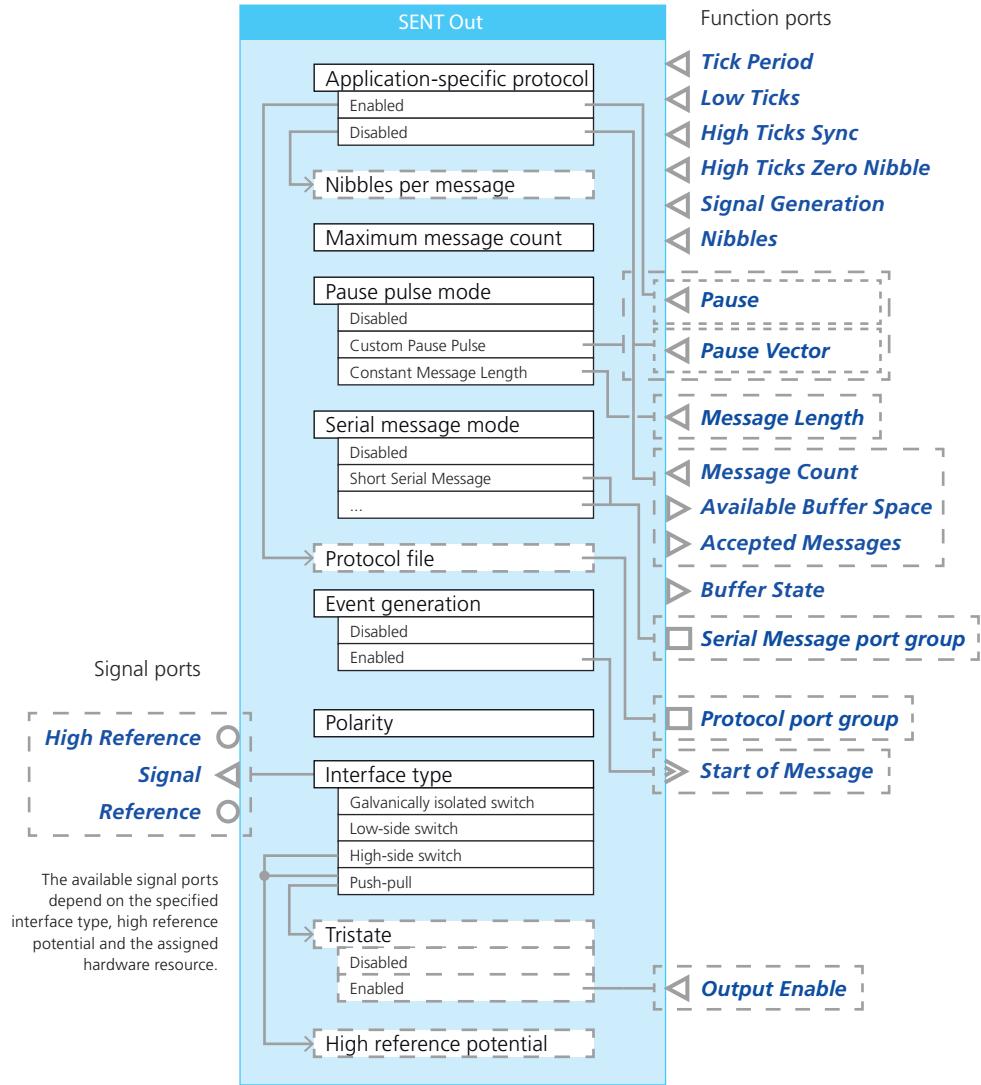
Information in this section

Overview of Ports and Basic Properties (SENT Out).....	1308
Overview of Tunable Properties (SENT Out).....	1320

Overview of Ports and Basic Properties (SENT Out)

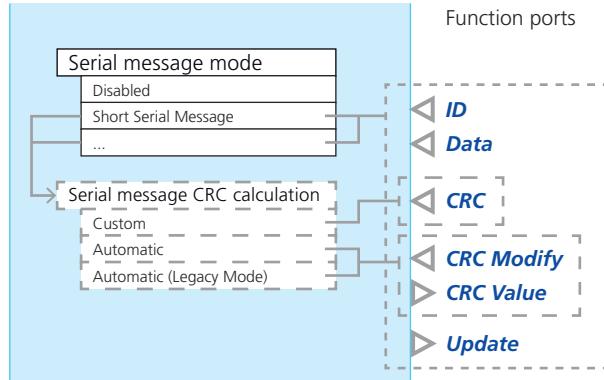
Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).

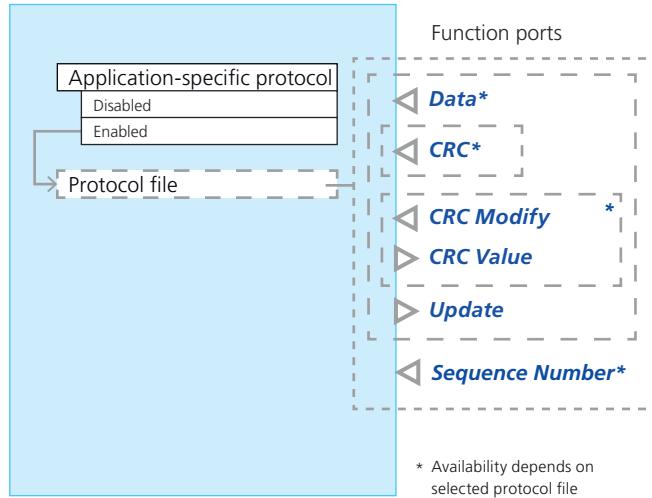


Serial Message port group The function ports of the Serial Message port group support the transmission of a serial message via a defined number of SENT messages. They are available only if the Serial Message Mode property is set to one of the available message formats or the application-specific protocol supports serial messages.

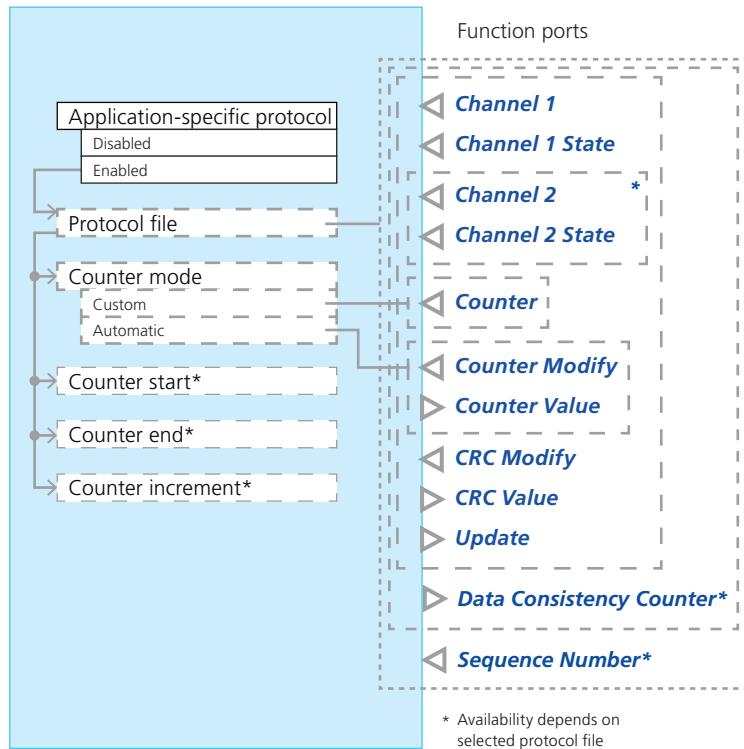
The following illustration shows the ports for serial messages if the Serial Message Mode property is set to one of the available message formats.



The following illustration shows the ports for serial messages if the application-specific protocol supports serial messages.



Protocol port group The function ports of the Protocol port group support the use of certain predefined data protocols (according to the SAE J2716 APR2016 SENT standard) for specific sensor applications. The Protocol port group is available only if the Application-specific protocol property is set to Enabled and a protocol file is selected via the Protocol file property.

**Tick Period**

This function import receives the current tick period (in seconds). A new value is taken at the beginning of the next message.

Value range	1 µs ... 200 µs
Dependencies	–

Low Ticks

This function import receives the number of tick periods used for a low pulse of every SENT pulse (nibble pulse, sync pulse, and pause pulse). A new value is considered at the beginning of the next SENT message.

Value range	1 ... 15 ticks
Dependencies	–

High Ticks Sync

This function import receives the number of tick periods used for the high part of the synchronization pulse. A new value is considered at the beginning of the next SENT message.

Value range	<ul style="list-style-type: none"> 0 ... 255 ticks 0: Synchronization pulse missing
Dependencies	–

High Ticks Zero Nibble

This function import receives the number of tick periods used for the high part of a nibble pulse with a value of 0 and for the high part of the pause pulse with a value of 0. A new value is considered at the beginning of the next SENT message.

Nibbles or pause pulses with values greater than 0 will result in a high pulse with the duration

High ticks zero nibble + Nibble

or

High ticks zero nibble + Pause

You can adapt the signal generation to different SENT sensor types or use the specified values to simulate failures.

Value range	1 ... 15 ticks
Dependencies	–

Signal Generation

This function import is used to enable/disable the generation of SENT messages by the behavior model. A new value is considered at the beginning of the next SENT message.

This function port can be used, for example, to simulate failures.

Value range	<ul style="list-style-type: none"> ▪ 1: Enabled. SENT messages which are generated by the behavior model are transmitted to the connected SENT receiver. ▪ 0: Disabled. SENT messages are not transmitted to the connected SENT receiver.
Dependencies	–

Nibbles

This function import receives nibble values from the behavior model. New values are considered at the beginning of the next SENT message.

Value range	<ul style="list-style-type: none"> ▪ -128 ... +127 ▪ 0 ... 15: Allowed value range for nibbles. <p>You can provide a nibble value outside the allowed range of 0 ... 15 to use these nibbles for failure simulation. For example: -128 = Missing nibble.</p> <ul style="list-style-type: none"> ▪ The vector size is the product of the Nibbles per message property value and the Maximum message count property value. ▪ The vector begins with the nibble of the first SENT message to be transmitted. For example, the vector {1; 2; 3; 4; 5; 6} with the property settings Nibbles per message = 3 and Maximum message count = 2 includes the nibble values of
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	two messages: Message 1 = {1; 2; 3} and message 2 = {4; 5; 6}.
Dependencies	Available only if the Application-specific protocol property is set to Disabled.

Pause

This function import receives a pause value from the behavior model. The value at this port is independent of the SENT message data content. New values are considered at the beginning of the next SENT message.

Value range	<ul style="list-style-type: none"> ▪ -32768 ticks ... +32767 ticks ▪ You can provide a negative pause value for failure simulation. For example: -32768 = Missing nibble.
Dependencies	Available only if the Pause pulse mode property is set to Custom Pause Pulse and the Application-specific protocol property is set to Enabled.

Pause Vector

This function import receives a vector with pause values from the behavior model for each SENT message that is provided with the Nibbles function port. The value at the Pause Vector function port is independent from the SENT message data content. New values are considered at the beginning of the next SENT messages that are provided by the behavior model.

Value range	<ul style="list-style-type: none"> ▪ -32768 ticks ... +32767 ticks ▪ You can provide negative pause values for failure simulation. For example: -32768 = Missing nibble. ▪ The vector size depends on the Maximum message count property.
Dependencies	Available only if the Pause pulse mode property is set to Custom Pause Pulse and the Application-specific protocol property is set to Disabled.

Message Length

This function import receives a message length value from the behavior model. The value at this port is evaluated from the SENT message data content to create SENT messages with a constant number of tick periods. New values are considered at the beginning of the next SENT message.

A pause pulse is generated if *Message Length Ticks* - (*Sync pulse ticks* + sum of all *Nibble pulse ticks*) is higher than 0. In this case a pause pulse is generated with a time length to reach the required message length. If the message length is greater than the value at the Message Length port, the pause pulse is not generated.

Value range	<ul style="list-style-type: none"> ▪ -32768 ... +32767 ▪ -32768: Missing pause pulse
Dependencies	Available only if the Pause pulse mode property is set to Constant Message Length.

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled ▪ 1: Enabled
Dependencies	Available only if the Interface type property is set to Push-pull and the Tristate property is set to Enabled.

Start of Message

This event port provides an I/O event when the function block starts to output the first SENT message of the message buffer.

Value range	—
Dependencies	Available only if the Event generation property is set to Enabled.

Message Count

This function import reads the number of SENT messages that can be forwarded to the message buffer from the behavior model during the current model step.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 8 ▪ Depends on the Maximum message count property and the Available Buffer Space function port.
Dependencies	Available only if the Application-specific protocol property is set to Disabled.

Available Buffer Space

This function outport provides the maximum number of SENT messages that can be written from the behavior model to the message buffer during the current model step.

Value range	0 ... 8
Dependencies	Available only if the Application-specific protocol property is set to Disabled.

Accepted Messages

This function outport provides the number of messages that were forwarded from the behavior model to the message buffer during the last model step.

Value range	0 ... 8
Dependencies	Available only if the Application-specific protocol property is set to Disabled.

Buffer State

This function outport provides status information on the message buffer.

Value range	<ul style="list-style-type: none"> ▪ 0: No error occurred. ▪ 1: Buffer underflow. During the last model step, the message buffer did not provide enough SENT messages to continuously transmit messages. To ensure a continuous transmission, the function block repeated the transmission of the last provided messages, starting with the first message. ▪ 2: Buffer overflow. During the last model step, too many SENT messages were provided to the message buffer. The remaining messages were discarded. <p>The number of discarded messages is the difference between the values of the Message count function port and the Accepted Messages function port.</p>
Dependencies	—

ID

This function import receives the value for the message ID of the serial message from the behavior model.

Value range	Depends on the selected serial message mode as follows: <ul style="list-style-type: none"> ▪ Short Serial Message: 0 ... 15 ▪ Enhanced Serial Message (4-bit ID, 16-bit data): 0 ... 15 ▪ Enhanced Serial Message (8-bit ID, 12-bit data): 0 ... 255
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats and no protocol file that specifies serial messages is selected.

Data

This function import receives the value for the data field of the serial message from the behavior model.

Value range	Depends on the selected serial message mode as follows: <ul style="list-style-type: none"> ▪ Short Serial Message: 0 ... 255 ▪ Enhanced Serial Message (4-bit ID, 16-bit data): 0 ... 65535 ▪ Enhanced Serial Message (8-bit ID, 12-bit data): 0 ... 4095
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats or a protocol file is selected that specifies a serial message without constant data.

CRC

This function import receives the value for the CRC checksum of the serial message provided by the behavior model.

Value range	0 ... 15
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats and the CRC calculation property is set to Custom .

CRC Modify (of Serial Message port group)

This function import receives a value, which is used to modify the CRC checksum value calculated by ConfigurationDesk. You can use this import, for example, to simulate faulty CRC calculations.

Value range	0 ... 15
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats and the CRC calculation property is set to Automatic or to Automatic (Legacy) .

CRC Value (of Serial Message port group)

This function outport provides the sum of the calculated checksum value and the value received via the **CRC Modify** function port calculated at run time from the system.

Value range	0 ... 15
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats and the CRC calculation property is set to Automatic or to Automatic (Legacy) .

Update (of Serial Message port group)

This function outport provides an update flag to the behavior model every time enough SENT messages have been sent to complete a serial message according to the SAE J2716 APR2016 SENT standard. For details, refer to [Configuring the Basic Functionality \(SENT Out\)](#) on page 1321.

Value range	<ul style="list-style-type: none"> ▪ 1: Update New data is taken and will be processed. ▪ 0: No Update Previous data is still processing, new data is ignored.
Dependencies	Available only if the Serial Message Mode property is set to one of the available message formats.

Sequence Number (of Serial Message port group)

This function import selects a sequence definition from the application-specific protocol file. A sequence defines the transmission order of serial messages in the data stream.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 15 ▪ The valid values depends on the sequences that are defined by the protocol file. Invalid values stops the transmission of SENT messages.
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property which requires the multiplexing of serial messages.

Channel 1 / Channel 2

This function import receives the value for the data field of the related channel from the behavior model.

Value range	Depends on the selected application-specific protocol. For specific values, refer to Hardware Dependencies (SENT Out) on page 1329.
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property. The Channel 2 function port is available for protocol files which require two channels.

Channel 1 State / Channel 2 State

This function import receives a diagnostic flag from the behavior model.

The value of Channel 1 State function port is mapped to bit 0 of the status nibble, the value of Channel 2 State function port to bit 1 of this nibble.

Value range	<ul style="list-style-type: none"> ▪ 0: OK. Channel x is OK ▪ 1: Error. Channel x is in error state.
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property. The Channel 2 function port is available for protocol files which require two channels.

CRC Modify (of Protocol port group)

This function import receives a value, which is used to modify the CRC checksum value calculated by ConfigurationDesk. You can use this import, for example, to simulate faulty CRC calculations.

Value range	0 ... 15
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property.

CRC Value (of Protocol port group)

This function outport provides the sum of the calculated checksum value and the value received via the CRC Modify function port calculated at run time from the system.

Value range	0 ... 15
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property.

Counter

This function import receives a rolling counter value which is provided from the behavior model.

Value range	0 ... 255
Dependencies	Available only if the following points are all fulfilled: <ul style="list-style-type: none"> ▪ The Application-specific protocol property is set to Enabled. ▪ A protocol file is selected by the Protocol file property, which requires a rolling counter. ▪ The Counter mode property is set to Custom.

Counter Modify

This function import receives a counter value, which is used to modify the rolling counter value provided by ConfigurationDesk. You can use this import, for example, to simulate faulty rolling counter values.

Value range	0 ... 15
Dependencies	Available only if the following points are all fulfilled: <ul style="list-style-type: none"> ▪ The Application-specific protocol property is set to Enabled. ▪ A protocol file is selected by the Protocol file property, which requires a rolling counter. ▪ The Counter mode property is set to Automatic.

Counter Value

This function outport provides the sum of the rolling counter value provided by ConfigurationDesk and the rolling counter value received via the Counter Modify function port calculated at run time from the system.

Value range	0 ... 15
Dependencies	Available only if the following points are all fulfilled: <ul style="list-style-type: none"> ▪ The Application-specific protocol property is set to Enabled. ▪ A protocol file is selected by the Protocol file property, which requires a rolling counter. ▪ The Counter mode property is set to Automatic.

Update (of Protocol port group)

This function outport provides an update flag to the behavior model every time new data can be sent via fast messages.

Value range	<ul style="list-style-type: none"> ▪ 1: Update New data is taken and will be processed. ▪ 0: No Update Previous data is still processing, new data is ignored.
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property.

Data Consistency Counter

This function outport provides the value of the data consistency counter that is incremented before the fast messages of a new data consistency group is sent. Fast messages of a data consistency group are sent with the same counter value to check which values of fast messages belong together.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 15 ▪ The valid range depends on the number of data consistency groups that are defined by the protocol file.
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property which requires the implementation of data consistency groups.

Sequence Number (of Protocol port group)

This function import selects a sequence definition from the application-specific protocol file. A sequence defines the transmission order of fast messages in the data stream.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 15 ▪ The valid values depends on the sequences that are defined by the protocol file. Invalid values stops the transmission of SENT messages.
Dependencies	Available only if the Application-specific protocol property is set to Enabled and a protocol file is selected by the Protocol file property which requires the multiplexing of fast messages.

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SENT Out) on page 1329
Dependencies	Available only if the port is supported by the settings of the Interface type and High reference potential properties and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SENT Out) on page 1329
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SENT Out) on page 1329
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Overview of Tunable Properties (SENT Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

	Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality			
Counter start	✓	-	-
Counter end	✓	-	-
Counter increment	✓	-	-
Function Ports			
Initial switch setting (Test Automation)	-	✓	-
Initial substitute value (Test Automation)	-	✓	-
Electrical Interface			
Polarity	✓	-	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (SENT Out)

Where to go from here

Information in this section

Configuring the Basic Functionality (SENT Out).....	1321
Using Application-Specific Protocols to Encode SENT Messages.....	1323
Encoding SENT Messages in the Behavior Model.....	1326
Configuring Standard Features (SENT Out).....	1327

Configuring the Basic Functionality (SENT Out)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the timing of a SENT message.
- Transmitting pause pulses.
- Encoding the SENT messages with one of the following methods:
 - [Using Application-Specific Protocols to Encode SENT Messages](#) on page 1323
 - [Encoding SENT Messages in the Behavior Model](#) on page 1326
- Transmitting serial data via SENT messages
- Transmitting faulty SENT messages.
- Providing I/O events.

Specifying the timing of a SENT message

To transmit SENT message, the sync pulse must be longer than the maximum possible duration of a nibble pulse:

$$\text{Minimal sync pulse length} \geq \text{Maximal nibble pulse length}$$

The timing of low pulses, zero nibble pulses and synchronization high pulses is defined by a number of tick periods. These tick periods are provided via separate function imports from the behavior model. Thus, the timing can be changed during run time, for example, to simulate faulty SENT messages or to adapt signal generation to different sensor types.

The tick period can also be changed by the behavior model during run time. This can be used to simulate a clockdrift.

Transmitting pause pulses

Every SENT message can contain a pause pulse transmitted at the end of the message after the CRC nibble and before the sync pulse of the next message.

The pause pulse can be used, for example, to create SENT messages with a constant number of tick periods.

To transmit pause pulses, you have to select the mode for the pause pulse generation via the **Pause pulse mode** property.

The pause value must be provided by the behavior model. You can select how to evaluate and provide the pause value:

- The pause value is evaluated from the SENT message data content to create SENT messages with a constant message length (= constant number of tick periods). In this case the pause value must be provided at the **Message Length** function import.
- The pause value provided by the behavior model is independent of the SENT message data content. In this case the pause value must be provided at the **Pause/ Pause Vector** function import.

Transmitting serial data via SENT messages

The function block can encode the first nibble of consecutive SENT messages to transmit serial messages. The encoding can be activated by setting the function block to a serial message mode or by importing a user-defined protocol that specifies serial messages.

According to the SAE J2716 APR2016 SENT standard, the **SENT Out** function block supports the following formats:

- *Short Serial Message (4-bit ID, 8-bit data, 4-bit CRC)*: A serial message frame comprises 16 SENT messages.
- *Enhanced Serial Message (4-bit ID, 16-bit data)*: A serial message frame comprises 18 SENT messages with the specified bit arrangement.
- *Enhanced Serial Message (8-bit ID, 12-bit data)*: A serial message frame comprises 18 SENT messages with the specified bit arrangement.

The function block codes the SENT messages with the serial data values when the SENT messages are written to the message buffer. If a buffer underflow occurs and the function block repeats the generation of fast messages, the serial message transmission becomes faulty. Therefore, set the buffer size to a value that avoids a buffer underflow or write the data values asynchronously. The **Buffer State** function port provides status information on the message buffer.

For more information on configuring the use of the message buffer and self-triggered systems, refer to one of the following topics:

- [Using Application-Specific Protocols to Encode SENT Messages](#) on page 1323
- [Encoding SENT Messages in the Behavior Model](#) on page 1326

Update functionality To ensure that only complete serial messages are transmitted to the SENT receiver, the function block counts the generated SENT messages. Depending on the selected serial message format, the function block provides an update flag via the **Update** function port, if the required number of SENT messages (16 or 18) has been transmitted.

Only after this update flag has been written to the behavior model, new data for the serial message is taken into account.

Information on the coding of serial data For basics on the coding of the SENT messages to transmit serial data, refer to [Basics on the SENT Protocol](#) on page 1256.

Transmitting faulty SENT messages

The SENT Out function block supports the simulation of missing nibbles in a SENT message, for example, to simulate failures during run time. If a nibble (low pulse and high pulse) is missing, the transmission is continued with the following nibble, a synchronization pulse or a pause pulse. If all the pulses of a SENT message are missing, the transmitter behaves as if Signal generation is disabled.

To simulate failures, the function block also supports the transmission of nibbles which have a value outside the valid range (0 ... 15). This valid range is specified in the SENT standard. For more information on nibble and pause values for faulty SENT messages, refer to [Basics on Simulating Faulty SENT Messages](#) on page 1306.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The SENT Out function block provides an I/O event when the function block starts to output the first message of the message buffer.

If event generation is enabled and if signal generation is disabled (either by disabling the Signal generation property or by transmitting a message with all pulses missing), event generation continues with a fixed period of 1 ms. Thus, the behavior model is able to re-enable signal generation again.

To use the I/O events in the behavior model, you have to assign them to a task via the Start of Message property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Using Application-Specific Protocols to Encode SENT Messages

Introduction

Each SENT message provides a sequence of nibble pulses to transmit data. You can use an application-specific protocol to evaluate the values of the nibble pulses. The behavior model writes the data to specific function ports: To the Channel 1/Channel 2 function ports for fast messages and to the Data function ports for serial channels. The function block evaluates the required SENT messages to transmit the values.

Protocol types

The following protocol types are supported:

- Predefined protocol files that can be used to support specific sensor applications. For more information, refer to [Using Application-Specific Protocols](#) on page 1260.
- User-defined protocol files that you created. For more information, refer to [Basics on User-Defined Protocol Files](#) on page 1262.

Importing an application-specific protocol

The Protocol file property lets you import a protocol file. To import a protocol file, you can enter its file path or you can open the standard Open dialog to browse for and select a protocol file.

There are no restrictions on the location of SENT protocol files. Predefined protocol files are located in the
`<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles` folder.

Changing user-defined protocols Each time you change a protocol file, you must reimport the changed file so that the changes become effective.

After a reimport, you must map and configure protocol-based function ports again.

Writing the data to the function block

The behavior model can write the data periodically with each model step or it can write the data asynchronously via I/O events. The Update function ports indicate which function ports can process new data.

Writing the data periodically At run time, the timer task periodically calls the SENT Out function block with each model step and writes the data values for transmission.

The SENT Out function block encodes the SENT messages with the provided data and writes the messages to the message buffer. To ensure a continuous transmission of SENT messages, the function block repeats the output of the buffered messages if a buffer underflow occurs until new values are available at the next model step.

To ensure that enough messages are buffered for one model step, you must specify the number of messages that can be written to the message buffer. You specify the buffer size with the Maximum message count property. The recommended setting can be calculated as follows:

$$\text{Maximum message count} > \frac{1.5 \cdot \text{Model step size}}{\text{Minimum message length}}$$

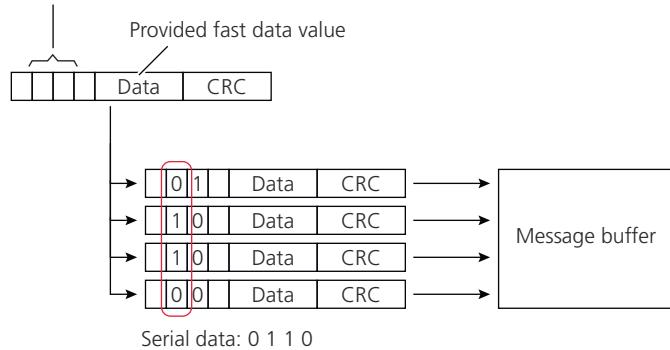
To calculate the minimum message length, add the ticks of all sync, nibble and pause pulses and multiply the value by the minimum tick period. All nibble pulses (status, data, CRC, etc.) represent the value 0 in this calculation unless a pause pulse ensures a constant message length. For messages with a constant length, use the constant length as minimum message length.

If you set the Maximum message count property much higher than required, new values can be written less often to the function block, but less likely a buffer underflow occurs.

However, the Channel function ports of the same port group read data only for one fast message. To fill the message buffer, data of fast messages is repeatedly written to the message buffer, but the status nibbles, CRC nibbles, optional counter nibbles, and optional DCC nibbles are updated. This makes the transmission of serial messages and multiplexed fast messages possible.

The following example shows you the principle of writing SENT messages to the message buffer. The protocol specifies one fast channel and a serial channel. The SENT messages in the buffer have the same data nibbles, but the function block codes the first nibble of the messages with the data values of the serial message.

Bits for serial data transmission



Writing SENT messages asynchronously The writing of SENT messages asynchronously to the model steps requires a self-triggered system: The I/O events of the SENT Out function block trigger a runnable function, and the task of the runnable function triggers the model port block that reads the provided data from the function block. To use this method, you have to enable event generation for the function block.

The I/O event of the SENT Out function block is generated when the function block starts to output the first message of the message buffer.

In a self-triggered system, set the Maximum message count property to 1 to transmit the latest data.

For information on providing I/O events, refer to [Configuring the Basic Functionality \(SENT Out\)](#) on page 1321.

More information on the message buffer For more information on the message buffer, refer to [Basics on the Message Buffer](#) on page 1304.

Related topics

Basics

Using Application-Specific Protocols.....	1260
-------------------------------------------	------

Encoding SENT Messages in the Behavior Model

Introduction

The encoding of SENT messages in the behavior model is the most flexible method to transmit data via SENT communication. In the behavior model, the value of each nibble must be evaluated and provided to the **SENT Out** function block.

Specifying number of nibbles

The number of nibbles is constant for all the SENT messages of one specific SENT application, but can vary between different applications. The number of nibbles includes all nibbles, such as data nibbles, the status nibble, and the CRC nibble.

To adapt the **SENT Out** function block to the connected SENT receiver, you have to use the same number of nibbles as used by the receiver.

Providing the nibble values for transmission

The SENT specification requires the continuous transmission of messages. To provide the required nibble values, the behavior model can write the nibble values of several SENT messages periodically or asynchronously via I/O events.

Writing nibble values periodically At run time, the timer task periodically calls the **SENT Out** function block with each model step and reads the data values for transmission. To avoid a buffer underflow, always provide the maximum number of messages to the function block that can be handled. Specify the number of messages that can be handled via the **Maximum message count** property:

$$\text{Maximum message count} > \frac{1.5 \cdot \text{Model step size}}{\text{Minimum message length}}$$

To calculate the minimum message length, add the ticks of all sync, nibble and pause pulses and multiply the value by the minimum tick period. All nibble pulses (status, data, CRC, etc.) represent the value 0 in this calculation unless a pause pulse ensures a constant message length. For messages with a constant length, use the constant length as minimum message length.

If you set the **Maximum message count** property much higher than required, it takes more time to provide new values, but a buffer underflow is less likely.

To avoid a buffer overflow, observe the **Available Buffer Space/Accepted Messages** function ports and use the **Message Count** function port to specify a number of provided messages that matches the available buffer space.

The following example shows the provided information and the generated output signal if nibble values for three SENT messages are provided from the behavior model per model step.

Model step	1	2	3	4	
Available buffer space	3	3	3	3	3
Accepted messages	0	3	3	3	3
Buffer state	Ok	Ok	Ok	Ok	Ok
Provided messages	3 (1;2;3)	3 (4;5;6)	3 (7;8;9)	3 (10;11;12)	3
Generated messages	1	2	3	4	5

For more information on the timing of SENT messages, refer to [Basics on the Message Buffer](#) on page 1304.

Writing nibble values asynchronously Asynchronously writing of nibble values requires a self-triggered system: The I/O events of the SENT Out function block trigger a runnable function and the task of the runnable function must trigger the model port block that writes the new nibble values to the function block. To use this method, you have to enable event generation for the function block.

The I/O event of the SENT Out function block is generated when the function block starts to output the first message of the message buffer. Set the Maximum message count property to 1 to transmit only the most recent nibble values.

For information on providing I/O events, refer to [Configuring the Basic Functionality \(SENT Out\)](#) on page 1321.

Related topics

Basics

[Basics on the Message Buffer](#)..... 1304

Configuring Standard Features (SENT Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Digital Out 1	Digital Out 3	Digital InOut 5	Flexible Out 1	Further Information
Interface type	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<p>Depending on the channel type, refer to:</p> <ul style="list-style-type: none"> ▪ Specifying Digital Output Signals (Digital Out 1) on page 102 ▪ Specifying Digital Output Signals (Digital Out 3, Digital In/Out 3) on page 109 ▪ Specifying Digital Output Signals (Digital In/Out 5) on page 112 ▪ Specifying Digital Output Signals (Flexible Out 1) on page 98
High impedance (tristate)	Configurable in push-pull configuration to disconnect the digital outputs completely by setting the channel to high impedance (tristate) from the behavior model at run time.				–
Polarity of output signals	✓	✓	✓	✓	–
Channel multiplication	–	–	–	–	<p>Specifying Current and Voltage Values for Channel Multiplication on page 95</p> <p>Note: The SENT Out function block type does not support channel multiplication. Therefore the voltage and current ranges of a single hardware channel apply.</p>
Signal Ports					
Trigger level of electronic fuse	–	–	–	✓	Specifying Fuse Settings on page 122
Failure simulation support	✓	–	–	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital Out 1 on page 1575](#)
- [Digital Out 3 on page 1577](#)
- [Digital In/Out 5 on page 1586](#)
- [Flexible Out 1 on page 1604](#)

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88

Configuration Feature	Further Information
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Note

The support of stop values is limited to the output of the last run time value for the SENT Out function block. Stop values are output when the application status changes from running to stopped.

- Leave the setting of the Stopped status output property at Keep last run-time values.

Hardware Dependencies (SENT Out)

SCALEXIO Hardware Dependencies (SENT Out)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 3	Digital In/Out 5
Hardware	DS2680 I/O Unit	DS2621 Signal Generation Board	DS6101 Multi-I/O Board	DS6202 Digital I/O Board
Event generation	✓	✓	✓	✓
Interface type for digital outputs	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Galvanically isolated switch ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ Low-side switch ▪ High-side switch to internal reference ▪ Push-pull
In push-pull configuration the outputs can be set to high impedance (tristate) at run time.				
Output current range	0 ... +80 mA _{RMS}	0 ... +40 mA _{RMS}	<ul style="list-style-type: none"> ▪ 0 ... +140 mA_{RMS} (each channel) ▪ To avoid overloading on VBAT pins, the total load on each VBAT pin must not exceed 1 A_{RMS}. Connect the two VBAT pins in parallel if the maximum current of all the channels exceeds 1 A_{RMS}. 	0 ... ±40 mA (each channel)

Channel Type	Digital Out 1	Flexible Out 1	Digital Out 3	Digital In/Out 5
High-side reference voltage	+5 V ... +60 V	-60 V ... +60 V	+5 V ... +60 V	+3.3 V and +5 V
Configurable fuse	–	<ul style="list-style-type: none"> ▪ Load side ▪ 5 mA_{RMS} ... 40 mA_{RMS} 	–	–
Circuit diagrams	Digital Out 1 on page 1575	Flexible Out 1 on page 1604	Digital Out 3 on page 1577	Digital In/Out 5 on page 1586
Required channels	1	1 (additional channels are required for operation in push/pull mode.)	1	1

General limitations

The SENT Out function block does not support channel multiplication.

DS6202 Digital I/O Board Only 8 SENT Out function blocks can be assigned to one DS6202 Digital I/O Board.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2621 Signal Generation Board For more board-specific data, refer to [Data Sheet of the DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6101 Multi-I/O Board For more board-specific data, refer to [Data Sheet of the DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS6202 Digital I/O Board For more board-specific data, refer to [Data Sheet of the DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

SPI Master

Where to go from here

Information in this section

Introduction (SPI Master).....	1332
Overviews (SPI Master).....	1336
Configuring the Function Block (SPI Master).....	1342
Hardware Dependencies (SPI Master).....	1350

Introduction (SPI Master)

Where to go from here

Information in this section

- | | |
|--------------------------------------------------------------------------|------|
| Introduction to the Function Block (SPI Master)..... | 1332 |
| Basics on Serial Peripheral Interfaces (SPI Master)..... | 1333 |

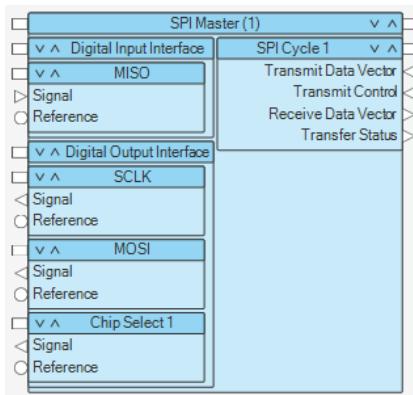
Introduction to the Function Block (SPI Master)

Function block purpose

The SPI Master function block controls and performs a short-distance communication via the serial peripheral interface (SPI). SPI communication is a master-slave architecture with a single master.

Default display

The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Sending data provided from within the behavior model to selected SPI slaves.
- Receiving data from selected SPI slaves and providing them to the behavior model.
- Providing the configuration features for the data transfer for each SPI slave.
- Providing information on the status of the data transfer for each SPI slave.
- Generating and providing I/O events at the end of each data transfer.

Supported channel types

The SPI Master function block type supports the following channel types:

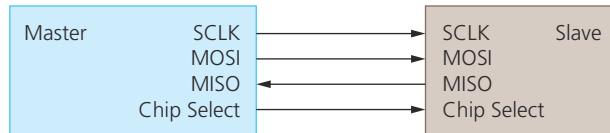
	SCALEXIO	MicroAutoBox III		
Channel type	–	Digital In 4	Digital Out 4	Digital In/Out 6
Hardware	–	▪ DS1511 ▪ DS1511B1 ▪ DS1513	▪ DS1552 ▪ DS1552B1	

Basics on Serial Peripheral Interfaces (SPI Master)

SPI topology

The SPI is a synchronous serial interface for short-distance communication. It allows for full-duplex communication between an SPI master and a selected SPI slave. Communication via SPI requires a dedicated chip select signal for each SPI slave plus three shared signals for clock and data transfer to which the SPI master and all the SPI slaves are connected.

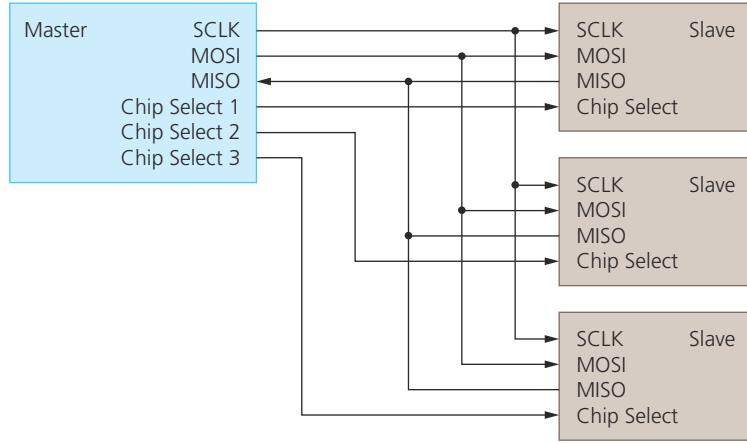
The following illustration shows the basic configuration with an SPI master connected to one SPI slave:



SPI Signal	Full Name	Purpose
SCLK	Serial clock	To synchronize the data transmission.
MOSI	Master Output Slave Input	To send data from the SPI master to an SPI slave.
MISO	Master Input Slave Output	To receive data from an SPI slave to the SPI master.
CS	Chip Select	To select the SPI slave for the point-to-point communication.

The SPI master generates the output signals according to the requirements of the selected SPI slave.

The following illustration shows an SPI master and three slaves connected in a star topology. The SPI master provides a dedicated chip select line for each SPI slave.

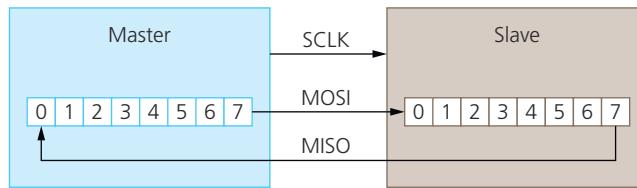
**Data transfer principle**

The data transfer is similar to shift registers working as a circular buffer. Therefore, the size of both registers must be the same and equal one data word.

The data transfer is processed in an SPI cycle as follows:

1. The SPI master selects an SPI slave.
2. The SPI master starts the transmission with the output of the first clock pulse.
3. During each clock cycle:
 - The SPI master shifts its register, outputs a bit value to the MOSI signal port, and reads a bit value from the MISO signal port.
 - The SPI slave shifts its register, outputs a bit value to the MISO signal port, and reads a bit value from the MOSI signal port.

The following illustration shows the principle: All the bits of an 8-bit data word are exchanged after eight clock cycles.



This sequence is also maintained when only a one-directional data transfer is intended.

4. After the word is transmitted, the SPI devices save the received data words to their receive buffers. To transmit the next word, the SPI devices copy new data words from their transmit buffers to the shift registers.
5. The SPI master pauses the clock generation between the transmission of consecutive words to ensure that the SPI slave is ready to transmit the next word.
5. Depending on the configuration, the SPI master deselects the SPI slave after all words are exchanged or after the transmission of every data word.

Data transfer setup

There are features, such as data format or SPI mode, that are not determined by the SPI standard. They must be specified for the SPI master according to the requirements of the connected SPI slaves.

The transfer of data of a specific amount and format between SPI master and a specific SPI slave is referred to as an *SPI cycle*. Specifications, for example, for data format, transmission speed, or chip select signal activation, are organized in an SPI Cycle function for each SPI cycle.

Oversviews (SPI Master)

Where to go from here

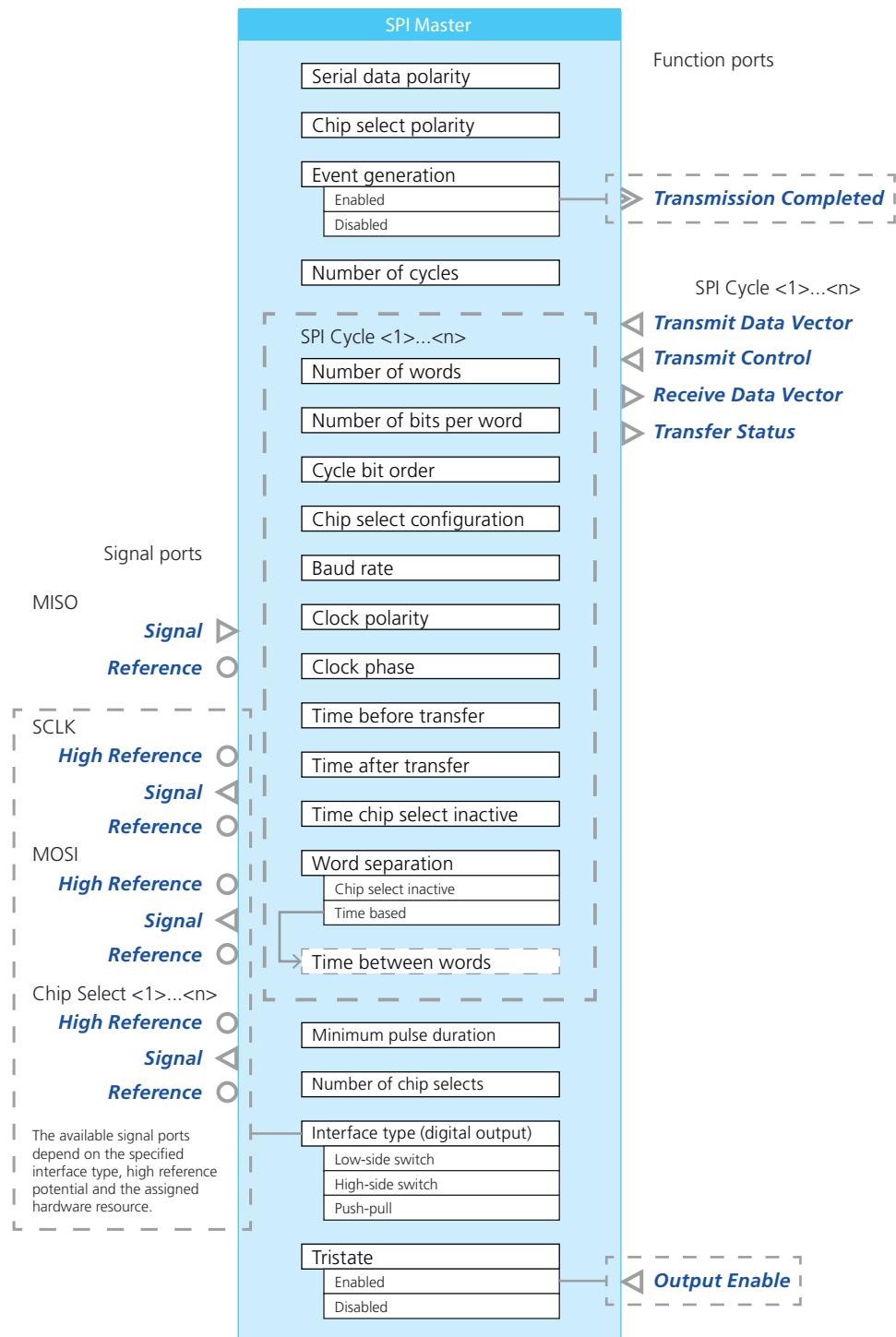
Information in this section

Overview of Ports and Basic Properties (SPI Master).....	1336
Overview of Tunable Properties (SPI Master)	1341

Overview of Ports and Basic Properties (SPI Master)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Transmission Completed

This event port provides an I/O event each time an SPI cycle is completed.

Value range	–
Dependencies	Available only if the Event generation property is set to Enabled.

Transmit Data Vector

This function import reads the data to be transmitted to the SPI slave from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry of the vector. ▪ The vector size depends on the Number of words and Number of bits per word properties. ▪ The bits of each word to be transmitted are represented via data bytes. For example, the property settings Number of words = 2 and Number of bits per word = 10 result in a data vector size of 4 bytes. Each word is represented by two bytes: The first byte for the bits 0 ... 7 and the second byte for the bits 8 and 9. ▪ The vector size is limited by the buffer size. <p>Refer to Configuring the Basic Functionality (SPI Master) on page 1342.</p>
Dependencies	–

Transmit Control

This function import enables its related SPI Cycle function to execute an SPI cycle for data transmission.

Value range	<ul style="list-style-type: none"> ▪ 0: Disabled The related SPI Cycle function is disabled. ▪ 1: Enabled The related SPI Cycle function is enabled. As long as the SPI Cycle function is enabled, a new SPI cycle starts with every model step.
Dependencies	–

Receive Data Vector

This function outport writes the data received from the SPI slave to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 255 for each entry of the vector. ▪ The vector size depends on the Number of words and Number of bits per word properties. ▪ Each received word is represented via data bytes. For example, the property settings Number of words = 2 and Number of bits per word = 10 result in a data vector size of 4 bytes. Each word is represented by two bytes: The first
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>byte for the bits 0 … 7 and the second byte for the bits 8 and 9.</p> <ul style="list-style-type: none"> The vector size is limited by the buffer size. Refer to Configuring the Basic Functionality (SPI Master) on page 1342.
Dependencies	–

Transfer Status

This function outport writes status information on the Data Vector (Transmit) function port of the related SPI Cycle function to the behavior model.

Value range	<ul style="list-style-type: none"> 0 … 15 0: Indicates that the function block successfully sent the data to the SPI slave and successfully captured the data from the SPI slave. The decimal value represents a 4-bit binary value. The following failure states or combinations of them are output: 1 (Bit 0 = 1): Indicates that the function block did not process the last request to transmit data values to the SPI slave. 2 (Bit 1 = 1): Indicates that the function block did not transmit any of the serial data values of the previous SPI cycle to the SPI slave. The values to be sent are lost and overwritten by new values of the current SPI cycle. 4 (Bit 2 = 1): Indicates that the function block cannot provide new data values to the behavior model, because the acquisition of new data values of the previous SPI cycle is not completed or no new data values were sent. 8 (Bit 3 = 1): Indicates that the function block overwrote received data values with newer values from the SPI slave before the behavior model read the old values. <p>Combinations of bits are set to indicate more than one failure state at a time. To get the specific failure states, you must evaluate the value of the Transfer Status function port in the behavior model.</p>
Dependencies	–

High Reference

This signal port is a reference port and provides the high-side reference for the Signal signal port when the digital outputs are operating as high-side switch or in push-pull configuration.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SPI Master) on page 1350.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SPI Master) on page 1350.
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (SPI Master) on page 1350.
Dependencies	Available only if the port is supported by the setting of the Interface type property and the assigned hardware resource.

Output Enable

This function import enables the signal ports of the function block from within the behavior model. If the signal ports are disabled, they are in a high impedance state. The enabling and disabling is independent of the current states of the function block's inputs and internal conditions.

Value range	<ul style="list-style-type: none">▪ 0: Disabled▪ 1: Enabled
Dependencies	Available only if the Tristate property is set to Enabled.

Overview of Tunable Properties (SPI Master)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Serial data polarity	✓	-
Chip select polarity	✓	-
Cycle bit order	✓	-
Chip select configuration	✓	-
Baud rate	✓	-
Clock polarity	✓	-
Clock phase	✓	-
Time before transfer	✓	-
Time after transfer	✓	-
Time chip select inactive	✓	-
Word separation	✓	-
Time between words	✓	-
Function Ports		
Initial switch setting (Test Automation)	-	✓
Initial substitute value (Test Automation)	-	✓
Electrical Interface		
Threshold	✓	-
Minimum pulse duration	✓	-

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (SPI Master)

Where to go from here

Information in this section

Configuring the Basic Functionality (SPI Master).....	1342
Configuring SPI Cycles (SPI Master).....	1343
Configuring Standard Features (SPI Master).....	1348

Configuring the Basic Functionality (SPI Master)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the number of SPI Cycle functions.
- Specifying the number of chip select signals.
- Conditioning input and output signals.
- Configuring SPI cycles. Refer to [Configuring SPI Cycles \(SPI Master\) on page 1343](#).
- Providing I/O events.

Specifying the number of SPI Cycle functions

An SPI Cycle function provides the configuration for data transmission between an SPI master and a specific SPI slave.

The number of required SPI Cycle functions depends on the number of SPI slaves that are connected to the SPI master. You have to specify at least one SPI Cycle function for each SPI slave. Additional SPI Cycle functions can be specified to support SPI slaves with different SPI cycle configurations.

Specifying the number of chip select signals

You have to specify chip select signals so that each connected SPI slave can be selected individually. The required number of chip selects signals depends on the number of slaves and the topology used to connect SPI master and slaves. For details on SPI topologies, refer to [Basics on Serial Peripheral Interfaces \(SPI Master\) on page 1333](#).

Conditioning input and output signals

The SPI Master function block provides the following features to condition the signals.

Setting signal polarity For the MOSI and MISO serial data signals and the chip select signals, the polarities are set commonly for all SPI slaves via the Serial data polarity and the Chip select polarity properties.

You can set the polarities of the signals to active high or active low.

- Active high:

A high voltage level represents a binary 1 at the MOSI/MISO signal ports or the active state of Chip select signal ports.

- Active low:

A low voltage level represents a binary 1 at the MOSI/MISO signal ports or the active state of Chip select signal ports.

The polarity of the clock signal is set individually for each SPI Cycle function via the Clock polarity property.

Filtering the MISO signal You can specify a pulse filter that evaluates the pulse duration. Pulses that have a duration shorter than specified at the Minimum pulse duration property are ignored.

For specific values, refer to [Hardware Dependencies \(SPI Master\)](#) on page 1350.

Providing I/O events

If Event generation is enabled, the function block generates I/O events to use them as a trigger source for runnable functions in the behavior model.

The SPI Master function block can generate an I/O event each time the transmission of serial data for an SPI cycle is completed.

To use the I/O events in the behavior model, you have to assign them to a task via the Transmission Completed property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Configuring SPI Cycles (SPI Master)

Overview

Each SPI Cycle function provides the following configuration features:

- Selecting the SPI slave.
- Specifying the time of data capturing.
- Specifying the SPI frame.
- Specifying the bit order for the data transfer.
- Specifying the timing of the SPI signals.

Selecting the SPI slave

For each SPI Cycle function, you have to define, which chip select lines are activated during the SPI cycle.

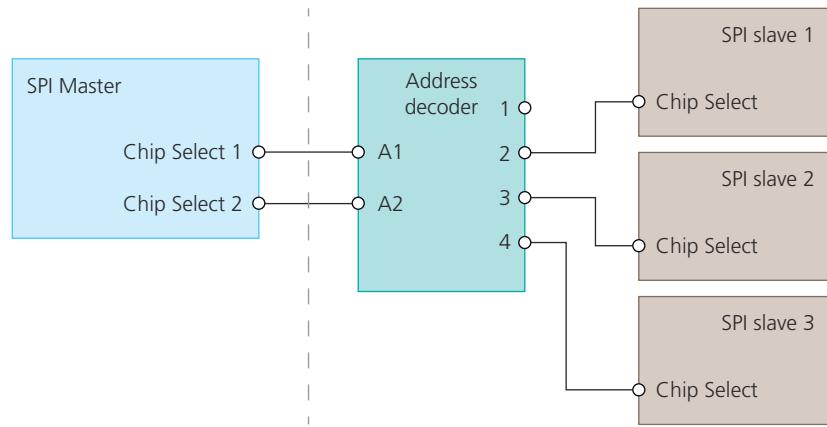
Up to four chip select signals let you select an SPI slave for communication. For chip select lines the polarity is often set to Active low. You can connect the SPI slaves as follows:

- Direct connection of chip select outputs to SPI slaves:

This lets you connect up to four SPI slaves to the SPI master. For this topology, you have to ensure that only one chip select output is activated at a time. For an example topology, refer to [Basics on Serial Peripheral Interfaces \(SPI Master\)](#) on page 1333.

- Connection of chip select outputs and SPI slaves via an additional address decoder (e.g., CD4428, SN74138, or SN74154):

For this topology, you have to specify an address with the chip select configuration. The address decoder decodes the specified address to select the desired SPI slave. Refer to the following example:



To select, for example, SPI slave 3, you have to activate chip select lines 1 and 2 via the **Chip select configuration** property.

The lowest decoder output representing address 0 (zero) cannot be used, because the function block uses address 0 to deselect SPI slaves. In total, up to 15 SPI slaves can be selected via an external address decoder by using all the four available chip select lines.

Note

Spikes affect SPI slaves

The chip select signals might not arrive synchronously at the address decoder inputs although all chip select outputs are synchronously switched. The address decoder might therefore produce spikes at its outputs that influence the SPI slaves.

- Avoid effects of spikes by increasing the values of the **Time before transfer** and/or **Time after transfer** properties.

A conflict is generated if you try to activate chip select lines that are not available because they are not specified in the **Number of chip selects** property.

Specifying the time of data capturing

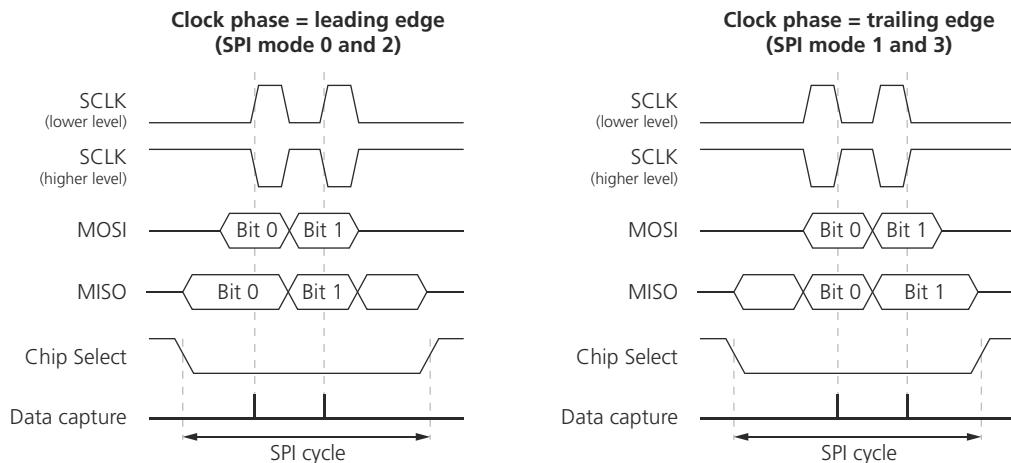
For each SPI Cycle function you have to specify the polarity and the phase of the clock signal (SCLK). The phase of the clock signal defines whether the SPI master captures the data bits from the MISO signal port at the leading (CPHA = 0) or trailing (CPHA = 1) edge of each clock pulse. If the SPI master captures data from the MISO signal port at the leading edge of a clock pulse, it writes data to the MOSI signal port at the falling edge of that clock pulse and vice versa.

The polarity of the clock signal defines whether the inactive clock signal is the lower (CPOL = 0) or the higher (CPOL = 1) voltage level. Depending on the polarity, the leading edge is a rising or falling edge. The polarity does not influence the point in time of data capturing.

The combinations of phase and polarity settings of the clock signal are referred to as SPI modes:

SPI Mode	Clock Polarity	Clock Phase
0	Lower level	Leading edge
1	Lower level	Trailing edge
2	Higher level	Leading edge
3	Higher level	Trailing edge

The following illustration shows the data transfer process of one 2-bit word in one SPI cycle for different SPI modes.



Specifying the timing of the SPI signals

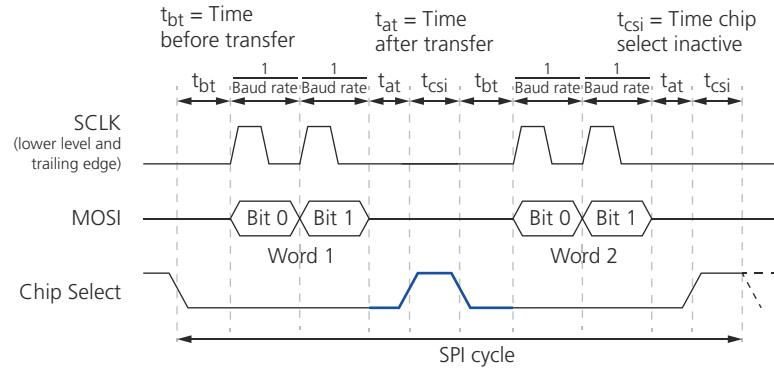
For each SPI Cycle function, you have to specify the baud rate and the timing of the clock signal and data signals according to the requirements of the selected SPI slave. The actual baud rate might differ from the specified baud rate and can be calculated as follows:

$$\text{Actual baud rate} = 20 \text{ MHz} / \text{floor}(20 \text{ MHz} / \text{specified baud rate} + 0.5).$$

To transmit multiple words with an SPI cycle, you must use one of the following methods to separate consecutive data words.

- Setting the chip select signal inactive between words.
- Time-based separation of words with a transmission pause.

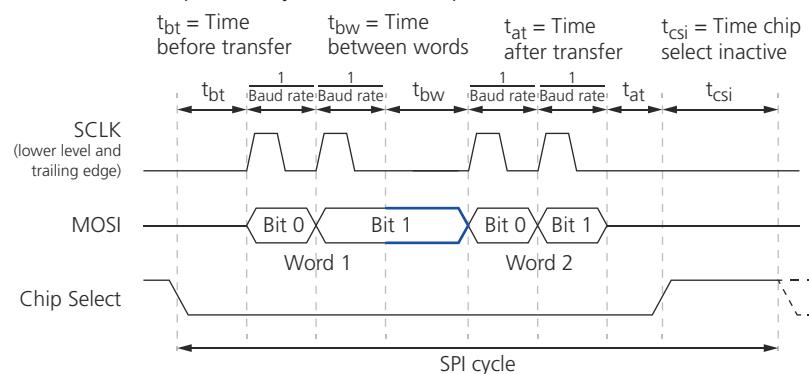
Signal timing with inactive chip select signal between words The following illustration shows how the timing-related properties influence the output signals. Consecutive data words are separated by an inactive chip select signal.



Note

- Times are also inserted between consecutive data words according to the settings of the Time after transfer, Time chip select inactive, and Time before transfer properties.
- Between the words of an SPI cycle, the MOSI channel is set to high impedance.

Signal timing with time-based word separation The following illustration shows how the timing-related properties influence the output signals. The words are time-based separated by a transmission pause between the words.



Note

Between the words of an SPI cycle, the MOSI channel outputs the last bit value.

Timing resolution The maximum value of each timing related property is 793.6 µs. The value range is internally separated into twelve intervals that are automatically assigned to the specified value. Each interval provides a different step size that is used to saturate a specified value to its next available value. For example, if the value of a timing related property is specified to 645 ns, interval 2 with a lower limit of 640 ns and a step size of 20 ns is assigned. As a result, the specified value of 645 ns is internally processed as 660 ns. The following table shows the twelve intervals and their related limits and step sizes:

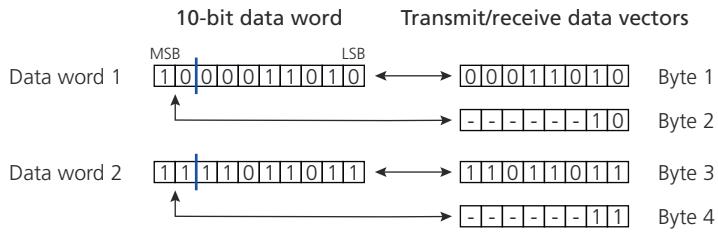
Interval Number	Lower Limit of Interval	Step Size
1	0 ns	10 ns
2	640 ns	20 ns
3	1.28 µs	40 ns
4	2.56 µs	80 ns
5	5.12 µs	160 ns
6	10.24 µs	320 ns
7	20.48 µs	640 ns
8	40.96 µs	1.28 µs
9	81.92 µs	2.56 µs
10	163.84 µs	5.12 µs
11	327.68 µs	10.24 µs
12	655.36 µs	20.48 µs

Specifying the SPI data frame

For each SPI Cycle function, you have to specify an SPI data frame which defines, how many data words with a specific number of bits are exchanged per SPI cycle.

You have to specify the data frame via the Number of words and the Bits per word properties according to the selected SPI slave.

Effects on the data function ports The specification of the data frame affects the vector size of the Transmit Data Vector and Receive Data Vector function ports. These function ports use vectors with data bytes to handle the words for one SPI cycle. The following illustration shows the assignment of 10-bit data words to the data vectors of the function ports, for example:



The following rules apply:

- Data words with more than eight bits are split to as many data bytes of the vector as required.
- If the number of bits per word is unequal eight or its multiple, the data byte of the vector carrying the most significant bits of the data word includes unused bits with undefined state. These bits have to be ignored for further processing.

Limited buffer size The buffer size is specified by the Number of words and Number of bits per word properties and can be calculated as follows:

$$\text{Buffer size} = \text{floor} ((\text{Number of bits per word} + 15) / 16) \cdot 16 \cdot \text{Number of words}.$$

The required buffer size must not exceed 2048 bits, otherwise a conflict is generated and displayed in the Conflicts Viewer.

Specifying the bit order for the data transfer

For each SPI Cycle function, you can specify one of the following bit orders for transferring data words:

- MSB to LSB: Transfer data starting with the most significant bit of each word.
- LSB to MSB: Transfer data starting with the least significant bit of each word.

This setting does not affect the bit order of the data bytes at the Transmit Data Vector and Receive Data Vector function ports.

Configuring Standard Features (SPI Master)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The functionality of the electrical interface depends on the channel type. The following table lists the available channel types for signal input and signal output with their configuration features. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature		Channel Type Combination		Digital In/Out 6	Further Information
		Digital In 4	Digital Out 4		
Digital input signals	Interface type	Ground-based	–	Ground-based	Specifying the Circuit Type for Voltage Input Signals on page 116
Digital output signals	Interface type	–	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	<ul style="list-style-type: none"> ▪ High-side switch ▪ Low-side switch ▪ Push-pull 	–
	Setting to high impedance (tristate)	–	To disconnect the digital outputs by setting the channel to high impedance (tristate) from the behavior model during run time.		–

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Digital In 4 on page 1572](#)
- [Digital Out 4 on page 1578](#)
- [Digital In/Out 6 on page 1588](#)

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (SPI Master)

MicroAutoBox III Hardware Dependencies (SPI Master)

Characteristics depending on channel types The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type.

Channel Type	Digital In 4/Digital Out 4	Digital In/Out 6
Hardware	<ul style="list-style-type: none"> ▪ DS1511 Multi-I/O Board ▪ DS1511B1 Multi-I/O Board ▪ DS1513 Multi-I/O Board 	<ul style="list-style-type: none"> ▪ DS1552 Multi-I/O Module ▪ DS1552B1 Multi-I/O Module
Event generation	✓	✓
Digital Input Interface (MISO)	Channel type	Digital In 4
	Input voltage range	0 ... +40 V
	Threshold	Input threshold voltage
		+2 V (typ.)
		Input hysteresis voltage
	Pulse filter	1 V (typ.)
	Minimum pulse duration	0.7 V (typ.)
	Supported interface type	12.5 ns ... 12.75 µs
Digital Output Interface (SCLK, MOSI, Chip Select <1> ... <n>)	Supported interface type	Ground-based
	Circuit diagram	Ground-based
	Required channels	Digital In 4 on page 1572
		Digital In/Out 6 on page 1588
	Required channels	1
	Channel type	1
	Baud rate range	Digital In/Out 6
	High-side reference voltage	306 Bd ... 2 MBd
	Supported Interface type	+4.5 V ... +40 V
		+5 V
		▪ Low-side switch
		▪ High-side switch
		▪ Push-pull
	The outputs can be set to high impedance (tristate) during run time.	▪ Low-side switch
	Circuit diagram	High-side switch
	Required channels	Push-pull
		The outputs can be set to high impedance (tristate) during run time.
		Digital Out 4 on page 1578
		Digital In/Out 6 on page 1588
	Required channels	3 ... 6 ¹⁾
		3 ... 6 ¹⁾

¹⁾ Depends on the number of chip select lines.

More hardware data

DS1511/DS1511B1 Multi-I/O Board For more board-specific data, refer to DS1511 Multi-I/O Board Data Sheet ([MicroAutoBox III Hardware Installation and Configuration](#) ).

DS1513 Multi-I/O Board For more board-specific data, refer to DS1513 Multi-I/O Board Data Sheet ([MicroAutoBox III Hardware Installation and Configuration](#) ).

DS1552/DS1552B1 Multi-I/O Module For more module-specific data, refer to DS1552 Multi-I/O Module Data Sheet ([MicroAutoBox III Hardware Installation and Configuration](#) ).

Gigalink

Where to go from here

Information in this section

Introduction (Gigalink).....	1354
Overviews (Gigalink).....	1355
Configuring the Function Block (Gigalink).....	1357

Introduction (Gigalink)

Introduction to the Function Block (Gigalink)

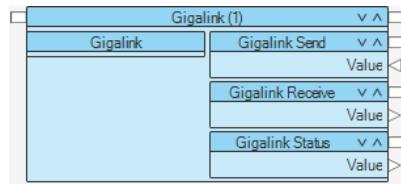
Function block purpose

The Gigalink function block type realizes communication between a SCALEXIO system and a PHS-bus based system (DS1006 or DS1007 modular system) or between two SCALEXIO systems.

A real-time application running on a SCALEXIO platform can send signals to and receive signals from another real-time application via Gigalink connection. Gigalink allows bidirectional serial data transmission at high speed.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

The main features are:

- Sending and receiving data via Gigalink connection.
- Sending and receiving Gigalink events for synchronization.
- Providing diagnostic information to the behavior model.

Overviews (Gigalink)

Where to go from here

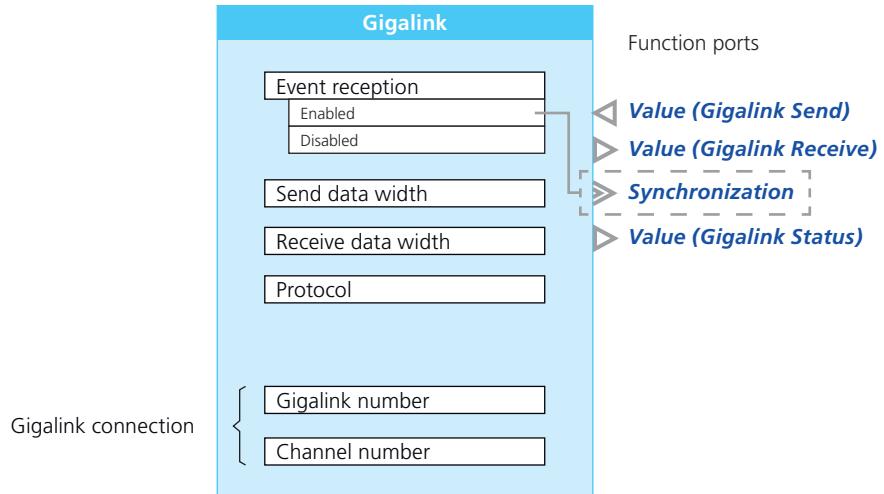
Information in this section

- | | |
|--------------------------------------------------------|------|
| Overview of Ports and Basic Properties (Gigalink)..... | 1355 |
| Overview of Tunable Properties (Gigalink)..... | 1356 |

Overview of Ports and Basic Properties (Gigalink)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Value (Gigalink Send)

This function import gets data from the behavior model and sends it to the connected Gigalink member.

Value range	-[Double.max] ... +[Double.max]
Dependencies	The vector size depends on the number of signals specified at the Send data width property. Different Send data width values between this function port and the Data Outport block in the behavior model are indicated by an orange mapping line (if the View - Highlight option is enabled).

Value (Gigalink Receive)

This function outport receives data from the connected Gigalink member and transmits it to the behavior model.

Value range	-[Double.max] ... +[Double.max]
Dependencies	The vector size depends on the number of signals specified at the Receive data width property. Different Receive data width values between this function port and the Data Import block in the behavior model are indicated by an orange mapping line (if the View - Highlight option is enabled).

Synchronization

This event port provides an I/O event each time a Gigalink event has been received. The received events can be used to trigger a task that is executed synchronously on the connected systems.

Value range	–
Dependencies	Available only if the Event reception property is set to Enabled.

Value (Gigalink Status)

This function outport is used for sending status evaluation results of the Gigalink members to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: The Gigalink member is not connected or is not working properly. ▪ 1: The Gigalink member is connected and working properly.
Dependencies	–

Note

The Gigalink status is always evaluated for all the channels of a Gigalink at once, not for each channel separately. This means that the status has a value of 1 at each Gigalink member, even if the specified channels at the connected systems do not match.

Overview of Tunable Properties (Gigalink)

Tunable properties

The Gigalink function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (Gigalink)

Where to go from here

Information in this section

Configuring the Basic Functionality (Gigalink).....	1357
Configuring Standard Features (Gigalink).....	1359

Configuring the Basic Functionality (Gigalink)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Specifying the Gigalink connection (Gigalink and Channel number).
- Specifying data widths for signal transmission.
- Enabling and specifying synchronized data transmission.

Basics on Gigalink connections

To simulate your real-time applications or parts of your real-time application on several platforms simultaneously, you have to define the signal transmission between the real-time application parts you want to assign to the different CPUs. To enable signal transmission between the Gigalink members (real-time applications), you have to configure the Gigalink function block, i.e., define clear interfaces. Using a Gigalink connection, system coupling is established as a bidirectional, serial point-to-point connection.

You can check if signal transmission is working properly, via the Gigalink Status function port.

Specifying the Gigalink connection

Preconditions for connecting two SCALEXIO systems If you connect two SCALEXIO systems, you have to change the IOCNET Link Board Configuration in the SCALEXIO System Configuration home page from IOCNET/GIGALINK (auto-detect) to GIGALINK (fixed) for both SCALEXIO systems. For details, refer to [How to Connect Another SCALEXIO System via Gigalink \(SCALEXIO Hardware Installation and Configuration\)](#).

Specifying ports and channels Internally, SCALEXIO provides up to eight Gigalink ports, but only one of them (default: port number four) is available on the SCALEXIO backplane. To enable signal transmission, you have to know this port number and configure the port (Gigalink number) in ConfigurationDesk. You can specify up to eight channels (Channel number) for a port. This wide range of configurable channels is useful, for example, if different tasks of a

real-time application require separate communication channels due to different timing characteristics.

Note

Limitation for multicore and multi-processing-unit applications

In a multicore or a multi-PU application with Gigalink communication all the channels of a specific Gigalink port must be mapped to data ports from model implementations that are assigned to the same application process.

If you map the channels of one Gigalink port to data ports of model implementations that are assigned to different application processes, this error is detected when the real-time application is executed on the hardware. The real-time application is then terminated and an error message is generated.

Note that ConfigurationDesk does not generate conflicts during signal chain implementation, Gigalink function block configuration, or the build process.

Specifying data widths for data transmission

You must configure matching Send and Receive data widths between ConfigurationDesk and the behavior model. A maximum of 1024 signals can be transmitted on each channel as a single vector which is sent in its entirety. The data type used for signal transmission is always double.

Specifying synchronized communication

If signal transmission is not (time) synchronized, there is no way to check if new data is available. New data sent on the same channel overrides previously sent data, and repeatedly reading signal data on the same channel without sending new data returns already received data.

Note

Synchronized communication via Gigalink is supported between SCALEXIO systems and between a SCALEXIO system and a PHS-bus-based system. The PHS-bus-based system can only act as an event receiver, while the SCALEXIO systems can be both event receiver and event sender.

ConfigurationDesk supports event-based synchronization of applications running on two or more SCALEXIO systems via Gigalink connection as follows:

- **Sending Gigalink events:**

You have to create a timer event via the task modeling feature (Executable Application table). To send the event, you have to enable the **Sent** event via Gigalink property, and specify the Gigalink port and the channel number.

Note that only timer events can be configured to be sent via Gigalink connection. Up to eight events can be used for each Gigalink port.

- **Receiving Gigalink events:**

If you enable event reception in the Gigalink function block, the received events can be used to trigger a task that is executed synchronously on the connected systems.

To use the I/O events in the behavior model, you have to assign them to a task via the **Synchronization** property. For more information, refer to [Modeling Asynchronous Tasks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Note

To achieve synchronized communication, you have to ensure that the timer task configured to be sent via Gigalink has no offset. If the offset is used, this can lead to the two applications having different simulation times.

Additionally, you can specify synchronized reading of data by using the blocking or non-blocking mode of the **Gigalink** function block:

- **Blocking:** If no new data is available the receiver waits for the sender to deliver new data. Data consistency and synchronized data transfer are especially important in control applications where a delay might cause stability problems.

Note

Note that the execution of the real-time application waits until the signal provides new values. Then the execution continues. Waiting for new signal values can cause task overrun and loop deadlock situations.

- **Non-Blocking:** If no new data is available the receiver reads the data vector of the previous sampling step. As a result there are no waiting times but the signal is delayed by one sampling.

Use scenario for using Gigalink events ControlDesk supports synchronized data acquisition across ECUs, and RCP and HIL systems. If you want to use simulation time groups to collect platforms/devices with synchronized simulation times, you have to specify Gigalink events in ConfigurationDesk. Simulation time groups can improve synchronization accuracy. For details, refer to [Using Simulation Time Groups \(ControlDesk Measurement and Recording\)](#).

Configuring Standard Features (Gigalink)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The electrical interface of the Gigalink function block does not provide standard configuration features.
-----------------------------	------------------------------------------------------------------------------------------------------------------

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Test automation support	Configuring Test Automation Support on page 92

ECU Interface Configuration

Where to go from here

Information in this section

Introduction (ECU Interface Configuration).....	1362
Overviews (ECU Interface Configuration).....	1364
Configuring the Function Block (ECU Interface Configuration).....	1377

Introduction (ECU Interface Configuration)

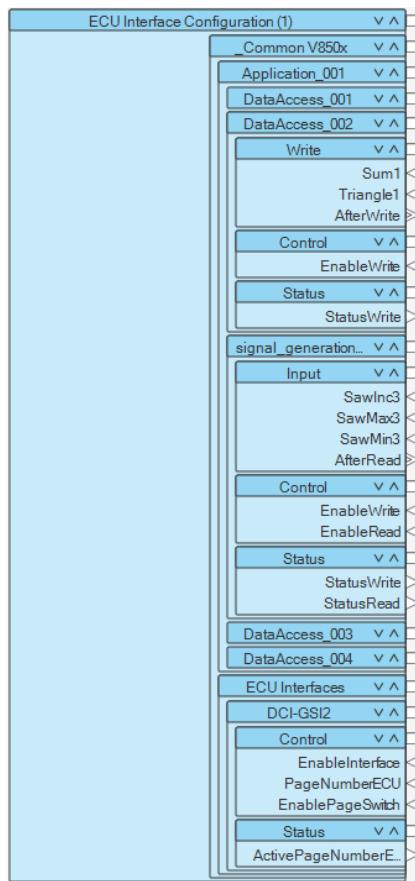
Introduction to the Function Block (ECU Interface Configuration)

Function block purpose

The ECU Interface Configuration function block configures the data exchange between a real-time application (executed on SCALEXIO or MicroAutoBox III) and an ECU application (executed on an ECU) for ECU interfacing.

Block overview

The following illustration shows an example of the function block with an imported ECU interface configuration (EIC) file.



Main features

These are the main features:

- Importing ECU interface container (EIC) files to the ConfigurationDesk application.
- Assigning asynchronous tasks to I/O events.

- Using status information.
- Referencing an ECU interface.

Oversviews (ECU Interface Configuration)

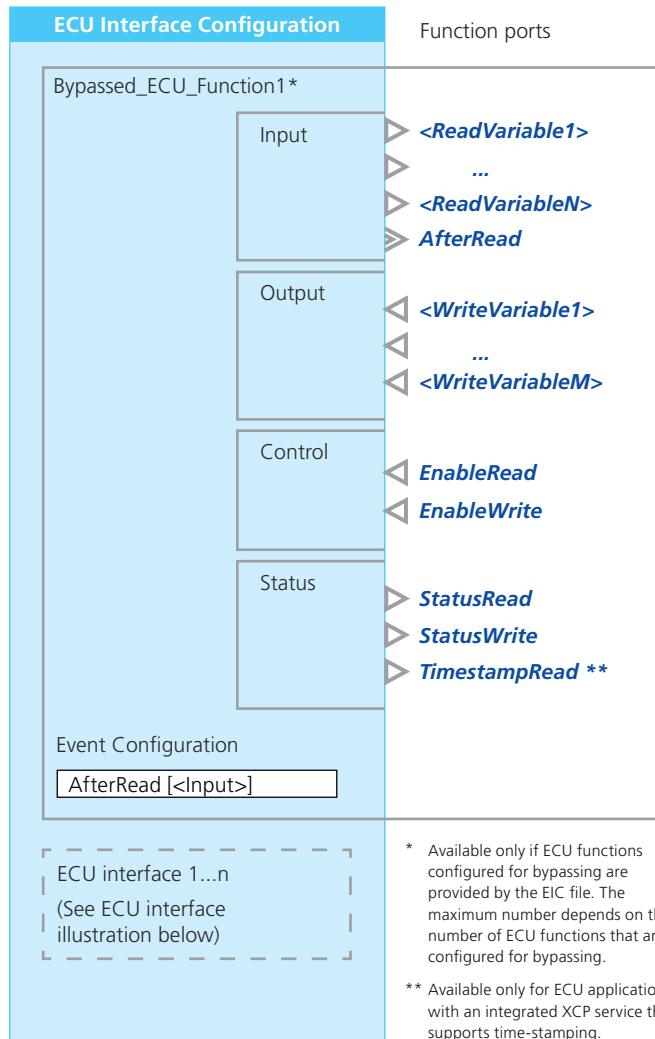
Overview of Ports and Basic Properties (ECU Interface Configuration)

Overview illustrations

The functions, function ports, event ports, and properties that are available for the ECU Interface Configuration function block depend on the imported ECU interface container (EIC) file. In general, the EIC file can provide one or more of the following elements:

- Access to ECU functions, e.g., for bypassing or function tests. For examples on the effects on the ECU Interface Configuration function block, refer to:
 - [Bypassing use scenario](#) on page 1364
 - [Function test use scenario](#) on page 1365
- Data accesses to various ECU variables. For an example on the effects on the ECU Interface Configuration function block, refer to [Data access use scenario](#) on page 1366.
- Access to ECU interfaces and ECU calibration page handling. For an example on the effects on the ECU Interface Configuration function block, refer to [ECU interfaces](#) on page 1367.

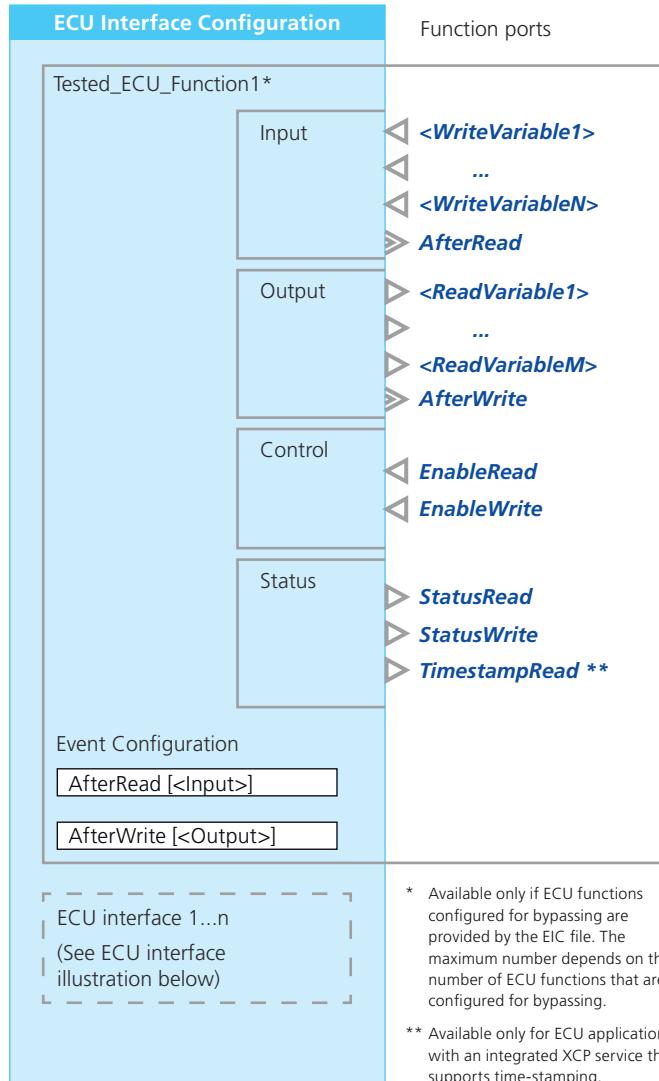
Bypassing use scenario The following illustration is an example of an ECU Interface Configuration function block with an imported EIC file that configures an ECU function for bypassing. The function port types of the Input and Output functions, and the available events are characteristic for this use scenario.



* Available only if ECU functions configured for bypassing are provided by the EIC file. The maximum number depends on the number of ECU functions that are configured for bypassing.

** Available only for ECU applications with an integrated XCP service that supports time-stamping.

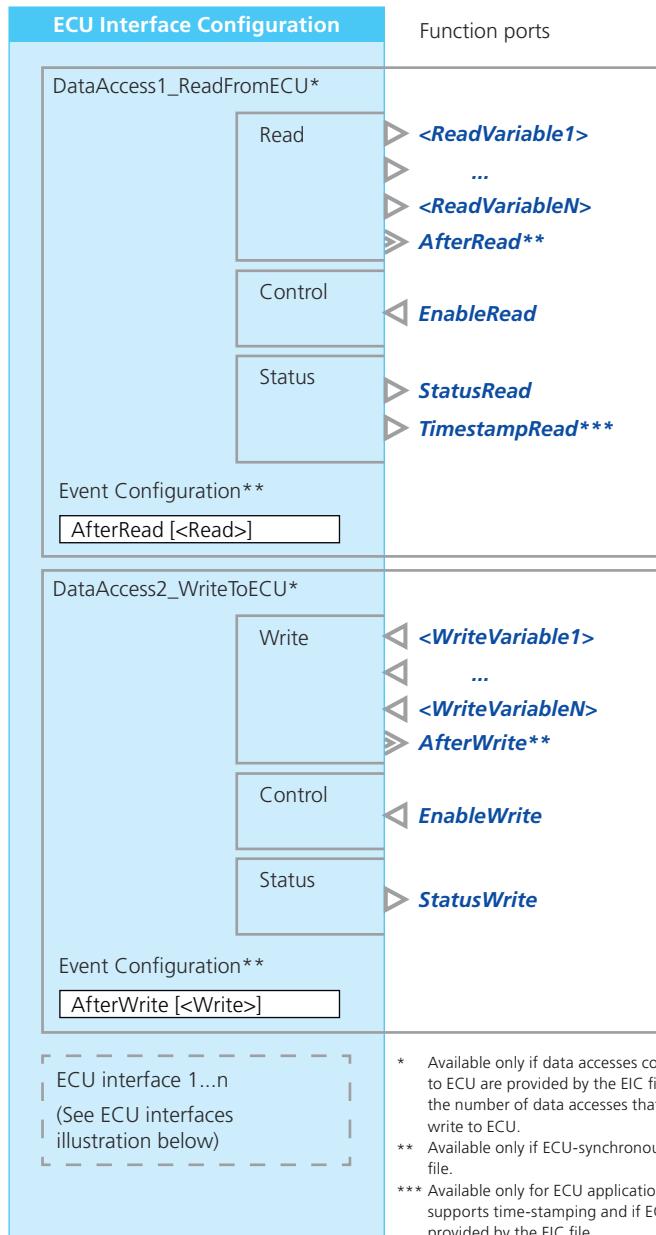
Function test use scenario The following illustration is an example of an ECU Interface Configuration function block with an imported EIC file that configures an ECU function for function test. The function port types of the Input and Output functions, and the available events are characteristic for this use scenario.



* Available only if ECU functions configured for bypassing are provided by the EIC file. The maximum number depends on the number of ECU functions that are configured for bypassing.

** Available only for ECU applications with an integrated XCP service that supports time-stamping.

Data access use scenario The following illustration is an example of an ECU Interface Configuration function block with an imported EIC file that configures ECU variables for data access. The EIC file configures two data accesses, one for reading data from the ECU and one for writing data to the ECU. The available functions, their function ports, and the available events are specific for each data access.

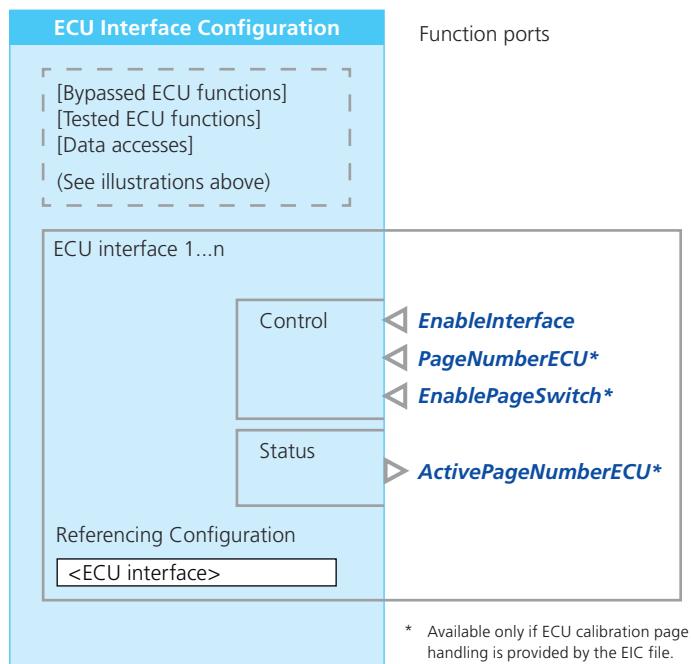


* Available only if data accesses configured for read from ECU or write to ECU are provided by the EIC file. The maximum number depends on the number of data accesses that are configured for read from ECU or write to ECU.

** Available only if ECU-synchronous data access is provided by the EIC file.

*** Available only for ECU applications with an integrated XCP service that supports time-stamping and if ECU-synchronous data access is provided by the EIC file.

ECU interfaces Each EIC file contains information on the ECU interfaces that are used to connect the target ECU to the SCALEXIO or MicroAutoBox III system. The following illustration provides an overview of the functions, function ports, and properties of an ECU Interface Configuration function block that are specific for accessing the ECU interfaces.



Input function

This function provides either function outports or function imports.

- Function outports: The input values of an ECU function are read and provided to a behavior model (e.g., in bypassing use scenarios).
- Function imports: The input values of an ECU function are overwritten with values provided by a behavior model (e.g., in function test use scenarios).

For each ECU function that can be accessed according to the imported EIC file, there can be only one Input function. The availability of the Input function and all its characteristics, e.g., number and types of function ports, depend on the settings specified in the ECU Interface Manager.

Note

When you extend the signal chain and you map the function ports of the Input function to multiple model port blocks, you must ensure that all the model port blocks are used in exactly one system or subsystem of one behavior model.

To avoid invalid model port block arrangements in the behavior model, set the function block's Model port block structure property to Function-based. In this case, only one model port block is created for all the function ports of the Input function when you generate model port blocks via the Generate New Simulink Model Interface or Propagate to Simulink Model commands. For more information, refer to [Specifying Options for Creating Model Port Blocks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Read function

This function provides function outports: Values of ECU variables are read and provided to a behavior model.

There is exactly one Read function for each data access that reads data from the ECU. The characteristics of the Read function, e.g., number of function ports, depend on the settings specified in the ECU Interface Manager.

Note

When you extend the signal chain and you map the function ports of the Read function to multiple model port blocks, you must ensure that all the model port blocks are used in exactly one system or subsystem of one behavior model.

To avoid invalid model port block arrangements in the behavior model, set the function block's Model port block structure property to Function-based. In this case, only one model port block is created for all the function ports of the Read function when you generate model port blocks via the Generate New Simulink Model Interface or Propagate to Simulink Model commands. For more information, refer to [Specifying Options for Creating Model Port Blocks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

AfterRead

This event port provides an I/O event each time the related AfterRead event is received from the ECU application. The settings specified in the ECU Interface Manager determine which AfterRead event is related to the event port. For example, a related AfterRead event can be generated by an ECU function after the complete data of the ECU function is provided to the real-time application. In the real-time application, the event can be used to trigger the execution of functions that bypass or test the ECU function, for example.

Value range	—
Dependencies	There is only one AfterRead event port for each Input or Read function if it is configured in the imported EIC file.

Output function

This function provides either function outports or function imports.

- Function outports: The output values of an ECU function are read and provided to a behavior model (e.g., in function test use scenarios).
- Function imports: The output values of an ECU function are overwritten with values provided by a behavior model (e.g., in bypassing use scenarios).

For each ECU function that can be accessed according to the imported EIC file and for each specified write data access, there can be only one Output function. The availability of the Output function and all its characteristics, e.g., number and types of function ports, depend on the settings specified in the ECU Interface Manager.

Note

When you extend the signal chain and you map the function ports of the Output function to multiple model port blocks, you must ensure that all the model port blocks are used in exactly one system or subsystem of one behavior model.

To avoid invalid model port block arrangements in the behavior model, set the function block's Model port block structure property to Function-based. In this case, only one model port block is created for all the function ports of the Output function when you generate model port blocks via the Generate New Simulink Model Interface or Propagate to Simulink Model commands. For more information, refer to [Specifying Options for Creating Model Port Blocks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Write function

This function provides function imports: Values provided by a behavior model are written to ECU variables.

There is exactly one Write function for each data access that writes data to the ECU. The characteristics of the Write function, e.g., number of function ports, depend on the settings specified in the ECU Interface Manager.

Note

When you extend the signal chain and you map the function ports of the Write function to multiple model port blocks, you must ensure that all the model port blocks are used in exactly one system or subsystem of one behavior model.

To avoid invalid model port block arrangements in the behavior model, set the function block's Model port block structure property to Function-based. In this case, only one model port block is created for all the function ports of the Write function when you generate model port blocks via the Generate New Simulink Model Interface or Propagate to Simulink Model commands. For more information, refer to [Specifying Options for Creating Model Port Blocks \(ConfigurationDesk Real-Time Implementation Guide\)](#).

AfterWrite

This event port provides an I/O event each time the related AfterWrite event is generated. The settings specified in the ECU Interface Manager determine which AfterWrite event is related to the event port. For example, a related AfterWrite event can be generated after data of the behavior model is written completely to an ECU function. In the real-time application, the event can be used to trigger the execution of functions that test the ECU function, for example.

Value range	—
Dependencies	There is only one AfterWrite event port for each Output or Write function if it is configured in the imported EIC file.

EnableRead

This function import controls the read access from within the behavior model. The default state of the read access depends on whether the EnableRead function port and function ports of the related Input, Output, or Read function are used, i.e., if test automation support is enabled and/or the function ports are mapped to model ports:

- If none of the function outports of the related Input, Output, or Read function is used (i.e., test automation support is disabled and no function port is mapped to a model port) read access is disabled permanently. The state of the EnableRead function port is ignored.
- If the EnableRead function port is not used (i.e., test automation support is disabled and the function port is not mapped to a model port) but at least one of the function outports of the related Input, Output, or Read function is used, read access is enabled permanently.
- If the EnableRead function port and at least one of the function outports of the related Input, Output, or Read function is used, read access is disabled by default. To read data from the ECU function, you must enable the read access explicitly.

Note

If you place the model port block that is mapped to the EnableRead function port in a function-call subsystem in your behavior model, it is recommended that the function-call subsystem is not triggered by I/O events provided by the ECU Interface Configuration function block. Depending on the states of the ECU interfaces and on the read and/or write accesses of the ECU Interface Configuration function block, I/O events might be disabled. In this case, you cannot enable the read access from within the behavior model if the EnableRead-related model port block is placed in a function-call subsystem that is triggered by such an I/O event. Instead, place the model port block in a function-call subsystem that is triggered by a cyclic timer event, for example.

Value range	<ul style="list-style-type: none"> ▪ 0: Read access is disabled. No data is read from the related ECU function or ECU variables on the target ECU. ▪ 1: Read access is enabled. Data is read from the related ECU function or ECU variables on the target ECU.
Dependencies	According to the imported EIC file, there is one EnableRead function port for each ECU function and data access that reads data from the ECU application.

EnableWrite

This function import controls the write access from within the behavior model. The default state of the read access depends on whether the EnableWrite function port and function ports of the related Input, Output, or Write

function are used, i.e., if test automation support is enabled and/or the function ports are mapped to model ports:

- If none of the function inputs of the related Input, Output, or Write function is used (i.e., test automation support is disabled and no function port is mapped to a model port) write access is disabled permanently. The state of the EnableWrite function port is ignored.
- If the EnableWrite function port is not used (i.e., test automation support is disabled and the function port is not mapped to a model port) but at least one of the function inputs of the related Input, Output, or Write function is used, write access is enabled permanently.
- If the EnableWrite function port and at least one of the function inputs of the related Input, Output, or Write function is used, write access is disabled by default. To write data to the ECU function, you must enable the write access explicitly.

Note

If you place the model port block that is mapped to the EnableWrite function port in a function-call subsystem in your behavior model, it is recommended that the function-call subsystem is not triggered by I/O events provided by the ECU Interface Configuration function block. Depending on the states of the ECU interfaces and on the read and/or write accesses of the ECU Interface Configuration function block, I/O events might be disabled. In this case, you cannot enable the write access from within the behavior model if the EnableWrite-related model port block is placed in a function-call subsystem that is triggered by such an I/O event. Instead, place the model port block in a function-call subsystem that is triggered by a cyclic timer event, for example.

Value range	<ul style="list-style-type: none"> ▪ 0: Write access is disabled. No data is written to the related ECU function or ECU variables on the target ECU. ▪ 1: Write access is enabled. Data is written to the related ECU function or ECU variables on the target ECU.
Dependencies	According to the imported EIC file, there is one EnableWrite function port for each ECU function and data access that writes data to the ECU application.

StatusRead

This function outport provides the state of the read access of the Input, Output, or Read function to the behavior model. The state indicates if data has been read from the related ECU variables on the target ECU since the last time the Input, Output, or Read function was called. For more information, refer to [Configuring the Basic Functionality \(ECU Interface Configuration\)](#) on page 1377.

The state is calculated immediately after the function call, i.e., before the function values are used by other functions in the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: New data is available Data has been read from the related ECU variables on the target ECU since the last time the Input, Output, or Read function was called. The function outports of the Input, Output, or Read function provide the newest read data. ▪ -1: Old data is available No data has been read from the related ECU variables on the target ECU since the last time the Input, Output, or Read function was called. The function outports of the Input, Output, or Read function provide the data of the previous read access. ▪ -4: No data is available No data is available from the related ECU variables on the target ECU (e.g., because the read access is disabled, no ECU interface is connected, or the data reception failed).
Dependencies	According to the imported EIC file, there is one StatusRead function port for each ECU function and data access that reads data from the ECU application.

StatusWrite

This function outport provides the state of the write access of the Input, Output, or Write function to the behavior model. The state indicates if data has been written to the related ECU function or ECU variables on the target ECU since the last time the Input, Output, or Write function was called. For more information, refer to [Configuring the Basic Functionality \(ECU Interface Configuration\)](#) on page 1377.

The state is calculated after all the related functions of the behavior model have written their data to the Input, Output, or Write function.

Value range	<ul style="list-style-type: none"> ▪ 0: New data is available New data has been written to the ECU function or ECU variables on the target ECU since the last time the Input, Output, or Write function was called. ▪ -4: No data is available No data was written to the related ECU function or ECU variables on the target ECU (e.g., because the write access is disabled, no ECU interface is connected, or the data transmission failed).
Dependencies	According to the imported EIC file, there is one StatusWrite function port for each ECU function and data access that writes data to the ECU application.

TimestampRead

This function outport provides a time stamp for the read access of the Input, Output, or Read function to the behavior model.

However, the `TimestampRead` function ports provides time stamps only if you map the port to a model port. When you do this, the ECU sends time stamps to the real-time hardware. The function port receives the time stamps and provides them as an absolute time in seconds, starting from the first data package that is read by the `Input`, `Output`, or `Read` function.

Value range	0 ... Double _{max} s
Dependencies	<ul style="list-style-type: none"> ▪ Available only if an XCP service is integrated in the ECU application that supports time-stamping, i.e., the A2L file of the ECU application has the <code>TIMESTAMP_SUPPORTED</code> entry. ▪ Available only for ECU-synchronous read operations, i.e.: <ul style="list-style-type: none"> ▪ There is one <code>TimestampRead</code> function port for each ECU function whose data is read from the ECU application. ▪ There is one <code>TimestampRead</code> function port for each ECU-synchronous data access that reads data from the ECU application.

EnableInterface

This function import enables and disables the related ECU interface (e.g., an XCP on UDP/IP interface) from within the behavior model. The default state of the ECU interface depends on whether the `EnableInterface` function port is used, i.e., if test automation support is enabled or the function port is mapped to a model port:

- If the `EnableInterface` function port is not used (i.e., test automation support is disabled and the function port is not mapped to a model port), the related ECU interface is enabled.
- If the `EnableInterface` function port is used (i.e., test automation support is disabled and/or the function port is mapped to a model port), the related ECU interface is disabled by default.

Note

If you place the model port block that is mapped to the `EnableInterface` function port in a function-call subsystem in your behavior model, it is recommended that the function-call subsystem is not triggered by I/O events provided by the `ECU Interface Configuration` function block.

Depending on the states of the ECU interfaces and on the read and/or write accesses of the `ECU Interface Configuration` function block, I/O events might be disabled. In this case, you cannot enable the ECU interface from within the behavior model if the `EnableInterface`-related model port block is placed in a function-call subsystem that is triggered by such an I/O event. Instead, place the model port block in a function-call subsystem that is triggered by a cyclic timer event, for example.

Value range	<ul style="list-style-type: none"> ▪ 0: The ECU interface is disabled. <p>No data can be exchanged between the ECU application and the SCALEXIO or MicroAutoBox III system via the related ECU interface.</p>
-------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 1: The ECU interface is enabled. Data can be exchanged between the ECU application and the SCALEXIO or MicroAutoBox III system via the related ECU interface.
Dependencies	There is one EnableInterface function port for each ECU interface that is specified in the imported EIC file.

PageNumberECU

This function import lets you switch the calibration page that is accessed by the ECU application from within the behavior model. The calibration page with the specified page number is the active calibration page of the ECU application on the target ECU.

The calibration page that is accessed by the ECU application by default is specified via the ECU Interface Manager. This calibration page is the active calibration page of the ECU application even if the function port is not mapped to a model port.

Note

The calibration page that is accessed by the real-time application is specified via the ECU Interface Manager. You cannot change this setting at run time.

For basic information on ECU calibration page handling, refer to [Preparing ECU Calibration Page Handling \(ECU Interface Manager Manual\)](#).

Value range	0 ... maximum number of calibration pages
Dependencies	Available only if ECU calibration page handling is prepared with the ECU Interface Manager.

EnablePageSwitch

This function import lets you enable and disable to switch the active ECU calibration page from within the behavior model. The default state depends on whether the EnablePageSwitch and PageNumberECU function ports are used, i.e., if test automation support is enabled and/or the function ports are mapped to model ports:

- If the PageNumberECU function port is not used (i.e., test automation support is disabled and the function port is not mapped to a model port), switching the active ECU calibration page is disabled permanently. The state of the EnablePageSwitch function port is ignored.
- If the EnablePageSwitch function port is not used (i.e., test automation support is disabled and the function port is not mapped to a model port) but the PageNumberECU function port is used, switching the active ECU calibration page is enabled permanently.
- If the EnablePageSwitch function port and the PageNumberECU function port are used, switching the active ECU calibration page is disabled by default. To switch the active ECU calibration page, you must enable it explicitly.

For basic information on ECU calibration page handling, refer to [Preparing ECU Calibration Page Handling \(ECU Interface Manager Manual\)](#).

Value range	<ul style="list-style-type: none">▪ 0: Switching the active ECU calibration page is disabled.▪ 1: Switching the active ECU calibration page is enabled.
Dependencies	Available only if ECU calibration page handling is prepared with the ECU Interface Manager.

ActivePageNumberECU

This function output provides the number of the active calibration page to the behavior model. The calibration page with the related page number is the calibration page that is accessed by the ECU application on the target ECU.

For basic information on ECU calibration page handling, refer to [Preparing ECU Calibration Page Handling \(ECU Interface Manager Manual\)](#).

Value range	0 ... maximum number of calibration pages
Dependencies	Available only if ECU calibration page handling is prepared with the ECU Interface Manager.

Related topics

Basics

[ECU Interfacing with SCALEXIO or MicroAutoBox III Systems \(ConfigurationDesk Real-Time Implementation Guide\)](#)

Configuring the Function Block (ECU Interface Configuration)

Where to go from here

Information in this section

- [Configuring the Basic Functionality \(ECU Interface Configuration\)..... 1377](#)
- [Configuring Standard Features \(ECU Interface Configuration\)..... 1379](#)

Configuring the Basic Functionality (ECU Interface Configuration)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Assigning asynchronous tasks to I/O events.
- Using status information.
- Referencing an ECU interface.

Assigning asynchronous tasks to I/O events

Depending on the specifications in the imported EIC file, the ECU Interface Configuration function block can provide the following I/O events for each accessible ECU function and each data access that groups accessible ECU variables:

- AfterRead event

The AfterRead event is set when the data of the related ECU function or ECU variables that are grouped by a data access is read completely.

- AfterWrite event

The AfterWrite event is set when the data of the behavior model is written completely to the related ECU function or ECU variables that are grouped by a data access.

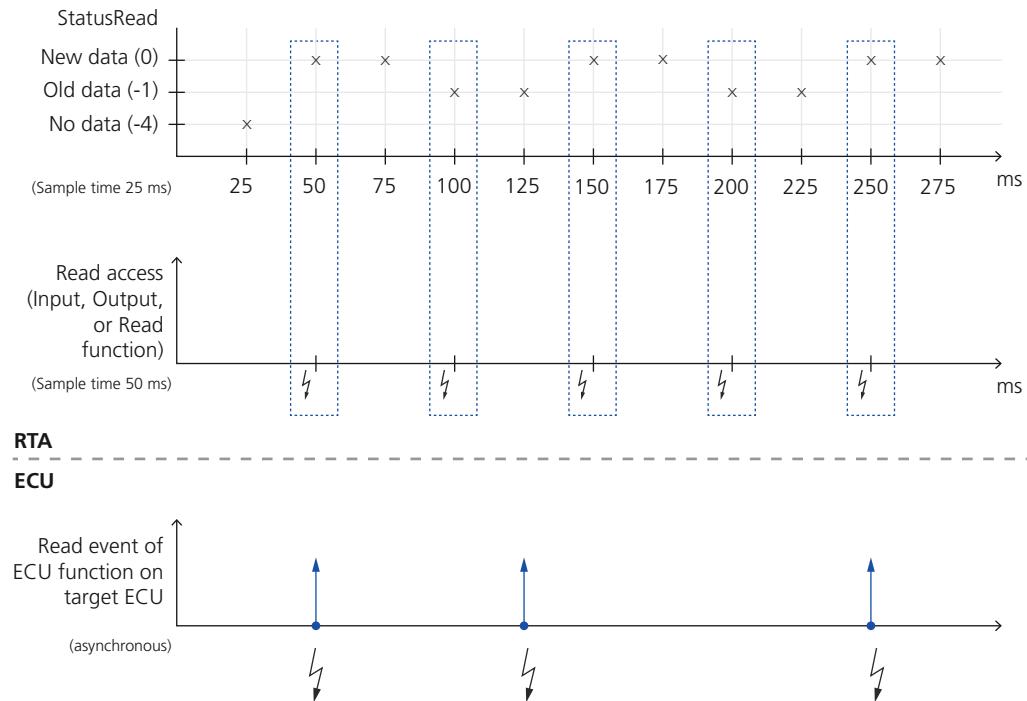
You can use the I/O events as trigger sources for functions in the behavior model. For more information, refer to [Workflow for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink \(ConfigurationDesk Real-Time Implementation Guide](#) .

Using status information

Depending on the specifications in the imported EIC file, the ECU Interface Configuration function block can provide one StatusRead and/or one StatusWrite function port for each configured ECU function access or data access.

The states provided by these function ports indicate if data has been read from or written to the affected ECU variables on the target ECU since the last time the related Input, Output, Read, or Write function was called in the behavior

model. The following illustration is an example of the states provided by the StatusRead function port.



In the example above, the ECU function on the target ECU provides the data to be read non-cyclically. The related reading function (Input, Output, or Read) is not triggered by the events of the ECU function but by a cyclic timer event with a sample time of 50 ms. The StatusRead is triggered by another timer event with a sample time of 25 ms. Before the reading function is sampled for the first time, the state is -4 (no data available). Then, the state changes each time the reading function is sampled, regardless if the ECU function or the StatusRead are triggered in the meantime. The states indicate either if new data (0) or no data was read from the ECU function since the reading function was sampled for the last time. In the latter case, the reading function provides old data (-1) to the behavior model.

Specifying the ECU interface

Importing an EIC file to an ECU Interface Configuration function block adds requirements regarding the ECU interface to the function block (such as XCP on UDP/IP compliance). To exchange data configured with the ECU Interface Configuration function block with the target ECU, you must specify the ECU interface. To do so, you must reference an Ethernet Setup or CAN function block.

Referencing an Ethernet Setup function block To specify an Ethernet ECU interface, you must reference an Ethernet Setup function block in the ECU Interface Configuration function block. The Ethernet Setup function block lets you set up and configure the connection between the ECU interface and an Ethernet controller of a SCALEXIO or MicroAutoBox III system.

For more information on the Ethernet Setup function block, refer to [Ethernet Setup](#) on page 1168.

Referencing a CAN function block To specify a CAN ECU interface, you must reference a CAN function block in the ECU Interface Configuration function block. The CAN function block lets you set up and configure the connection between the ECU interface and different channel types of a SCALEXIO or MicroAutoBox III system.

Depending on the channel type, you can configure various bus-related settings, such as the baud rate, CAN FD support, or partial networking.

- For more information on specifying a CAN ECU interface, refer to [Notes on Specifying CAN ECU Interfaces \(ConfigurationDesk Real-Time Implementation Guide\)](#).
- For more information on the CAN function block and the characteristics of the supported channel types, refer to [CAN](#) on page 1091.

Referencing one Ethernet Setup or CAN function block in several ECU Interface Configuration function blocks If several ECU Interface Configuration function blocks reference the same Ethernet Setup or CAN function block, you must ensure that all the ECU Interface Configuration function blocks belong to only one application process (e.g., all their mapped model port blocks must be elements of the same behavior model).

Related topics

Basics

[ECU Interfacing with SCALEXIO or MicroAutoBox III Systems \(ConfigurationDesk Real-Time Implementation Guide\)](#)

Configuring Standard Features (ECU Interface Configuration)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The ECU Interface Configuration function block does not provide an electrical interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Test automation support	Configuring Test Automation Support on page 92

Power Supply Control

Where to go from here

Information in this section

Introduction (Power Supply Control).....	1382
Overviews (Power Supply Control).....	1383
Configuring the Function Block (Power Supply Control).....	1388

Information in other sections

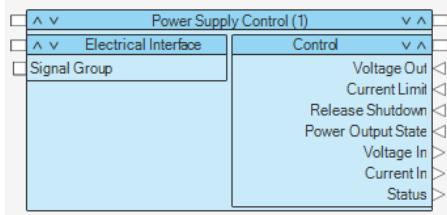
Channels for Battery Simulation Control and Power Switch (DS2680) (SCALEXIO Hardware Installation and Configuration  <td></td>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

Introduction (Power Supply Control)

Introduction to the Function Block (Power Supply Control)

Function block purpose The Power Supply Control function block type gives you access to the battery simulation power supply controller.

Default display The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features These are the main features:

- Controlling the assigned power supply (output voltage, max. output current, etc.) from within the behavior model.
- Specifying the behavior of overcurrent protection.
- Providing voltage and current values and the operating status of the controlled power supply to the behavior model.

Supported channel types The Power Supply Control function block type supports the following channel type:

	SCALEXIO	MicroAutoBox III
Channel type	Power Control 1	–
Hardware	DS2907	–

Overviews (Power Supply Control)

Where to go from here

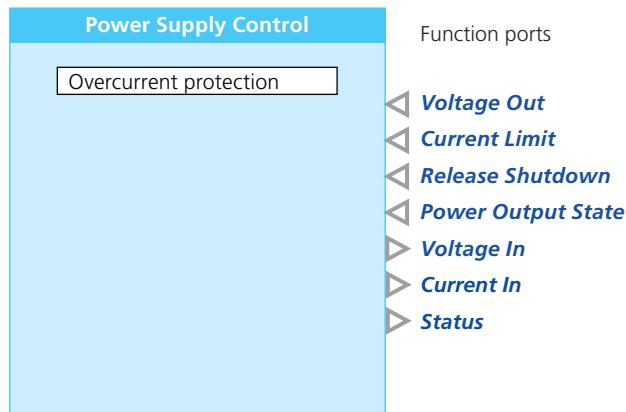
Information in this section

- | | |
|------------------------------------------------------------------------------------|----------------------|
| Overview of Ports and Basic Properties (Power Supply Control)..... | 1383 |
| Overview of Tunable Properties (Power Supply Control)..... | 1386 |

Overview of Ports and Basic Properties (Power Supply Control)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



The block has an electrical interface but does not provide signal ports.

Voltage Out

This function import is used to define the output voltage of the power supply from within the behavior model during run time.

Value range	<ul style="list-style-type: none"> ▪ 0 ... U_{max} (in Volt) ▪ The maximum value depends on the battery simulation power supply unit that is controlled by the assigned battery simulation controller. ▪ If you use a dSPACE battery simulation power supply unit, refer to Data Sheet of a Typical SCALEXIO Rack (SCALEXIO Hardware Installation and Configuration).
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ If you use a customer-specific power supply unit, refer to its data sheet for particular maximum values. These maximum values also have to be specified at the channel type properties of the assigned battery simulation controller. ▪ If you create a hardware topology from scratch, the maximum value is set to 15 V. This also applies if a connected and assigned hardware resource does not provide specific maximum values to ConfigurationDesk. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the value range.
Dependencies	–

Current Limit

This function import is used to define the current limit of the power supply from within the behavior model during run time.

Value range	<ul style="list-style-type: none"> ▪ 0 ... I_{max} (in Ampere) ▪ The maximum value depends on the battery simulation power supply unit that is controlled by the assigned battery simulation controller. ▪ If you use a dSPACE battery simulation power supply unit, refer to Data Sheet of a Typical SCALEXIO Rack (SCALEXIO Hardware Installation and Configuration). ▪ If you use a customer-specific power supply unit, refer to its data sheet for particular maximum values. These maximum values also have to be specified at the channel type properties of the assigned battery simulation controller. ▪ If you create a hardware topology from scratch, the maximum value is set to 25 A. This also applies if a connected and assigned hardware resource does not provide specific maximum values to ConfigurationDesk. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the value range.
Dependencies	–

Release Shutdown

This function import is used to re-enable the power supply's output from within the behavior model during run time after an automatic shutdown has occurred. The new value takes effect immediately.

Value range	<ul style="list-style-type: none"> ▪ 0: Inactive Deactivate release shutdown. ▪ 1: Active Activate release shutdown.
-------------	------------------------------------------------------------------------------------------------------------------------------------------------------

	Note, that only the change from Inactive to Active re-enables the power supply's output.
Dependencies	The function port values are considered only if the Overcurrent protection property is set to Shutdown.

Power Output State

This function import is used to enable/disable the output of the power supply from within the behavior model during run time. The new value takes effect immediately.

Value range	<ul style="list-style-type: none"> ▪ 0: Off Power supply output is disabled. ▪ 1: On Power supply output is enabled.
Dependencies	–

Voltage In

This function outport returns the voltage value of the power supply to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... U_{max} (in Volt) ▪ The maximum value depends on the battery simulation power supply unit that is controlled by the assigned battery simulation controller. ▪ If you use a dSPACE battery simulation power supply unit, refer to Data Sheet of a Typical SCALEXIO Rack (SCALEXIO Hardware Installation and Configuration). ▪ If you use a customer-specific power supply unit, refer to its data sheet for particular maximum values. These maximum values also have to be specified at the channel type properties of the assigned battery simulation controller. ▪ If you create a hardware topology from scratch, the maximum value is set to 15 V. This also applies if a connected and assigned hardware resource does not provide specific maximum values to ConfigurationDesk. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the value range.
Dependencies	–

Current In

This function outport returns the current value of the power supply to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... I_{max} (in Ampere) ▪ The maximum value depends on the battery simulation power supply unit that is controlled by the assigned battery simulation controller.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ If you use a dSPACE battery simulation power supply unit, refer to Data Sheet of a Typical SCALEXIO Rack (SCALEXIO Hardware Installation and Configuration). ▪ If you use a customer-specific power supply unit, refer to its data sheet for particular maximum values. These maximum values also have to be specified at the channel type properties of the assigned battery simulation controller. ▪ If you create a hardware topology from scratch, the maximum value is set to 25 A. This also applies if a connected and assigned hardware resource does not provide specific maximum values to ConfigurationDesk. ▪ If you apply user saturation min/max values for saturation, the specified values might reduce the value range.
Dependencies	–

Status This function outport returns the status of the power supply to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: OK The power supply has no failure and keeps the voltage constant. The current limit is not exceeded. The output is enabled. ▪ 1: Current saturation The current limit is reached. ▪ 2: Output shutdown The power supply's output is disabled. This can be if a fault occurred or the output was disabled using the Power Output State function port.
Dependencies	–

Overview of Tunable Properties (Power Supply Control)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality		
Overcurrent protection	✓	–

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Power Supply Control)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------------|------|
| Configuring the Basic Functionality (Power Supply Control)..... | 1388 |
| Configuring Standard Features (Power Supply Control)..... | 1389 |

Configuring the Basic Functionality (Power Supply Control)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Specifying the behavior of overcurrent protection

Interface to Controller

The Power Supply Control function port is the interface to the dSPACE battery simulation controller that sets the battery simulation power supply (e.g., Genesys™ battery simulation power supply). For more information, refer to [Battery Simulation Concept \(SCALEXIO Hardware Installation and Configuration](#) (PDF)).

Specifying the behavior of overcurrent protection

For the overcurrent protection you can select the following options:

- Saturation

When reaching the current limit, the power supply holds the current value and reduces the output voltage. Saturation is reported via the Status function port.

- Shutdown

When exceeding the current limit, the power supply's outputs are shut down. A shutdown is reported via the Status function port

Note

The Overcurrent protection is only supported by the TDK-Lambda power supplies. Power supplies of other manufacturers do not assess this property. You get information on the used power supply via the Platform Manager.

Tip

Using the Release Shutdown function port:

You can use the Release Shutdown function port to re-enable the power supply's output after an automatic shutdown has occurred. Therefore you must set the value at the port from Inactive to Active. Keep in mind, only the change from Inactive to Active re-enables the power supply's output. You can do this from within the behavior model during run time. The new value takes effect immediately. The function port values are considered only if the Overcurrent protection property is set to Shutdown.

Configuring Standard Features (Power Supply Control)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.
-----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Configuration Feature	Power Control 1	Further Information
Channel multiplication	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports		
–	–	–

Mapping signal ports The Power Supply Control function block does not provide signal ports. You do not have to select a connection to an external device (e.g., ECU) or a load. The connection to the power supply is internally fixed. However, you have to assign the hardware resource.

Model interface	The interface to the behavior model of the function block provides the following standard features.
------------------------	-----------------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
User saturation	Specifying User Saturation on page 90
Test automation support	Configuring Test Automation Support on page 92

Power Switch

Where to go from here

Information in this section

Introduction (Power Switch).....	1392
Overviews (Power Switch).....	1393
Configuring the Function Block (Power Switch).....	1396
Hardware Dependencies (Power Switch).....	1398

Information in other sections

Channels for Battery Simulation Control and Power Switch
(DS2680) (SCALEXIO Hardware Installation and
Configuration 

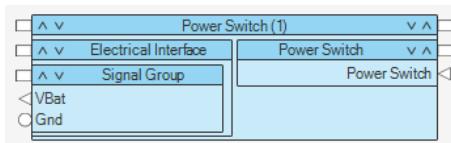
Introduction (Power Switch)

Introduction to the Function Block (Power Switch)

Function block purpose The Power Switch function block type provides switched battery simulation supply outputs for connected external devices (e.g., ECU).

The function block is part of the dSPACE battery simulation concept and switches the battery simulation supply voltage to connected external devices.

Default display The function block provides signal ports and function ports according to the settings. The following illustration shows the default display of the function block.



Main features These are the main features:

- Connecting/disconnecting the power supply to external devices from within the behavior model.
- Measuring the current of the switched signal path and providing the averaged measurement results to the behavior model.

Supported channel types The Power Switch function block type supports the following channel types:

	SCALEXIO		MicroAutoBox III
Channel type	Power Switch 1	Power Switch 2	–
Hardware	DS2642	DS2680	–

Overviews (Power Switch)

Where to go from here

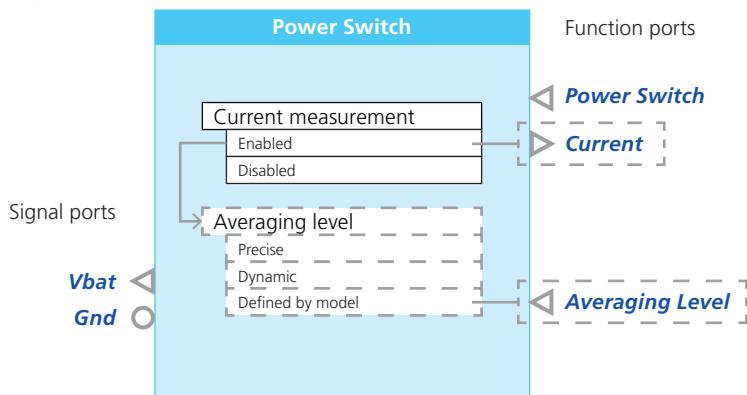
Information in this section

- | | |
|------------------------------------------------------------|------|
| Overview of Ports and Basic Properties (Power Switch)..... | 1393 |
| Overview of Tunable Properties (Power Switch)..... | 1394 |

Overview of Ports and Basic Properties (Power Switch)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Power Switch

This function import is used to define the switch setting from within the behavior model during run time.

Value range	<ul style="list-style-type: none"> ▪ 0: Off The power supply is disconnected from the external device (e.g., ECU). ▪ 1: On The power supply is connected to the external device.
Dependencies	-

Current

This function outport provides the current value measured at the signal ports to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... I_{max} (in Ampere)
-------------	--------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> Depends on the assigned hardware. For specific values, refer to Hardware Dependencies (Power Switch) on page 1398.
Dependencies	<ul style="list-style-type: none"> Available only if the Current measurement property is set to Enabled. Supported only for the Power Switch 1 channel type.

Averaging Level

This function import is used to switch the averaging level for the current measurement from within the behavior model during run time.

Value range	<ul style="list-style-type: none"> 1: Precise The current values are measured with lower sample rate and higher resolution. 2: Dynamic The current values are measured with higher sample rate and lower resolution.
Dependencies	<ul style="list-style-type: none"> Available only if the Current measurement property is set to Enabled and the Averaging level property is set to Defined by model. Supported only for the Power Switch 1 channel type.

Vbat

This signal port represents the electrical connection point to the switched power.

Value range	Depends on the assigned hardware resource. For specific values, refer to Hardware Dependencies (Power Switch) on page 1398.
Dependencies	–

Gnd

This signal port represents the ground potential of the dSPACE hardware.

Value range	–
Dependencies	–

Overview of Tunable Properties (Power Switch)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
Averaging level ³⁾	✓	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

³⁾ Is not available as a parameter if the averaging level is provided by the behavior model via function port.

Configuring the Function Block (Power Switch)

Where to go from here

Information in this section

Configuring the Basic Functionality (Power Switch).....	1396
Configuring Standard Features (Power Switch).....	1397

Configuring the Basic Functionality (Power Switch)

Overview

The function block type provides the following configuration features which influence the behavior of the basic functionality:

- Enabling and specifying current measurement

Enabling and specifying current measurement

You can measure the current of the switched signal path and provide the current values via the Current function port to the behavior model. There are two measurement modes you can choose for this purpose via the Averaging level property. These modes differ in their sample rates and resolutions.

- Dynamic

The Dynamic mode has a higher sample rate and a lower resolution than the Precise mode. You can measure, for example, the outgoing current of a standard signal.

- Precise

The Precise mode has a lower sample rate and a higher resolution. However, the measurable current range is restricted in this mode. For example, the DS2642 FIU & Power Switch Board allows for measuring 0 ... 1.6 A per channel. You can measure, for example, a leakage current.

The sample rate and resolution depend on the assigned hardware resource. Refer to [Hardware Dependencies \(Power Switch\)](#) on page 1398.

The measurement mode (dynamic or precise) can also be defined from within the behavior model and therefore be switched during run time. For using this option, the Averaging Level function port has to be enabled via the Averaging level property.

Limitation The current measurement functionality is supported only for the Power Switch 1 channel type.

Configuring Standard Features (Power Switch)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface The functionality of the electrical interface depends on the channel type. The following table lists the available channel types, the configuration features, and important channel-type-specific differences in property settings. The table shows only the property settings that do not cause a conflict in the Conflicts Viewer. You can configure all property settings in ConfigurationDesk. A conflict occurs if a property setting does not match a specific channel type.

Configuration Feature	Power Switch 1	Power Switch 2	Further Information
Channel multiplication	–	–	Specifying Current and Voltage Values for Channel Multiplication on page 95
Signal Ports			
Failure simulation support	✓	✓	Specifying Settings for Failure Simulation on page 123

Mapping signal ports To map signal ports to device ports, it is helpful to know the circuit diagrams of the channels on the real-time hardware. For more information, refer to:

- [Power Switch 1 on page 1617](#)
- [Power Switch 2 on page 1618](#)

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
Test automation support	Configuring Test Automation Support on page 92

Hardware Dependencies (Power Switch)

SCALEXIO Hardware Dependencies (Power Switch)

Characteristics depending on channel types

The following table shows the characteristics of the hardware resources (for different channel types) that are supported by the function block type. You can use this table to select a suitable channel type.

Channel Type			Power Switch 1	Power Switch 2
Hardware		DS2642 FIU & Power Switch Board	DS2680 I/O Unit	
Output current range		0 ... +10 A _{RMS}	0 ... +24 A _{RMS}	
Output voltage range		0 ... +60 V	0 ... +60 V	
Current measurement	Dynamic averaging	Range	0 A ... 39 A	–
		Resolution	150 µA	–
		Offset error	±5 mA (typ.) (max., 0°C ... 55°C)	–
		Gain error	±1 % (typ.) (max., 0°C ... 55°C)	–
		Sample rate	3.815 kS/s	–
	Precise averaging	Range	0 ... 1.6 A	–
		Resolution	6.25 µA	–
		Offset error	±50 µA (typ.) (max., 0°C ... 55°C)	–
		Gain error	±1 % (typ.) (max., 0°C ... 55°C)	–
		Sample rate	7.4 S/s	–
Circuit diagrams		Power Switch 1 on page 1617	Power Switch 2 on page 1618	
Required channels		1	1	

General limitations

DS2642 FIU & Power Switch Board

The following restrictions apply:

- Channel multiplication across several I/O boards is not supported.

More hardware data

DS2680 I/O Unit For more board-specific data, refer to [Data Sheet of the DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2642 FIU & Power Switch Board For more board-specific data, refer to [Data Sheet of the DS2642 FIU & Power Switch Board \(SCALEXIO Hardware Installation and Configuration\)](#).

Power On Signal In

Where to go from here

Information in this section

Introduction (Power On Signal In).....	1400
Overviews (Power On Signal In).....	1401
Configuring the Function Block (Power On Signal In).....	1404

Introduction (Power On Signal In)

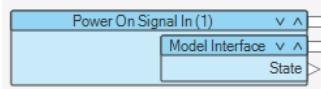
Introduction to the Function Block (Power On Signal In)

Function block purpose

The Power On Signal In function block type monitors hardware based shutdown requests for the SCALEXIO LabBox (with DS6001 Processor Board), SCALEXIO AutoBox (with DS6001 Processor Board) and MicroAutoBox III. The request state can be provided to the behavior model and/or can be accessed in the experiment software.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

This is the main feature:

- Providing manually triggered shutdown requests for dSPACE processing hardware to the behavior model.
-

Supported hardware

The Power On Signal In function block supports the following hardware:

- SCALEXIO LabBox with a DS6001 Processor Board installed
- SCALEXIO AutoBox with a DS6001 Processor Board installed
- MicroAutoBox III

SCALEXIO Processing Units are *not* supported.

Overviews (Power On Signal In)

Where to go from here

Information in this section

- [Overview of Ports and Basic Properties \(Power On Signal In\).....](#) 1401
- [Overview of Tunable Properties \(Power On Signal In\)](#) 1402

Overview of Ports and Basic Properties (Power On Signal In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



State	The function outport provides the state for shutdown requests for the dSPACE hardware to the behavior model. Depending on the hardware used, the shutdown request is triggered by: <ul style="list-style-type: none"> ▪ The on/off button of SCALEXIO LabBox with DS6001 Processor Board. ▪ The remote control signal at the REMOTE pin (pin 4 of the power supply connector) for MicroAutoBox III and SCALEXIO AutoBox with DS6001 Processor Board. The remote control signal comes, for example, from the ignition switch of the vehicle in which the processing hardware is installed.
--------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Value range	<ul style="list-style-type: none"> ▪ 0: No power off request There is no shutdown request from the dSPACE hardware. ▪ 1: Power off request There is a shutdown request from the dSPACE hardware.
Dependencies	—

Tip

When you activate test automation support (TA) for the State function port, the request state of the On/off button or the REMOTE pin is also generated as a variable in the variable description file (*.TRC). This lets you evaluate the request state in the experiment software without having a behavior model connected.

Signal ports

The Power On Signal In function block type does not provide signal ports.

Overview of Tunable Properties (Power On Signal In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	—	—
Function Ports		
Initial switch setting (Test Automation)	—	✓
Initial substitute value (Test Automation)	—	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Power On Signal In)

Configuring Standard Features (Power On Signal In)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface This function block does not provide an electrical interface.

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

System Shutdown

Where to go from here

Information in this section

Introduction (System Shutdown).....	1406
Overviews (System Shutdown).....	1411
Configuring the Function Block (System Shutdown).....	1413

Introduction (System Shutdown)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------|------|
| Introduction to the Function Block (System Shutdown)..... | 1406 |
| Basics on Using the System Shutdown Functionality..... | 1407 |

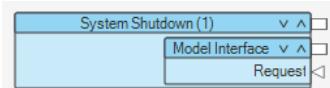
Introduction to the Function Block (System Shutdown)

Function block purpose

The System Shutdown function block type lets you control the shutdown of a SCALEXIO LabBox or SCALEXIO AutoBox (both with a DS6001 Processor Board installed) or a MicroAutoBox III from within the behavior model and/or the experiment software. Hardware based shutdown requests can be ignored or delayed.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Enabling control of the shutdown of SCALEXIO LabBoxes, SCALEXIO AutoBoxes, and MicroAutoBox III from within the behavior model and/or the experiment software.
- Disabling immediate system shutdown from hardware based shutdown requests or other sleep triggers via CAN bus etc.

Supported hardware

The System Shutdown function block supports the following hardware:

- SCALEXIO LabBox with a DS6001 Processor Board installed
- SCALEXIO AutoBox with a DS6001 Processor Board installed
- MicroAutoBox III

SCALEXIO Processing Units are *not* supported.

Basics on Using the System Shutdown Functionality

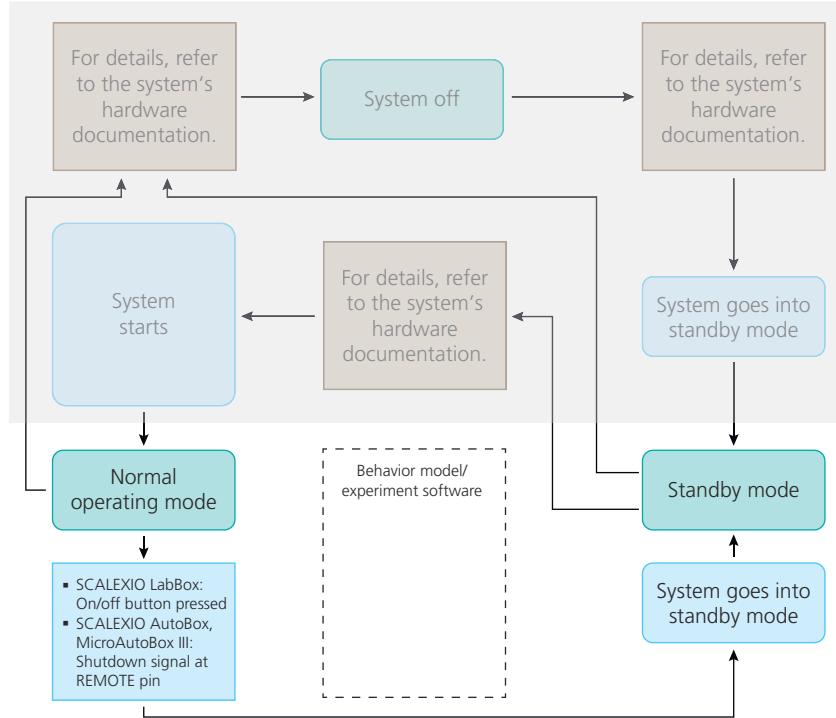
Usage and effects

If you use a System Shutdown function block, the system shutdown is controlled from within the behavior model and/or the experiment software. The system shuts down only if the request function inport is set to *shutdown*. You can let the system enter a defined state before executing shutdown requests. For example, data can be stored and the system can be logged off from a network prior to system shutdown.

The following illustrations show the usage and the effects of the System Shutdown and the related Power On Signal In function block for MicroAutoBox III and single-PU systems with SCALEXIO LabBox or SCALEXIO AutoBox.

Application without System Shutdown function block The system goes from normal operating mode into standby mode when the on/off button of a SCALEXIO LabBox is pressed or a shutdown signal is present at the REMOTE pin of a SCALEXIO AutoBox or MicroAutoBox III. The behavior model/experiment software has no information about these hardware based shutdown requests and cannot control the shutdown of the system. As a result, the real-time application is interrupted and there is no guarantee that the system is in a defined state at that point.

Refer to the following illustration (the upper part is included only for the sake of completeness).



Standby mode: In standby mode, the system is awake but not operating. Applications are not executed but the system can respond to wake-up triggers and go into normal operating mode.

Normal operating mode: In normal operating mode, the system is fully operational. For example, applications can be loaded, started, and terminated.

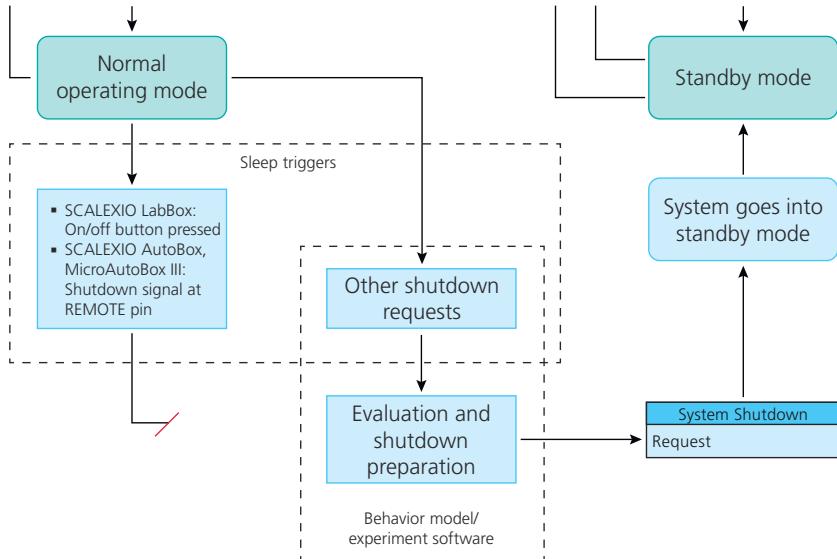
For details on system-specific states and operating modes, refer to

- [Powering Up and Shutting Down the LabBox \(SCALEXIO Hardware Installation and Configuration\)](#)
- [Powering Up and Shutting Down SCALEXIO AutoBox \(SCALEXIO Hardware Installation and Configuration\)](#)
- [Operating Modes when Switching On/Off the MicroAutoBox III \(MicroAutoBox III Hardware Installation and Configuration\)](#)

Application with System Shutdown function block The shutdown of the system is controlled from within the behavior model and/or the experiment software. You can evaluate sleep triggers and bring the system into a defined

state before shutting it down. Afterwards, the real-time application is terminated and the system goes into standby mode.

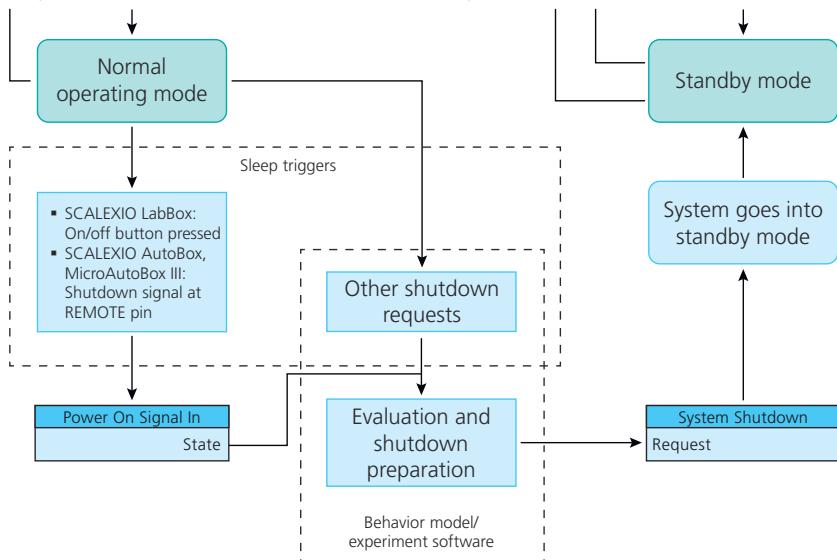
Hardware based shutdown requests are not provided to the behavior model or the experiment software and you cannot use them to shut down the system.



Application with System Shutdown and Power On Signal In function block

block This system uses a System Shutdown function block and a Power On Signal In function block, which provides hardware based shutdown requests to the behavior model and/or the experiment software.

Hardware based shutdown requests can be evaluated like any other sleep trigger, so you also can use them to shut down the system.



Working with single-PU systems with several cores

If you work with a multicore system, use a System Shutdown function block in at least one of the running application processes. It does not matter in which of

the running applications you use the block. The system shuts down only after *all* running application processes are terminated.

Working with multi-PU systems (only for SCALEXIO LabBox and SCALEXIO AutoBox)

If you work with a multi-PU system, use a System Shutdown function block in at least one application process of every processing unit to ensure that all the processing units shut down.

Each System Shutdown function block in applications of a multi-PU system must be connected to a behavior model of the respective processing unit. In multi-PU systems, the system shutdown cannot be controlled via TRC variables.

Systems using the power control bus (only for SCALEXIO LabBox and SCALEXIO AutoBox)

If you work with SCALEXIO LabBoxes or SCALEXIO AutoBoxes that are connected via the power control bus, all the SCALEXIO LabBoxes/AutoBoxes shut down when at least one of them is shut down. This also applies to SCALEXIO LabBoxes/AutoBoxes in the system that have no DS6001 Processor Board installed.

For more information on the power control bus, refer to

- [Powering Up and Shutting Down the LabBox \(SCALEXIO LabBox Getting Started !\[\]\(d8db9837ce3837c8263ff51d82e9b941_img.jpg\)\).](#)
- [Powering Up and Shutting Down SCALEXIO AutoBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(bc29dd0f30802db5707fd8d97c9e8de7_img.jpg\).](#)

Overviews (System Shutdown)

Where to go from here

Information in this section

- | | |
|-------------------------------------------------------------------------------|----------------------|
| Overview of Ports and Basic Properties (System Shutdown)..... | 1411 |
| Tunable Properties (System Shutdown) | 1412 |

Overview of Ports and Basic Properties (System Shutdown)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Request

This function import lets you trigger the shutdown of the system from within the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: No shutdown The system stays in normal operating mode. ▪ 1: Shutdown The system shuts down properly, i.e., running real-time applications are terminated and the system goes into standby mode.
Dependencies	—

Tip

When you activate test automation support (TA) for the Request function import, the state of the request port is also generated as a variable in the variable description file (*.TRC). This lets you control the system shutdown from within the experiment software without having a behavior model connected.

Signal ports

The System Shutdown function block does not provide signal ports.

Tunable Properties (System Shutdown)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (System Shutdown)

Configuring Standard Features (System Shutdown)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface This function block does not provide an electrical interface.

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

System Temperature Monitoring

Where to go from here

Information in this section

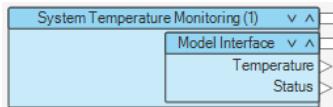
Introduction (System Temperature Monitoring).....	1416
Overviews (System Temperature Monitoring).....	1417
Configuration (System Temperature Monitoring).....	1419

Introduction (System Temperature Monitoring)

Introduction to the Function Block (System Temperature Monitoring)

Function block purpose With the System Temperature Monitoring function block type, you can monitor the internal temperature of MicroAutoBox III via a real-time application.

Default display The function block provides function ports according to the settings. The following illustration shows the default display of the function block.



Main features These are the main features:

- Monitors the internal temperature of the MicroAutoBox III and provides the measured value to the behavior model.
- Provides a status flag to the behavior model if the system reaches overtemperature.

Supported hardware The System Temperature Monitoring function block refers to the processing unit, on which the assigned application process is running. The function block type supports the following hardware:

SCALEXIO	MicroAutoBox III
-	DS1403 Processor Board

Overviews (System Temperature Monitoring)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------------------------|------|
| Overview of Ports and Basic Properties (System Temperature Monitoring)..... | 1417 |
| Overview of Tunable Properties (System Temperature Monitoring)..... | 1418 |

Overview of Ports and Basic Properties (System Temperature Monitoring)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Temperature

This function outport writes the measured internal temperature of MicroAutoBox III to the behavior model.

Value range	-30 °C ... +100 °C
Dependencies	-

Status

This function outport writes a status value regarding overtemperature of the hardware to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: There is no critical overtemperature. ▪ 1: The internal temperature of the MicroAutoBox III has reached a critical value of approx. +100 °C. The shutdown countdown of the MicroAutoBox III is running and MicroAutoBox III is powered down within approx. 5 minutes. <p>In addition the PWR LED on the front panel of MicroAutoBox III lights up red. Refer to LED States of the DS1403 Panel (MicroAutoBox III Hardware Installation and Configuration).</p>
Dependencies	-

Overview of Tunable Properties (System Temperature Monitoring)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuration (System Temperature Monitoring)

Configuring the Standard Features (System Temperature Monitoring)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface This function block does not provide an electrical interface.

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	More Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

USB Eject

Where to go from here

Information in this section

Introduction.....	1422
Overviews.....	1423
Configuration.....	1425

Introduction

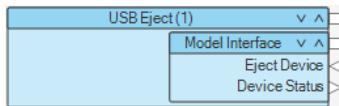
Introduction to the Function Block (USB Eject)

Function block purpose

The USB Eject function block provides a trigger from the behavior model to eject (unmount) a USB mass storage device from the dSPACE system. Removing a USB device from the USB port without unmounting can result in data loss.

Default display

The function block provides function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Provides a trigger from the behavior model to safely unmount a USB mass storage device from the dSPACE system.
- Provides a status flag to the behavior model that indicates whether a USB mass storage device is connected and mounted to the dSPACE system.

Supported hardware

The USB Eject function block refers to the processing unit, on which the assigned application process is running. The function block type supports the following hardware:

SCALEXIO	MicroAutoBox III
DS6001 Processor Board	DS1403 Processor Board

Overviews

Where to go from here

Information in this section

- | | |
|-------------------------------------------------------------------------|----------------------|
| Overview of Ports and Basic Properties (USB Eject)..... | 1423 |
| Overview of Tunable Properties (USB Eject)..... | 1424 |

Overview of Ports and Basic Properties (USB Eject)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Eject Device

This function import provides a trigger from the behavior model to eject (unmount) a USB mass storage device from the USB port of the dSPACE system.

Value range	<ul style="list-style-type: none"> ▪ 0: No trigger is available to eject (unmount) the USB device from the dSPACE system. ▪ 1: A trigger is available to eject (unmount) the USB device from the dSPACE system. <p>Any data values still to be written are saved on the USB device before the dSPACE system unmounts the device from its file system. An unmounted USB device cannot be used by the dSPACE system.</p>
Dependencies	–

Device Status

This function outport writes the status to the behavior model that indicates whether a USB mass storage device is mounted to the dSPACE system.

Value range	<ul style="list-style-type: none"> ▪ 0: A USB device is not connected in general or the device is ejected (unmounted) from the dSPACE system.
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

	An unmounted USB device cannot be used by the dSPACE system. ▪ 1: A USB device is connected and mounted to the dSPACE system.
Dependencies	–

Signal ports

The USB Eject function block type does not provide signal ports.

Overview of Tunable Properties (USB Eject)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuration

Configuring the Standard Features (USB Eject)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface This function block does not provide an electrical interface.

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	More Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

FuSa Setup

Where to go from here

Information in this section

Introduction (FuSa Setup).....	1428
Overviews (FuSa Setup).....	1432
Configuring the Function Block (FuSa Setup).....	1434

Introduction (FuSa Setup)

Where to go from here

Information in this section

Introduction to the Function Block (FuSa Setup).....	1428
Basics on Using FuSa with the MicroAutoBox III.....	1429

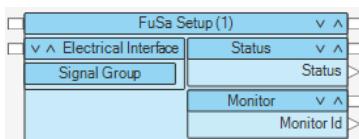
Introduction to the Function Block (FuSa Setup)

Function block purpose

The FuSa Setup function block provides the basic functionality for implementing functional safety in your system. The function block triggers basic error responses and lets you enable additional error responses.

Default display

The function block provides function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Triggering basic error responses
- Enabling additional error responses
- Providing status information and the ID of the monitor that has detected the FuSa error to the behavior model

Limitation of used instances

Only one instance of the FuSa Setup function block can be used in your ConfigurationDesk application.

Supported channel types

The FuSa Setup function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	–	FuSa Unit 1
Hardware	–	DS1403 Processor Board

Available demo project

ConfigurationDesk provides the *FunctionalSafetyDemo* project. This project shows an implementation of the FuSa functionality also using the FuSa Response Trigger and FuSa Challenge-Response Monitoring function blocks. You can use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to [FunctionalSafetyDemo Project: Example of Using MicroAutoBox III Functional Safety Features \(ConfigurationDesk Demo Projects\)](#).

Basics on Using FuSa with the MicroAutoBox III

Functional safety with the MicroAutoBox III

The MicroAutoBox III FuSa concept includes a set of functional safety (FuSa) functionalities. These let you implement elements of functional safety in a prototyping system.

Note

The MicroAutoBox III does not comply with a safety integrity level (SIL/ASIL) or provides functional safety as required by common standards, such as ISO 61508 or ISO 26262. The MicroAutoBox III is a prototyping device for a wide range of applications for which the required safety functions cannot be generally identified.

The MicroAutoBox III is designed on the basis of the three-layer safety concept that is common for electronic control units (ECUs). For more information on this concept and on the hardware components for the FuSa functionality, refer to [Functional Safety \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Activating the FuSa functionality

The MicroAutoBox III automatically activates the FuSa functionality when the real-time application uses the FuSa functionality of ConfigurationDesk, i.e., when you implement specific FuSa function blocks in a ConfigurationDesk application.

If a safety function detects a FuSa error, it triggers the MicroAutoBox III FuSa unit. The FuSa unit includes the functionality to respond to a detected FuSa error and to report it.

Error responses

If a FuSa error is detected, the MicroAutoBox III triggers basic error responses and, if enabled, additional error responses.

Basic error responses You cannot configure or disable the following error responses:

- The FuSa relay of the MicroAutoBox III opens.
- The FuSa LED on the MicroAutoBox III lights up red.

- An error message is generated and stored in the non-volatile memory. The message is displayed on the FUNCTIONAL SAFETY page of the MicroAutoBox III web interface. Refer to [FUNCTIONAL SAFETY Page \(MicroAutoBox III Hardware Installation and Configuration\)](#).
- The generated error message is also sent to a connected host PC (if available). The message is displayed in on the MESSAGES page of the MicroAutoBox III web interface. Refer to [MESSAGES Page \(MicroAutoBox III Hardware Installation and Configuration\)](#).

Additional error responses You can enable and configure the following additional error responses:

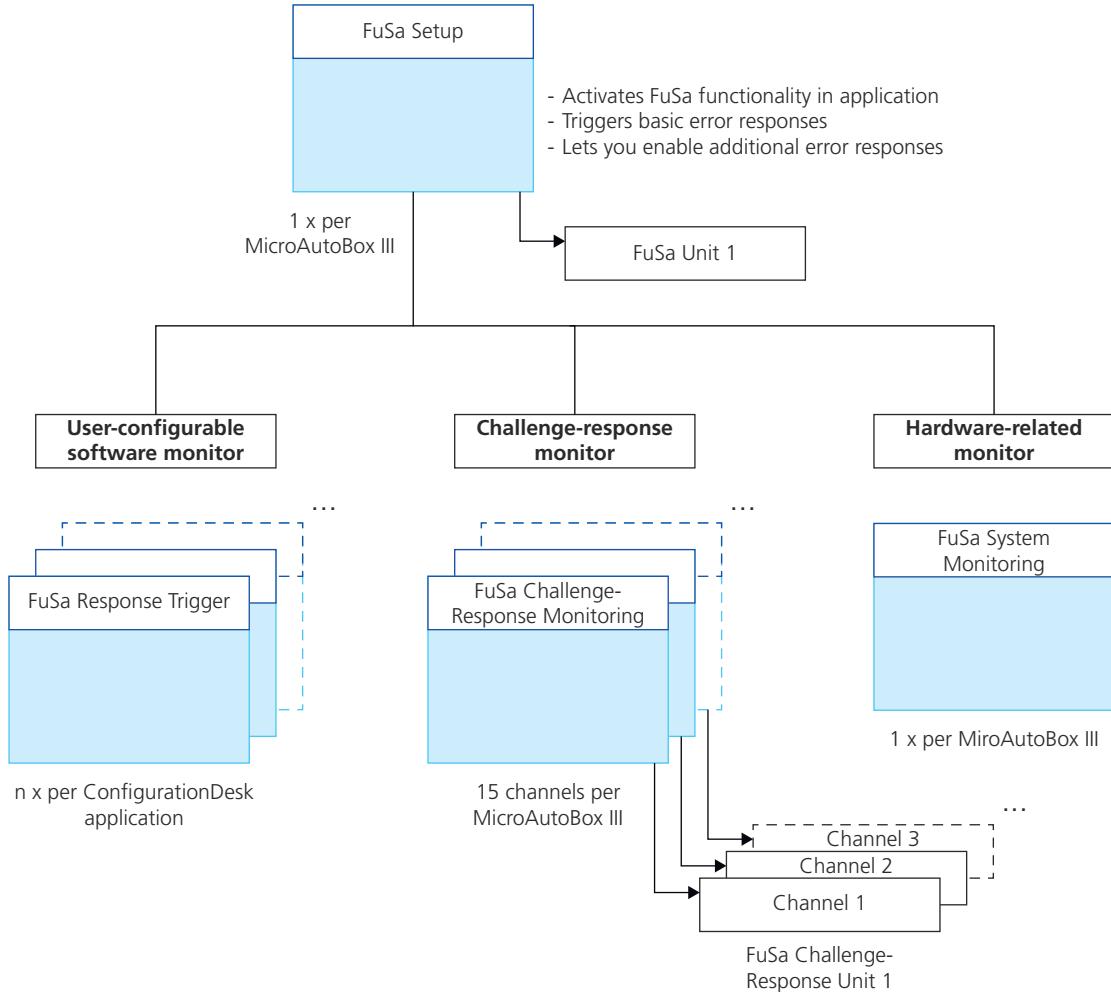
- Providing an I/O event to the behavior model
- Rebooting the MicroAutoBox III
- Terminating the currently running real-time application

Termination means the real-time application stops and the I/O is reset to its initial states. No more actions are performed.

You cannot use the latter two responses simultaneously in a real-time application.

Implementing the FuSa functionality

ConfigurationDesk provides specific function blocks to support the FuSa functionality of the MicroAutoBox III. Refer to the following illustration.



- The FuSa Setup function block (one per MicroAutoBox III) provides the basic functionality for implementing functional safety in your system.
- Each FuSa Response Trigger function block works as a user-configurable software monitor and can be used to trigger an error response directly by software from within the behavior model.
- Up to 15 FuSa Challenge-Response Monitoring function blocks can be used per MicroAutoBox III. The function block works as a challenge-response monitor. Based on the challenge and response principle, you can implement advanced monitoring services, for example, monitor periodic tasks or the correct execution of subsystems.
- Only one FuSa System Monitoring function block can be used for each MicroAutoBox III. The function block can be used to detect hardware-related problems, for example, if the internal operating temperature of the MicroAutoBox III exceeds a defined upper limit or the operating voltage is out of a defined range.

Overviews (FuSa Setup)

Where to go from here

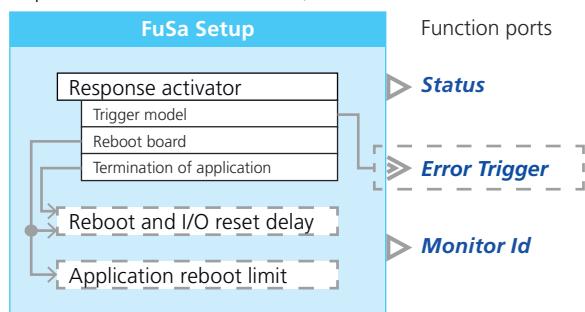
Information in this section

- | | |
|--------------------------------------------------------------------------|------|
| Overview of Ports and Basic Properties (FuSa Setup)..... | 1432 |
| Overview of Tunable Properties (FuSa Setup)..... | 1433 |

Overview of Ports and Basic Properties (FuSa Setup)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Status

This function output port writes status information concerning the functional safety to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: Inactive or error Functional safety is not activated on your system or a FuSa error has been detected. ▪ 1: Active and no error Functional safety is activated on your system and no FuSa error has been detected. This value also is used as the initial value.
Dependencies	—

Error Trigger

This event port provides an I/O event as soon as a FuSa error is detected.

Value range	—
Dependencies	Available only if the Trigger model setting is selected via the Response activator property.

Monitor Id

This function output writes the identification number of the monitor to the behavior model that detects the FuSa error.

If more than one monitor detects a FuSa error (almost simultaneously or consecutively), the function port provides only the ID of the monitor that actually detects the error first.

Value range	<ul style="list-style-type: none"> ▪ 1 ... 4,294,967,295 ▪ 0: A FuSa error is not detected. 0 (zero) cannot be used as an identification number for a monitor.
Dependencies	—

Overview of Tunable Properties (FuSa Setup)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	—	—
Function Ports		
Initial switch setting (Test Automation)	—	✓
Initial substitute value (Test Automation)	—	✓
Electrical Interface	—	—

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (FuSa Setup)

Where to go from here

Information in this section

Configuring the Basic Functionality (FuSa Setup).....	1434
Configuring Standard Features (FuSa Setup).....	1435

Configuring the Basic Functionality (FuSa Setup)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Enabling additional error responses
- Specifying a delay time for rebooting and resetting the I/O
- Specifying a reboot limit

Enabling additional error responses

If a FuSa error is detected, the basic error responses are triggered. You cannot configure them. For more information, refer to [Basics on Using FuSa with the MicroAutoBox III](#) on page 1429.

In addition to the basic error responses, you can enable additional error responses:

- Providing an I/O event to the behavior model
 - Rebooting MicroAutoBox III or terminating the running real-time application
- You cannot use the latter two responses simultaneously. If you select both related settings (Reboot board and Termination of application), a conflict is generated and displayed in the Conflicts Viewer.

Providing an I/O event to the behavior model An I/O event is generated as soon as a FuSa error is detected.

This I/O event can then be used as a trigger source for runnable functions in the behavior model. For this, the Error Trigger function port is added to the function block.

Rebooting the MicroAutoBox III If a real-time application is loaded to the flash memory and a FuSa error is detected, MicroAutoBox III is started again after a delay time specified via the Reboot and I/O reset delay property.

During the restart process the open FuSa relay is closed and the FuSa LED is lit green.

You can use the **Application reboot limit** property to define how often to restart the real-time application after the MicroAutoBox III was rebooted: never, each time, or up to a specified limit.

Terminating the real-time application If a FuSa error is detected, the real-time application is terminated after a delay specified via the **Reboot and I/O reset delay** property. Termination means the real-time application stops and the I/O is reset to its initial states. No more actions are performed. The application is still loaded, but you have to download it again to start it again.

The FuSa relay is closed and the FuSa LED lights up green if you download the real-time application to the hardware again and no FuSa error is detected.

Specifying a delay time for rebooting and resetting the I/O

You can specify a delay time between the detection of the FuSa error and the triggering of the following error responses:

- The MicroAutoBox III is rebooted.
- The real-time application is terminated and the I/O is reset to its initial states.

Note

Other error responses (e.g., switching the FuSa relay) are not delayed by this setting.

Specifying a reboot limit

You can specify how often to restart the real-time application (loaded to the flash memory) after the MicroAutoBox III was rebooted after a FuSa error. You can select from the following options:

- -1: The real-time application always starts again (unlimited restarts).
- 0: The real-time application never starts again.
- 1 ... 32767: The real-time application starts again until the specified number of restarts is exceeded.

Configuring Standard Features (FuSa Setup)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The electrical interface of the FuSa Setup function block does not provide standard configuration features.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

FuSa Response Trigger

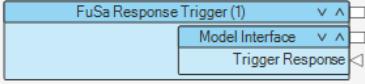
Where to go from here

Information in this section

Introduction (FuSa Response Trigger).....	1438
Overviews (FuSa Response Trigger).....	1439
Configuring the Function Block (FuSa Response Trigger).....	1441

Introduction (FuSa Response Trigger)

Introduction to the Function Block (FuSa Response Trigger)

Function block purpose	Each FuSa Response Trigger function block works as a user-configurable software monitor to implement functional safety in your application. If you use this function block, a Functional Safety (FuSa) error can be triggered directly by software from within the behavior model.
Default display	The following illustration shows the default display of the function block. 
Main features	These are the main features: <ul style="list-style-type: none">▪ Triggering the FuSa response from within the behavior model
Recommended reading	Before you implement the function block in your ConfigurationDesk application, you should be familiar with the basics on implementing functional safety. Refer to Basics on Using FuSa with the MicroAutoBox III on page 1429.
Available demo project	ConfigurationDesk provides the <i>FunctionalSafetyDemo</i> project. This project shows an implementation of the FuSa functionality also using the FuSa Setup and FuSa Challenge-Response Monitoring function blocks. You can use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to FunctionalSafetyDemo Project: Example of Using MicroAutoBox III Functional Safety Features (ConfigurationDesk Demo Projects) .

Overviews (FuSa Response Trigger)

Where to go from here

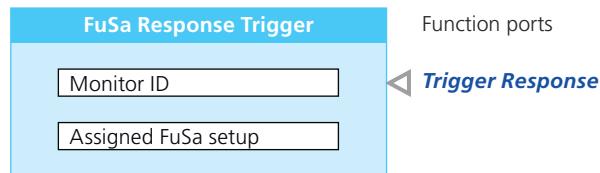
Information in this section

- | | |
|-------------------------------------------------------------------------------------|------|
| Overview of Ports and Basic Properties (FuSa Response Trigger)..... | 1439 |
| Overview of Tunable Properties (FuSa Response Trigger)..... | 1439 |

Overview of Ports and Basic Properties (FuSa Response Trigger)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Trigger Response

This function import receives a trigger from within the behavior model to trigger a FuSa response.

Value range	<ul style="list-style-type: none"> ▪ 0: The FuSa response is not triggered. ▪ 1: The FuSa response is triggered via the assigned FuSa Setup function block.
Dependencies	—

Overview of Tunable Properties (FuSa Response Trigger)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	—	—

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (FuSa Response Trigger)

Where to go from here

Information in this section

- | | |
|----------------------------------------------------------------------------------|------|
| Configuring the Basic Functionality (FuSa Response Trigger)..... | 1441 |
| Configuring Standard Features (FuSa Response Trigger)..... | 1442 |

Configuring the Basic Functionality (FuSa Response Trigger)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Specifying a monitor ID
- Assigning the FuSa Setup function block

Specifying a monitor ID

You have to specify an identification number for each monitor. After a FuSa error has been detected, you can use this number to identify the monitor that has detected the FuSa error and subsequently the cause of the error.

All monitors (such as challenge-response monitors or user-defined software monitors) used in your application for the FuSa functionality are treated identically. Their identification numbers do not have to be unique in your application. By specifying the same ID for multiple monitors, you can divide the monitors into categories of your choice. As a result, monitors with the same ID cannot be distinguished, for example, in error messages.

If a monitor detects a FuSa error, the ID of the affected monitor is reported to the behavior model via the Monitor Id function port (available at the FuSa Setup function block). If several monitors detect a FuSa error (almost simultaneously or consecutively), the function port provides only the ID of the monitor that actually detects the error first.

Assigning the FuSa Setup function block

You have to assign the FuSa Setup function block that resides in your active ConfigurationDesk application. The FuSa Setup function block:

- Provides access to the hardware resource supporting functional safety on the MicroAutoBox III.
- Triggers basic error responses and lets you configure additional error responses.

Configuring Standard Features (FuSa Response Trigger)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface This function block does not provide an electrical interface.

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

FuSa Challenge-Response Monitoring

Where to go from here

Information in this section

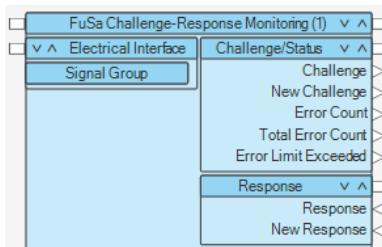
Introduction (FuSa Challenge-Response Monitoring).....	1444
Overviews (FuSa Challenge-Response Monitoring).....	1446
Configuring the Function Block (FuSa Challenge-Response Monitoring).....	1450

Introduction (FuSa Challenge-Response Monitoring)

Introduction to the Function Block (FuSa Challenge-Response Monitoring)

Function block purpose	Each FuSa Challenge-Response Monitoring function block works as a challenge-response monitor to support functional safety in your application. Based on the challenge and response principle, you can implement advanced monitoring services, for example, monitor periodic tasks or the correct execution of subsystems.
-------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Default display	The following illustration shows the default display of the function block.
------------------------	-----------------------------------------------------------------------------



Main features	<p>These are the main features:</p> <ul style="list-style-type: none"> ▪ Implementing a challenge-response monitor on the basis of configurable challenge and response values ▪ Specifying timing characteristics ▪ Triggering the FuSa response if timing errors are detected or invalid values are received from the behavior model
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Supported channel types	The FuSa Challenge-Response Monitoring function block type supports the following channel types:									
<table border="1"> <thead> <tr> <th></th> <th>SCALEXIO</th> <th>MicroAutoBox III</th> </tr> </thead> <tbody> <tr> <td>Channel type</td> <td>–</td> <td>FuSa Challenge-Response Monitoring Unit 1</td> </tr> <tr> <td>Hardware</td> <td>–</td> <td>DS1403 Processor Board</td> </tr> </tbody> </table>			SCALEXIO	MicroAutoBox III	Channel type	–	FuSa Challenge-Response Monitoring Unit 1	Hardware	–	DS1403 Processor Board
	SCALEXIO	MicroAutoBox III								
Channel type	–	FuSa Challenge-Response Monitoring Unit 1								
Hardware	–	DS1403 Processor Board								

Recommended reading	Before you implement the function block in your ConfigurationDesk application, you should be familiar with the basics on implementing functional safety. Refer to Basics on Using FuSa with the MicroAutoBox III on page 1429.
----------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Available demo project

ConfigurationDesk provides the *FunctionalSafetyDemo*. This project shows an implementation of the FuSa functionality also using the FuSa Response Trigger and FuSa Challenge-Response Monitoring function blocks. You can use the project as a template, for example, to copy parts of it to your application. For a description of the project, refer to [FunctionalSafetyDemo Project: Example of Using MicroAutoBox III Functional Safety Features \(ConfigurationDesk Demo Projects\)](#).

Overviews (FuSa Challenge-Response Monitoring)

Where to go from here

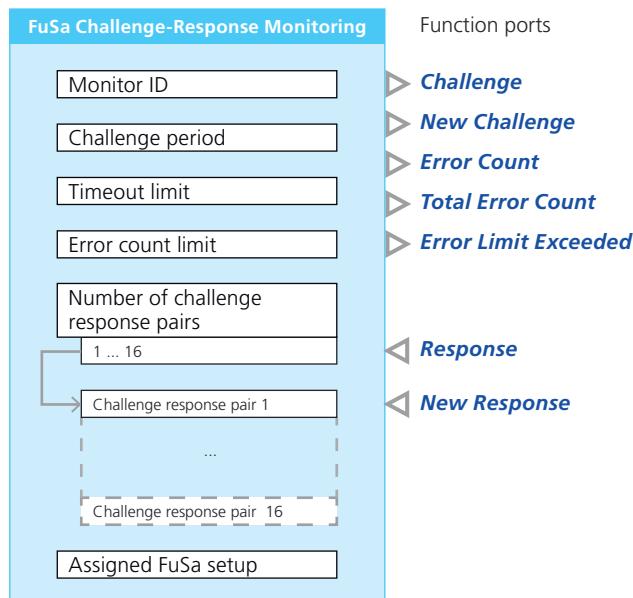
Information in this section

Overview of Ports and Basic Properties (FuSa Challenge-Response Monitoring).....	1446
Overview of Tunable Properties (FuSa Challenge-Response Monitoring).....	1449

Overview of Ports and Basic Properties (FuSa Challenge-Response Monitoring)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Challenge

This function outport writes the current challenge value to the behavior model. The value is updated only once every challenge period and kept for one model step. Then it is reset to 0. The challenge-response monitor strictly sequentially selects the value for the challenge from the defined challenge-response pair list.

Value range

- 1 ... 4,294,967,295: The value of the new challenge that is specified via challenge-response pair.

	<ul style="list-style-type: none"> ▪ 0: There is no new challenge value available since the last model step. 0 (zero) is also used as initial value.
Dependencies	—

New Challenge

This function outport provides a flag to the behavior model indicating whether the provided challenge value is new. The value is updated once every challenge period and kept for one model step. Then it is reset to 0.

Value range	<ul style="list-style-type: none"> ▪ 0: A new challenge value is not available. The provided value at the Challenge function port is not new compared to the value provided in the previous model step. The behavior model must not response to the value available at the Challenge function port. 0 (zero) is also used as initial value. ▪ 1: A new challenge value is available. The provided value at the Challenge function port is new compared to the value provided in the previous model step. The behavior model must response to the value available at the Challenge function port.
Dependencies	—

Error Count

This function outport writes the number of successive detected errors to the behavior model. The *error counter* is reset if the associated challenge-response monitor received a valid response value within the specified timeout (at **Timeout limit** property).

Only the first response value in the current challenge period is evaluated. The *error counter* is incremented in the following cases:

- The response value returned by the behavior model to the challenge-response monitor does not match the expected value.
- The behavior model did not return the response value to the challenge-response monitor in time, i.e., not before the specified timeout value (at **Timeout limit** property) expired.
- The behavior model did not return any value for the response to the challenge-response monitor within the current challenge period.

Value range	0 ... 4,294,967,295
Dependencies	—

Total Error Count

This function outport writes the total number of detected errors to the behavior model. In contrast to the *error counter*, the *total error counter* is only reset in the following cases:

- If you download the real-time application again.
- If MicroAutoBox III has been rebooted and the real-time application that is loaded to the flash memory starts running.

Only the first response value in the current challenge period is evaluated. The *total error counter* is incremented in the following cases:

- The response value returned by the behavior model to the challenge-response monitor does not match the expected value.
- The behavior model did not return the response value to the challenge-response monitor in time, i.e., not before the specified timeout value (at Timeout limit property) expired.
- The behavior model did not return any value for the response to the challenge-response monitor within the current challenge period.

Value range	0 ... 4,294,967,295
Dependencies	—

Error Limit Exceeded

This function outport provides a flag to the behavior model, indicating whether the current value of the *error counter* has exceeded the specified error limit for the associated challenge-response monitor.

Once set, the flag is not reset until the real-time application is reloaded. You can use the flag to control your own additional error response, for example, for debugging purposes or for other monitoring entities that are not safety-relevant.

Value range	<ul style="list-style-type: none"> ▪ 0: Error limit not exceeded. The value of the <i>error counter</i> has not exceeded the value specified at the Error count limit property. ▪ 1: Error limit exceeded. The value of the <i>error counter</i> exceeded the value specified at the Error count limit property.
Dependencies	—

Response

This function import reads the value that is the response to the current challenge from the behavior model.

Value range	0 ... 4,294,967,295
Dependencies	—

New Response

This function import controls, if the response value available at the **Response** function port must be processed by the associated challenge-response monitor.

The behavior model must send a new response value to the challenge-response monitor only once. It is therefore necessary that a response value is marked as a new value for only one model step.

Value range	<ul style="list-style-type: none"> ▪ 0: No new response value is available. The provided value at the Response function port is not new. The challenge-response monitor must not process the value available at the Response function port.
-------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ 1: A new response value is available. The provided value at the Response function port is new. The challenge-response monitor must process the value available at the Response function port.
Dependencies	—

Overview of Tunable Properties (FuSa Challenge-Response Monitoring)

Tunable properties

Tunable properties can be accessed and modified in the experiment software.
The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	—	—
Function Ports		
Initial switch setting (Test Automation)	—	✓
Initial substitute value (Test Automation)	—	✓
Electrical Interface	—	—

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (FuSa Challenge-Response Monitoring)

Where to go from here

Information in this section

Configuring the Basic Functionality (FuSa Challenge-Response Monitoring).....	1450
Configuring Standard Features (FuSa Challenge-Response Monitoring).....	1455

Configuring the Basic Functionality (FuSa Challenge-Response Monitoring)

Overview

The function block type provides the following configuration features, which influence the behavior of the basic functionality:

- Specifying challenge and response values
- Specifying timing characteristics (challenge period, timeout limit)
- Specifying an error count limit
- Specifying a monitor ID
- Assigning a FuSa Setup function block

Specifying challenge and response values

The challenge-response monitor works on the basis of a fixed list of value pairs. A value pair assigns the value for the challenge to the expected value for the response. You have to specify the number of challenge and response value pairs (up to 16) for the challenge-response monitor and then a challenge and response value for each available challenge-response pair.

At run time, the challenge response monitor takes the challenge value from the first line of the list and provides it to the behavior model via the **Challenge** function port. If the behavior model returns a value for the response in time (via the **Response** function port), the system compares the received value with the value for the response that is stored in the list in the same line as the challenge value.

At the beginning of the next challenge period, the system takes the challenge value from the second line of the list and also provides it to the behavior model, and so on. This way, the challenge-response monitor processes the complete list of challenge and response value pairs line by line. When the end of the list is reached, the processing starts again with the first line.

The following example, shows a list with 6 challenge-response pairs:

Number of challenge response pairs	6
Challenge response pair 1	1; 100
Challenge response pair 2	2; 200
Challenge response pair 3	3; 350
Challenge response pair 4	35; 400
Challenge response pair 5	5; 500
Challenge response pair 6	6; 6

For specifying challenge and response values the following rules apply:

- In the pair list, all values for the challenges must be different. Double values are not allowed.
- In the pair list, all values for the responses must be different. Double values are not allowed.
- In a value pair, the value for the challenge and the value for the response might be the same.
- None of the value pairs must contain the value 0 (zero).
- The list must not contain empty lines. This means that all lines in the list must be filled with a value greater than 0 (zero).

If you specify invalid values, a conflict is generated and displayed in the **Conflicts Viewer**.

Reading challenges and writing responses can be controlled via the **New Challenge** and **New Response** functions ports. The **New Challenge** function port indicates that a new challenge value is available for the behavior model. The **New Response** function ports indicates that a new response value is available and must be processed by the associated challenge-response monitor.

Specifying timing behavior

For implementing stable monitoring, you have to specify timing characteristics and timing constraints that must be fulfilled:

Specifying the challenge period You have to specify the challenge period, that is the time interval in which the associated challenge-response monitor provides a new challenge to the behavior model. For implementing stable monitoring, the following rules apply for the challenge period:

- The specified value must be greater than the value specified for the **Timeout limit** property.
- To ensure that each challenge can be processed in the behavior model, the challenge period must be at least twice as long as the model step of the task that is used to evaluate the challenge value. However, a factor significantly greater than twice is recommended.

Specifying a timeout limit You have to specify a timeout limit, that is the maximum time that may elapse between the time when the function block has provided a new value for the challenge (via the **Challenge** function port) and the time when the behavior model returns the expected value for the response to the function block (via the **Response** function port).

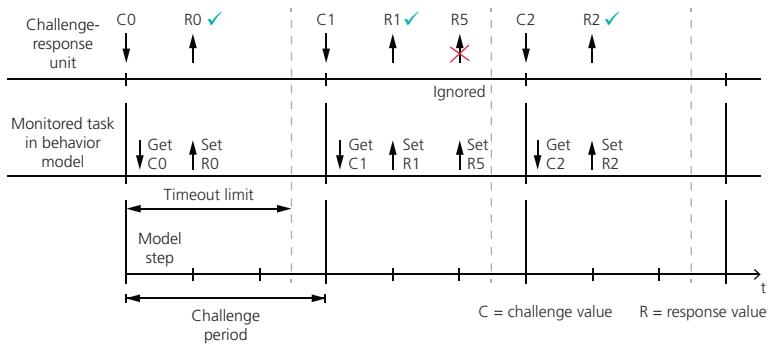
The following rules apply for the timeout limit:

- The specified value must be less than the value specified for the **Challenge period** property.

- The specified value must be greater than the model step of the task that is used to evaluate the challenge value.

At least one response is to be sent within the response timeout. If multiple responses were sent, only the first one is evaluated.

Timing constraints are fulfilled The expected behavior of sending and receiving response values is shown in the following example. In the second challenge period, you can see that only the first response value (R1) is received and evaluated.



Specifying error behavior

The FuSa hardware of MicroAutoBox III has two error counters to count the errors detected during challenge-response monitoring. Only the first response value in the current challenge period is evaluated. The error counters are incremented in the following cases:

- The response value returned by the behavior model to the challenge-response monitor does not match the expected value.
- The behavior model did not return the response value to the challenge-response monitor in time, i.e., not before the specified timeout value (at Timeout limit property) expired.
- The behavior model did not return any value for the response to the challenge-response monitor within the current challenge period.

The values of the error counters are provided to the behavior model as follows:

- Via Error Count function port

This function output writes the number of successive detected errors to the behavior model. The related *error counter* is reset if the associated challenge-response monitor received a valid response value.

For the successive detected errors, you have to specify the maximum number of successive errors to be ignored before the FuSa response is triggered (at Error count limit property). For example, if you specify the value 0, the first detected error immediately triggers the FuSa response.

If the number of successive detected errors exceeds the specified value the following applies:

- The FuSa response is triggered.
- A status flag is reported via Error Limit Exceeded function port to the behavior model.

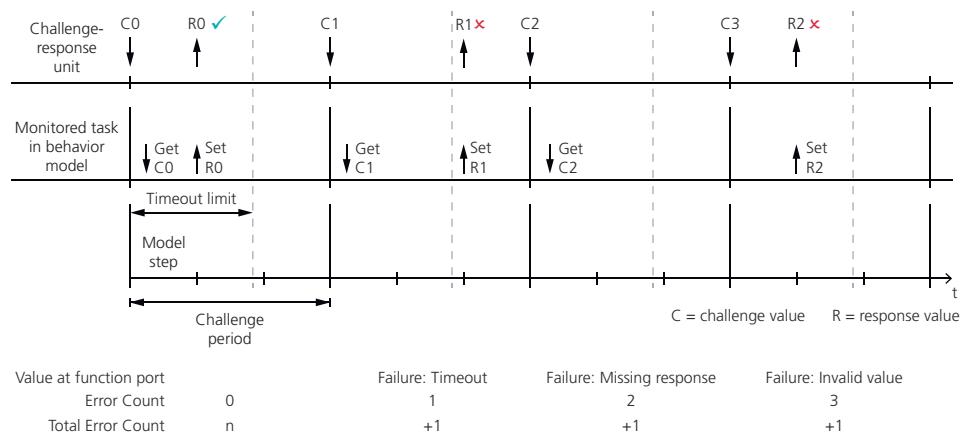
Once set, the flag is not reset until the real-time application is reloaded. You can use the flag to control your own additional error response, for example, for debugging purposes or for other monitoring entities that are not safety-relevant.

- **Via Total Error Count function port**

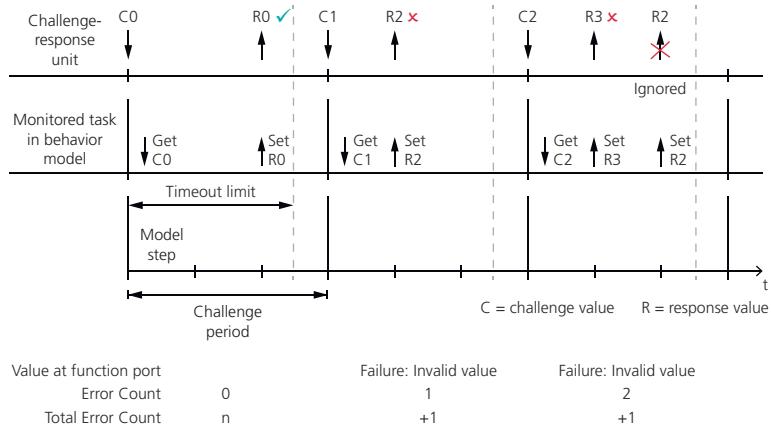
This function output writes the total number of detected errors to the behavior model. The related *total error counter* is reset only in the following cases:

- If you download the real-time application again.
- If MicroAutoBox III has been rebooted and the real-time application that is loaded to the flash memory starts running.

Timeout errors The following illustration shows two examples. In the second challenge period, the response value R1 is sent within the same challenge period but the response timeout is exceeded. This leads to an error and the error counters are incremented. In the third challenge response, the response value R2 exceeds not only the response timeout but also the challenge period. This leads to two errors because the response is missing in the related challenge period and the sent response value is invalid for the current challenge period.



Errors caused by invalid values The following illustration shows that an error is detected if the response value does not match the challenge value. In the third challenge period, you can see that only the first response value R3 is received and evaluated. The error status will not be reset by the following valid response value R2.



Specifying a monitor ID

You have to specify an identification number for each monitor. After a FuSa error has been detected, you can use this number to identify the monitor that has detected the FuSa error and subsequently the cause of the error.

All monitors (such as challenge-response monitors or user-defined software monitors) used in your application for the FuSa functionality are treated identically. Their identification numbers do not have to be unique in your application. By specifying the same ID for multiple monitors, you can divide the monitors into categories of your choice. As a result, monitors with the same ID cannot be distinguished, for example, in error messages.

If a monitor detects a FuSa error, the ID of the affected monitor is reported to the behavior model via the **Monitor Id** function port (available at the **FuSa Setup** function block). If several monitors detect a FuSa error (almost simultaneously or consecutively), the function port provides only the ID of the monitor that actually detects the error first.

Assigning the FuSa Setup function block

You have to assign the **FuSa Setup** function block that resides in your active ConfigurationDesk application. The **FuSa Setup** function block:

- Provides access to the hardware resource supporting functional safety on the MicroAutoBox III.
- Triggers basic error responses and lets you configure additional error responses.

Settings for the build process

To avoid timeout errors, do not use the following settings in the build process for your real-time application:

- Do not specify an offset for the timer event of the task that is used to evaluate the challenge value. The offset is the delay time between the simulation starting and the timer event occurring.

You can access the **Offset** property via Task view set – Task Configuration – Period Task – Timer Event – Run-Time Behavior.

- Do not change the default values for the task startup behavior for the task that is used to evaluate the challenge value:

- Time-scaled period: 0 (default setting)
- Time scale factor: 1 (default setting)

You can access the properties via Build view set – Build configuration – Global Build Settings.

The properties also can be accessed and modified in the experiment software. Do not change the default settings there either.

Configuring Standard Features (FuSa Challenge-Response Monitoring)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
--------------	------------------------------------------------------------------------------------------------------------------------------

Electrical interface	The electrical interface of the FuSa Challenge-Response Monitoring function block does not provide standard configuration features.
-----------------------------	-------------------------------------------------------------------------------------------------------------------------------------

Model interface	The interface to the behavior model of the function block provides the following standard features.
------------------------	-----------------------------------------------------------------------------------------------------

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

FuSa System Monitoring

Where to go from here

Information in this section

Introduction (FuSa System Monitoring).....	1458
Overviews (FuSa System Monitoring).....	1459
Configuring the Function Block (FuSa System Monitoring).....	1460

Introduction (FuSa System Monitoring)

Introduction to the Function Block (FuSa System Monitoring)

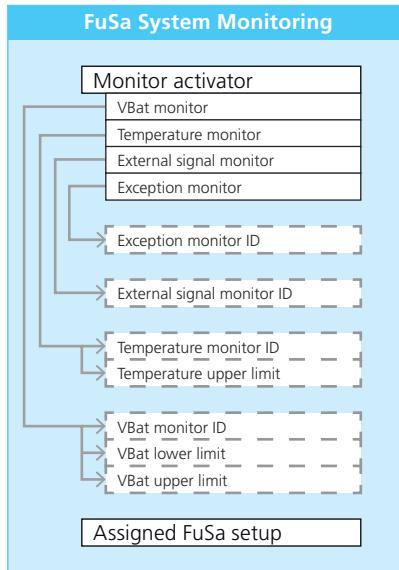
Function block purpose	The FuSa System Monitoring function block works as a hardware-related monitor to enhance functional safety in your application. If you use this function block, a Functional Safety (FuSa) error can be triggered directly by any activated monitor, for example, if the internal operating temperature of the MicroAutoBox III exceeds a defined upper limit or the operating voltage is out of a defined range.
Default display	The following illustration shows the display of the function block. 
Main features	These are the main features: <ul style="list-style-type: none">▪ Activating one or more hardware-related monitors▪ Triggering the FuSa response from the activated hardware-related monitor
Recommended reading	Before you implement the function block in your ConfigurationDesk application, you should be familiar with the basics on implementing functional safety. Refer to Basics on Using FuSa with the MicroAutoBox III on page 1429.

Overviews (FuSa System Monitoring)

Overview of Ports and Basic Properties (FuSa System Monitoring)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Ports

The FuSa System Monitoring function block type does not provide signal ports or function ports.

Tunable properties

The FuSa System Monitoring function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuring the Function Block (FuSa System Monitoring)

Configuring the Basic Functionality (FuSa System Monitoring)

Overview

The function block type provides the following configuration features, that influence the behavior of the basic functionality:

- Activating one or more hardware-related system monitors
- Specifying a monitor ID
- Assigning the FuSa Setup function block

Activating a system monitor

You can activate one or more of the following monitors. If activated, a FuSa response is triggered immediately when the corresponding monitor detects a FuSa error.

- VBat monitor:

This monitor constantly checks the operating voltage of the MicroAutoBox III. This VBAT input voltage is measured at the power input connector of the DS1403 Processor Board.

Via the VBat upper limit and the VBat lower limit properties you have to specify a range required for stable operation. If the measured voltage value is outside the specified range, a FuSa response is triggered immediately. To specify voltage limits you must follow the recommendations below.

- Temperature monitor:

This monitor constantly checks the internal operating temperature of the MicroAutoBox III. With the Temperature upper limit property, you have to specify a maximum temperature. If the measured temperature exceeds the specified limit, a FuSa response is triggered immediately.

Note

The upper temperature limit (default setting = 90 °C) applies only for triggering a FuSa response. Regardless of this response, the MicroAutoBox III has a further upper temperature limit of 104 °C. If the internal operating temperature of MicroAutoBox III exceeds this limit, a warning message is sent to the host PC. If the temperature remains this high for more than 5 minutes, MicroAutoBox III switches itself off to prevent damage to the hardware.

- External signal monitor:

This monitor can be used to monitor the external FuSa input of MicroAutoBox III. The external input is the SAFETY IN pin of the MicroAutoBox III power input connector. If the signal at this pin switches from a low to a high potential (>+ 4.7 V) a FuSa response is triggered immediately.

By using this monitor you can integrate an external device in the FuSa concept of MicroAutoBox III.

- Exception monitor:

This monitor can be used to monitor the processing in a real-time application. If a serious exception occurs, for example, an exception caused by a memory violation (also called segmentation fault), an internal bus error, or an invalid instruction, a FuSa response is triggered immediately.

Note

When the exception monitor triggers a FuSa response, the additional error responses (specified via the Response activator property of the FuSa Setup function block), may behave differently than configured. The following behavior applies:

- If an I/O event is to be generated when such a FuSa error is detected (Trigger model setting), the connected executable function (task) in the behavior model is not executed.
- If such a FuSa error is detected, the real-time application is always terminated (stopped) and the I/O is reset to its initial states. This happens even if you did not select the Termination of application error response setting.
- If you specified an I/O reset delay via the Reboot and I/O reset delay property, the specified value is ignored. The I/O is always reset immediately after such a FuSa error is detected.

Recommendations for specifying the VBat voltage limits

By default, the VBat voltage range is set from 10 V (VBat lower limit) to 36 V (VBat upper limit). You can change this range according to the requirements of the electrical system in your vehicle. However, you have to consider the following recommendations:

- **Note**

The maximum recommended operating voltage of the MicroAutoBox III is 36 V (= default setting). However, it is possible to run the MicroAutoBox III with an operating voltage of up to 40 V.

- The minimum recommended operating voltage of the MicroAutoBox III is 10 V (= default setting). You can also run the MicroAutoBox III with an operating voltage of only 6 V. However, due to the increased current requirement at this voltage, this is not permitted for continuous operation.

Specifying a monitor ID

You have to specify an identification number for each monitor. After a FuSa error has been detected, you can use this number to identify the monitor that has detected the FuSa error and subsequently the cause of the error.

All monitors (such as challenge-response monitors or user-defined software monitors) used in your application for the FuSa functionality are treated identically. Their identification numbers do not have to be unique in your

application. By specifying the same ID for multiple monitors, you can divide the monitors into categories of your choice. As a result, monitors with the same ID cannot be distinguished, for example, in error messages.

If a monitor detects a FuSa error, the ID of the affected monitor is reported to the behavior model via the **Monitor Id** function port (available at the **FuSa Setup** function block). If several monitors detect a FuSa error (almost simultaneously or consecutively), the function port provides only the ID of the monitor that actually detects the error first.

Assigning the FuSa Setup function block

You have to assign the **FuSa Setup** function block that resides in your active ConfigurationDesk application. The **FuSa Setup** function block:

- Provides access to the hardware resource supporting functional safety on the MicroAutoBox III.
- Triggers basic error responses and lets you configure additional error responses.

Domain Clock

Where to go from here

Information in this section

Introduction (Domain Clock).....	1464
Overviews (Domain Clock).....	1467
Configuration (Domain Clock).....	1469

Introduction (Domain Clock)

Where to go from here

Information in this section

Introduction to the Function Block (Domain Clock).....	1464
Basics on Using the Domain Clock Function Block.....	1465
bas	

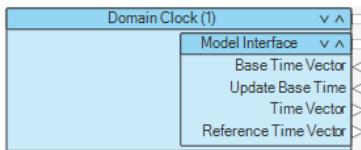
Introduction to the Function Block (Domain Clock)

Function block purpose

The Domain Clock function block represents a domain clock. The base time can be set and is updated by the processing hardware.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Providing a local domain clock.
- Time base can be set via the behavior model.
- Providing reference time values to compensate for delays.

Supported hardware

The Domain Clock function block type supports the following processing hardware:

- SCALEXIO Processing Unit
- SCALEXIO LabBox with a DS6001 Processor Board installed
- SCALEXIO AutoBox with a DS6001 Processor Board installed
- MicroAutoBox III

Basics on Using the Domain Clock Function Block

Presentation of time values

ConfigurationDesk uses a time vector to represent a time value with high accuracy. Two entries represent one time value:

- The first entry represents the seconds.
- The second entry represents the nanoseconds.

The following example shows a time vector and the represented time value.

$$\left(\begin{array}{c} 1\,234\,\text{s} \\ 567\,000\,000\,\text{ns} \end{array} \right) \rightarrow 1,234\,\text{s} + 567,000,000\,\text{ns} = 1,234.567\,\text{s}$$

Value range of the nanosecond value If you provide a time vector, you have to make sure that only time values $< 1\,\text{s}$ are provided by the second entry of the time vector.

The function block uses only the last 9 digits of the nanosecond value and discards higher order digits as shown in the following illustration.

$$\left(\begin{array}{c} 5\,\text{s} \\ 1\,200\,000\,000\,\text{ns} \end{array} \right) \rightarrow \left(\begin{array}{c} 5\,\text{s} \\ \textcolor{red}{+}200\,000\,000\,\text{ns} \end{array} \right) \rightarrow 5.2\,\text{s}$$

Specifyinge the base time

With the default settings, the domain clock outputs the local time.

To specify a specific base time, you can set a new time value $> 0\,\text{s}$ via the Base Time Vector function port. A 0 to 1 transition at the Update Base Time function port updates the base time and sets the new time value.

Note

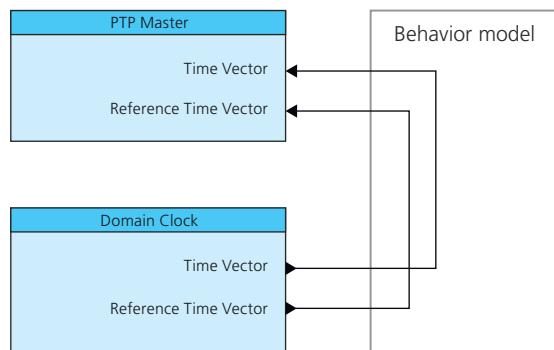
The Base Time Vector function port interprets the time vector $(0;0)$ as local time.

- To set the base time back to the local time, provide the time vector $(0;0)$ to the Base Time Vector function port.

Use of reference time values

A reference time value represents the point in time when a time value is captured. The reference time is intended to compensate for time delays caused by the processing of the signal.

The following example shows a Domain Clock function block used as time source for a PTP Master function block.



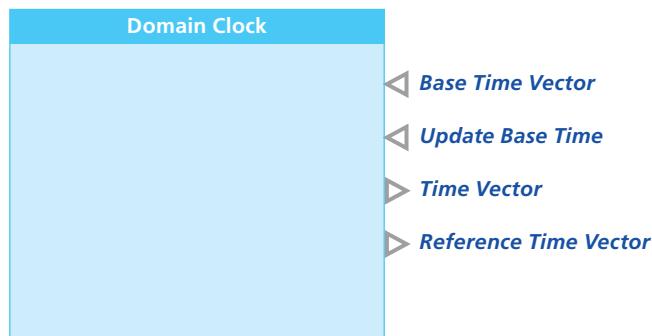
With the reference time values, the PTP Master function block can calculate the time it takes to transmit the time values through the behavior model.

Overviews (Domain Clock)

Overview of Ports and Basic Properties (Domain Clock)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Base Time Vector

This function import reads the starting time to set a new base time of the domain clock.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 281,474,976,710,655 for each entry of the vector ▪ The vector size is 2. Both entries represent one time value: ▪ The first entry represents the seconds. ▪ The second entry represents the nanoseconds. ▪ The function block reads only the last 9 digits of the nanosecond value and discards higher order digits. ▪ The time vector (0;0) specifies the local time as starting time.
Dependencies	–

Update Base Time

This function import lets you set a new base time for the domain clock.

Value range	<ul style="list-style-type: none"> ▪ 0, 1 ▪ A 0 to 1 transition updates the domain clock with the time value of the Base Time Vector port.
Dependencies	–

Time Vector

This function outport writes the current time value of the domain clock to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 2^{48} for each entry of the vector.
-------------	-------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ The vector size is 2. Both entries represent one time value: <ul style="list-style-type: none"> ▪ The first entry represents the seconds. ▪ The second entry represents the nanoseconds. <p>Only the last 9 digits of the nanosecond value are used.</p>
Dependencies	–

Reference Time Vector

This function outport writes the reference time value to the behavior model. A reference time value represents the point in time when the time value of the Time Vector function port was captured. The reference time value can be used for compensating time delays when processing the signals.

Value range	<ul style="list-style-type: none"> ▪ 0 ... 2^{48} for each entry of the vector. ▪ The vector size is 2. Both entries represent one time value: <ul style="list-style-type: none"> ▪ The first entry represents the seconds. ▪ The second entry represents the nanoseconds. <p>Only the last 9 digits of the nanosecond value are used.</p>
Dependencies	–

Tunable properties

The Domain Clock function block type does not provide tunable properties.

Tunable properties can also be accessed and modified in the experiment software.

Configuration (Domain Clock)

Configuring Standard Features (Domain Clock)

Scope Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface This function block does not provide an electrical interface.

Model interface The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Non-Volatile Memory Access

Where to go from here

Information in this section

Introduction (Non-Volatile Memory Access).....	1472
Overviews (Non-Volatile Memory Access).....	1475
Configuring the Function Block (Non-Volatile Memory Access).....	1478

Introduction (Non-Volatile Memory Access)

Where to go from here

Information in this section

Introduction to the Function Block (Non-Volatile Memory Access)..... 1472

Basics on Using the Non-Volatile Memory of dSPACE Processing Hardware..... 1473

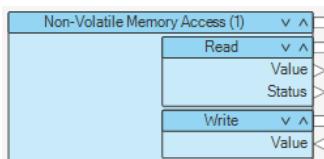
Introduction to the Function Block (Non-Volatile Memory Access)

Function block purpose

The Non-Volatile Memory Access function block provides access to the non-volatile memory of the processing hardware. The function block creates a data set in the non-volatile memory and handles the data transfer between the behavior model and the non-volatile memory for this data set.

Default display

The function block provides function ports but no signal ports. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Creating a data set with a unique name and specified data width and data type in the non-volatile memory of the processing hardware.
- Reading data from the non-volatile memory and providing the data to the behavior model.
- Writing data provided from within the behavior model to the non-volatile memory.

Supported hardware

The Non-Volatile Memory Access function block supports the following processing hardware:

- SCALEXIO LabBox with a DS6001 Processor Board installed
- SCALEXIO AutoBox with a DS6001 Processor Board installed
- MicroAutoBox III
- SCALEXIO Processing Unit is supported with the following limitation:

Data is written to the non-volatile memory when the real-time application is stopped, not during run time.

Basics on Using the Non-Volatile Memory of dSPACE Processing Hardware

Purpose of the non-volatile memory	The non-volatile memory is part of the memory provided by the processing hardware. Non-volatile memory keeps data after the processing hardware is powered down. You can use non-volatile memory, for example, to store the mileage or driver-specific settings, such as the seat position.
Characteristics	<p>Data that you want to write to the non-volatile memory or that you want to read from it must be provided by a data set. The non-volatile memory is structured like a flat file system of limited capacity where each file stores the content of a unique data set.</p> <p>The following characteristics and rules apply:</p> <ul style="list-style-type: none">▪ A data set in the non-volatile memory is identified by its name that must be unique in the non-volatile memory of a processing unit. The name of the data set is identical to the name of its related Non-Volatile Memory Access function block.▪ A data set can include up to 4,096 entries.▪ All entries of a data set are of the same data type.▪ The data rate for writing or reading a data set is approx. 10 MB/s.▪ Processing hardware such as SCALEXIO LabBox, SCALEXIO AutoBox, SCALEXIO Processing Units, and MicroAutoBox III can handle up to 64 data sets per processing unit application. <p>If you assign more than 64 Non-Volatile Memory Access function blocks to a processing unit application, a conflict is generated and displayed in the Conflicts Viewer.</p>
Working principle	<p>Writing data to the non-volatile memory Data is written to the non-volatile memory as follows:</p> <ol style="list-style-type: none">1. Data provided at the Value (Write) function port from within the behavior model is periodically stored in a RAM buffer of the processing hardware.2. Depending on the workload of the processing hardware and the number of data sets, the content of the non-volatile memory is updated every 10 ... 1,000 ms.

The update process of a data set is performed as a whole (atomically). Therefore, no corrupted data sets are created, even in the case of power loss.

Reading data from the non-volatile memory Data is read from the non-volatile memory only during application start and provided to the behavior model as follows:

1. Depending on the content of the non-volatile memory, the RAM buffer is filled with either of the following data:
 - If a data set with matching name, type, and size exists, data of that set is transferred to the RAM buffer.
 - If no data set with an identical name exists or if a data set with a matching name but of different type or size exists, the initial value from the Value (Read) function port is stored in every entry of the RAM buffer.
2. Data from the RAM buffer is available at the Value (Read) function port.

Overviews (Non-Volatile Memory Access)

Where to go from here

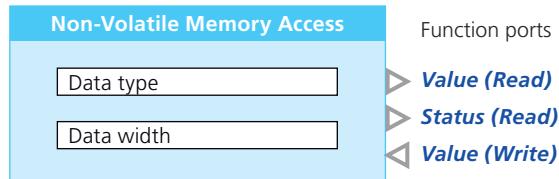
Information in this section

Overview of Ports and Basic Properties (Non-Volatile Memory Access).....	1475
Overview of Tunable Properties (Non-Volatile Memory Access)	1476

Overview of Ports and Basic Properties (Non-Volatile Memory Access)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Value (Read)

This function outport reads the data from the non-volatile memory and provides it to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ Value_{min} ... value_{max} for each entry of the vector. ▪ The minimum and maximum values depend on the data type specified at the Data type property. ▪ The vector size (the number of data elements of the vector) depends on the number specified at the Data width property.
Dependencies	–

Status (Read)

This function outport indicates whether the data set was created and provides this information to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ True: The specified data set was created. ▪ False: The specified data set was <i>not</i> created. As a result, the function block does not write data to the non-volatile
-------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>memory. At the Value (Read) function port, only data specified at the function ports Initial value property is provided.</p> <p>The Status (Read) function port is set to false in the following cases:</p> <ul style="list-style-type: none"> ▪ A data set of identical name but with different a data width and/or data type exists already in the non-volatile memory. ▪ The specified data set does not fit into the free non-volatile memory. <p>Both cases can occur if data sets from previously loaded applications still exist in the non-volatile memory of the processing hardware. To view and delete data sets in the non-volatile memory via the MicroAutoBox III web interface, refer to NVDATA Page (MicroAutoBox III Hardware Installation and Configuration).</p>
Dependencies	–

Value (Write)

This function import writes data provided from within the behavior model to the non-volatile memory.

Value range	<ul style="list-style-type: none"> ▪ $\text{Value}_{\min} \dots \text{value}_{\max}$ for each entry of the vector. ▪ The minimum and maximum values depend on the data type specified at the Data type property. ▪ The vector size depends on the number specified at the Data width property.
Dependencies	–

Overview of Tunable Properties (Non-Volatile Memory Access)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
----------	------------------------------------------	------------------------------------------

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Non-Volatile Memory Access)

Where to go from here

Information in this section

- [Configuring the Basic Functionality \(Non-Volatile Memory Access\)..... 1478](#)
- [Configuring the Standard Features \(Non-Volatile Memory Access\)..... 1478](#)

Configuring the Basic Functionality (Non-Volatile Memory Access)

Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Specifying the characteristics of the data set to be created.

Specifying the characteristics of the data set

You have to specify the data type and the data width of the data set. The function block creates a data set with a number of entries corresponding to the specified data width, each of the specified data type.

A specified data set might be too large for the non-volatile memory if previously created data sets still exist in the non-volatile memory. In this case, the new data set is not created and the Status function port is set accordingly. Refer to [Status \(Read\) on page 1475](#).

Configuring the Standard Features (Non-Volatile Memory Access)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

This function block does not provide an electrical interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Acceleration In

Where to go from here

Information in this section

Introduction (Acceleration In).....	1482
Overviews (Acceleration In).....	1483
Configuring the Function Block (Acceleration In).....	1485

Introduction (Acceleration In)

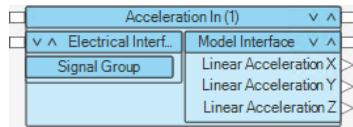
Introduction to the Function Block (Acceleration In)

Function block purpose

The Acceleration In function block type reads the measured acceleration and angular velocity values on three axes (x, y, z) from an onboard acceleration sensor and provides the values to the behavior model.

Default display

The function block provides function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Reading the measured acceleration values on three axes (x, y, z) from an onboard acceleration sensor.
- Using the gyroscope functionality of the onboard acceleration sensor to read angular velocity values on three axes (x, y, z).
- Providing the measured values to the behavior model.
- Specifying a sample rate and value ranges for the measurement.

Supported channel types

The Acceleration In function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	–	Acceleration Sensor Unit 1
Hardware	–	DS1403 Processor Board

Overviews (Acceleration In)

Where to go from here

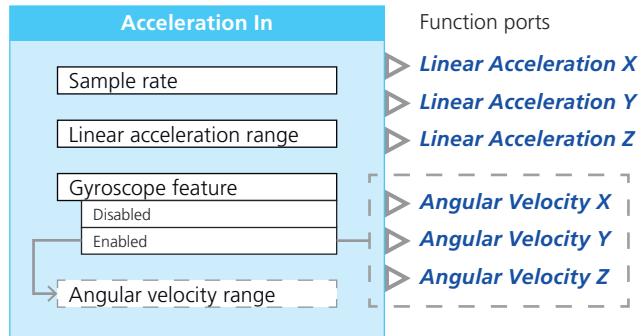
Information in this section

- [Overview of Ports and Basic Properties \(Acceleration In\).....](#) 1483
- [Overview of Tunable Properties \(Acceleration In\).....](#) 1484

Overview of Ports and Basic Properties (Acceleration In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Linear Acceleration X, Linear Acceleration Y, Linear Acceleration Z

This function outport writes the current measured acceleration value (of the x-, y-, or z-axis) from the acceleration sensor to the behavior model.

For the assignment of the axes to the housing of the MicroAutoBox III, refer to [Configuring the Standard Features \(Acceleration In\)](#) on page 1487.

Value range	Depends on the setting of the Linear acceleration range property: <ul style="list-style-type: none"> ▪ -19.6133 m/s² ... +19.6133 m/s² (at 2 g) ▪ -39.2266 m/s² ... +39.2266 m/s² (at 4 g) ▪ -78.4532 m/s² ... +78.4532 m/s² (at 8 g) ▪ -156.9064 m/s² ... +156.9064 m/s² (at 16 g)
Dependencies	-

Angular Velocity X, Angular Velocity Y, Angular Velocity Z

This function outport writes the current measured angular velocity value (of the x-, y-, or z-axis) from the acceleration sensor to the behavior model.

For the assignment of the axes to the housing of the MicroAutoBox III, refer to [Configuring the Standard Features \(Acceleration In\)](#) on page 1487.

Value range	Depends on the setting of the Angular velocity range property: <ul style="list-style-type: none">▪ -245 °/s ... +245 °/s (at 245 °/s)▪ -500 °/s ... +500 °/s (at 500 °/s)▪ -2.000 °/s ... +2.000 °/s (at 2000 °/s)
Dependencies	Available only if the Gyroscope feature property is set to Enabled.

Signal ports

The Acceleration In function block type does not provide signal ports.

Overview of Tunable Properties (Acceleration In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (Acceleration In)

Where to go from here

Information in this section

- | | |
|----------------------------------------------------------------------------|----------------------|
| Configuring the Basic Functionality (Acceleration In)..... | 1485 |
| Configuring the Standard Features (Acceleration In)..... | 1487 |

Configuring the Basic Functionality (Acceleration In)

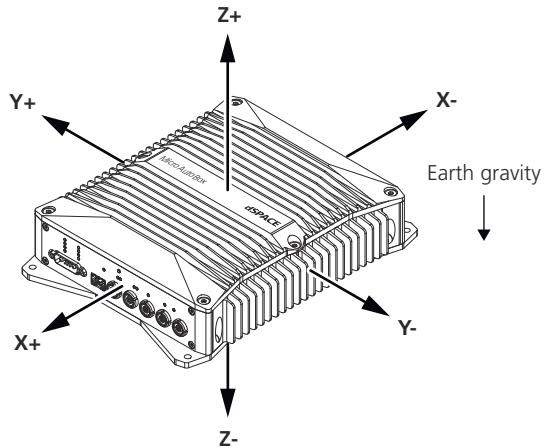
Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Using the gyroscope functionality
- Selecting a sampling rate for the measurement
- Selecting measurement ranges for expected values

Measuring acceleration

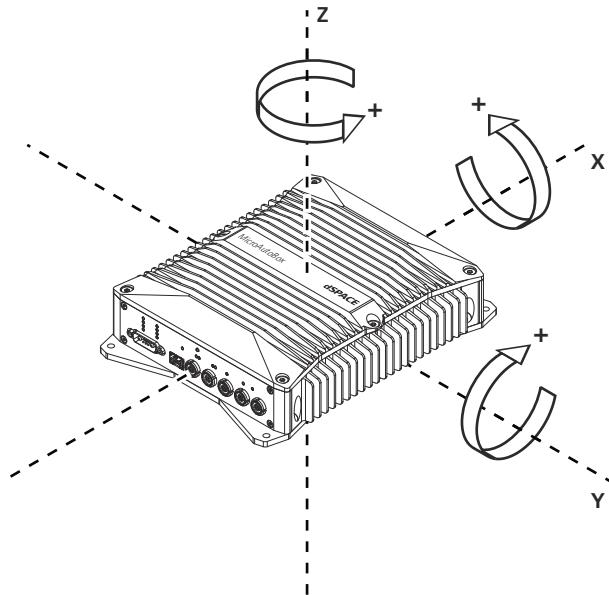
The onboard acceleration sensor measures gravity on its three axes with a specified sample rate. For the assignment of the axes to the housing of the MicroAutoBox III, refer to the following illustration.



When the MicroAutoBox III is exactly horizontal, the acceleration sensor provides $+9.81 \text{ m/s}^2$ for the z-axis, and 0 for the x-axis and the y-axis. This means that a force of 1 g impacts on the MicroAutoBox III. Thus, any position or movement of the MicroAutoBox III can be detected and then calculated in the behavior model.

Using the gyroscope functionality

If you want to measure the angular velocities on the axes of the MicroAutoBox III, you have to enable the gyroscope feature. The acceleration sensor includes a gyroscope, which measures the rotation around the three axes: x, y and z. For the assignment of the axes to the housing of the MicroAutoBox III, refer to the following illustration.



The arrowhead in the illustration indicates in which direction of rotation positive values are measured.

Selecting a sample rate

You have to select a sample rate at which the acceleration and angular velocity values are measured with the onboard acceleration sensor. The sensor then continuously measures at the selected sample rate.

Select the sample rate according to the expected change in acceleration and angular velocity:

- With a lower sample rate, fast changes will not be detected, but the measured values do not have larger deviations.
- With a higher sample rate, fast/strong changes are captured, but the deviations in the measured values become larger.

Note

With each model step, the function block reads the last available measured value from the acceleration sensor and provides the value to the behavior model. Note the following behavior:

- If the sample rate of the model task (in the connected behavior model) is higher than the sample rate for the acceleration sensor, the same measured values are provided to the behavior model several times.
- If the sample rate of the model task (in the connected behavior model) is lower than the sample rate for the acceleration sensor, measured values are lost and cannot be provided to the behavior model.

Selecting measurement ranges

You have to specify a measurement range for the acceleration values and a measurement range for the angular velocity values. Select the smallest possible range with respect to the expected values. The larger the value range, the lower the resolution. Refer to the following tables:

Linear Acceleration Range	Resolution ¹⁾
±2 g	$0.59820565 \times 10^{-3} \text{ m/s}^2$
±4 g	$1.1964113 \times 10^{-3} \text{ m/s}^2$
±8 g	$2.3928226 \times 10^{-3} \text{ m/s}^2$
±16 g	$7.1784678 \times 10^{-3} \text{ m/s}^2$

¹⁾ The resolution value refers to the value range at the function ports.

Angular Velocity Range	Resolution ¹⁾
±245 °/s	0.00875 °/s
±500 °/s	0.0175 °/s
±2.000 °/s	0.07 °/s

¹⁾ The resolution value refers to the value range at the function ports.

Configuring the Standard Features (Acceleration In)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The electrical interface of the Acceleration In function block does not provide standard configuration features.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Atmospheric Pressure In

Where to go from here

Information in this section

Introduction.....	1490
Overviews.....	1491
Configuration.....	1493

Introduction

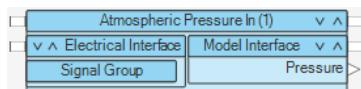
Introduction to the Function Block (Atmospheric Pressure In)

Function block purpose

The Atmospheric Pressure In function block type reads the measured atmospheric pressure value from an onboard pressure sensor and provides the value to the behavior model.

Default display

The function block provides function ports according to the settings. The following illustration shows the default display of the function block.



Main features

These are the main features:

- Providing the current atmospheric pressure value (measured with the onboard sensor) to the behavior model.

Supported channel types

The Atmospheric Pressure In function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	–	Pressure Sensor Unit 1
Hardware	–	DS1403 Processor Board

Overviews

Where to go from here

Information in this section

- [Overview of Ports and Basic Properties \(Atmospheric Pressure In\)..... 1491](#)
- [Overview of Tunable Properties \(Atmospheric Pressure In\)..... 1491](#)

Overview of Ports and Basic Properties (Atmospheric Pressure In)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



Pressure

This function outport writes the current measured atmospheric pressure value to the behavior model in Pascal.

Value range	20.000 Pa ... 110.000 Pa
Dependencies	-

Signal ports

The Atmospheric Pressure In function block type does not provide signal ports.

Overview of Tunable Properties (Atmospheric Pressure In)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	-	-

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface		
–	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuration

Configuring the Standard Features (Atmospheric Pressure In)

Scope	Each function block offers properties for configuring standard features of the electrical interface and the model interface.
Electrical interface	The electrical interface of the Atmospheric Pressure In function block does not provide standard configuration features.
Model interface	The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	More Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

LED Out

Where to go from here

Information in this section

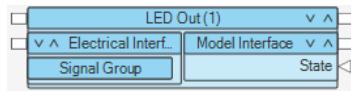
Introduction (LED Out).....	1496
Overviews (LED Out).....	1497
Configuring the Function Block (LED Out).....	1499

Introduction (LED Out)

Introduction to the Function Block (LED Out)

Function block purpose Each LED Out function block type provides access to one of the 4 user LEDs on the MicroAutoBox III. The LEDs can be controlled from the behavior model, for example, to display status information on the real-time application.

Default display The function block provides function ports according to the settings. The following illustration shows the default display of the function block.



Main features These are the main features:

- Accessing one of the four user LEDs (USR 1 ... USR 4).
- Controlling the color of the assigned LED from the behavior model.

Supported channel types The LED Out function block type supports the following channel types:

	SCALEXIO	MicroAutoBox III
Channel type	–	LED Out 1
Hardware	–	DS1403 Processor Board

Overviews (LED Out)

Where to go from here

Information in this section

- | | |
|-----------------------------------------------------------------------|----------------------|
| Overview of Ports and Basic Properties (LED Out)..... | 1497 |
| Overview of Tunable Properties (LED Out)..... | 1497 |

Overview of Ports and Basic Properties (LED Out)

Overview illustration

The following illustration shows the ports and the properties that support the basic functionality of the function block and their dependencies (channel type dependencies are not shown).



State

This function import defines the color of the assigned user LED from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ 0: LED does not light up. ▪ 1: LED lights up red. ▪ 2: LED lights up yellow. ▪ 3: LED lights up green. ▪ 4: LED lights up blue.
Dependencies	-

Overview of Tunable Properties (LED Out)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	-	-

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Function Ports		
Initial switch setting (Test Automation)	–	✓
Initial substitute value (Test Automation)	–	✓
Electrical Interface	–	–

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (LED Out)

Where to go from here

Information in this section

- | | |
|----------------------------------------------------|------|
| Configuring the Basic Functionality (LED Out)..... | 1499 |
| Configuring the Standard Features (LED Out)..... | 1499 |

Configuring the Basic Functionality (LED Out)

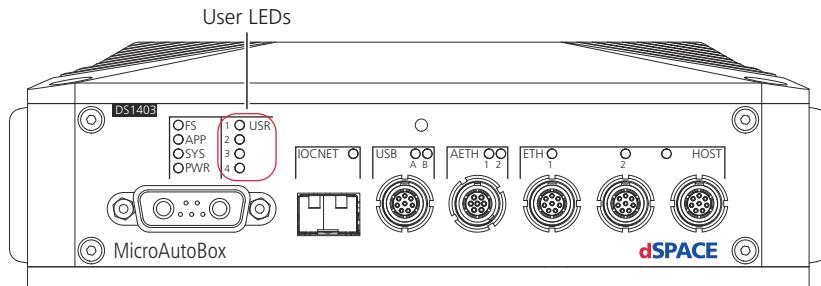
Overview

The function block type provides the following configuration features that influence the behavior of the basic functionality:

- Assigning a specific user LED

Assigning a specific user LED

MicroAutoBox III provides four user LEDs (USR 1 ... USR 4). Refer to the following illustration.



Each LED Out function block can access one of the four LEDs. You have to assign a specific LED via the hardware assignment properties. The LED numbering on the front panel of MicroAutoBox III matches the channel number in ConfigurationDesk.

Configuring the Standard Features (LED Out)

Scope

Each function block offers properties for configuring standard features of the electrical interface and the model interface.

Electrical interface

The electrical interface of the LED Out function block does not provide standard configuration features.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Stop behavior	
Test automation support	Configuring Test Automation Support on page 92

Implementing FPGA Custom Function Blocks

Where to go from here

Information in this section

Introduction (FPGA).....	1502
Handling FPGA Custom Function Blocks in ConfigurationDesk.....	1511
Overviews (FPGA).....	1520
Configuring the Function Block (FPGA).....	1525
Accessing FPGA Applications At Run Time.....	1532
Initializing of the RAM Used by the FPGA Application.....	1536

Information in other sections

Designing and building an FPGA application:

[RTI FPGA Programming Blockset Guide](#)

Provides basic information and detailed instructions on working with the RTI FPGA Programming Blockset.

[RTI FPGA Programming Blockset Handcode Interface Guide](#)

Provides basic information and detailed instructions on handcoding HDL and Verilog code for use with dSPACE hardware.

Supported hardware:

[Hardware for FPGA Applications \(SCALEXIO Hardware Installation and Configuration](#)

To execute FPGA applications, a SCALEXIO system must have an FPGA base board with I/O modules.

[DS1514 FPGA Base Board Data Sheet \(MicroAutoBox III Hardware Installation and Configuration](#)

Introduction (FPGA)

Where to go from here

Information in this section

Introduction to the Function Block (FPGA).....	1502
Steps to Implement an FPGA Application.....	1502
Types of FPGA Applications.....	1505
Identifying Elements of the RTI FPGA Programming Blockset.....	1509

Introduction to the Function Block (FPGA)

Function block purpose

FPGA custom function blocks contain the functionality of an FPGA application that must be defined with the RTI FPGA Programming Blockset.

Main feature

Implementing a custom-coded FPGA application.

Supported dSPACE hardware

FPGA custom function blocks support the following hardware:

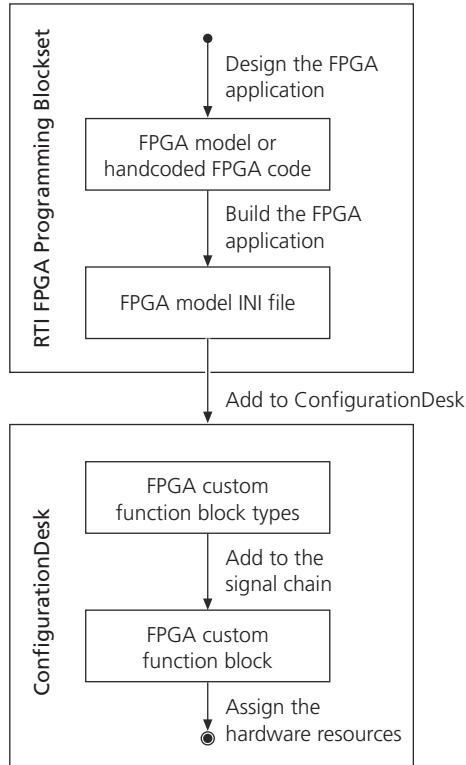
	SCALEXIO	MicroAutoBox III
Hardware	<ul style="list-style-type: none"> ▪ DS2655 ▪ DS6601 ▪ DS6602 ▪ DS2655M1 ▪ DS2655M2 ▪ DS6651 ▪ MGT modules ▪ Inter-FPGA 	<ul style="list-style-type: none"> ▪ DS1514 ▪ DS1552 ▪ DS1552B1 ▪ DS1554

Steps to Implement an FPGA Application

Introduction

The custom-coded FPGA application that you want to use in ConfigurationDesk has to be generated with the RTI FPGA Programming Blockset. Then, you add the FPGA application to ConfigurationDesk as FPGA custom function blocks.

Implementation overview



Designing FPGA applications

There are two methods to design FPGA applications:

- Modeling FPGA applications in Simulink:

The RTI FPGA Programming Blockset provides blocks to model the interface to the external devices and the interface to the behavior model. The I/O functionality itself must be modeled with Xilinx® blocks. Refer to [Modeling the FPGA Application \(RTI FPGA Programming Blockset Guide\)](#).

- Handcoding FPGA applications:

The RTI FPGA Programming Blockset provides templates to handcode the interface to the external devices and the interface to the behavior model. The I/O functionality itself must be designed with VHDL code. Refer to [Detailed Instructions on the Handcode Workflow \(RTI FPGA Programming Blockset Handcode Interface Guide\)](#).

FPGA applications for the following use cases are supported in ConfigurationDesk:

- FPGA applications supporting single-core real-time applications.
- FPGA applications supporting multicore real-time applications.
- FPGA applications supporting inter-FPGA communication.

For more information on how the specific FPGA application types are represented by function blocks, refer to [Types of FPGA Applications](#) on page 1505.

Building FPGA applications	You start the build process via the RTI FPGA Programming Blockset. The build results consist of the generated FPGA bitstream to program the FPGA and files that are needed to add the FPGA application to ConfigurationDesk. The required build result files are archived in the FPGA model INI file. Adding port-specific functions to the function ports The FPGA model INI file also provides templates for user functions that can be used in ConfigurationDesk to trigger the execution of port-specific functions, for example, the conversion of values or other computations. For more information, refer to How to Build FPGA Applications (MicroAutoBox III, SCALEXIO) (RTI FPGA Programming Blockset Guide) .
Adding FPGA applications to ConfigurationDesk	You can export the FPGA applications to ConfigurationDesk or you import the FPGA model INI file including the FPGA application. Added FPGA applications are available as FPGA custom function block types in the Function Browser. Depending on the selected file directory to add the FPGA application, the FPGA custom function block types are available in the Function Browser of one ConfigurationDesk project or in all ConfigurationDesk projects. For more information on the file directory to add FPGA applications and instructions on adding, refer to Handling FPGA Custom Function Blocks in ConfigurationDesk on page 1511.
Adding FPGA custom function blocks to the signal chain	You can add FPGA custom function blocks to the signal chain, for example, via drag & drop from the Function Browser to a working view.
Assigning the hardware resources	With the RTI FPGA Programming Blockset, you design an FPGA application for a certain arrangement of I/O modules that are inserted to the I/O module slots of the FPGA board. Therefore, you can assign the hardware resources only if the connected I/O modules matches by module type and I/O module slot. Refer to Hardware Dependencies of FPGA Custom Function Blocks on page 1511.
Loading the bitstream to the FPGA board	The build process of ConfigurationDesk adds the FPGA bitstream from the FPGA model INI file to the real-time application. When you download the real-time application to the dSPACE hardware, the FPGA bitstream is automatically loaded to the FPGA base board that is assigned by the FPGA custom function block.

Types of FPGA Applications

Types derived from use cases

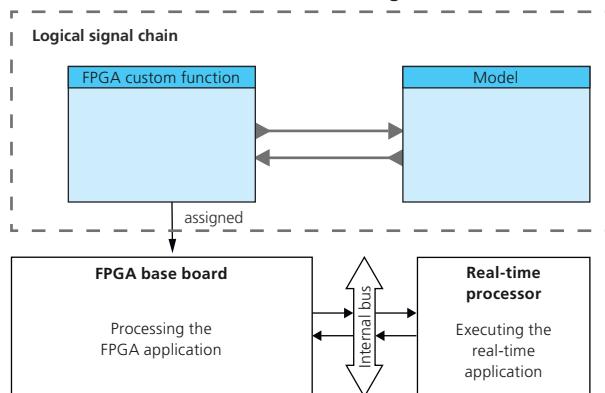
The different types can be derived from use cases. FPGA applications for the following use cases are supported in ConfigurationDesk:

- FPGA applications supporting single-core real-time applications.
- FPGA applications supporting multicore real-time applications.
- FPGA applications supporting inter-FPGA communication.

FPGA applications supporting single-core real-time applications

One FPGA custom function block represents the entire FPGA application in the logical signal chain. The function block lets you configure and initialize the access to the FPGA base board and provides the interfaces to map the behavior model and external devices.

The illustration below is an example to show you the block in the logical signal chain and the hardware resource assignment.

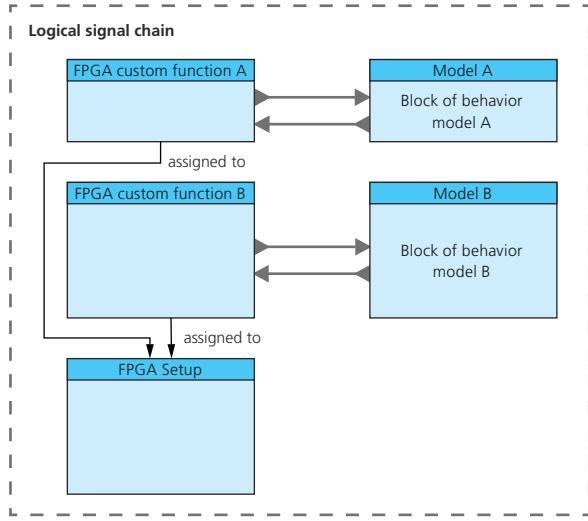


FPGA applications supporting multicore real-time applications

To support multicore processor applications, at least three function blocks represent the entire FPGA application. The separation of the FPGA application into one FPGA Setup block (<application name>_Setup) and at least two FPGA custom function blocks let you use one FPGA application by several application processes (behavior models).

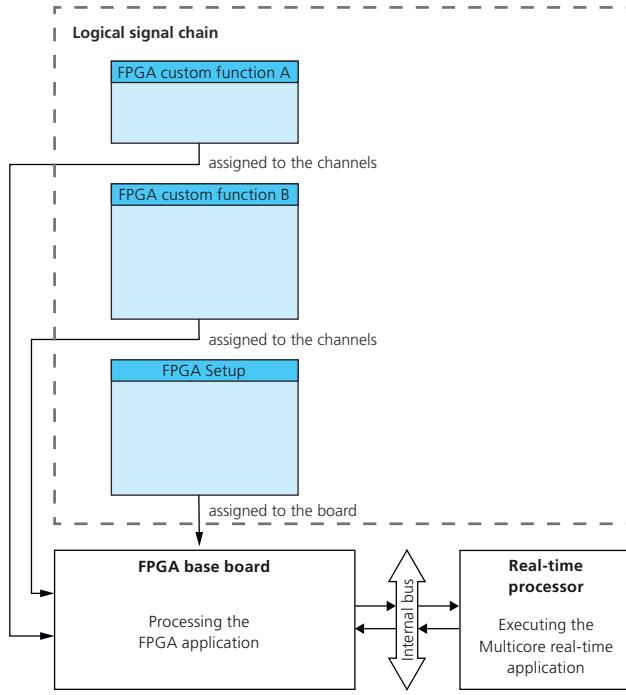
In the logical signal chain, the FPGA Setup function block lets you configure and initialize the access to the FPGA base board. At least two FPGA custom function blocks provide the interfaces to map behavior models and external devices.

Assigning the FPGA Setup function block All FPGA custom function blocks must be assigned to their FPGA Setup function block that initializes the access to the hardware. The illustration below is an example to show you the blocks used in a multicore processor application and the assignment to the FPGA Setup function block.



Hardware resource assignment The FPGA Setup function block and also the FPGA custom function blocks must be assigned to the FPGA hardware resources as shown below:

- The FPGA Setup function block must be assigned to the FPGA board.
- The FPGA custom function blocks must be assigned to the channels of the I/O modules.



FPGA applications supporting inter-FPGA communication

If your SCALEXIO system provides several FPGA base boards, you can exchange data directly between the board's FPGA applications. The following methods are supported:

- Inter-FPGA communication via IOCNET

IOCNET can be used to transfer data directly between FPGA boards of a SCALEXIO system.

- Inter-FPGA communication via MGT modules

Multi-gigabit transceiver (MGT) modules can be used to transfer data directly between FPGA boards of a SCALEXIO system.

- Inter-FPGA communication via I/O module slots

Inter-FPGA communication via I/O module slots is done with a direct connection between I/O module slots of each affected FPGA base board.

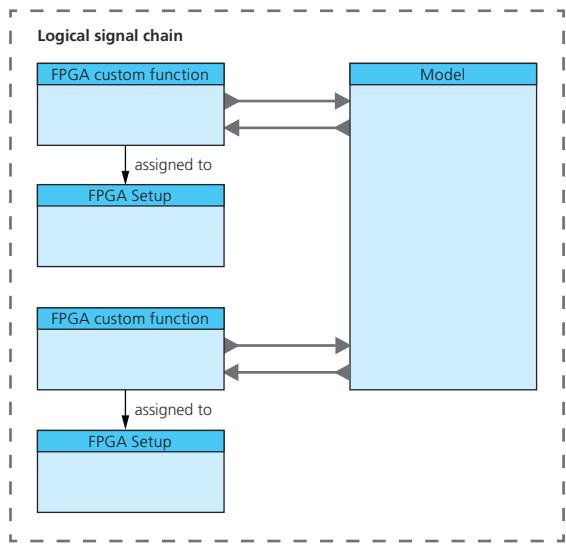
Therefore, the FPGA application supports an inter-FPGA interface for the I/O module slots as specified in the FPGA model/handcode.

Inter-FPGA via IOCNET and MGT modules have no influence on the FPGA application type, in contrast to inter-FPGA communication via I/O module slots.

Each FPGA application that supports inter-FPGA communication via I/O module slots is represented by an **FPGA Setup** function block (<application name>_Setup) and at least one **FPGA custom** function block.

In the logical signal chain, the **FPGA Setup** function blocks let you configure and initialize the access to the FPGA base boards. At least two **FPGA custom** function blocks provide the interfaces to map behavior models and external devices.

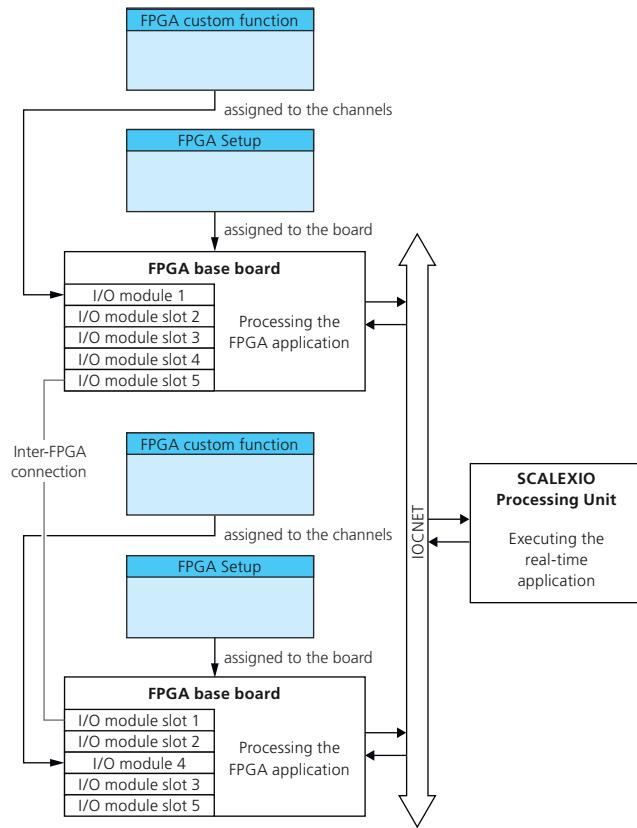
Assigning the **FPGA Setup function block** All **FPGA custom** function blocks must be assigned to their **FPGA Setup** function blocks that initialize the access to the hardware. The illustration below is an example to show you the function blocks used in a single-core real-time application with inter-FPGA communication.



Hardware resource assignment The FPGA Setup function blocks and also the FPGA custom function blocks must be assigned to the FPGA hardware resources:

- The FPGA Setup function blocks must be assigned to the FPGA boards.
- The FPGA custom function blocks must be assigned to the channels of the I/O modules.

In the example below, the FPGA applications support the inter-FPGA communication on I/O module slots 1 and 5.



Related topics

Basics

- [Assigning Hardware Resources to Function Blocks \(ConfigurationDesk Real-Time Implementation Guide\)](#)
- [Rules for Model Port Mapping \(ConfigurationDesk Real-Time Implementation Guide\)](#)

Identifying Elements of the RTI FPGA Programming Blockset

Introduction

Naming conventions for FPGA custom function block types and their functions and logical signals let you identify the FPGA application and the corresponding signals in the FPGA model or FPGA code. This helps you to map the ports of the FPGA custom function block to the correct device ports and model ports.

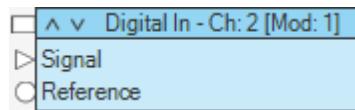
Naming convention for FPGA custom function block types

The naming of the function block type depends on the use case:

- For single-core real-time applications, the FPGA application name is also the FPGA custom function block type.
- For all other use cases, the FPGA custom function block type is <application name>_<FPGA subsystem>. The FPGA Setup function block type is <application name>_Setup.

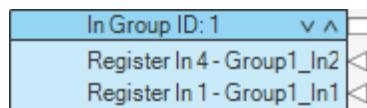
Naming convention for logical signals

The name of a logical signal is based on the I/O function block used in the FPGA model or I/O function used in the FPGA code: <unit type> - Ch: <channel number> <- optional custom name> [Mod: <module number>]. For example, if you specified the second channel of the first module to be used for a digital input signal, it is named Digital In - Ch: 2 [Mod: 1] as shown below.

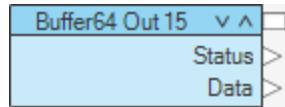
**Naming convention for functions**

Register functions One register function of an FPGA custom function block consists of the registers of one register group as provided by the FPGA application. The function name is <data direction> Group ID: <register group ID>. For example, the function In Group ID: 1 consists of the input registers with the group ID 1. Ungrouped registers are added to the function Ungrouped.

The naming convention of the function ports is <register type> <channel number> <- optional custom name>. For example, the Register In 1 - Group 1_In1 function port is the input register of channel 1 with the custom name Group1_In1.



Buffer functions The name of the function is <buffer type> <channel number> <- optional custom name>. For example, the Buffer64 Out 15 function is the 64-bit output buffer of channel 15 and has no custom name.



Handling FPGA Custom Function Blocks in ConfigurationDesk

Where to go from here

Information in this section

Hardware Dependencies of FPGA Custom Function Blocks.....	1511
How to Add FPGA Applications to ConfigurationDesk.....	1513
How to Check Hardware Resources Required for FPGA Custom Function Blocks (SCALEXIO).....	1515
How to Update FPGA Custom Function Blocks.....	1518
How to Open the FPGA Model in MATLAB/Simulink.....	1519

Hardware Dependencies of FPGA Custom Function Blocks

Introduction

An FPGA custom function is modeled or handcoded for a certain combination of FPGA base board, I/O boards, and connections. To assign the hardware resources to an FPGA custom function block, the hardware resources must match to the requirements of the FPGA custom function block.

Common dependencies

FPGA applications are built for a certain FPGA base board and certain I/O modules. The FPGA application cannot be downloaded to a dSPACE hardware platform with a different FPGA base board or I/O modules.

During the hardware resource assignment ConfigurationDesk checks whether the hardware resource matches the FPGA application.

DS2655 FPGA Base Board dependencies

There are two variants of the DS2655 FPGA Base Board that are supported by two different frameworks:

- DS2655 (7K160) FPGA Base Board
- DS2655 (7K410) FPGA Base Board

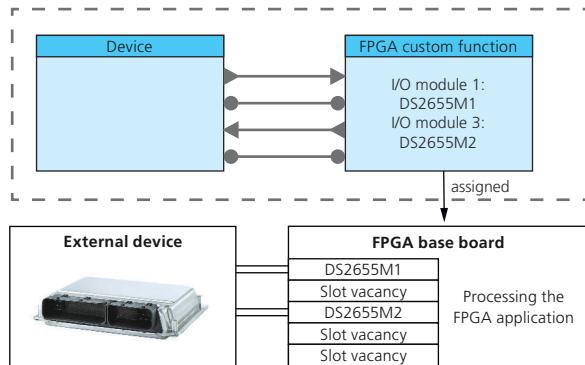
FPGA applications that are built for a DS2655 (7K160) FPGA Base Board cannot be downloaded to a DS2655 (7K410) FPGA Base Board and vice versa. ConfigurationDesk does not check the variant of the FPGA base board when you assign the hardware or you build the real-time application. No conflict occurs if you assign a wrong variant.

I/O module slot dependencies of SCALEXIO systems

A SCALEXIO FPGA base board lets you connect up to five I/O modules to the board's I/O module slots. The FPGA custom function block supports the base

board and the connected I/O modules only if the connected I/O modules matches the requirements of the FPGA custom function block by module type and module slot.

For example, an FPGA custom function block requires a DS2655M1 Multi-I/O Module inserted in I/O module slot 1 and a DS2655M2 Digital I/O Module inserted in I/O module slot 3 as illustrated below. To be able to assign the hardware resources in ConfigurationDesk, the SCALEXIO FPGA base board must provide at least a DS2655M1 Multi-I/O Module inserted in I/O module slot 1 and a DS2655M2 Digital I/O Module inserted in I/O module slot 3.



Note

I/O modules inserted in I/O module slots that are not supported by the FPGA custom function block have no effect on the hardware resource assignment.

I/O module slot dependencies of the inter-FPGA communication bus

A SCALEXIO FPGA base board lets you connect up to five inter-FPGA communication cables to the board's I/O module slots. The FPGA custom function block supports the inter-FPGA communication only if the inter-FPGA communication cables are connected to the I/O module slots as modeled or handcoded for the FPGA custom function block. The function block does not display the required connections.

Note

ConfigurationDesk does not check the connection of the inter-FPGA communication cables when you assign the FPGA boards or build the real-time application.

However, during the download process, ConfigurationDesk checks if the implemented inter-FPGA communication of the FPGA custom function block matches the inter-FPGA connections of the target hardware.

Related topics**HowTos**

How to Check Hardware Resources Required for FPGA Custom Function Blocks (SCALEXIO).....	1515
---------------------------------------------------------------------------------------------	------

How to Add FPGA Applications to ConfigurationDesk

Objective

To use an FPGA application, you must add the FPGA application as an FPGA custom function block to ConfigurationDesk.

File directories to add FPGA applications

You can add an FPGA application to the following directories:

- To the project-specific custom functions directory
`<DocumentsFolder>\<Project>\CustomFunctions`.

Only the current ConfigurationDesk project then provide the FPGA custom function block in its Function Browser.

- To the global custom functions directory.

Each ConfigurationDesk project then provides the FPGA custom function block in its Function Browser. The FPGA custom function is not included in a project backup. For more details, refer to [Configuration Page](#) ([ConfigurationDesk User Interface Reference](#)).

The default global custom functions directory is
`<DocumentsFolder>\UserFiles`. You can find and configure the global custom functions directory on the Configuration page of the ConfigurationDesk Options dialog.

Possible methods

Possible methods when you model the FPGA application with RTI blocks:

- As of RTI FPGA Programming Blockset 3.1, you can export a built FPGA application via the blockset. Refer to Method 1.
- You can import the FPGA model INI file of a built FPGA application. Refer to Method 2.

Method when you handcode the FPGA application:

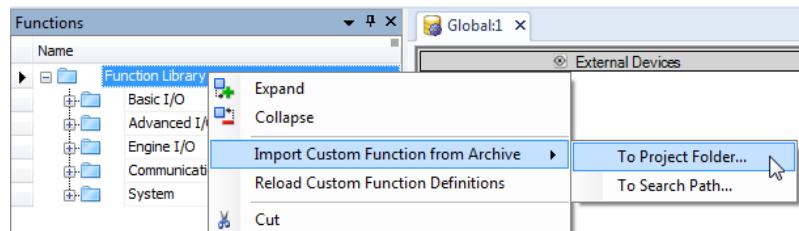
- You import the FPGA model INI file of a built FPGA application. Refer to Method 2.

Method 1**To add an FPGA application via the RTI FPGA Programming Blockset**

- 1 In the FPGA model, open the FPGA SETUP BL block dialog and export the FPGA application. Refer to [How to Export Build Results and Processor Models to ConfigurationDesk Projects](#) ([RTI FPGA Programming Blockset Guide](#)).

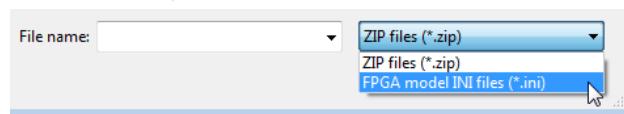
Method 2**To add an FPGA application via ConfigurationDesk**

- 1 In the Function Browser, right-click a free area.
- 2 Select one of the following options to add the FPGA application to ConfigurationDesk:
 - Select Import Custom Function from Archive - To Project Folder to add the FPGA application to the project-specific custom functions directory.
 - Select Import Custom Function from Archive - To Search Path to add the FPGA application to the global custom functions directory.



ConfigurationDesk opens an Open dialog.

- 3 Switch the file type to FPGA model INI files (*.ini).



- 4 Select the FPGA model INI file of the FPGA application and import it.

Result

You added an FPGA application to ConfigurationDesk. Depending on the use case, one or several FPGA custom function block types are added to the Function Browser. The naming of the function block types is based on the FPGA application name.

For more information on the naming conventions of FPGA custom function blocks, refer to [Identifying Elements of the RTI FPGA Programming Blockset](#) on page 1509. For more information on the function blocks derived from the use case, refer to [Types of FPGA Applications](#) on page 1505.

If you exported the FPGA application with the RTI FPGA Programming Blockset, the export process also performed the following steps:

- FPGA custom function blocks are added to the signal chain.
- The model interface (processor interface) are exported to ConfigurationDesk.
- The function ports of the added FPGA custom function blocks are mapped to the model ports.

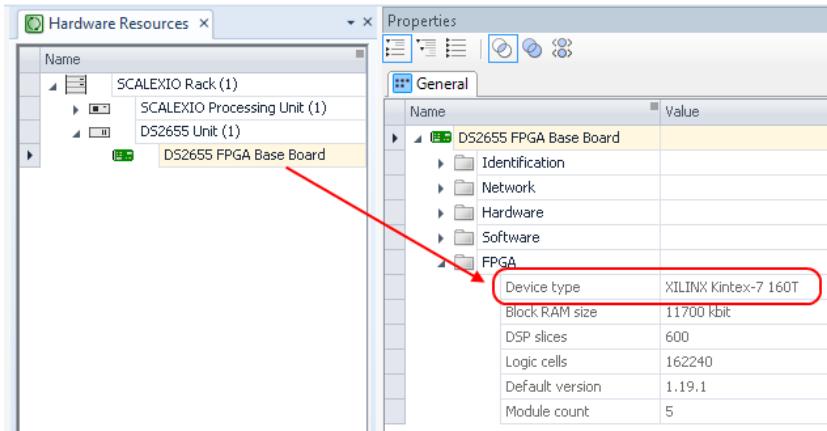
Related topics**Basics**

[Basics on Instantiating Function Blocks.....](#) 62

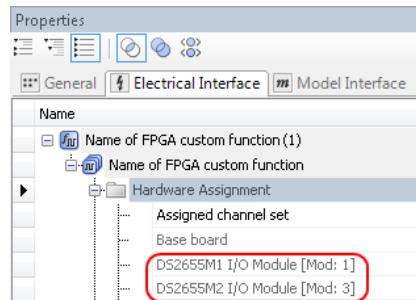
How to Check Hardware Resources Required for FPGA Custom Function Blocks (SCALEXIO)

Objective	Before you assign an SCALEXIO FPGA base board to an FPGA custom function block, make sure that the FPGA base board, the I/O modules, and the wiring of optional inter-FPGA connection cables match the requirements of the FPGA custom function block.
Hardware dependencies	For basics on hardware dependencies of FPGA custom function blocks, refer to Hardware Dependencies of FPGA Custom Function Blocks on page 1511.
Avoiding damage	<p>NOTICE</p> <p>Inter-FPGA communication between FPGA boards that are connected to different processors can damage the FPGA boards.</p> <ul style="list-style-type: none">Do not use inter-FPGA communication in a multiprocessor application to transmit data between FPGA boards that are connected to different processors via LOCNET. <p>You can use inter-FPGA communication only for FPGA boards that are connected to the same processor.</p>
Precondition	The FPGA custom function block is added to the signal chain. For instructions, refer to How to Add FPGA Applications to ConfigurationDesk on page 1513.

Method	To check hardware resources required for an FPGA custom function block
	<p>1 Only if you use a DS2655 FPGA Base Board, check the following:</p> <p>In the Hardware Resource Browser, make sure that the variant of the DS2655 FPGA Base Board matches with the framework that is used for the FPGA custom function.</p>

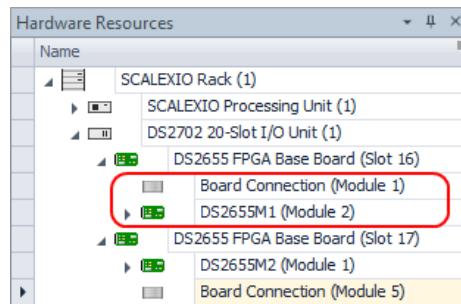


- 2 In the signal chain, select the FPGA custom function block or the **FPGA Setup** function block. For more information on the different kinds of FPGA custom function blocks, refer to [Types of FPGA Applications](#) on page 1505.
- 3 In the Electrical Interface page of the Properties Browser, browse for the hardware resources requirements of the FPGA custom function block as shown below.



The required I/O modules are displayed as <I/O module type> [Mod: <I/O module slot>].

- 4 In the Hardware Resource Browser, make sure that your hardware topology matches to the requirements of the FPGA custom function block by I/O module type and I/O module slot.

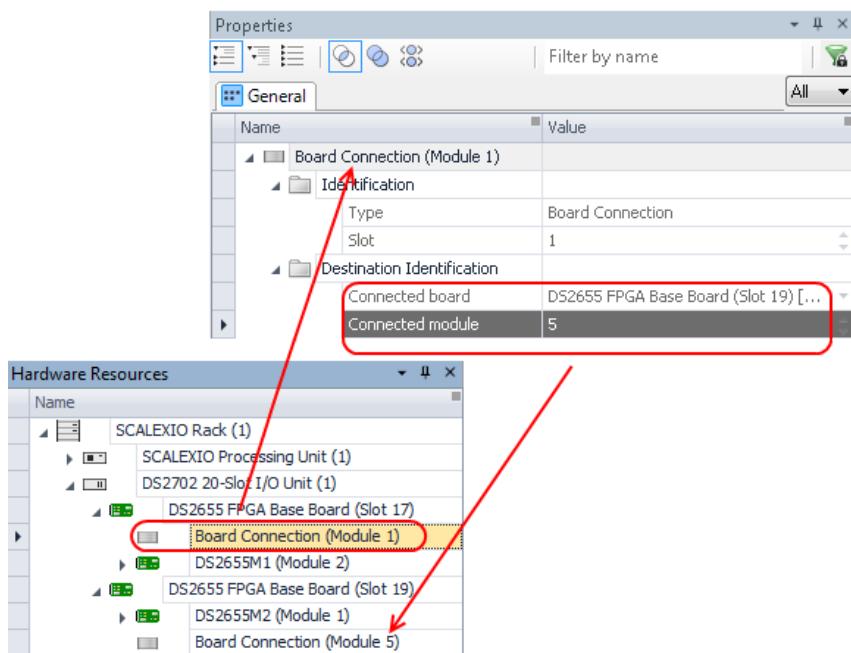


Measures to be taken if the hardware topology does not match the displayed hardware requirements:

- Change the hardware configuration of the FPGA model or handcode and update the FPGA custom function block in ConfigurationDesk. For instructions on updating, refer to [How to Update FPGA Custom Function Blocks](#) on page 1518.
- Change the assembly order of the I/O modules on the FPGA base board according to the requirements of the FPGA custom function block and replace the hardware topology. For more information on the assembly, refer to [Hardware for FPGA Applications \(SCALEXIO Hardware Installation and Configuration\)](#).

5 Only if you use inter-FPGA communication, check the following:

Make sure that the inter-FPGA communication cables connect the correct I/O module slots of the FPGA boards. The Properties Browser displays the connection of the inter-FPGA communication bus as shown below.



Result

You checked and made sure that the SCALEXIO system matches the requirements of the FPGA custom function block.

Related topics

Basics

[Hardware Dependencies of FPGA Custom Function Blocks.....](#) 1511

How to Update FPGA Custom Function Blocks

Objective	When you modify the FPGA application you must update the FPGA custom function blocks in ConfigurationDesk to take effect the modifications.
Preconditions	<ul style="list-style-type: none">▪ The FPGA application name must not be modified in the FPGA model or handcode. The FPGA application name is used as function block type in ConfigurationDesk. If you modify the name, new FPGA custom function block types will be added to ConfigurationDesk.▪ You generated the FPGA model INI file of the modified FPGA application.
Possible methods	You can update FPGA custom function blocks as follows: <ul style="list-style-type: none">▪ You can update an FPGA custom function block via the Function Browser. Refer to method Method 1.▪ You can update an FPGA custom function block by reloading the FPGA model INI file. Refer to method Method 2.
Method 1	To update FPGA custom function blocks via the Function Browser <ol style="list-style-type: none">1 Add the modified FPGA model INI file to ConfigurationDesk. Refer to How to Add FPGA Applications to ConfigurationDesk on page 1513.
Method 2	To update FPGA custom function blocks by reloading the FPGA model INI file <ol style="list-style-type: none">1 Right-click on the FPGA custom function block and select Reload INI File from Archive from the context menu. The Reload custom function type from archive dialog opens.2 Double-click the modified FPGA model INI file.
Result	ConfigurationDesk updated the FPGA custom function blocks. If updated FPGA custom function blocks include modifications of the interfaces to the neighboring blocks, the port mapping might be affected. You have to update the signal chain in ConfigurationDesk for the following modifications in the FPGA application: <ul style="list-style-type: none">▪ Changing the electrical interface You have to manually remap the signal ports with the related device in ConfigurationDesk.▪ Changing the channel number of registers and buffers Check, if you have to manually remap the function port with the related model port in ConfigurationDesk.

- Changing the channel name of registers and buffers
Check, if you have to manually remap the function port with the related model port in ConfigurationDesk.
- Adding RTI FPGA blocks
Check if you have to generate new model ports in ConfigurationDesk.
- Deleting RTI FPGA blocks
There are unconnected model ports that can generally be ignored.

How to Open the FPGA Model in MATLAB/Simulink

Objective	You can open the Simulink model of the FPGA custom function block from ConfigurationDesk.
------------------	-------------------------------------------------------------------------------------------

Preconditions	<ul style="list-style-type: none">▪ The FPGA custom function is modeled with the Xilinx System Generator Blockset.▪ MATLAB/Simulink must be started with the Xilinx System Generator, otherwise the Xilinx System Generator Blockset cannot be displayed correctly.▪ The FPGA model must be a part of the overall behavior model.
----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Method	To open the FPGA Model in MATLAB/Simulink <ol style="list-style-type: none">1 Right-click on the FPGA custom function block.2 Select Show Model from the context menu.
---------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Result	ConfigurationDesk opens the FPGA model in MATLAB/Simulink. If the FPGA model of the selected FPGA custom function block is no longer available or has been moved to another folder, an error message is displayed in MATLAB.
---------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Related topics	HowTos
	How to Update FPGA Custom Function Blocks..... 1518

Oversviews (FPGA)

Where to go from here

Information in this section

Overview of Ports and Properties (FPGA).....	1520
Overview of Tunable Properties (FPGA)	1524

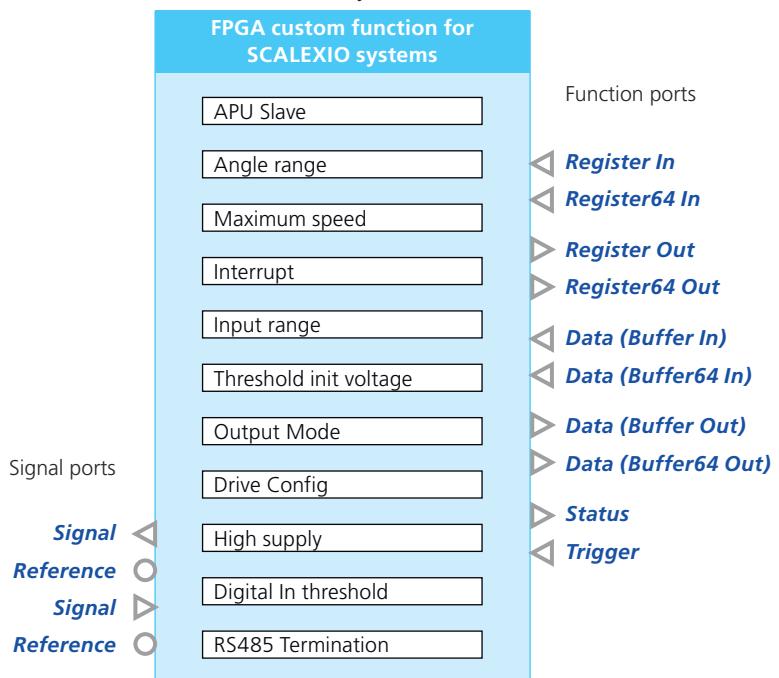
Overview of Ports and Properties (FPGA)

Overview illustrations

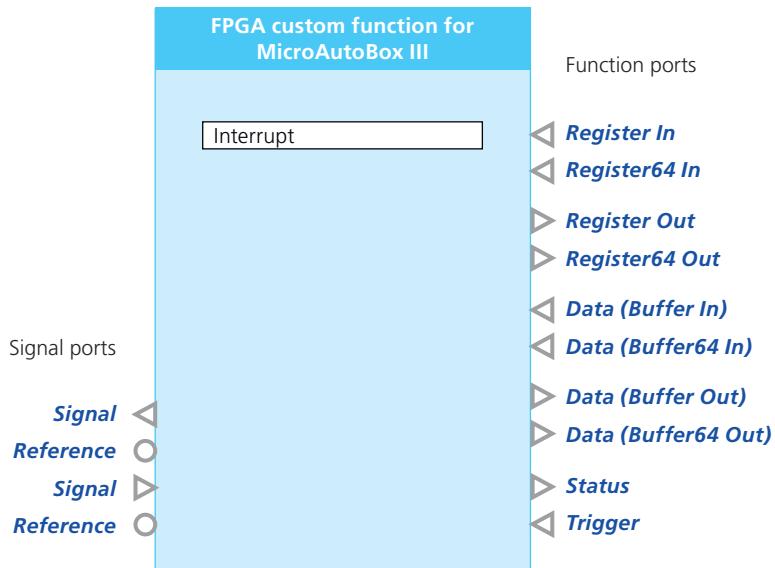
FPGA custom function blocks represent FPGA application for different use cases. The illustrations give you an overview which logical signals, function ports, and properties can be provided by the FPGA application and which kinds of function blocks are used to represent the FPGA application in ConfigurationDesk.

FPGA custom function block Depending on the use case, an FPGA custom function block represents the FPGA interface of the entire FPGA application or parts of the FPGA interface. For more information , refer to [Types of FPGA Applications](#) on page 1505.

The following overview shows the possible ports and properties of an FPGA custom function for a SCALEXIO system.



The following overview shows the possible ports and properties of an FPGA custom function for a MicroAutoBox III.



FPGA Setup function block FPGA application supporting multicore real-time applications or inter-FPGA communication are represented by FPGA custom function blocks and an additional FPGA Setup function block. The FPGA Setup function block provides the access to the FPGA base board and lets you assign angular clock values to the APU Slave. All FPGA custom function blocks of one FPGA application must be assigned to the FPGA application's FPGA Setup function block. For more information, refer to [Types of FPGA Applications](#) on page 1505.

The following illustration shows function ports influencing the basic functionality that might be available for an FPGA Setup function block.



Register In

This function import reads 32-bit data values from the behavior model.

Value range	The value range depends on the FPGA application's settings of the referenced Register In channel.
Dependencies	Available only if the FPGA application supports the function port.

Register64 In

This function import reads 64-bit data values from the behavior model.

Value range	The value range depends on the FPGA application's settings of the referenced Register64 In channel.
-------------	-----------------------------------------------------------------------------------------------------

Dependencies	Available only if the FPGA application supports the function port.
--------------	--------------------------------------------------------------------

Register Out

This function outport writes 32-bit data values to the behavior model.

Value range	The value range depends on the FPGA application's settings of the referenced Register Out channel.
Dependencies	Available only if the FPGA application supports the function port.

Register64 Out

This function outport writes 64-bit data values to the behavior model.

Value range	The value range depends on the FPGA application's settings of the referenced Register64 Out channel.
Dependencies	Available only if the FPGA application supports the function port.

Data (Buffer In)

This function import reads vectors with 32-bit data values from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ The value range depends on the FPGA application's settings of the referenced Buffer In channel. ▪ The Data width property displays the vector size of the buffer.
Dependencies	Available only if the FPGA application supports the function port.

Data (Buffer64 In)

This function import reads vectors with 64-bit data values from the behavior model.

Value range	<ul style="list-style-type: none"> ▪ The value range depends on the FPGA application's settings of the referenced Buffer64 In channel. ▪ The Data width property displays the vector size of the buffer.
Dependencies	Available only if the FPGA application supports the function port.

Data (Buffer Out)

This function import reads vectors with 32-bit data values to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ The value range depends on the FPGA application's settings of the referenced Buffer Out channel. ▪ The Data width property displays the vector size of the buffer.
Dependencies	Available only if the FPGA application supports the function port.

Dependencies	Available only if the FPGA application supports the function port.
--------------	--------------------------------------------------------------------

Data (Buffer64 Out)

This function outport writes vectors with 64-bit data values to the behavior model.

Value range	<ul style="list-style-type: none"> ▪ The value range depends on the FPGA application's settings of the referenced Buffer64 Out channel. ▪ The Data width property displays the vector size of the buffer.
Dependencies	Available only if the FPGA application supports the function port.

Status

This function outport writes a vector with status information on the data exchange.

Value range	<p>Data width: 3</p> <ul style="list-style-type: none"> ▪ Status[0]: Contains the number of valid elements in the Data vector of the buffer output. ▪ Status[1]: Indicates whether the current buffer contains new or old values. 1 means that the data in the buffer is new, 0 means that the data in the buffer is old. ▪ Status[2]: Indicates whether a buffer overflow occurred. If it is 1, at least one buffer was not read and its data was lost before the currently read buffer was filled. If no overflow occurs, the output is 0.
Dependencies	Available only for Buffer Out and Buffer64 Out functions.

Trigger

This function import enables the data acquisition to trace FPGA signals.

Value range	If the port is mapped to a model port, the data acquisition is triggered with the shortest model task period of the mapped behavior model.
Dependencies	Available only if the FPGA application provides traceable FPGA signals.

Signal

This signal port represents the electrical connection point of the logical signal of the function block.

Value range	<p>Depends on the assigned hardware resource.</p> <p>For specific values, refer to one of the following topics:</p> <ul style="list-style-type: none"> ▪ Data Sheet of the DS2655M1 Multi-I/O Module (SCALEIO Hardware Installation and Configuration) 
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> ▪ Data Sheet of the DS2655M2 Digital I/O Module (SCALEXIO Hardware Installation and Configuration  ▪ Data Sheet of the DS6651 Multi-I/O Module (SCALEXIO Hardware Installation and Configuration  ▪ DS1552 Multi-I/O Module Data Sheet (MicroAutoBox III Hardware Installation and Configuration  ▪ DS1554 Engine Control I/O Module Data Sheet (MicroAutoBox III Hardware Installation and Configuration 
Dependencies	–

Reference

This signal port is a reference port and provides the reference signal for the Signal signal port.

Related topics**Basics**

Types of FPGA Applications..... 1505

Overview of Tunable Properties (FPGA)

Tunable properties

Tunable properties can be accessed and modified in the experiment software. The following table shows all the tunable properties of the function block.

Property	Stop-Run-Tunable Parameter ¹⁾	Run-Time-Tunable Parameter ²⁾
Basic Functionality	–	–
Function Ports	Initial switch setting (Test Automation) Initial substitute value (Test Automation)	✓ ✓
Electrical Interface	–	–
FPGA internal signals and constants	Refer to Accessing the FPGA Application with the Experiment Software on page 1532.	

¹⁾ You can change the setting of this parameter when the real-time application is running or stopped. For the changes to take effect, you must restart the real-time application.

²⁾ You can change the setting of this parameter when the real-time application is running or stopped. The changes take effect immediately.

Configuring the Function Block (FPGA)

Where to go from here

Information in this section

Configuring the Basic Functionality (FPGA).....	1525
Configuring Standard Features (FPGA).....	1531

Information in other sections

Handling FPGA Custom Function Blocks in ConfigurationDesk.....	1511
----------------------------------------------------------------	------

Configuring the Basic Functionality (FPGA)

Overview

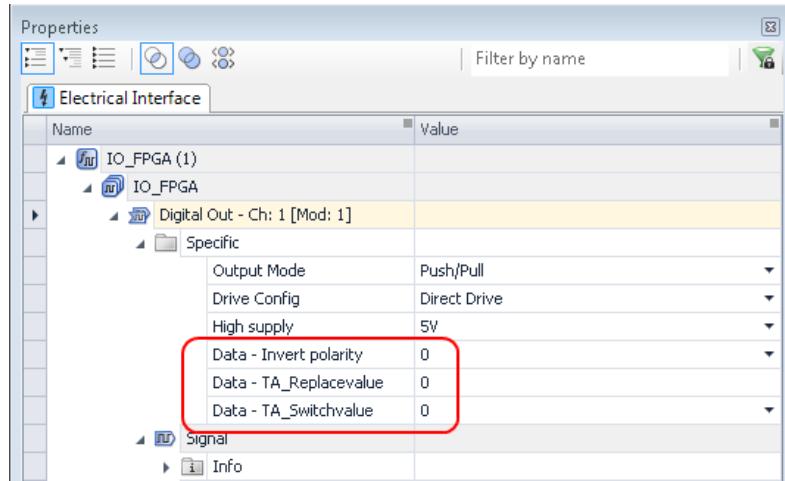
The function block provides the following configuration features which influence the behavior of the basic functionality:

- Setting the initial values for FPGA test access and scaling
- Using Interrupts of the FPGA custom function block
- Assigning FPGA custom function blocks to the FPGA Setup function block
- Enabling the FPGA access for experiment software products
- Configuring the inter-FPGA communication via IOCNET (SCALEXIO)
- Configuring APU masters and slaves (SCALEXIO)
- Configuring characteristics of the electrical interface (SCALEXIO)
- Initializing of the DS6602 DDR4 RAM (SCALEXIO)

Setting the initial values for FPGA test access and scaling

If your FPGA application supports FPGA test access and scaling, FPGA variables can be accessed with the experiment software to modify the FPGA interface signals.

In the Properties Browser, you set the initial values for FPGA test access and scaling. The FPGA application uses the initial values until they are changed in the experiment software.



For more information on FPGA test access and scaling as well as a description of the properties or FPGA variables, refer to [Accessing the FPGA Application with the Experiment Software](#) on page 1532.

Using interrupts of the FPGA custom function block

If you have specified an interrupt in your FPGA application, it is available in ConfigurationDesk as an I/O event. You can use the I/O event as trigger source for runnable functions in the behavior model.

To use the I/O events in the behavior model, you have to assign them to a task via event port or property. For more information, refer to [User-Friendly Connection of ConfigurationDesk and Simulink Models \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Assigning FPGA custom function blocks to the FPGA Setup function block

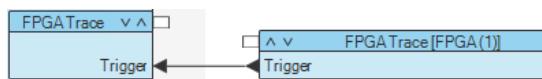
If you add an FPGA application to the signal chain that supports multicore real-time applications or inter-FPGA communication, the FPGA application is represented by at least one FPGA custom function function and one FPGA Setup function block. In the signal chain, all FPGA custom functions must be assigned to the application's FPGA Setup function block.

If you export the FPGA application via the `FPGA_SETUP_BL` block of the FPGA model, the assignment is done automatically. Otherwise, you have to specify the FPGA Setup function block for each FPGA custom function block via the `Assigned Setup` property of the FPGA custom function blocks.

For more information on the assignment of FPGA custom function blocks, refer to [Types of FPGA Applications](#) on page 1505.

Enabling the FPGA access for experiment software products

If the FPGA application provides access to FPGA signals and constants, you can enable the data acquisition by mapping the Trigger function port of an accessible FPGA custom function block or FPGA Setup function block to a model port.

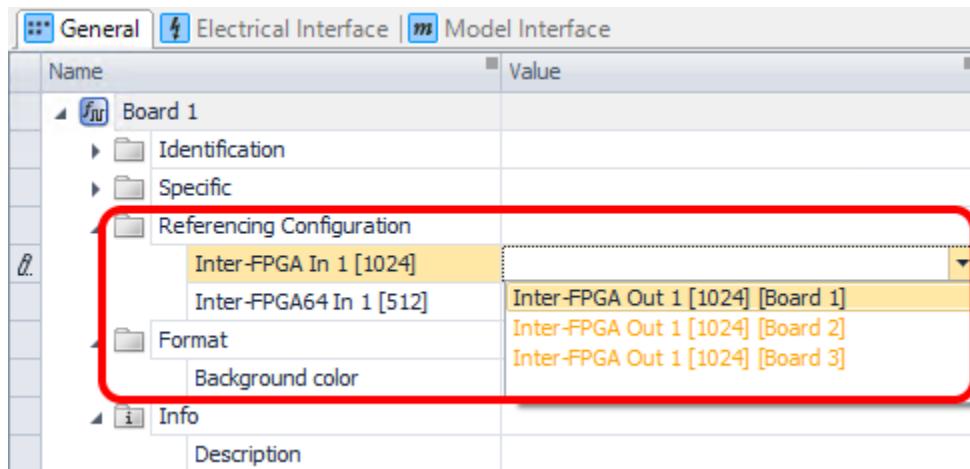


For more information on experimenting with the FPGA application, refer to [Accessing the FPGA Application with the Experiment Software](#) on page 1532.

For more information on implementing the FPGA access to the FPGA application, refer to [Accessing FPGA Applications with your Experiment Software \(RTI FPGA Programming Blockset Guide\)](#).

Configuring the inter-FPGA communication via IOCNET (SCALEXIO)

If the FPGA application supports inter-FPGA communication via IOCNET, you have to reference the Inter-FPGA In functions of the transmitting FPGA custom function block to the Inter-FPGA Out functions of the receiving FPGA custom function block in the Properties Browser.

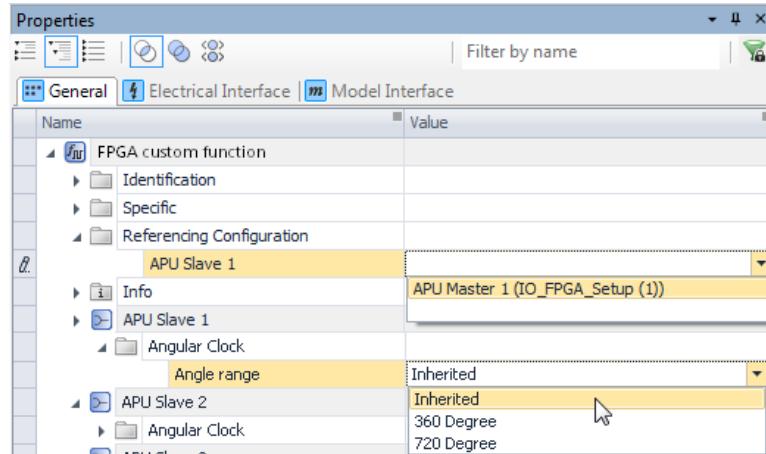


Configuring APU slaves (SCALEXIO)

An FPGA custom function block for a SCALEXIO system can provide up to 6 APU slaves.

In the Properties Browser, you assign a master APU provider to the APU slave of the FPGA custom function block via the APU Slave property. For more

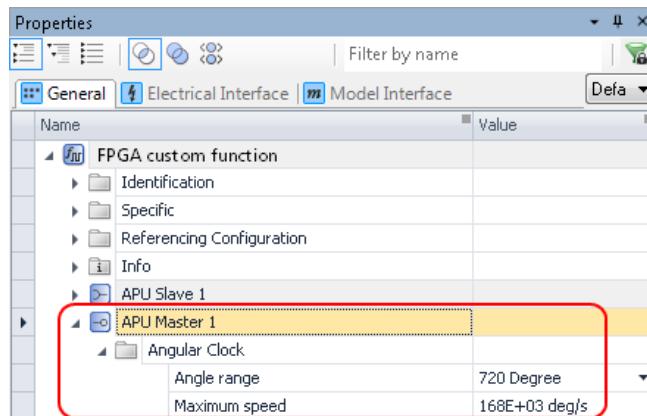
information on using the provided angle values, refer to [Using Angular Processing Units \(APUs\) on page 126](#).



Configuring APU masters (SCALEXIO)

An FPGA custom function block for a SCALEXIO system can provide up to 6 APU masters.

As of RTI FPGA Programming Blockset 3.6, you can configure the maximum speed and the angle range of the APU masters as shown below:



Configuring the maximum speed The angular speed of the APU master might exceed the angular speed that can be processed by an APU slave of a hardware resource. Therefore, you can specify a maximum speed that fulfills the requirements of your application. ConfigurationDesk uses the specified maximum speed to optimize function blocks and to check if the hardware resources support the maximum speed.

Note

The FPGA custom function block does not saturate the angular speed of the APU master to the specified maximum speed.

Configuring the angle range If the FPGA custom function block is modeled or handcoded with configurable angle ranges for the APU masters, you can select the angle range that relates to one engine cycle. You can select either 360° or 720°. The engine cycle of a four-stroke piston engine, for example, covers 720°.

The angle range does not influence the step size of the angular processing unit (APU). The angle step size is always 0.011°, rounded to 3 decimal digits.

Configuring characteristics of the electrical interface (SCALEXIO)

NOTICE**Original I/O configuration of the FPGA custom function block might damage the external device interface.**

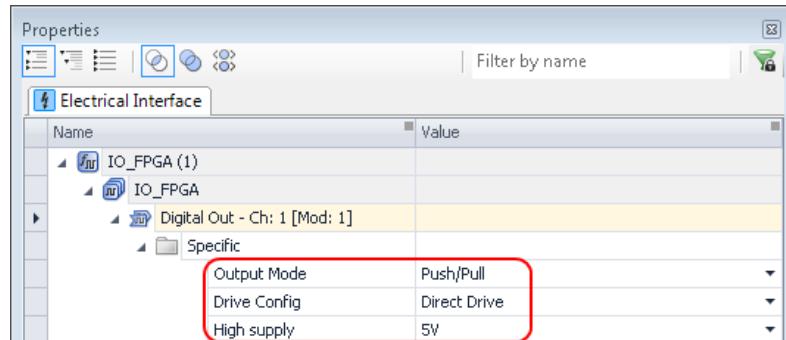
During the initialization of the FPGA, the original I/O configuration of the FPGA custom function block is active for a few milliseconds until the I/O configuration configured in ConfigurationDesk is effective.

- Make sure that the external device interface cannot be damaged by the original I/O configuration.
The Message Viewer displays changes of the original I/O configuration when you download and execute the real-time application.

Note

You can configure the electrical interface of FPGA custom function blocks only if they are built with RTI FPGA Programming Blockset as of version 3.2.

In the Properties Browser, you set the characteristics of the electrical interface. The settings cannot be changed after the real-time application is built.



The following table gives you an overview of configurable I/O functionalities and the properties you can change.

Logical Signal	Property	Value/Description
Analog In	Input range	Lets you specify the input voltage range that can be converted from analog to digital for the chosen ADC channel. Input voltages outside the specified range are ignored.
Digital In	Threshold init voltage	Lets you specify the voltage value that is used for the threshold in mV. This setting can be overwritten by the FPGA application during run time.
▪ Digital InOut ▪ Digital In/Out-Z ▪ Digital Out ▪ Digital Out-Z	Output Mode	<ul style="list-style-type: none"> ▪ Low-side switch Lets you actively drive the output to GND to output a low-level signal. An external load to VCC is required to output a high-level signal. ▪ High-side switch Lets you actively drive the output to VCC to output a high-level signal. An external load to GND is necessary to output a low-level signal. ▪ Push-pull Lets you drive the output between VCC and GND. An external load is not required.
	Drive Config	<p>Lets you enable/disable the termination of the signal line by an internal resistor.</p> <ul style="list-style-type: none"> ▪ Direct Drive Lets you directly drive the I/O signal. The internal termination resistor is disabled. ▪ 68 Ohm Terminated Lets you terminate the I/O signal with an internal 68 Ω resistor.
	High supply	Lets you select the VCC voltage that determines the high-level voltage for the high-side switch.
	Threshold init voltage	<p>Lets you set the initial threshold voltage for a digital input signal in 100 mV steps.</p> <p>This setting can be overwritten by the FPGA application during run time.</p>
▪ RS485 Rx ▪ RS485 Rx/Tx ▪ RS485 Tx	RS485 Termination	<p>Lets you enable an internal termination between the signal lines.</p> <ul style="list-style-type: none"> ▪ Open The signal lines are not terminated. ▪ Terminated The internal termination is enabled. <p>This setting can be overwritten by the FPGA application during run time.</p>
	High Supply	DS6651 only: Lets you set the differential output voltage. This setting can be overwritten by the FPGA application during run time.

Initializing of the DS6602 DDR4 RAM (SCALEXIO)

The DS6602 FPGA Base Board for SCALEXIO systems provides a 4 GB DDR4 RAM that can be used by the FPGA application. To start the FPGA application with defined DDR4 RAM data values, you can initialize the DDR4 RAM.

For instructions, refer to [How to Initialize the DDR4 RAM of the DS6602](#) on page 1536.

Related topics**Basics**

Accessing the FPGA Application with the Experiment Software.....	1532
Introduction (FPGA).....	1502
Using Angular Processing Units (APUs).....	126

Configuring Standard Features (FPGA)

Purpose

The FPGA custom function block offers properties to configure standard features of the model interface.

Model interface

The interface to the behavior model of the function block provides the following standard features.

Configuration Feature	Further Information
Function Ports	
Initialization behavior	Specifying Initialization and Stop Behavior on page 88
Test automation support	Configuring Test Automation Support on page 92

Related topics**Basics**

Introduction (FPGA).....	1502
--------------------------	------

Accessing FPGA Applications At Run Time

Accessing the FPGA Application with the Experiment Software

Use cases	To access an FPGA application with your experiment software is useful in the following cases, for example: <ul style="list-style-type: none">▪ To trace signals and adjust control parameters for optimization.▪ To use the FPGA application with various settings, e.g., you can scale analog I/O signals to support different sensors and actors.▪ To debug the application, e.g., you can test the cable harness by switching the I/O interface to defined values.
Accessible FPGA custom function blocks	Signals and constants of an FPGA custom function block are accessible if you enable the access in the FPGA model before you build the FPGA application. Then, the imported FPGA custom function block provides the FPGA Trace model port group. RTI FPGA Programming Blocksets that support accessible FPGA applications The RTI FPGA Programming Blockset must support the build of accessible FPGA applications: <ul style="list-style-type: none">▪ You can trace FPGA signals as of RTI FPGA Programming Blockset 3.1.▪ You can adjust tunable FPGA constants and substitute values of the FPGA interfaces (FPGA test access) as of RTI FPGA Programming Blockset 3.4.▪ You can modify I/O signals as of RTI FPGA Programming Blockset 3.5. The handcode interface of the RTI FPGA Programming Blockset does not support the build of accessible FPGA applications.

Accessing FPGA applications with the experiment software

Note

Running applications might stop, if too many FPGA signals are traced

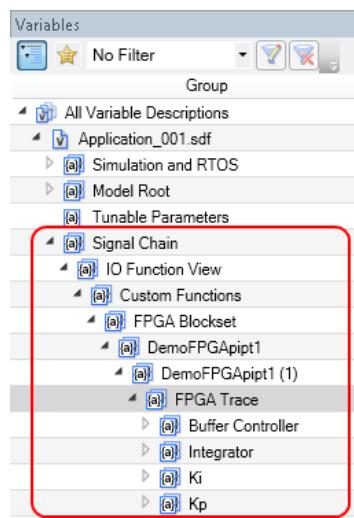
If you trace more than 100 signals with 32-bit values (or 50 signals with 64-bit values) every millisecond with your experiment software, tracing might cause a task overrun that stops the application.

- Reduce the number of signals that you trace with the experiment software. Only the values of signals traced with an experiment software are sent to the real-time processor and can cause a task overrun.

The data acquisition of all traceable FPGA signals is triggered by the shortest model task period of the mapped behavior model, so all signals of an FPGA application are captured at the same time. It is not possible to capture signals at different points in time.

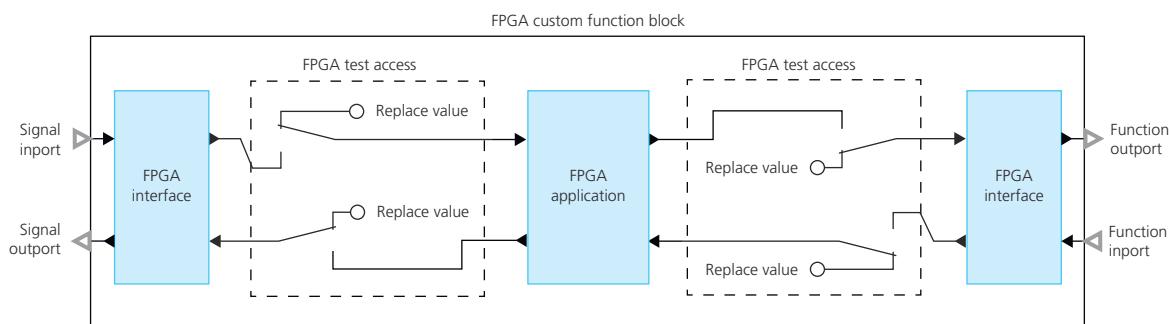
The experiment software can trace only FPGA signals with integer multiples of the trigger period for tracing FPGA signals.

When you build the real-time application, an SDF file will be generated. Refer to [Build Result Files of the Build Process \(ConfigurationDesk Real-Time Implementation Guide\)](#). You use the SDF file to access the FPGA signals and constants with your experiment software, for example, with ControlDesk. In ControlDesk, you can access the signals and constants in the Variables controlbar under [Signal Chain/I/O Function View/Custom Functions/FPGA Blockset/<CustomFunctionName>/<CustomFunctionInstance>/FPGA Trace](#).



Substituting values of the function block ports

An FPGA application with FPGA test access provides intervention points at the FPGA interface. These intervention points let you substitute values that are exchanged with the FPGA interface.



With the experiment software, you can set a value that replaces the original signal. A replace value becomes active when you switch the input or output from the original signal to the replace value.

In contrast to the test automation support that you enable in ConfigurationDesk, the FPGA test access is executed on the FPGA board. Therefore, you cannot enable FPGA test access in ConfigurationDesk. You must enable FPGA test access in the FPGA model before you build the FPGA application.

Added FPGA variables for FPGA test access The following table shows you the types of FPGA variables that are added to support FPGA test access.

Variable	Description
<FPGASignal> -TA_Switchvalue	This variable lets you switch between the actual values used as input or output for the FPGA interface and the substitute value. <ul style="list-style-type: none"> ▪ 0: The actual values are used. ▪ 1: The substitute value is used.
<FPGASignal> - TA_Replacevalue	This variable provides the substitute value that can be used instead the actual values.

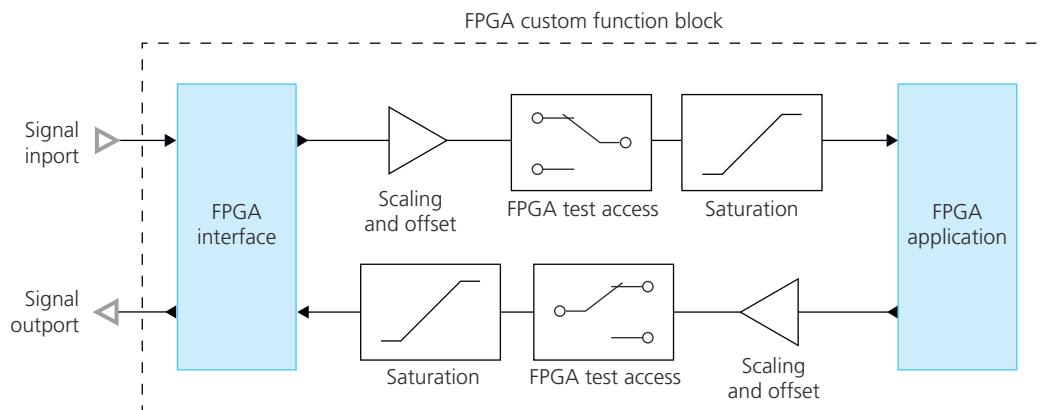
Modifying the I/O signals

An FPGA application that supports FPGA scaling lets you access and modify the interface signals of the FPGA application. To support FPGA scaling, FPGA scaling must be enabled in the FPGA model before you build the FPGA application.

FPGA scaling lets you modify the I/O signal of the electrical interface as follows:

- Scaling analog signals with a scaling factor.
- Adding signal offsets to analog signals.
- Saturating analog signals.
- Inverting digital I/O signals, including RS232/485 signals.

Modifying analog I/O signals The following illustration shows you how analog signals of the electrical interface are modified.

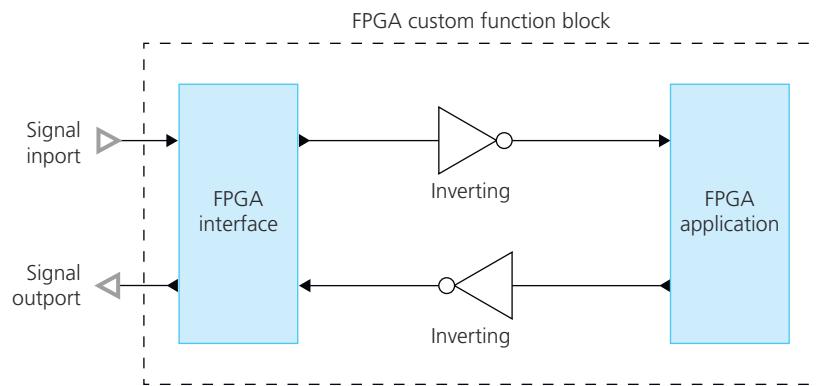


The table shows you the types of FPGA variables that are added to modify analog signals.

Variable	Description
<FPGASignal> - Scaling factor	Lets you specify the scaling factor. The scaling factor amplifies the signal before it is saturated or replaced via FPGA test access.
<FPGASignal> - Scaling offset	Lets you add an offset after the signal is scaled.

Variable	Description
<FPGASignal> - Saturation minimum value	Lets you set the minimum and maximum values to which the analog signals are saturated.
<FPGASignal> - Saturation maximum value	
<FPGASignal> - Invert polarity	Lets you invert the digital I/O signal. <ul style="list-style-type: none"> ▪ 0: The signal is not inverted: A high-level voltage or a low-high transition represents a 1 and vice versa. ▪ 1: The signal is inverted: A high-level voltage or a low-high transition represents a 0 and vice versa.

Inverting digital I/O signals and RS232/485 signals The following illustration shows you how digital and RS232/485 signals of the electrical interface are inverted.



The table shows you the types of FPGA variables that are added to invert digital and RS232/485 signals.

Signal Type	Variable	Description
Digital	<FPGASignal> - Invert polarity	Lets you invert the digital I/O signal. <ul style="list-style-type: none"> ▪ 0: The signal is not inverted: A high-level voltage or a low-high transition represent a 1 and vice versa. ▪ 1: The signal is inverted: A high-level voltage or a low-high transition represent a 0 and vice versa.
RS232/485 ¹⁾	<FPGASignal> - Invert polarity	Lets you invert RS232/485 signals. <ul style="list-style-type: none"> ▪ 0: The signal is not inverted. ▪ 1: The signal is inverted.

¹⁾ Only SCALEXIO systems

Related topics

Basics

[Accessing FPGA Applications with your Experiment Software \(RTI FPGA Programming Blockset Guide\)](#)

Initializing of the RAM Used by the FPGA Application

How to Initialize the DDR4 RAM of the DS6602

Objective	Starting the FPGA application with defined DDR4 RAM data values.
Basics	<p>The DS6602 FPGA Board of a SCALEXIO system provides a DDR4 RAM that can be used by the FPGA application.</p> <p>You can provide a file with initial values for the RAM memory to start the FPGA application with defined DDR4 RAM data values. The processing hardware writes the initial values to the FPGA base board during the initialization phase of the SCALEXIO system.</p> <p>SCALEXIO system with multiple DS6602 You can use the same file or different files to initialize the RAM of multiple FPGA base boards. The initializing of multiple boards takes more time which can lead to an timeout error during the initialization phase of the SCALEXIO system. Therefore you have to change the default timeout setting of the SCALEXIO system.</p>
Generating an initialization file	<p>The initialization file is a binary file. You can use the following MATLAB script to generate an initialization file with MATLAB.</p> <pre>% generate sample 32 bit values (4GB - 1 Word) 1..2^30-1 (2^30*4 Byte = 4 GB) % generate sample 64 bit values (4GB - 1 Word) 1..2^29-1 (2^29*8 Byte = 4 GB) ddr4_4gb = 1:2^30-1; % open, write, close file fid = fopen('ddr4_4gb.bin', 'w'); fwrite(fid, ddr4_4gb, 'uint32'); % or as different data type % fwrite(fid, ddr4_4gb, 'single'); % fwrite(fid, ddr4_4gb, 'double'); % 64 bit -> use only half as many elements ... fclose(fid)</pre>
	<p>Note</p> <p>You can use any data type to initialize the DDR4 RAM.</p>
Preconditions	<p>The following preconditions must be fulfilled:</p> <ul style="list-style-type: none"> ▪ At least one initialization file must be available. For more information, refer to Generating an initialization file on page 1536. ▪ An FTP client must be installed on the host PC.

- One of the following SCALEXIO Processing Unit variants must be installed:
 - A SCALEXIO Processing Unit of the HCP product line with a Real-Time PC Version 2.0 or later.
 - A SCALEXIO Processing Unit of the HPP product line.
- For more information on the SCALEXIO Processing Unit variants, refer to [Variants of the SCALEXIO Processing Unit \(SCALEXIO Hardware Installation and Configuration\)](#).
- A SCALEXIO SSD must be installed to save the initialization file.

Workflow

Proceed the following workflow to initialize the DDR4 RAM:

- Provide the initialization file to the SCALEXIO system. Refer to Part 1.
 - If you initialize the RAM of multiple FPGA base boards, change the timeout setting of the SCALEXIO system. Refer to Part 2.
-

Part 1

To provide the initial values

- 1 Open the FTP client and copy the initialization files to the following folder on the SCALEXIO SSD:
`ftp://<SCALEXIO IP address>/userstorage1/`
You can use the same file to initialize the RAM of multiple DS6602 FPGA Base Boards.
 - 2 In ConfigurationDesk, enter the file name of the initialization file to the DDR4 filename property of each FPGA custom function block or FPGA Setup function block. The file name can be entered with or without the ending `.bin`.
-

Part 2

To change the timeout setting to initialize multiple FPGA base boards

- 1 Open the SCALEXIO web interface. Refer to [Web Interface for Configuring the SCALEXIO System \(SCALEXIO Hardware Installation and Configuration\)](#).

2 Open the Configuration - Custom Configuration page.

The screenshot shows the dSPACE SCALEXIO Configuration software interface. At the top, there is a navigation bar with tabs: MAIN, CONTROL, CONFIGURATION (which is highlighted in red), SUPPORT, NVDATA, FIRMWARE, and MESSAGES. Below the navigation bar is a horizontal menu with buttons: Network Configuration, IOCNET Configuration, Ethernet Board Configuration, Safety Configuration, Fan Monitoring, User Storage, and Custom Configuration. The 'Custom Configuration' button is also highlighted in red. The main content area has a title 'Custom Configuration' and a sub-section 'Options:' followed by an empty text input field. To the right of the input field is a note: 'Contact dSPACE before adding or changing any custom options.' and a blue 'Change' button. At the bottom of the interface, there is a footer with the text '© 2020, dSPACE GmbH. All rights reserved.' and 'dSPACE v5.0.1'.

3 Enter the following to the Options edit field:

```
[APPLICATION]
COMMAND_TIMEOUT=120
```

The unit of the timeout value is second. The default value is 60 s. Depending on the number of FPGA base boards, higher values than 120 s might be necessary.

4 Click Change and restart the SCALEXIO system.

Result

The SCALEXIO system initializes the DDR4 RAM during the initialization phase of the SCALEXIO system.

General Hardware Dependencies

Optimal Selection of Analog Output Channels on DS2680 and DS6101

Objective

If you need analog signals that must be updated synchronously, you have to use channels of the same multiplex cycle provided by the DAC on the real-time hardware.

Update phases of analog outputs on DS2680

The output signals of the Analog Out 1, Analog Out 3, and Analog Out 4 channel types are updated cyclically by means of multiplexed digital-to-analog converters (DACs). Depending on the channel numbers used, there can be differences of up to 5.1 µs when the signals are updated.

The following table shows which channels are updated synchronously.

Multiplex Cycle	Synchronous Output Channels		
	Analog Out 1	Analog Out 3	Analog Out 4
1	1, 5, 9, 13	1, 5	1, 5
2	2, 6, 10, 14	2, 6	2, 6
3	3, 7, 11, 15	3, 7	3, 7
4	4, 8, 12	4	4, 8

For example, if you use channels 1 and 5 from Analog Out 1 channel type in combination with channels 1 and 5 from Analog Out 3 channel type, all the related signals are updated synchronously.

Update phases of analog outputs on DS6101

The output signals of the Analog Out 6, Analog Out 8, and Analog Out 9 channel types are updated cyclically by means of multiplexed digital-to-analog converters (DACs). Depending on the channel numbers used, there can be differences of up to 5.1 µs when the signals are updated.

The following table shows which channels are updated synchronously.

Multiplex Cycle	Synchronous Output Channels		
	Analog Out 6	Analog Out 8	Analog Out 9
1	1, 5	1	1
2	2, 6	2	2
3	3, 7	3	3
4	4, 8	None	4

For example, if you use channels 1 and 5 from Analog Out 6 channel type in combination with channel 1 from Analog Out 9 channel type, all the related signals are updated synchronously.

Circuit Diagrams of Channel Types

Where to go from here

Information in this section

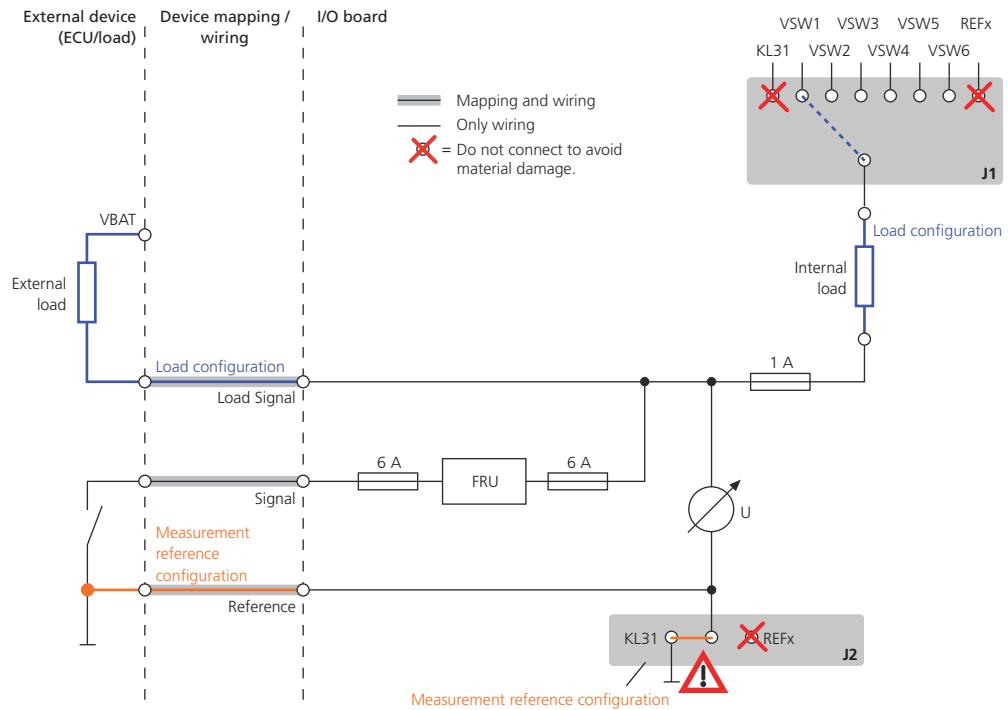
Analog In 1.....	1544
Analog In 2.....	1546
Analog In 4.....	1546
Analog In 5.....	1547
Analog In 6.....	1547
Analog In 7.....	1548
Analog In 8.....	1548
Analog In 9.....	1549
Analog In 10.....	1549
Analog In 11.....	1550
Analog In 12.....	1550
Analog In 14.....	1551
Analog In 15	1551
Analog In 16.....	1552
Analog In 17.....	1552
Analog Out 1.....	1553
Analog Out 2.....	1554
Analog Out 3.....	1554
Analog Out 4.....	1555
Analog Out 6.....	1556

Analog Out 7.....	1556
Analog Out 8.....	1557
Analog Out 9.....	1557
Analog Out 10.....	1558
Analog Out 11.....	1559
Analog Out 12.....	1559
Analog Out 13.....	1560
Bus 1	1560
CAN 1.....	1563
CAN 2.....	1564
CAN 3.....	1566
CAN 4.....	1566
CAN 5.....	1567
CAN 6.....	1568
Digital In 1.....	1569
Digital In 2.....	1571
Digital In 3.....	1572
Digital In 4.....	1572
Digital In 5.....	1573
Digital In 9.....	1573
Digital In 10.....	1574
Digital Out 1.....	1575
Digital Out 2.....	1576
Digital Out 3.....	1577
Digital Out 4.....	1578
Digital Out 5.....	1579
Digital Out 7.....	1580
Digital Out 8.....	1581
Digital In/Out 1.....	1582
Digital In/Out 3.....	1584
Digital In/Out 5.....	1586

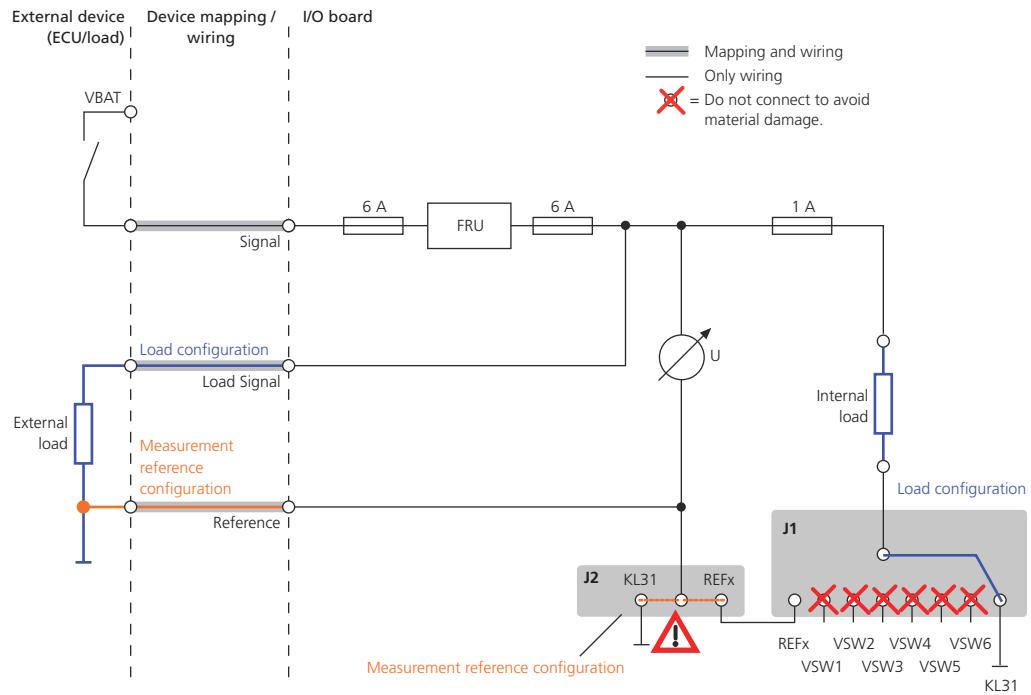
Digital In/Out 6.....	1588
Digital In/Out 8.....	1589
Digital In/Out 9.....	1590
Digital In/Out 10.....	1591
Flexible In 1.....	1593
Flexible In 2.....	1595
Flexible In 3.....	1603
Flexible Out 1.....	1604
Flexible In/Out 1.....	1608
FlexRay 1.....	1609
FlexRay 2.....	1610
FlexRay 3.....	1611
FlexRay 4.....	1613
LIN 1.....	1615
LIN 2.....	1615
LIN 3.....	1616
LIN 4.....	1616
Power Switch 1.....	1617
Power Switch 2.....	1618
Resistance Out 1.....	1619
Resistance Out 2.....	1620
Resolver In 2.....	1621
Trigger In 1.....	1621
Trigger In 2.....	1622
Trigger In 3.....	1622
UART 1.....	1623
UART 2.....	1624
UART 3.....	1625
UART 4.....	1626

Analog In 1

Voltage measurement of low-side controlled load via A/D converter



Voltage measurement of high-side controlled load via A/D converter



Load configuration:

Use external or internal load. If you use an internal load, define the reference voltage with J1 as follows:

- Jumper is set to KL31: Internally connected to KL31 reference.
- Jumper is set to REFx: Reference of internal load is connected to measurement reference.
In this case the reference port has to be connected externally and J2 has to be set to REFx (see Measurement reference configuration).

Measurement reference configuration:

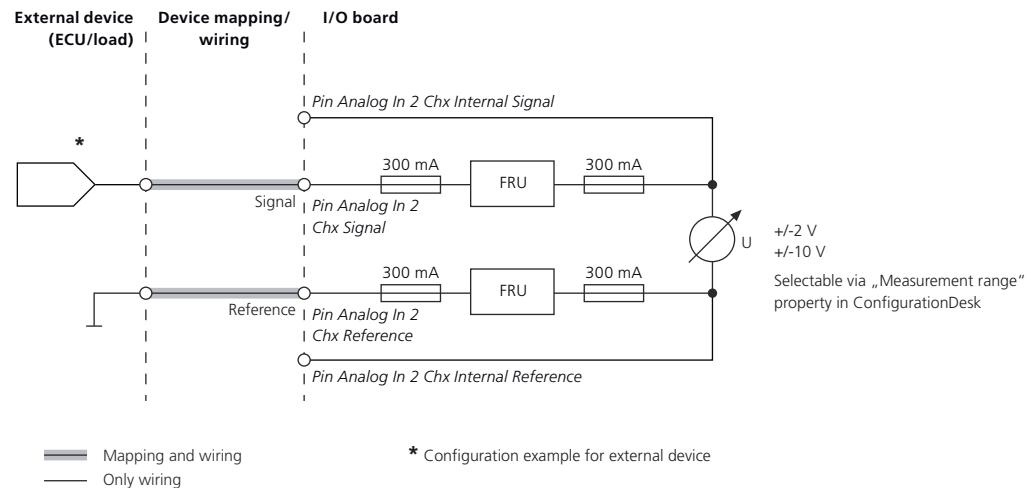
Measurement against:

- KL31: J2 is set to KL31
- External reference: - J2 is not present.
 - Reference port has to be connected externally.
- Internal load: - J2 is set to REFx.
 - Reference port has to be connected externally.
 - J1 is set to REFx (see Configuration 1).

⚠ To avoid ground loops do not set J2 to KL31 and connect an external potential to Reference at the same time.

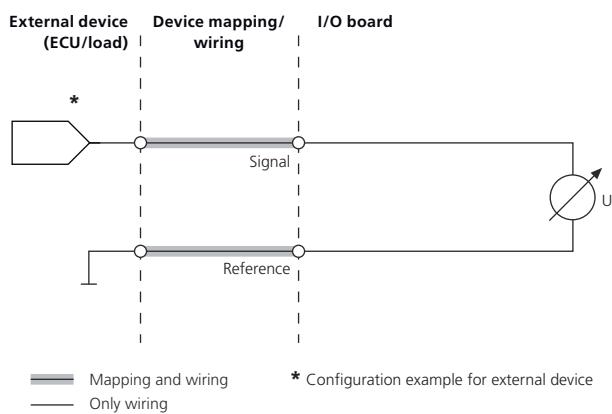
Analog In 2

Voltage measurement



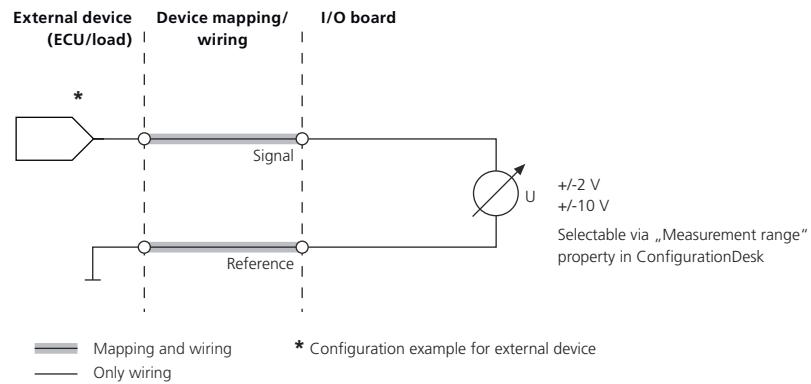
Analog In 4

Voltage measurement



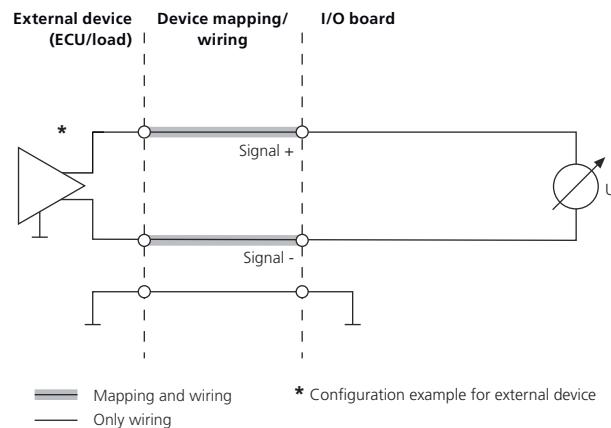
Analog In 5

Voltage measurement



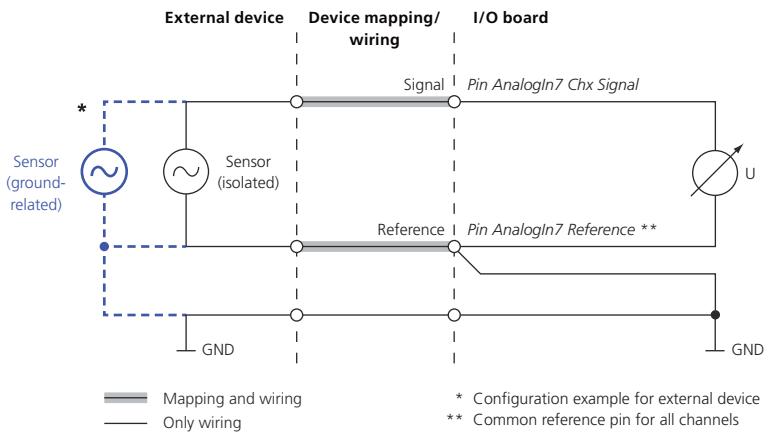
Analog In 6

Voltage measurement



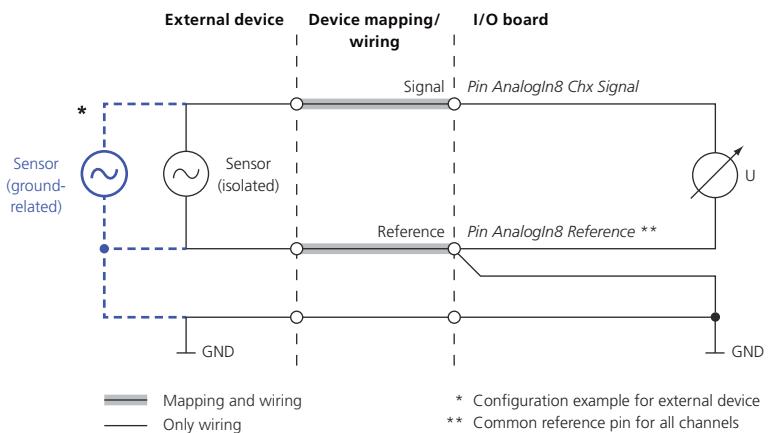
Analog In 7

Voltage measurement



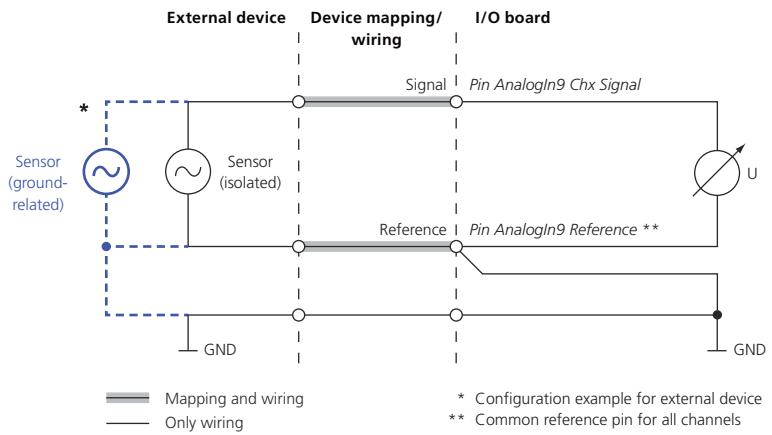
Analog In 8

Voltage measurement



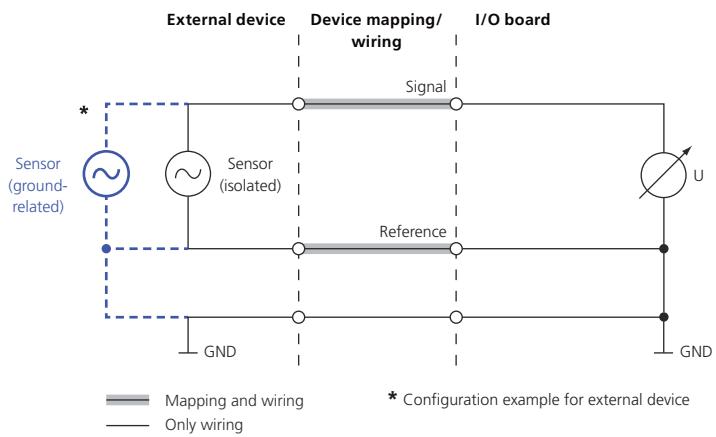
Analog In 9

Voltage measurement



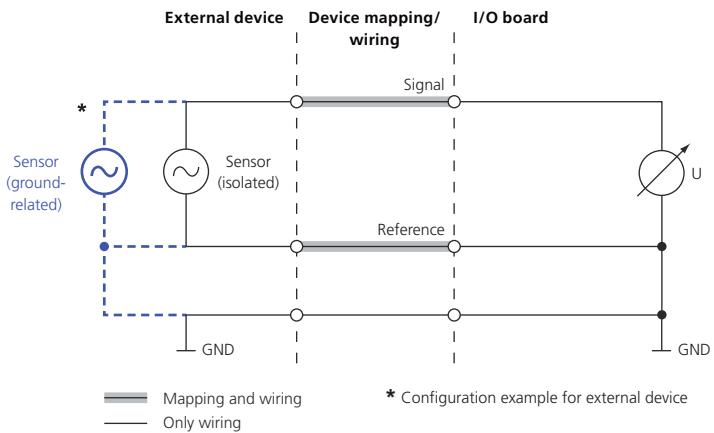
Analog In 10

Voltage measurement



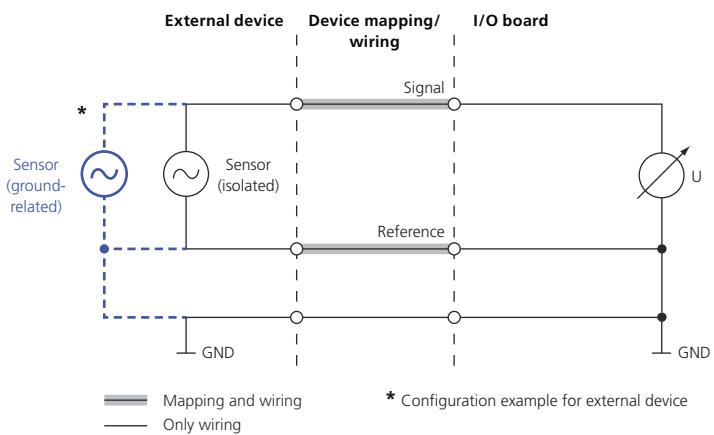
Analog In 11

Voltage measurement



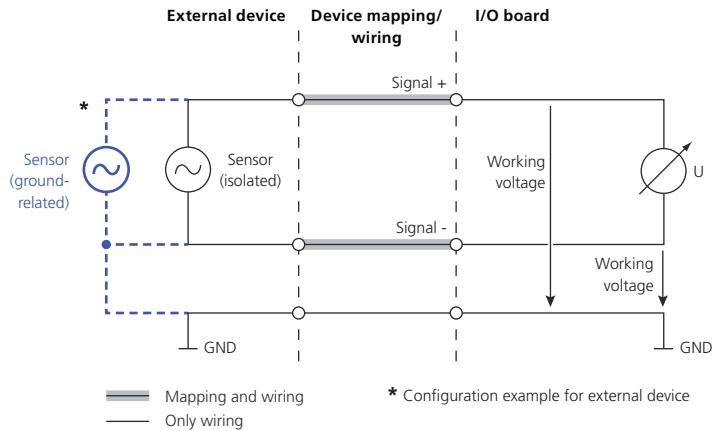
Analog In 12

Voltage measurement



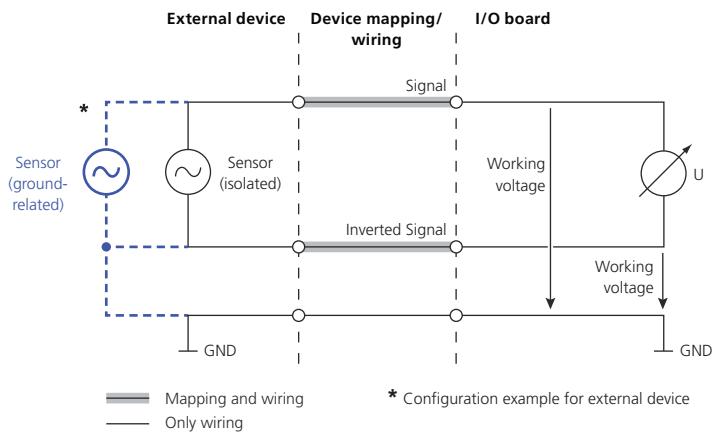
Analog In 14

Analog voltage measurement



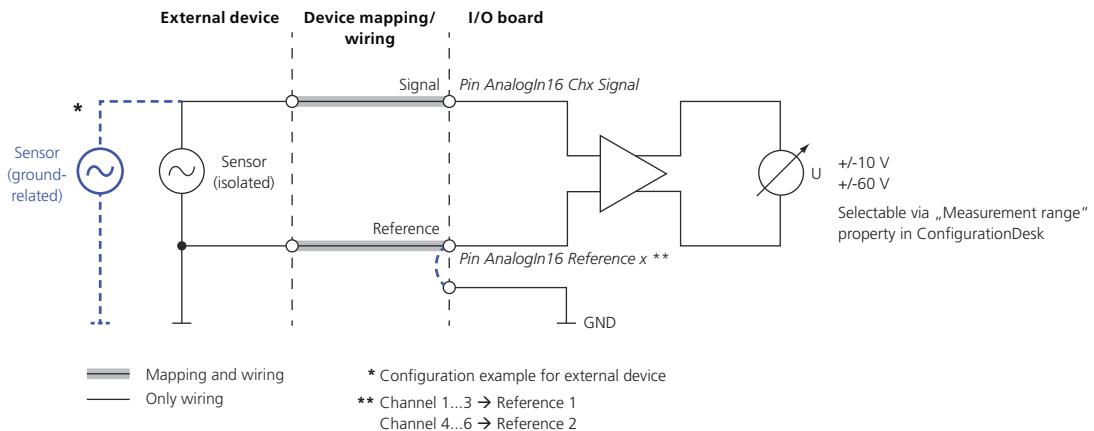
Analog In 15

Analog voltage measurement



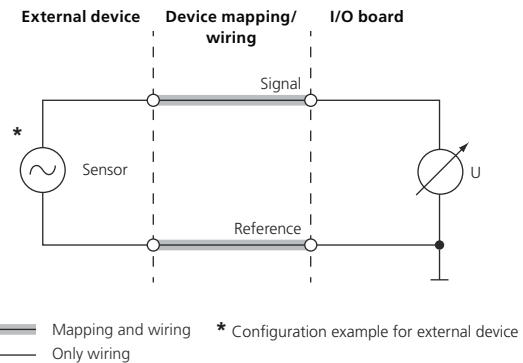
Analog In 16

Analog voltage measurement



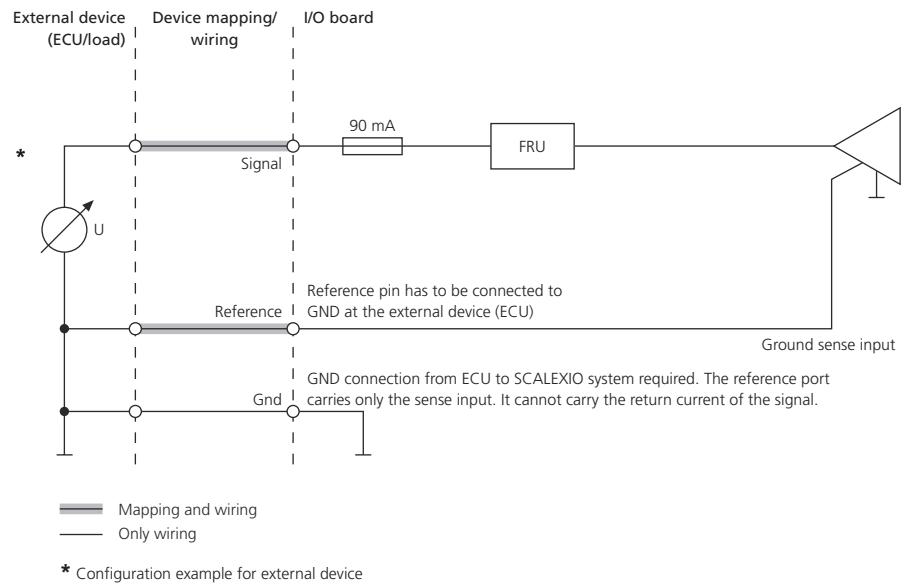
Analog In 17

Analog voltage measurement



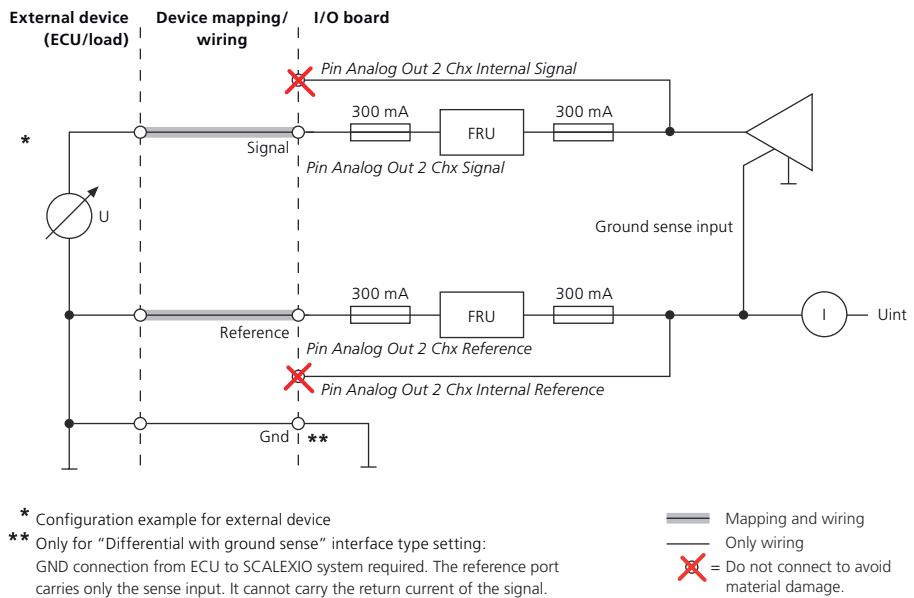
Analog Out 1

Voltage generation



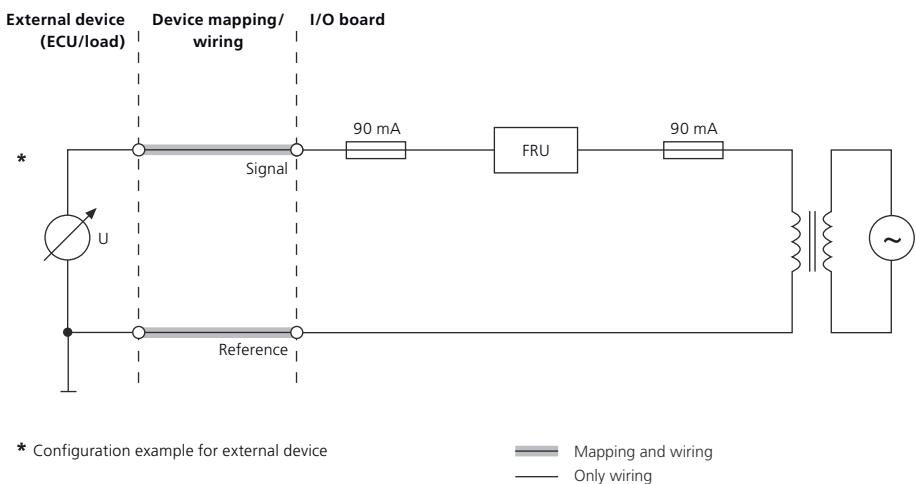
Analog Out 2

Voltage generation



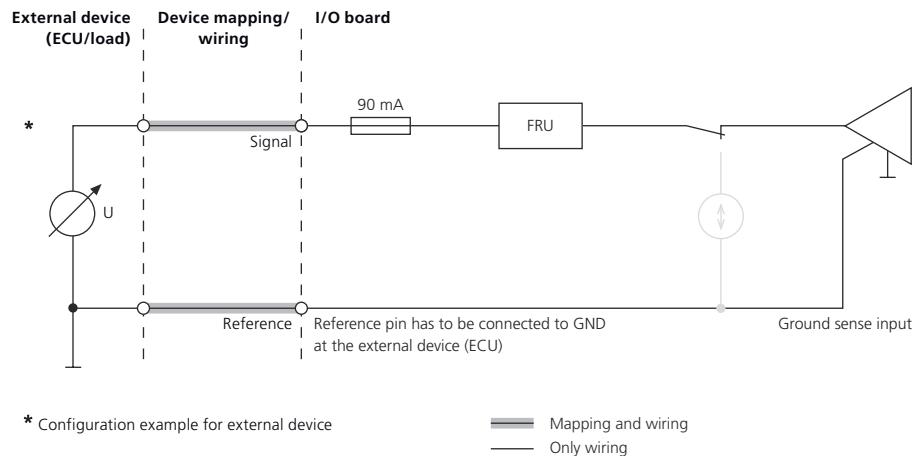
Analog Out 3

Isolated voltage generation

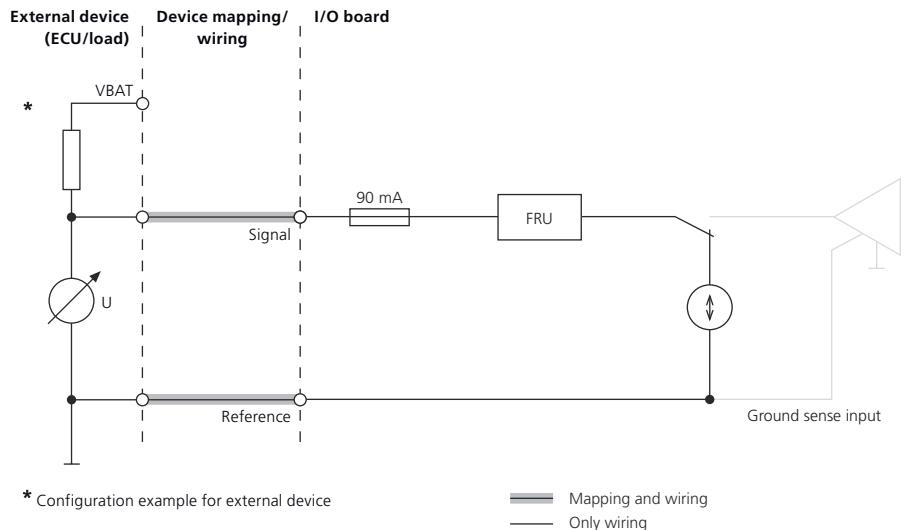


Analog Out 4

Voltage generation

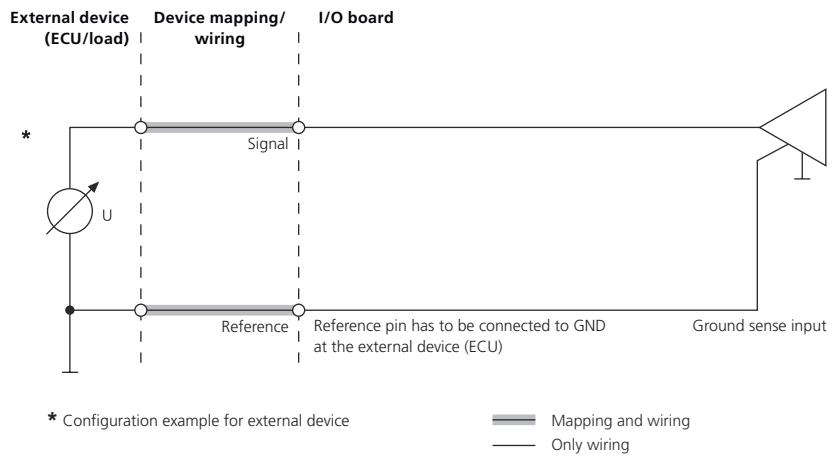


Current sink



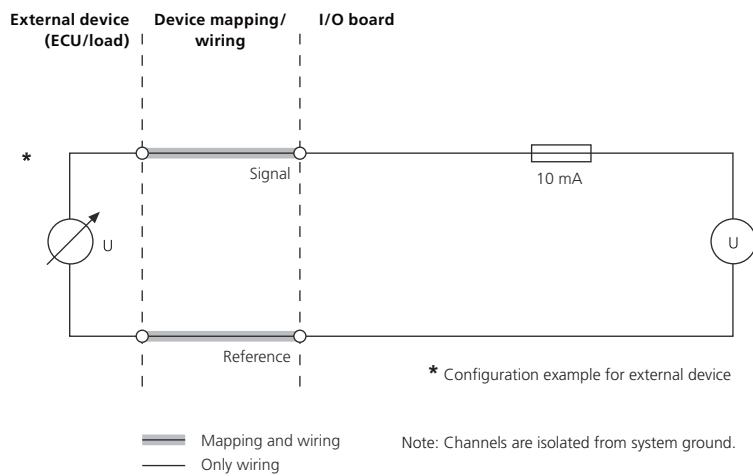
Analog Out 6

Voltage generation



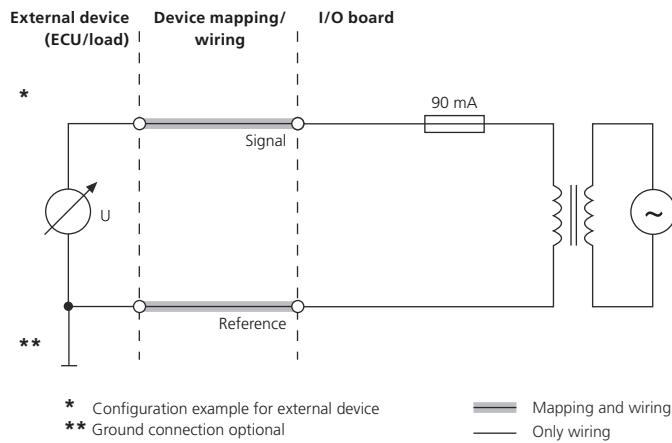
Analog Out 7

Voltage generation



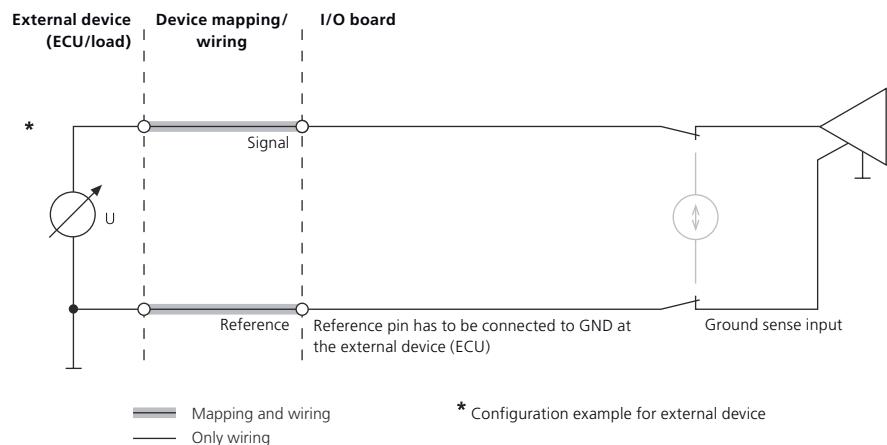
Analog Out 8

Voltage generation

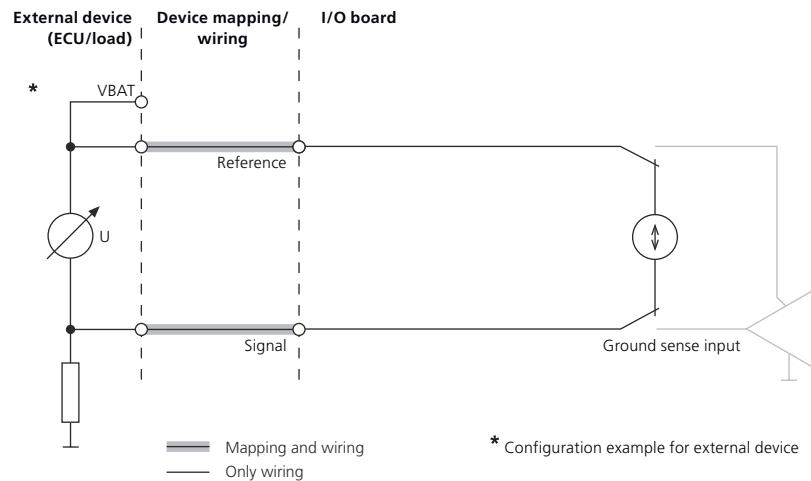


Analog Out 9

Voltage generation

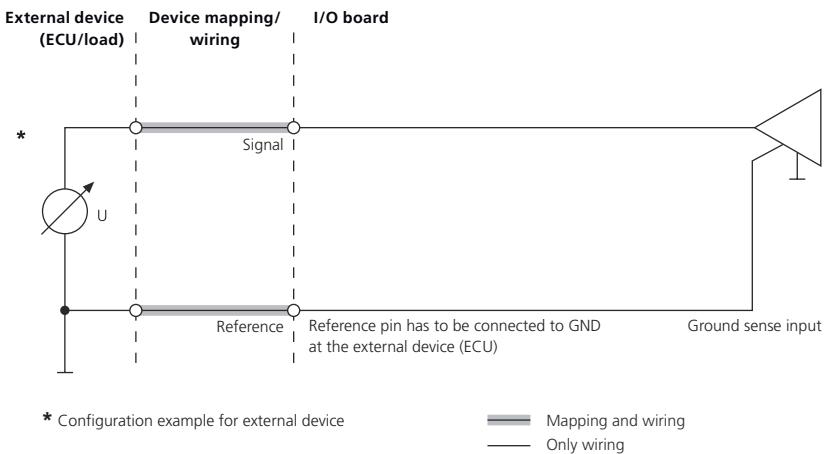


Current sink



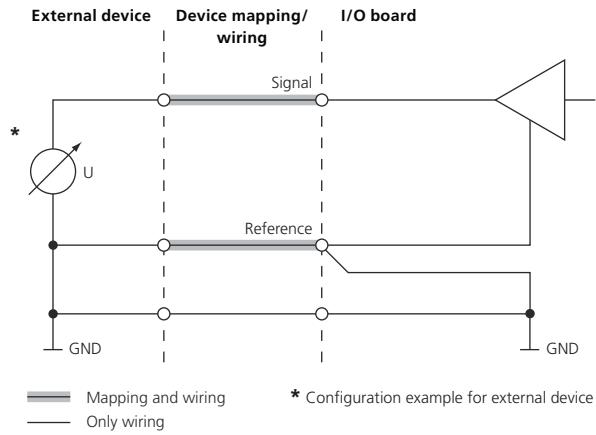
Analog Out 10

Voltage generation



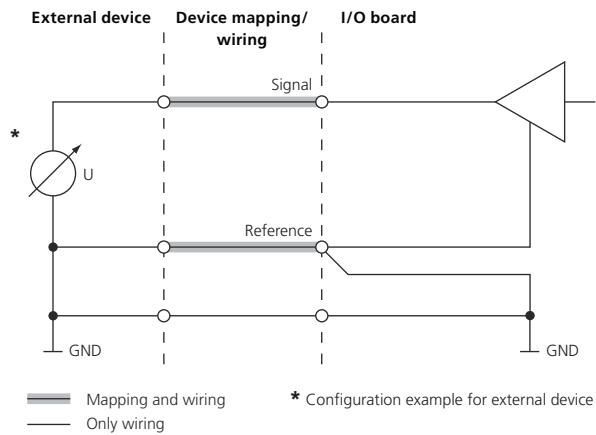
Analog Out 11

Voltage generation



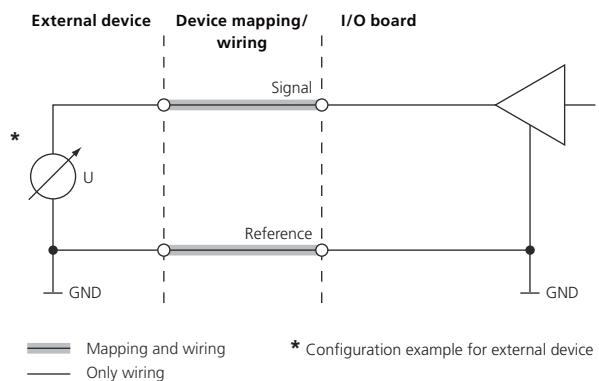
Analog Out 12

Voltage generation



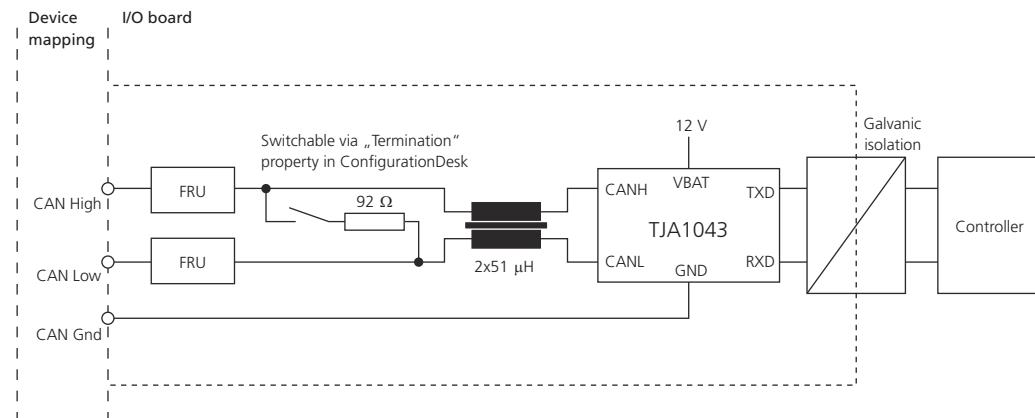
Analog Out 13

Voltage generation

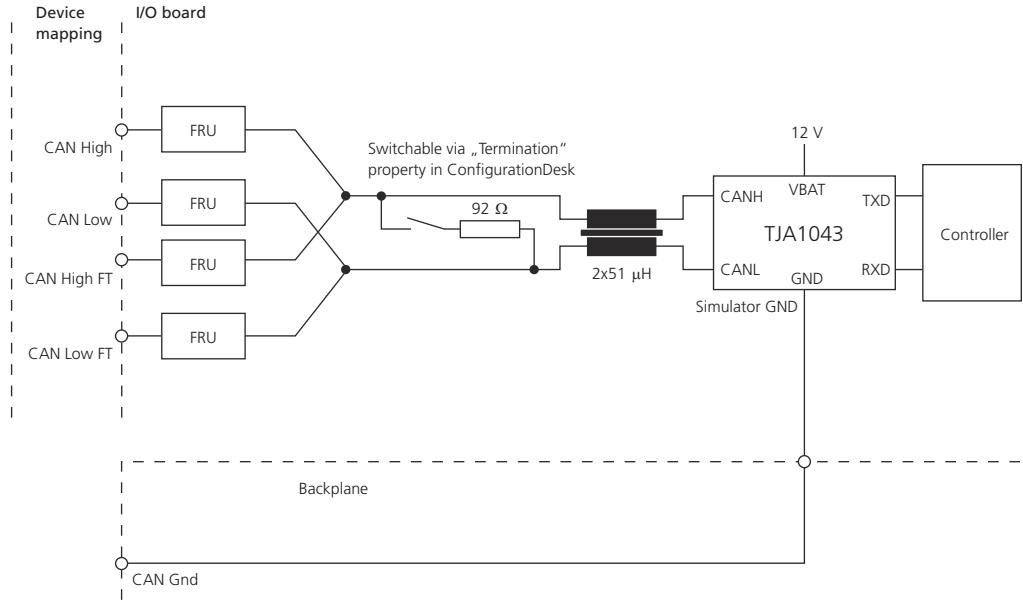


Bus 1

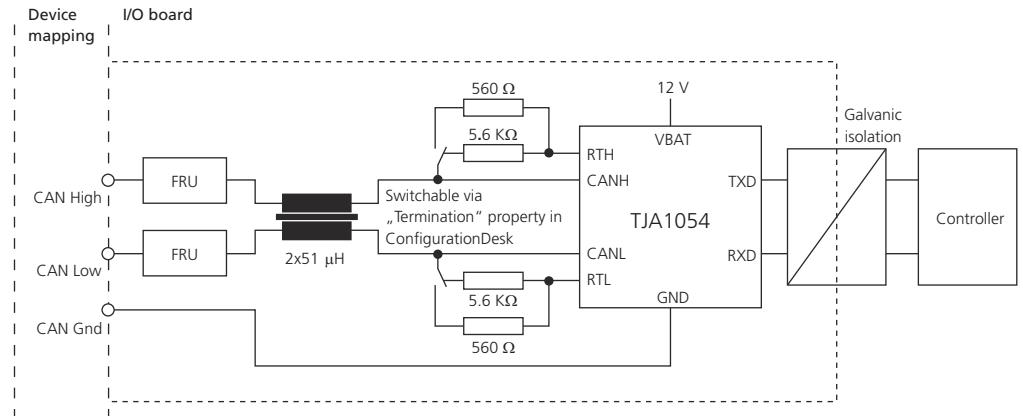
ISO 11898-2 High-Speed CAN without feedthrough



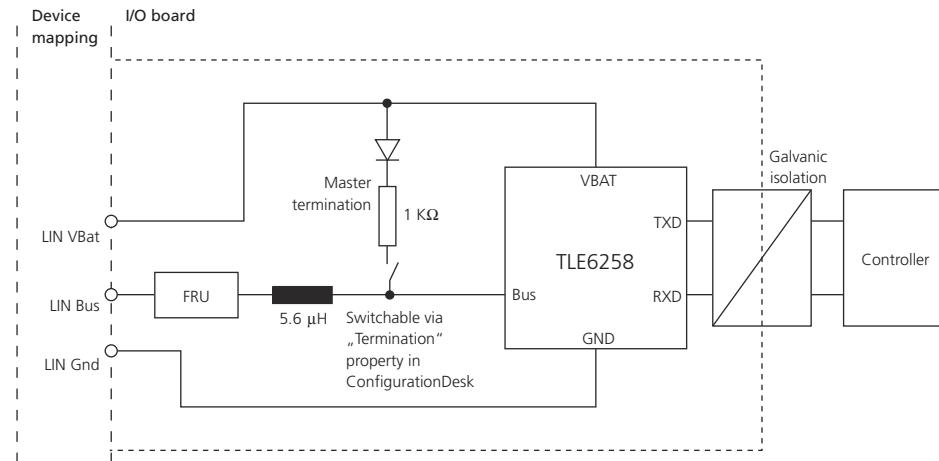
ISO 11898-2 High-Speed CAN with feedthrough



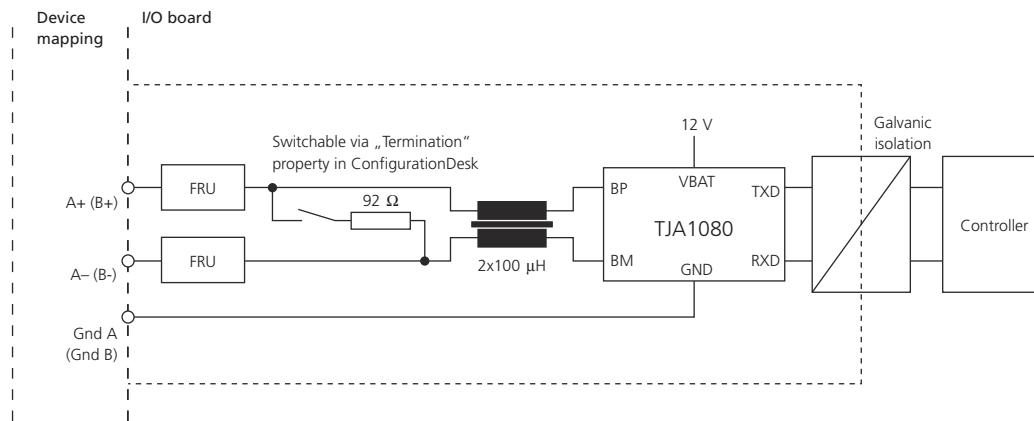
ISO 11898-3 Fault-Tolerant CAN without feedthrough



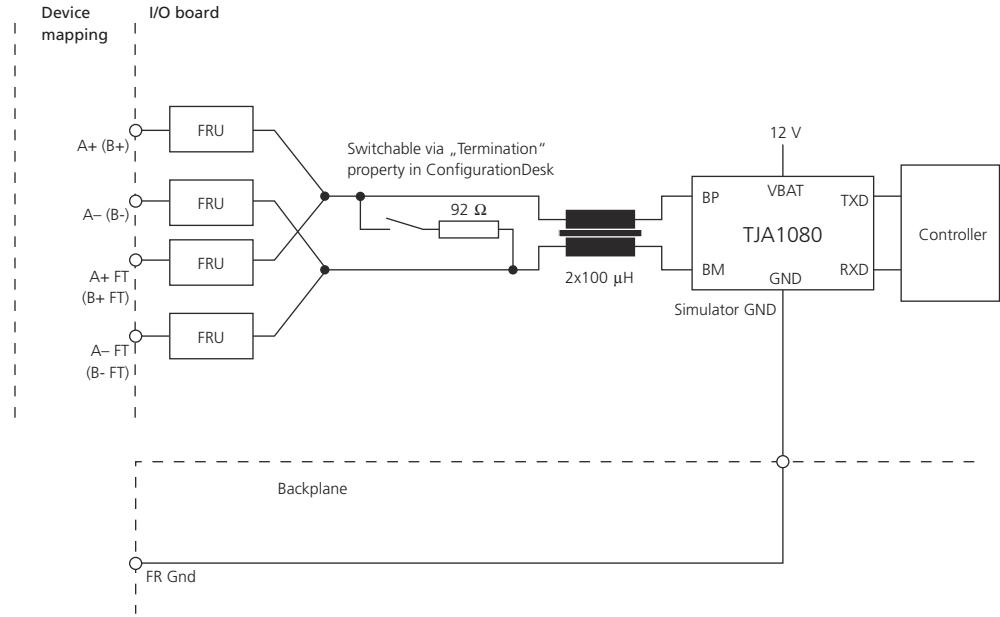
LIN



FlexRay without feedthrough

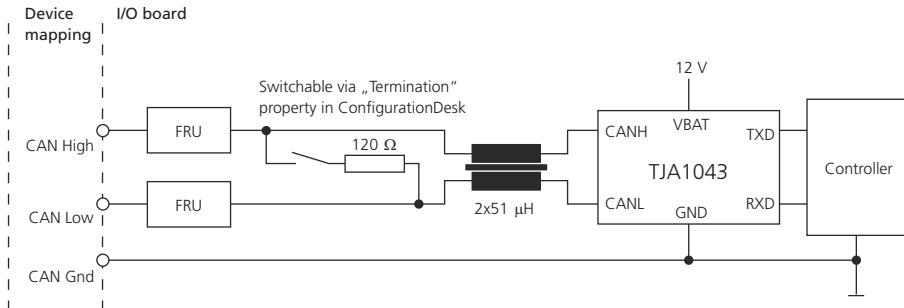


FlexRay with feedthrough

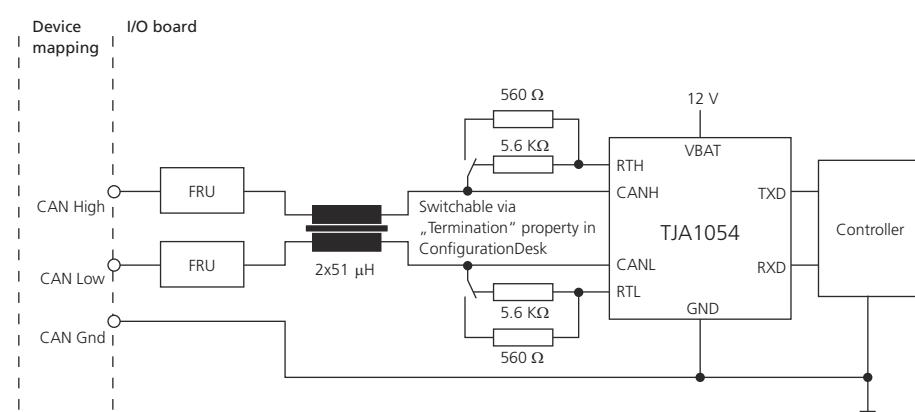


CAN 1

ISO 11898-2 High-Speed CAN

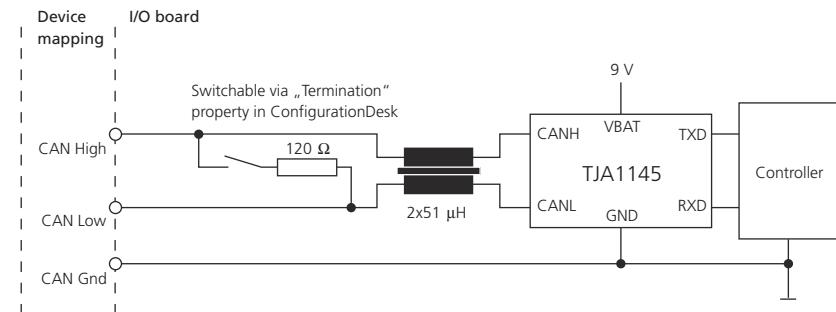


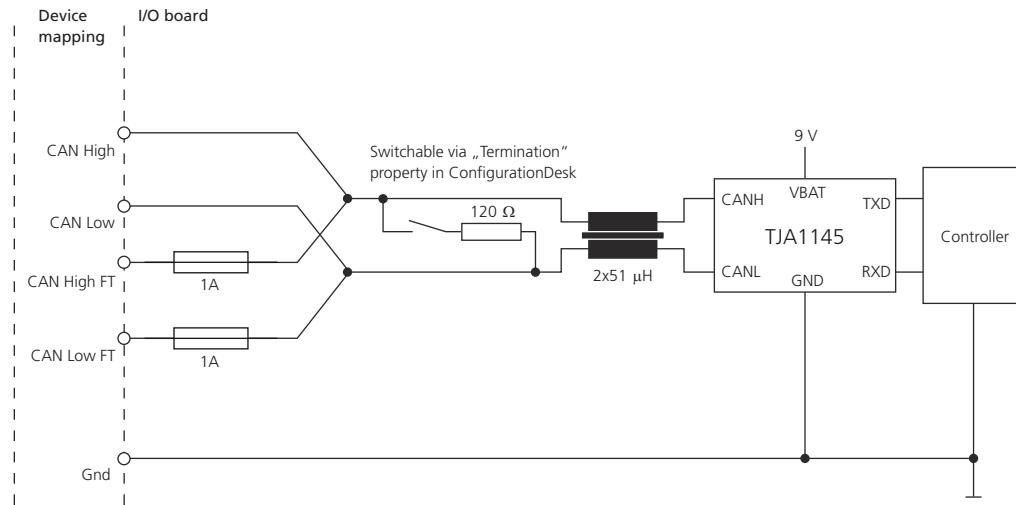
ISO 11898-3 Fault-Tolerant CAN

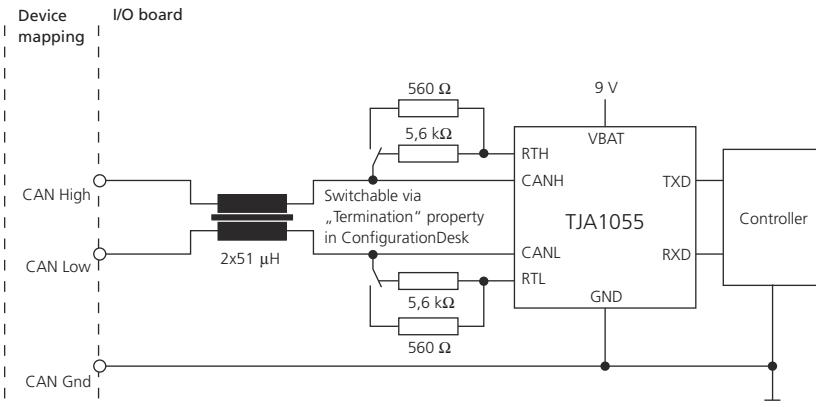


CAN 2

ISO 11898-2 High-Speed CAN without feedthrough

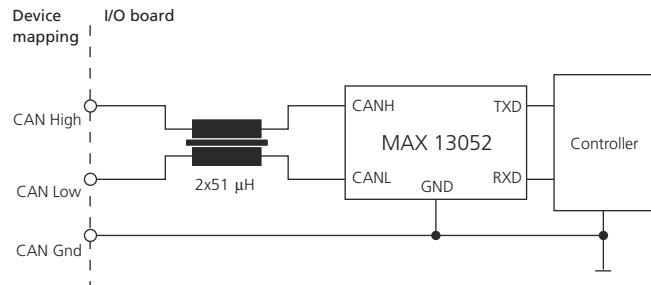


**ISO 11898-2 High-Speed CAN
with feedthrough**


**ISO 11898-3 Fault-Tolerant
CAN**


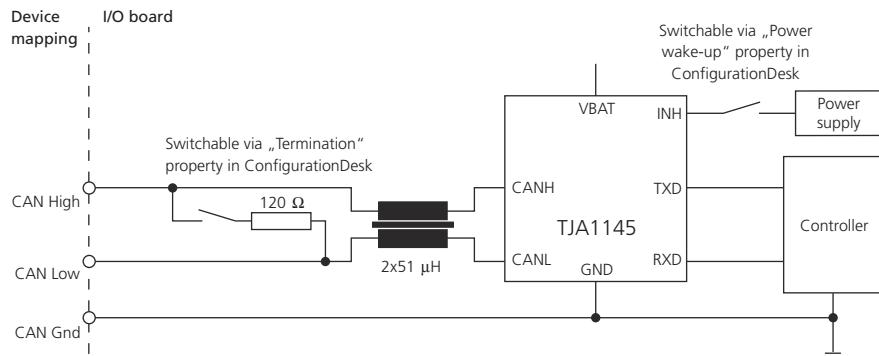
CAN 3

ISO 11898-2 High-Speed CAN without feedthrough



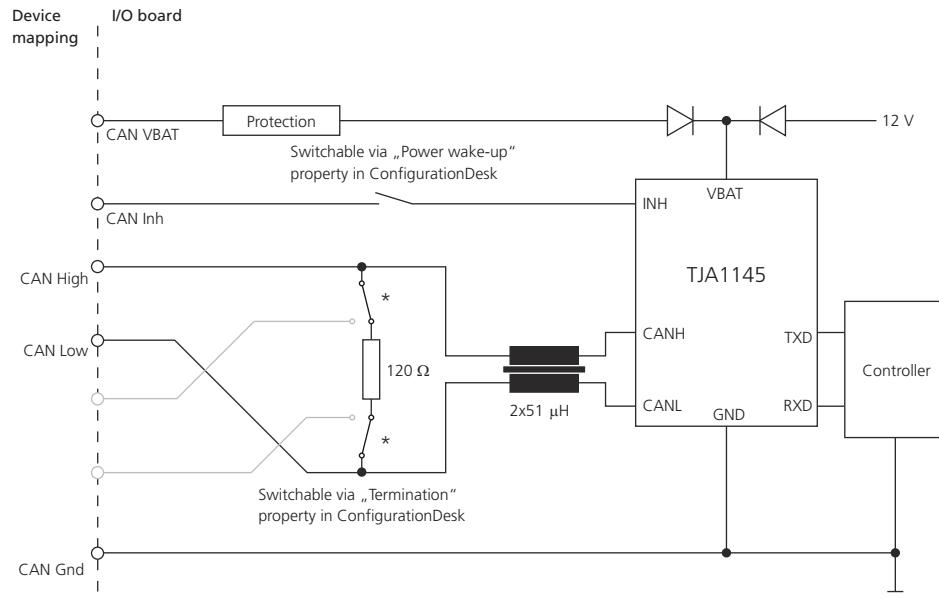
CAN 4

ISO 11898-2 High-Speed CAN without feedthrough

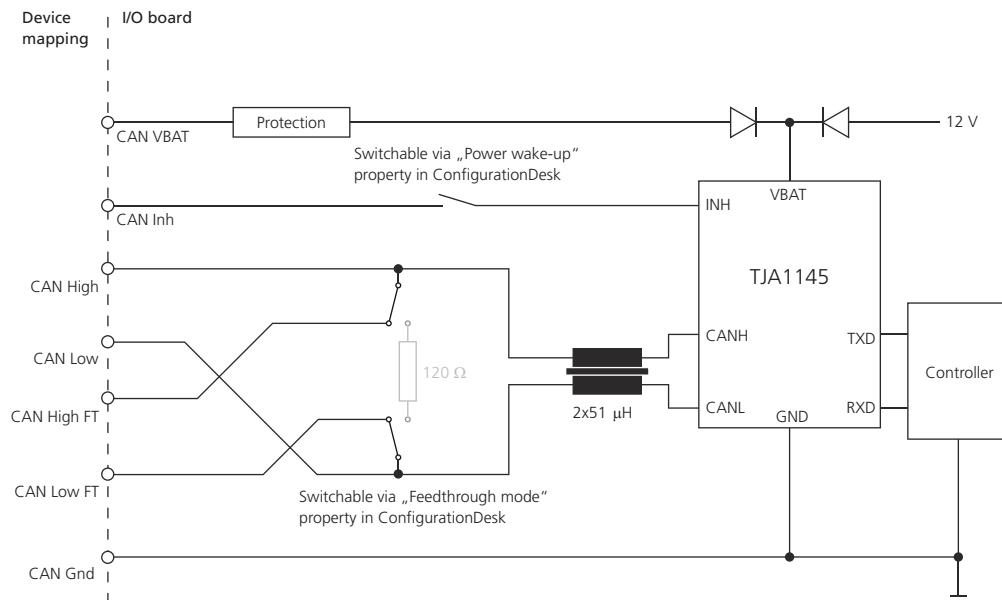


CAN 5

ISO 11898-2 High-Speed CAN without feedthrough

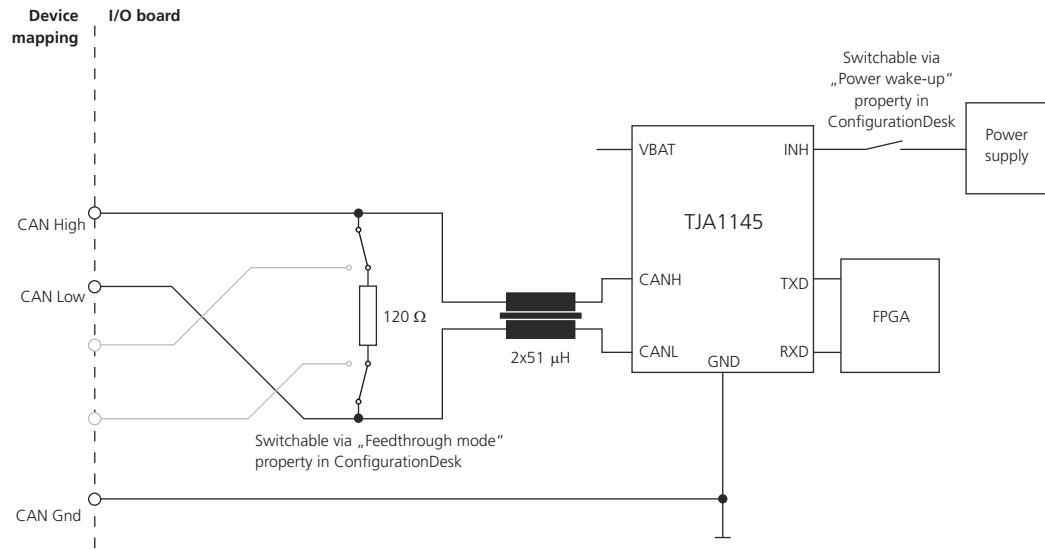


ISO 11898-2 High-Speed CAN with feedthrough

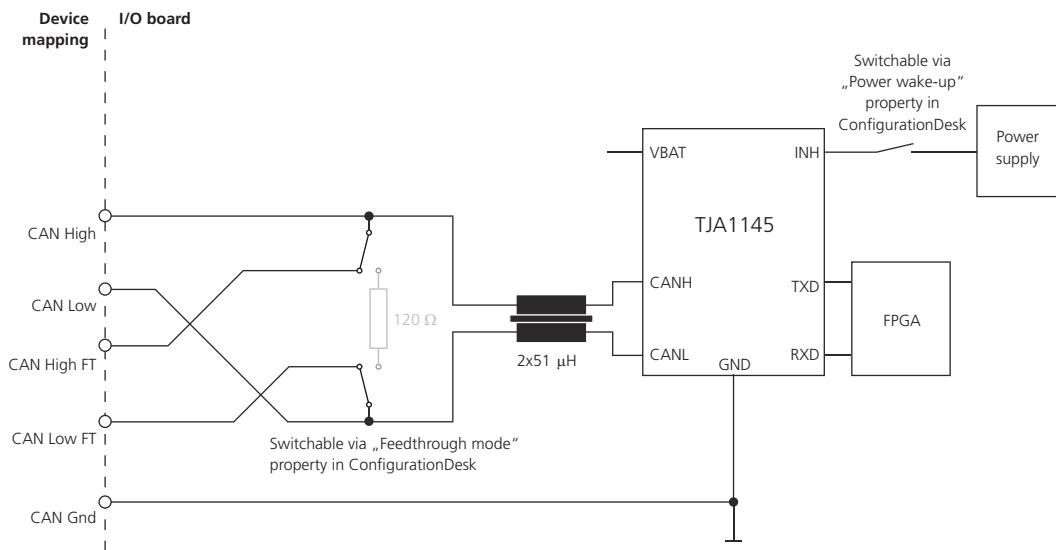


CAN 6

ISO 11898-2 High-Speed CAN without feedthrough

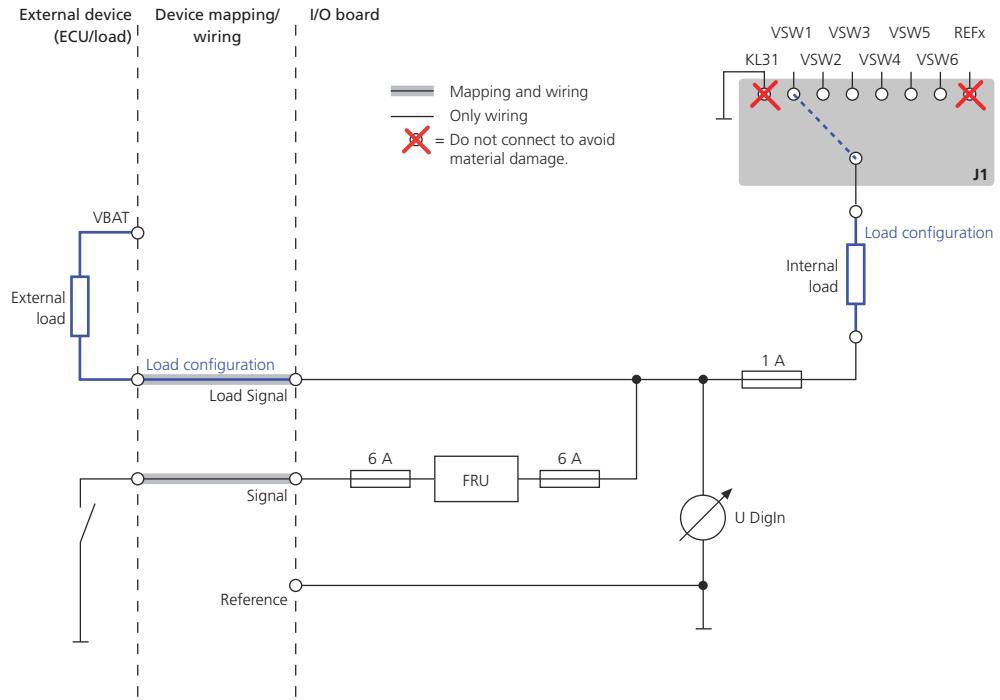


ISO 11898-2 High-Speed CAN with feedthrough



Digital In 1

Digital voltage measurement of low-side controlled load

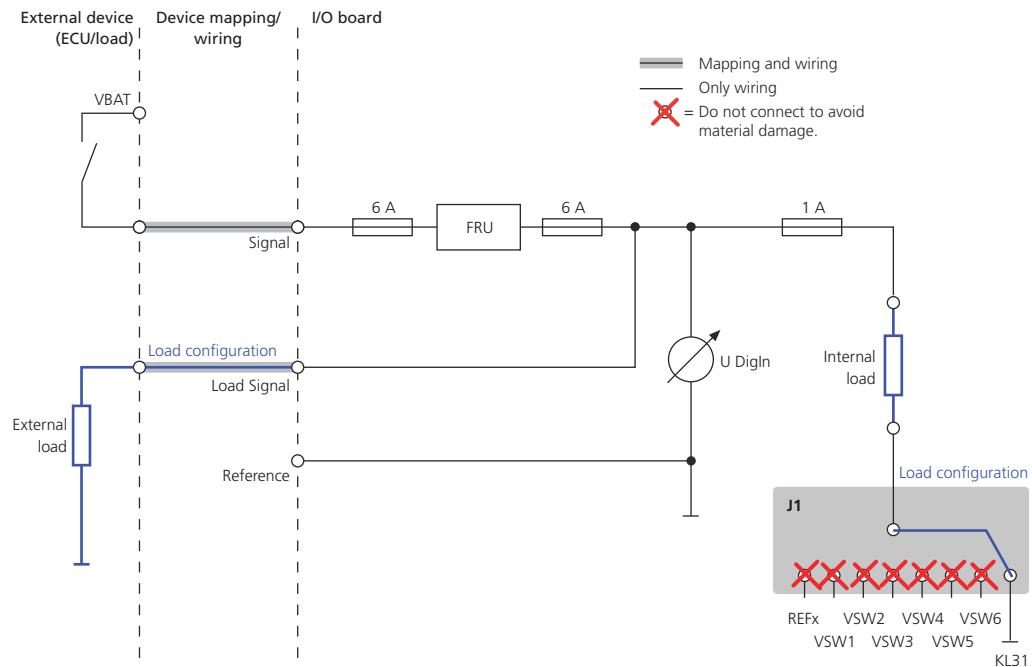


Load configuration:

Use external or internal load.

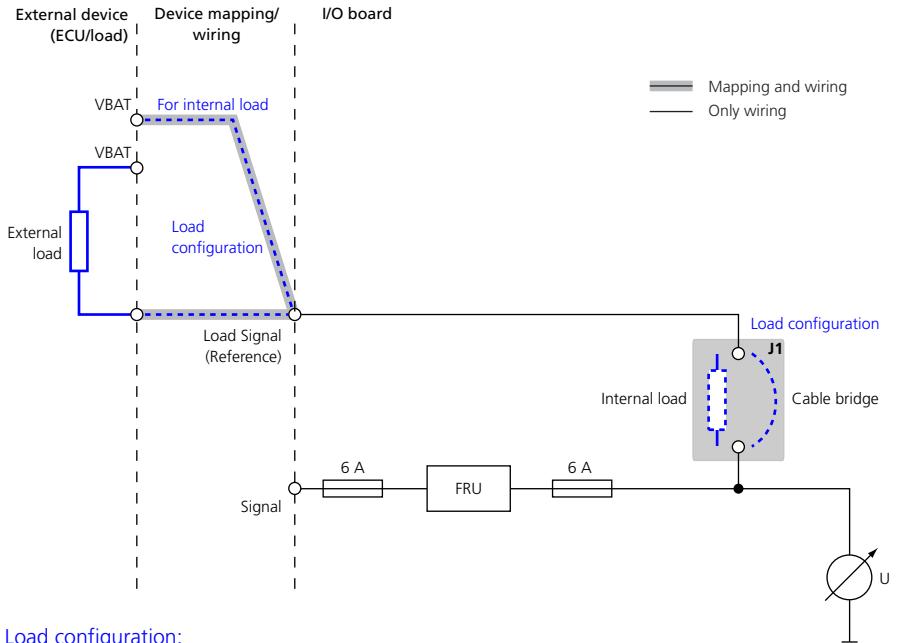
If you use an internal load, define the reference voltage with J1.

Digital voltage measurement of high-side controlled load



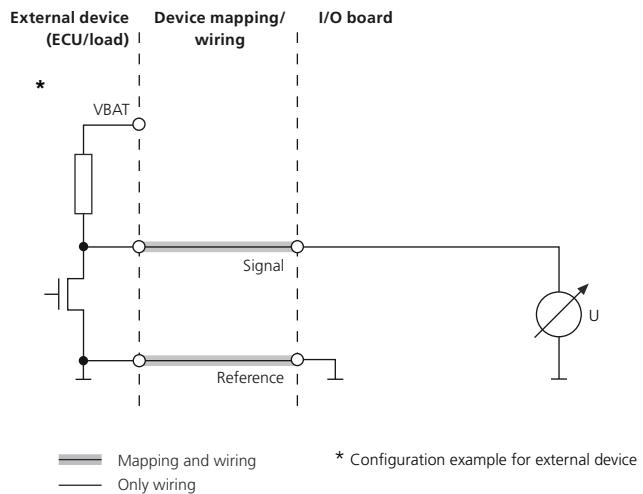
Digital In 2

Voltage measurement



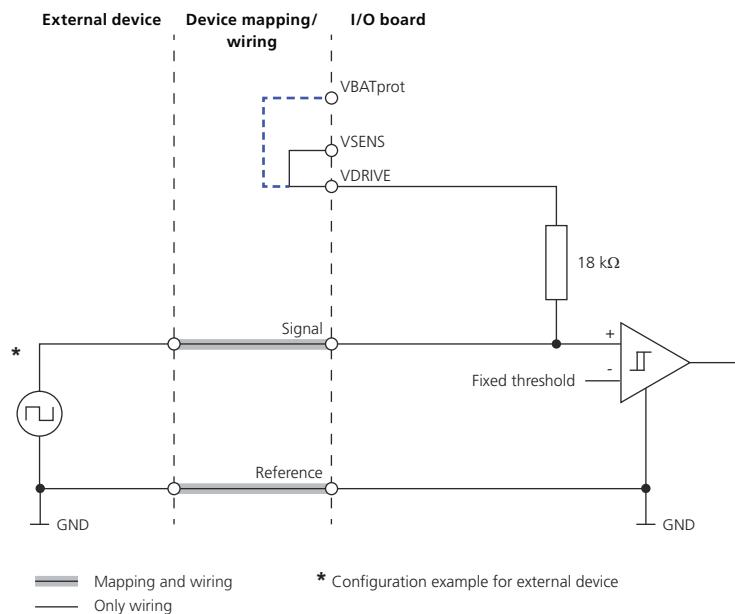
Digital In 3

Voltage measurement



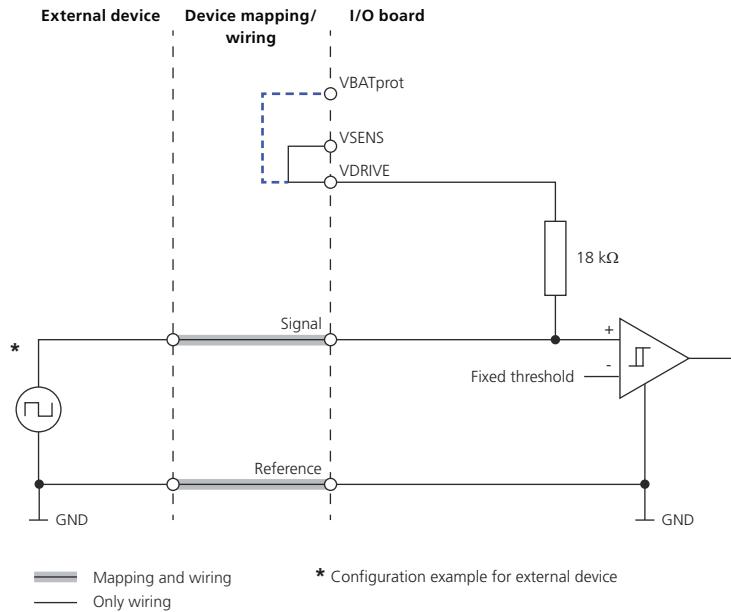
Digital In 4

Digital voltage measurement



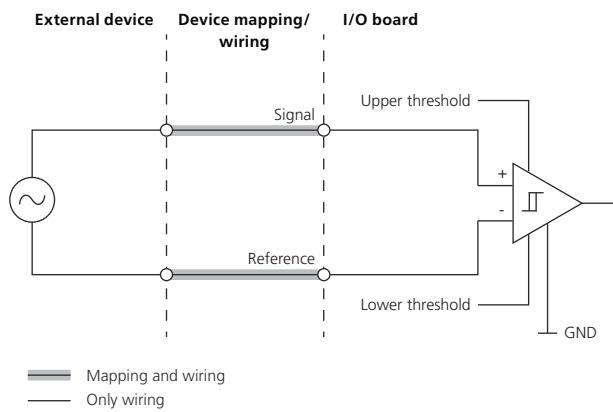
Digital In 5

Digital voltage measurement



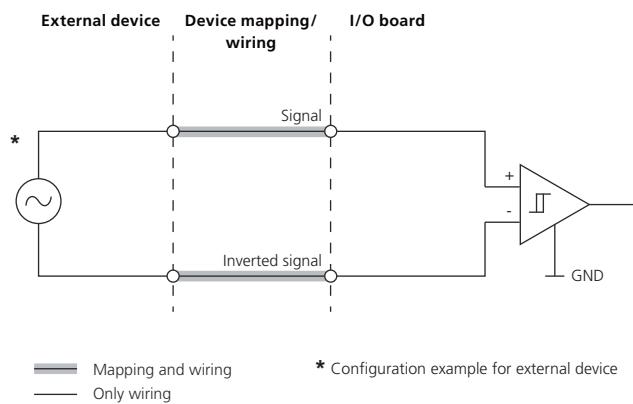
Digital In 9

Digital voltage measurement



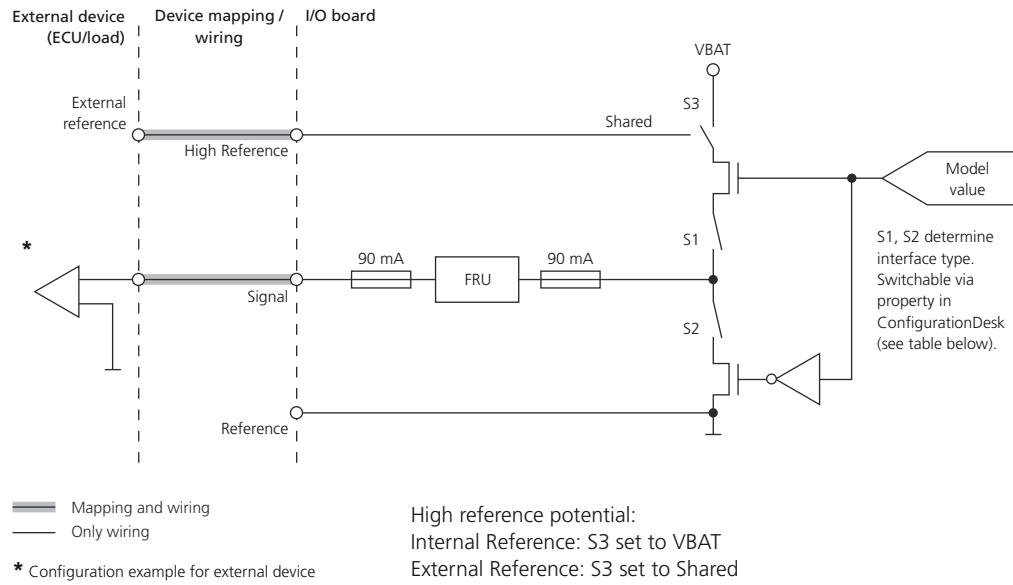
Digital In 10

Zero voltage detection



Digital Out 1

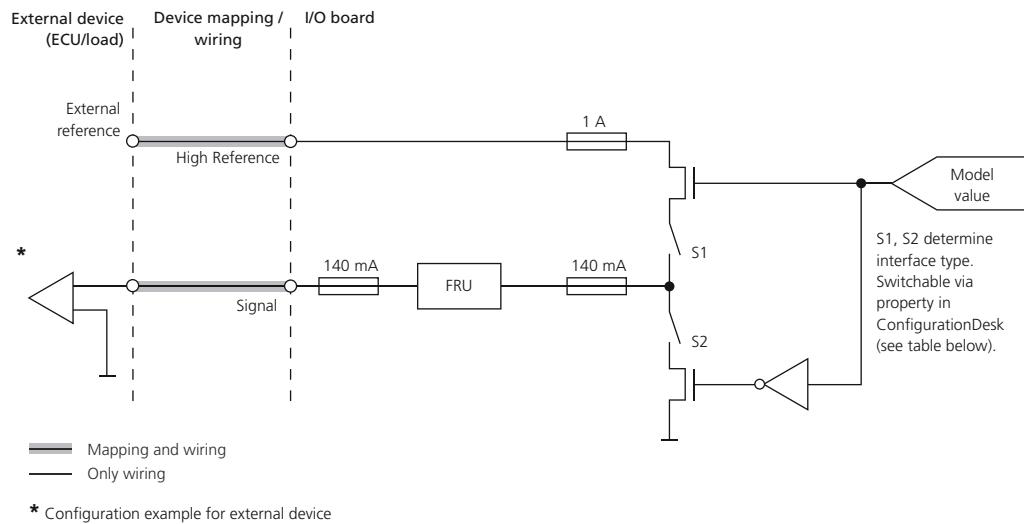
Generating digital output signals



Property Setting of...		Available Signal Ports	Switches
Interface Type	High Reference Potential		
High-side switch	Shared	■ High Reference ■ Signal	S1 closed S2 open
	VBat	■ Signal	
Low-side switch	-	■ Signal ■ Reference	S1 open, S2 closed
	Shared	■ High Reference ■ Signal ■ Reference	
Push-pull	VBat	■ Signal ■ Reference	S1 closed S2 closed

Digital Out 2

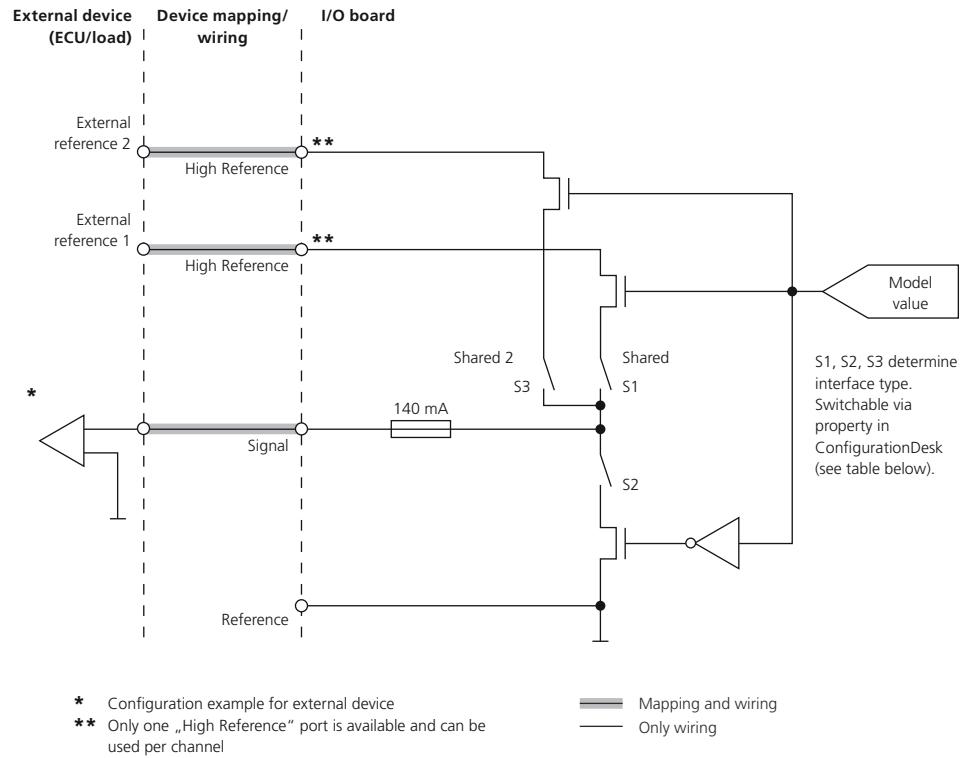
Generating digital output signals



Property Setting of...		Available Signal Ports	Switches
Interface Type	High Reference Potential		
High-side switch	<ul style="list-style-type: none"> ■ Individual ■ VBat ■ Shared ■ Shared 2 	<ul style="list-style-type: none"> ■ High Reference ■ Signal 	S1 closed S2 open
Low-side switch	-	<ul style="list-style-type: none"> ■ Signal 	S1 open, S2 closed
Push-pull	<ul style="list-style-type: none"> ■ Individual ■ VBat ■ Shared ■ Shared 2 	<ul style="list-style-type: none"> ■ High Reference ■ Signal 	S1 closed S2 closed

Digital Out 3

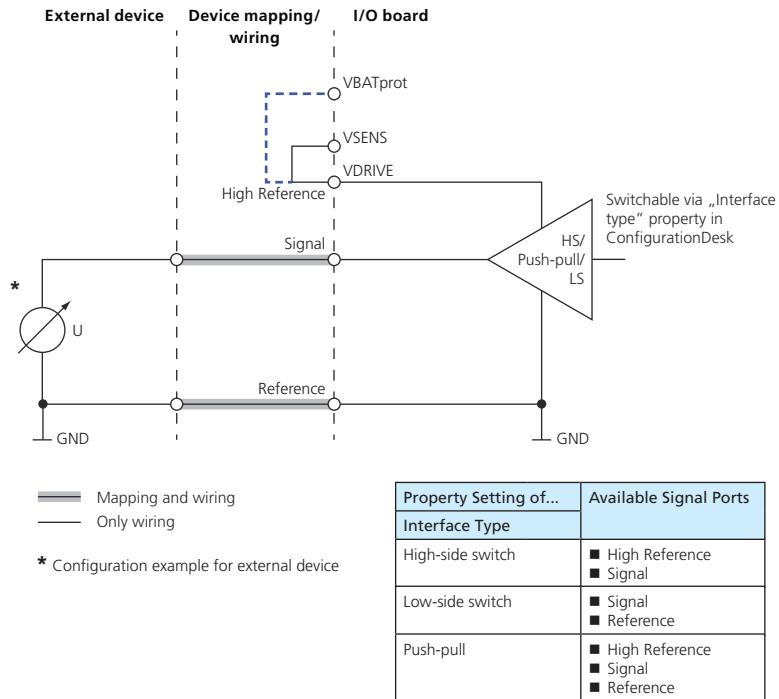
Generating digital output signals



Property Setting of...		Available Signal Ports	Switches
Interface Type	High Reference Potential		
High-side switch	■ VBat ■ Shared	■ High Reference ■ Signal	S1 closed, S2 open, S3 open
	■ Shared 2		S1 open, S2 open, S3 closed
Low-side switch	-	■ Signal ■ Reference	S1 open, S2 closed, S3 open
Push-pull	■ VBat ■ Shared	■ High Reference ■ Signal ■ Reference	S1 closed, S2 closed, S3 open
	■ Shared 2		S1 open, S2 closed, S3 closed

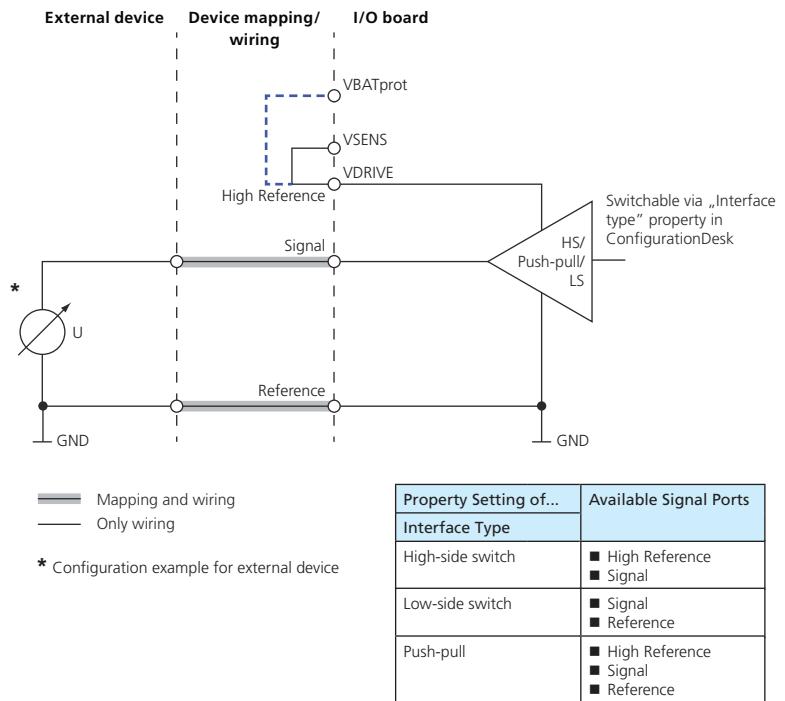
Digital Out 4

Generating digital output signals



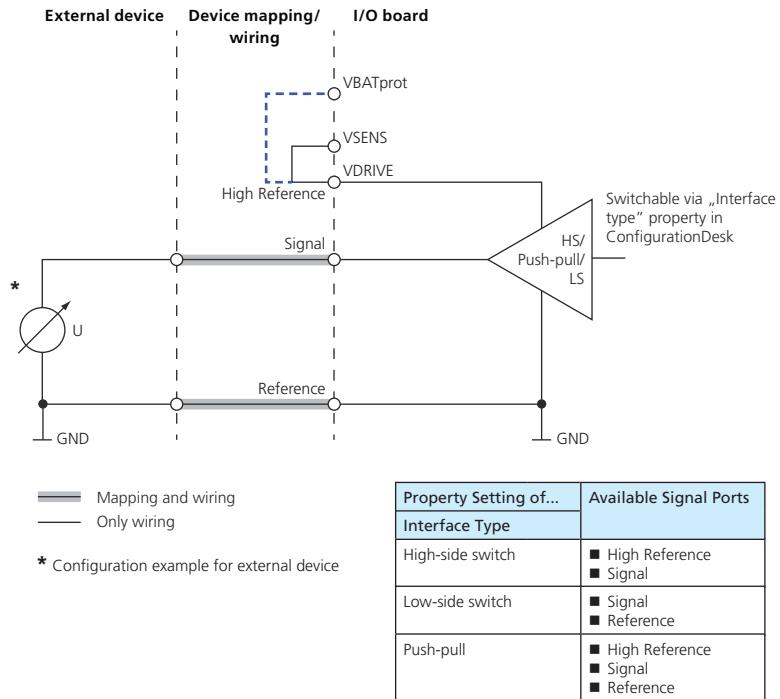
Digital Out 5

Generating digital output signals



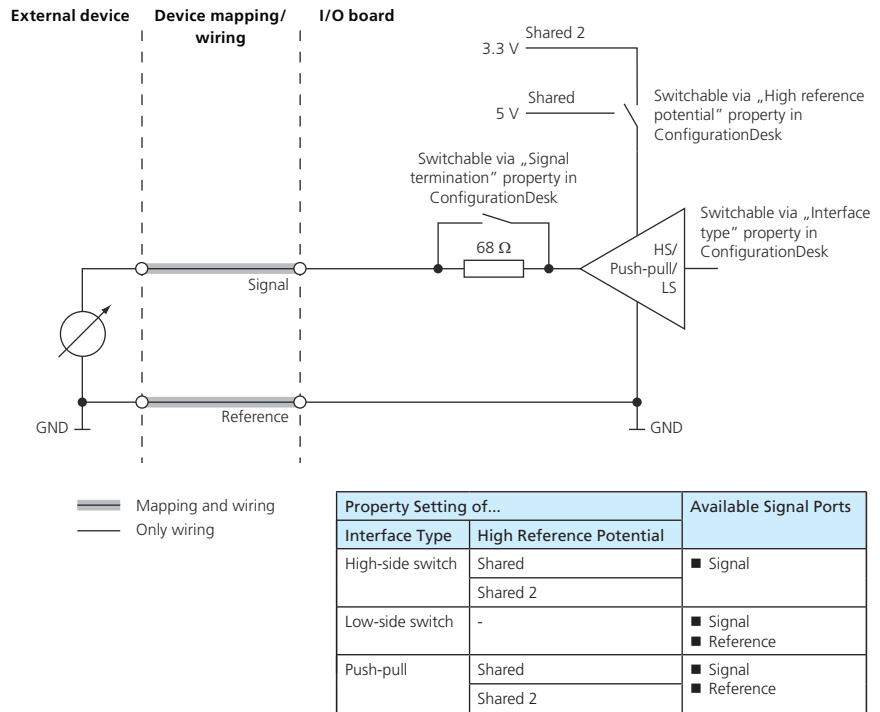
Digital Out 7

Generating digital output signals



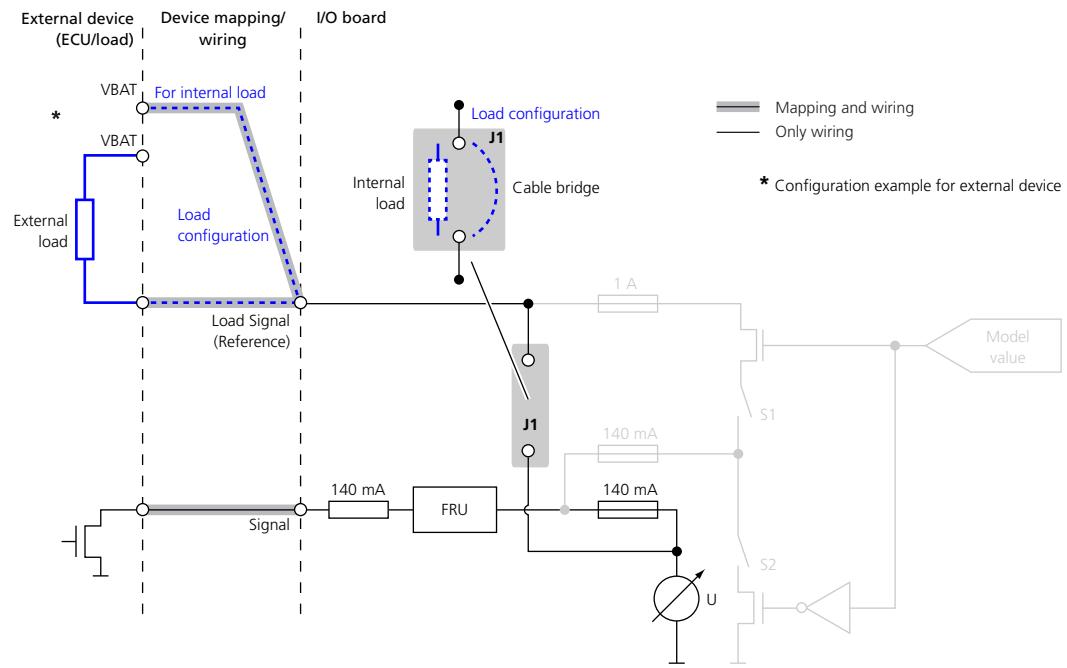
Digital Out 8

Generating digital output signals



Digital In/Out 1

Digital voltage measurement



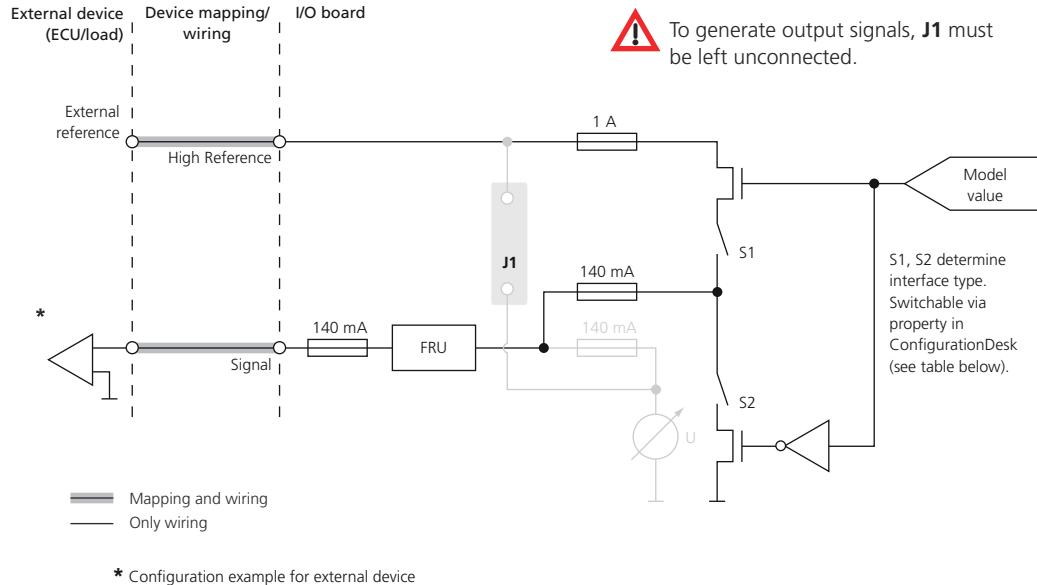
Load configuration:

Use external or internal load.

If you use an external load, use a cable bridge for J1.

If you do not use an external load, connect the Load Signal (Reference) port directly to an external reference, for example, VBAT.

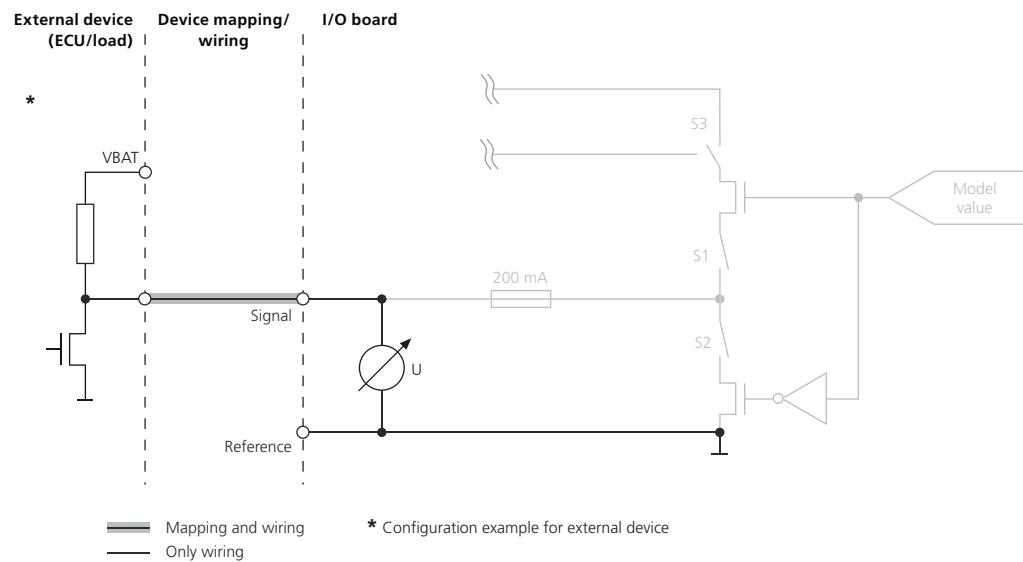
Generating digital output signals



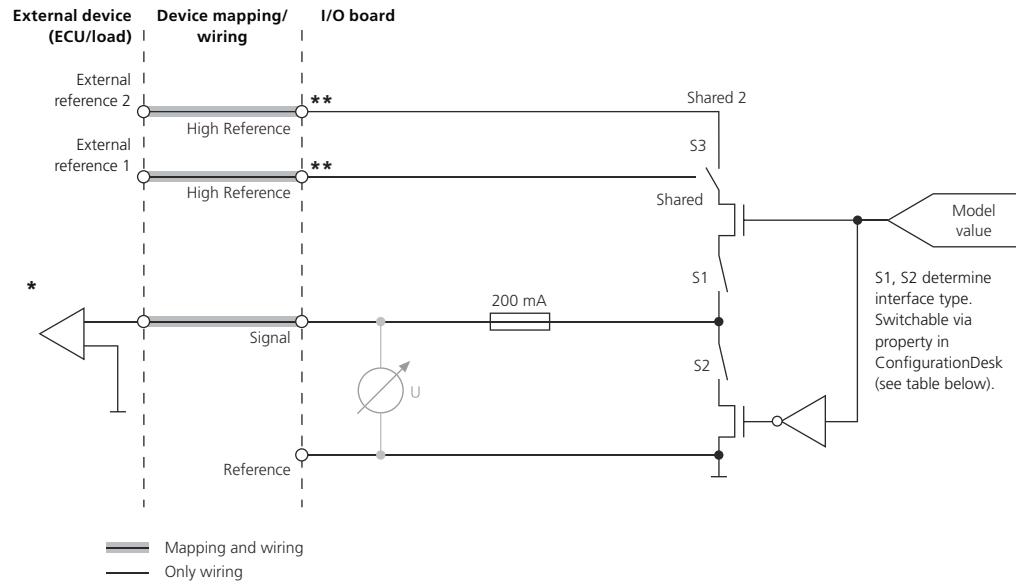
Property Setting of...		Available Signal Ports	Switches
Interface Type	High Reference Potential		
High-side switch	<ul style="list-style-type: none"> ■ Individual ■ VBat ■ Shared ■ Shared 2 	<ul style="list-style-type: none"> ■ High Reference ■ Signal 	S1 closed S2 open
Low-side switch	-	<ul style="list-style-type: none"> ■ Signal 	S1 open, S2 closed
Push-pull	<ul style="list-style-type: none"> ■ Individual ■ VBat ■ Shared ■ Shared 2 	<ul style="list-style-type: none"> ■ High Reference ■ Signal 	S1 closed S2 closed

Digital In/Out 3

Digital voltage measurement



Generating digital output signals



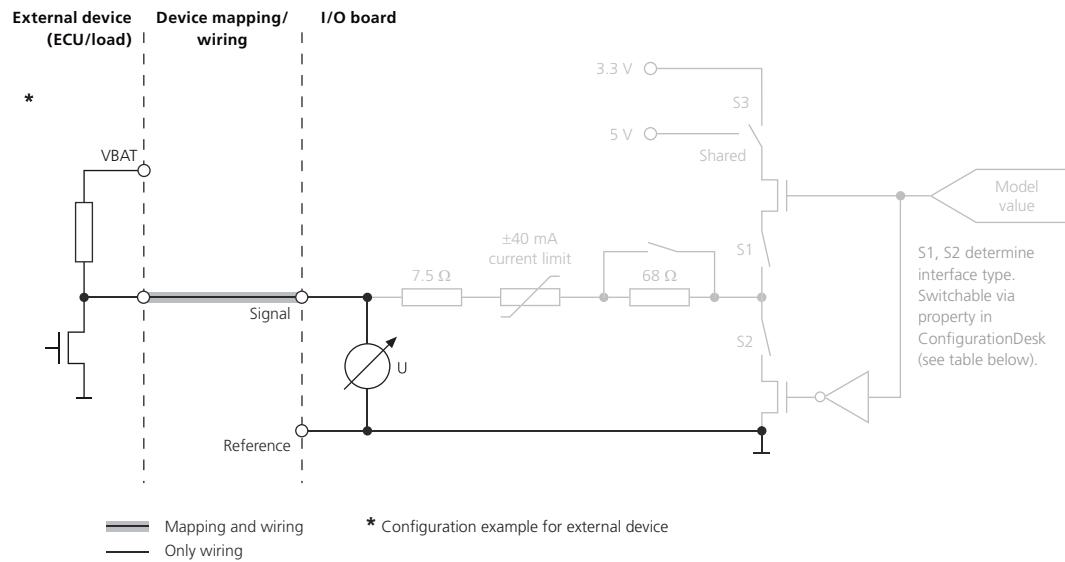
* Configuration example for external device
 ** Only one „High Reference“ port is available and can be used per channel

High reference potential:
 External Reference 1: S3 set to Shared
 External Reference 2: S3 set to Shared 2

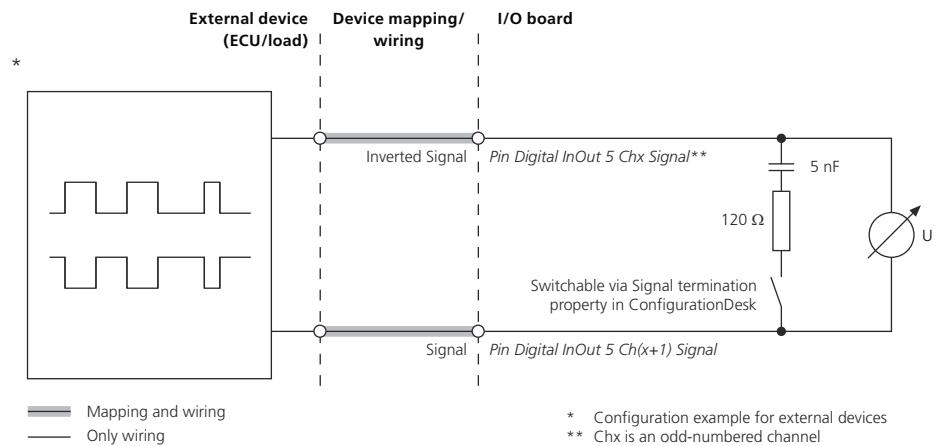
Property Setting of...		Available Signal Ports	Switches
Interface Type	High Reference Potential		
High-side switch	■ VBat ■ Shared	■ High Reference ■ Signal	S1 closed S2 open
	■ Shared 2		S1 closed S2 open
Low-side switch	-	■ Signal ■ Reference	S1 open S2 closed
	■ VBat ■ Shared		S1 closed S2 closed
Push-pull	■ Shared 2	■ High Reference ■ Signal ■ Reference	S1 closed S2 closed
	-		S1 closed S2 closed

Digital In/Out 5

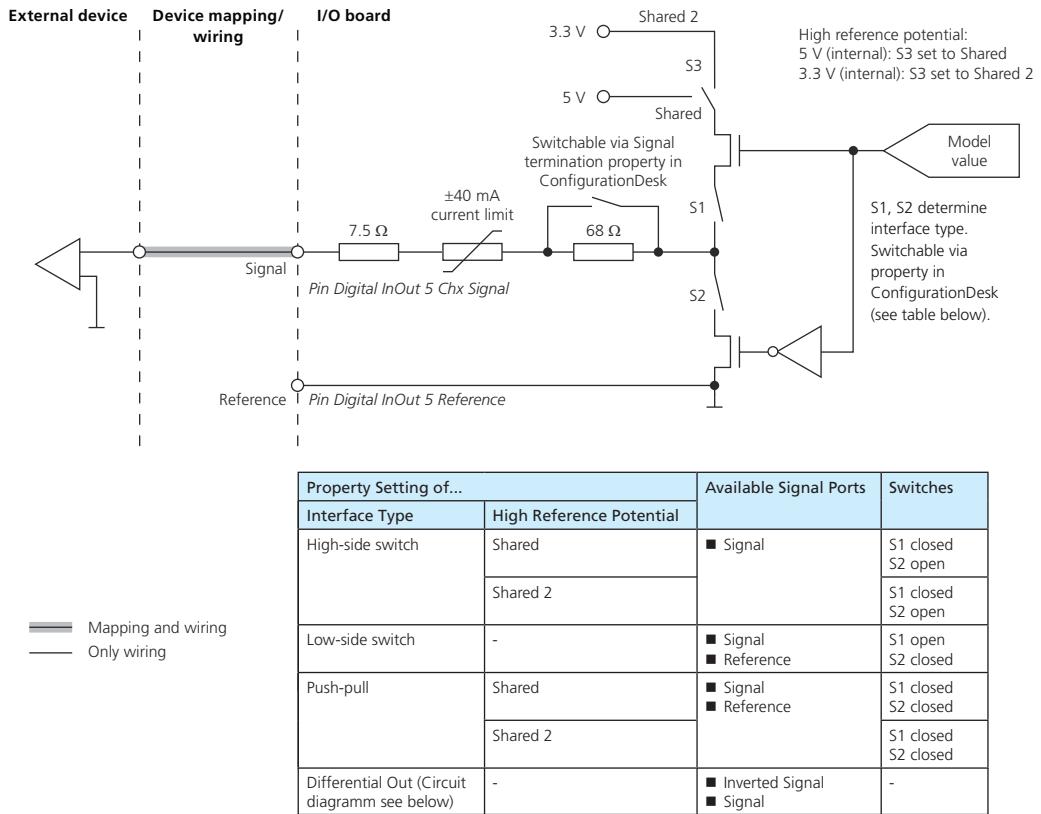
Digital voltage measurement



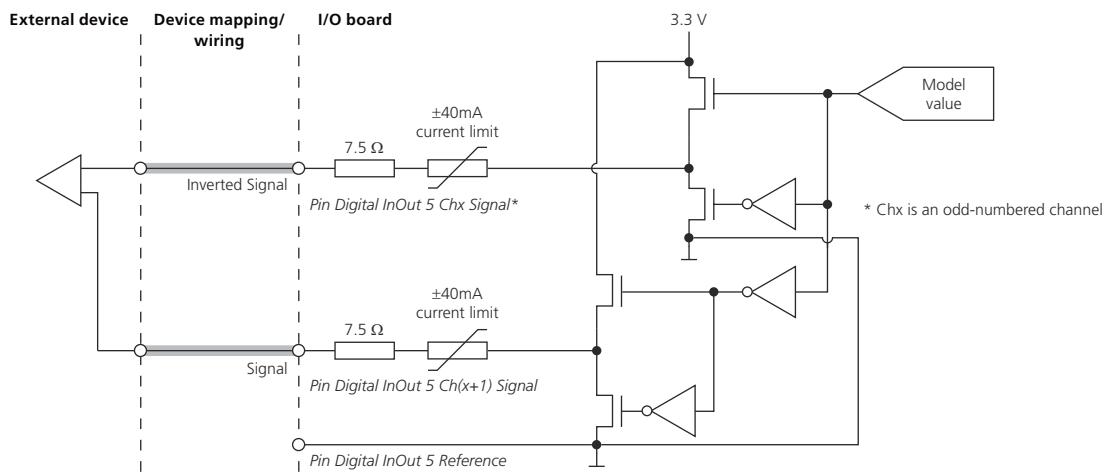
Differential digital voltage measurement



Generating digital output signals

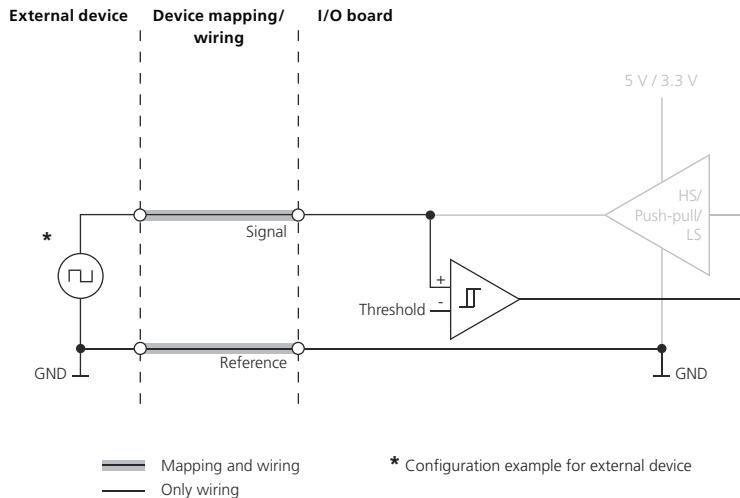


Differential Out

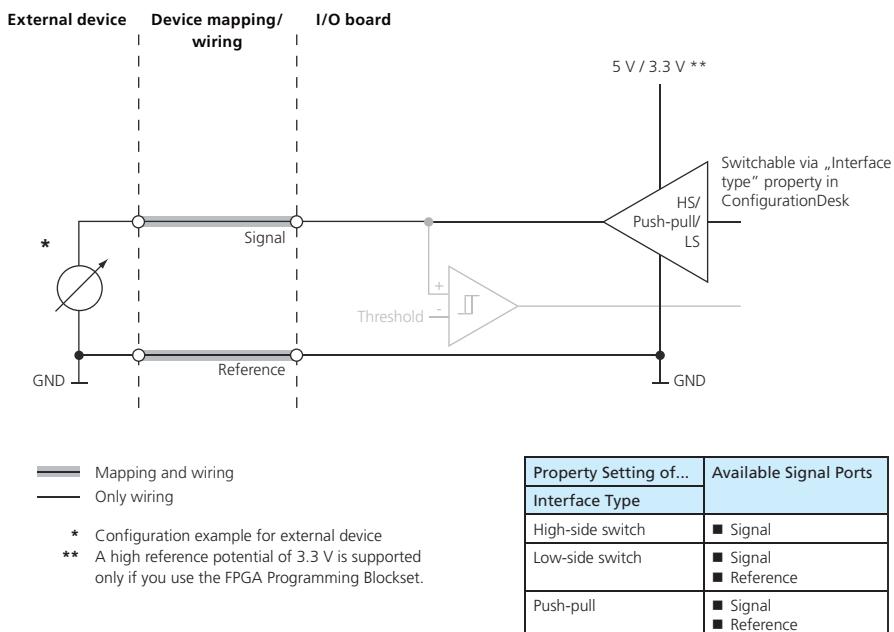


Digital In/Out 6

Digital voltage measurement

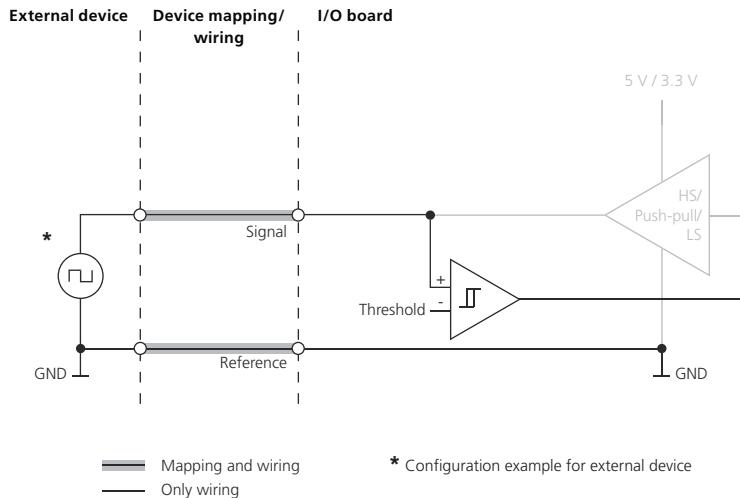


Generating digital output signals

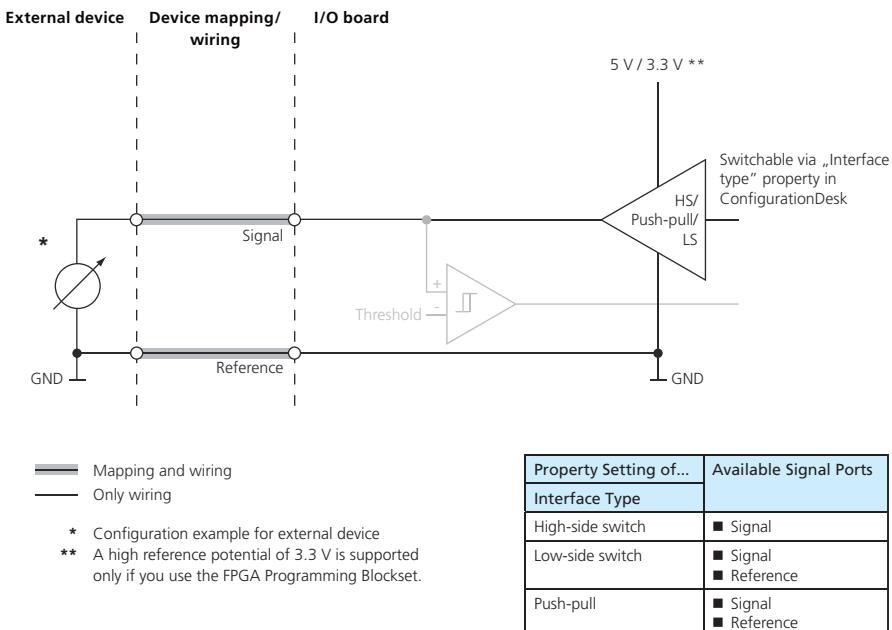


Digital In/Out 8

Digital voltage measurement

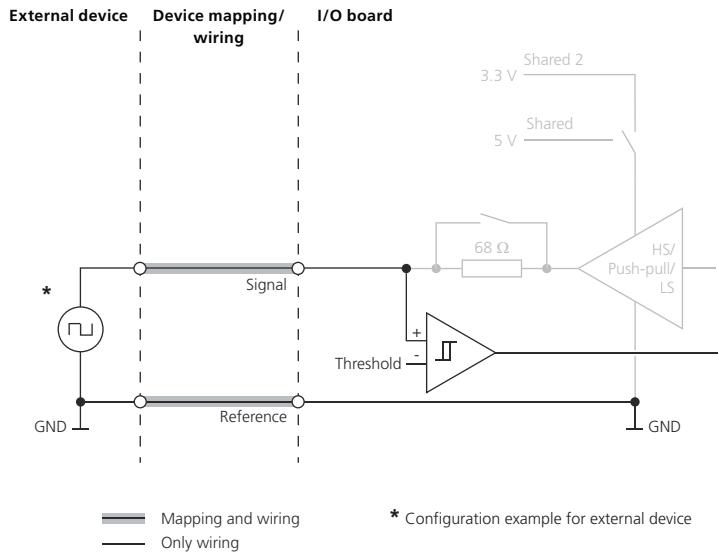


Generating digital output signals

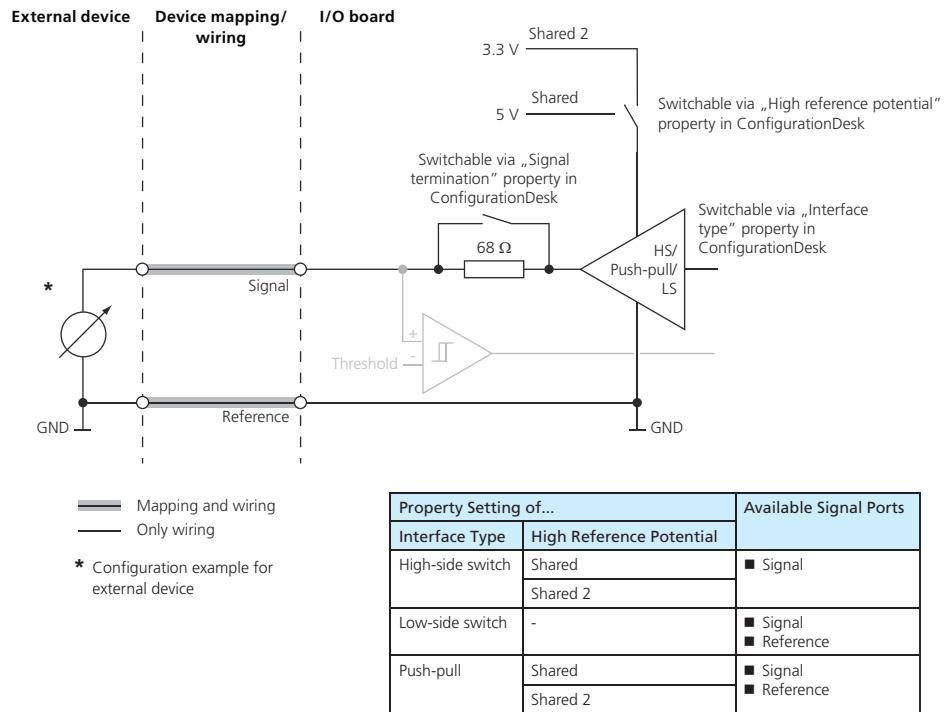


Digital In/Out 9

Digital voltage measurement

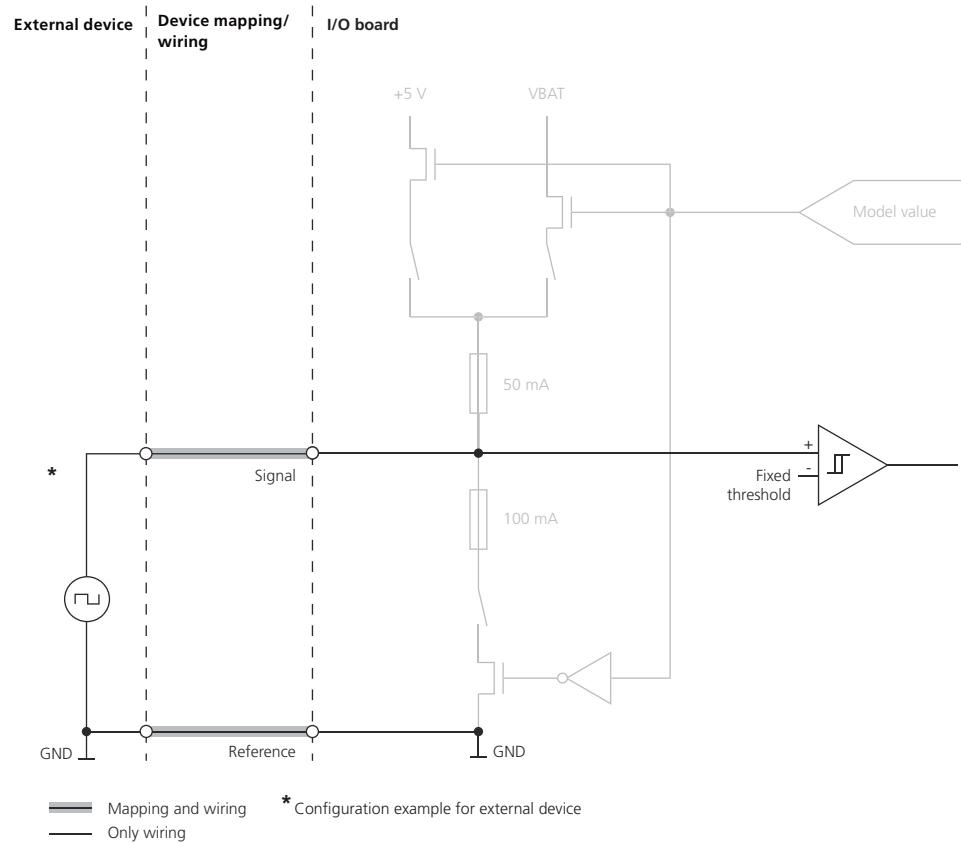


Generating digital output signals

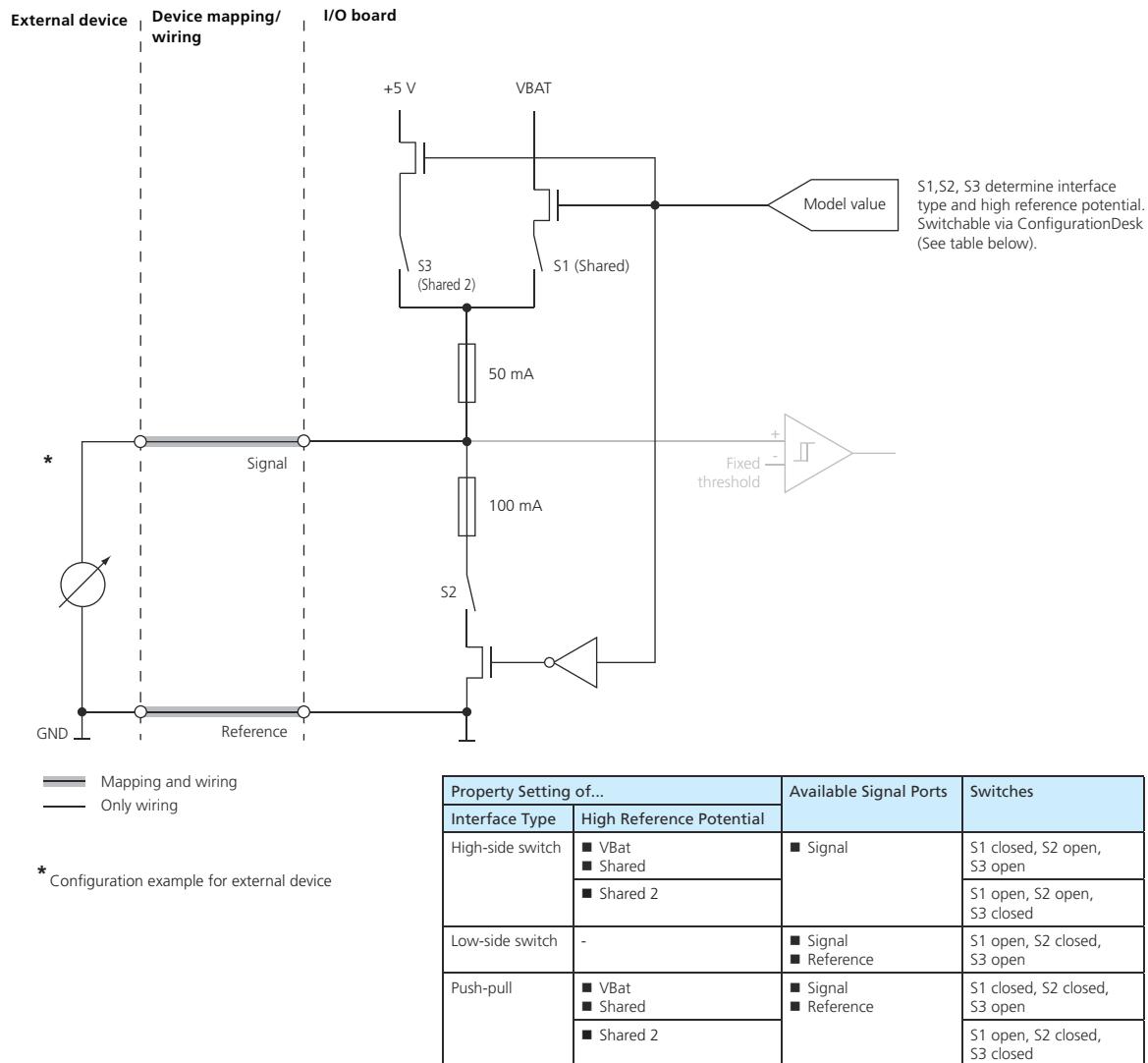


Digital In/Out 10

Digital voltage measurement

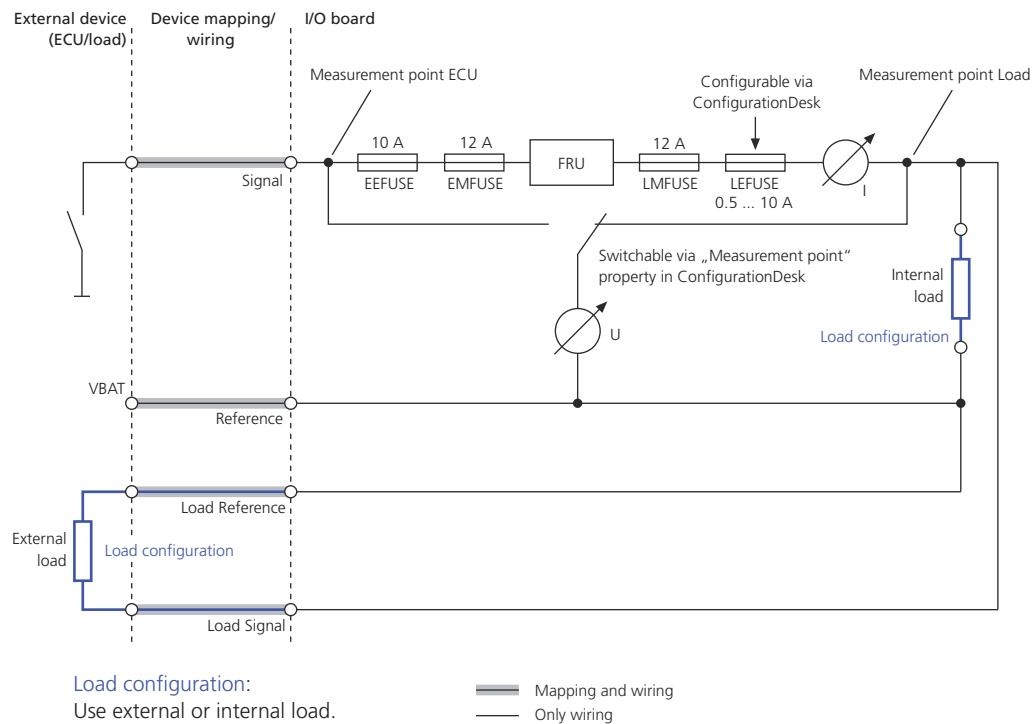


Generating digital output signals

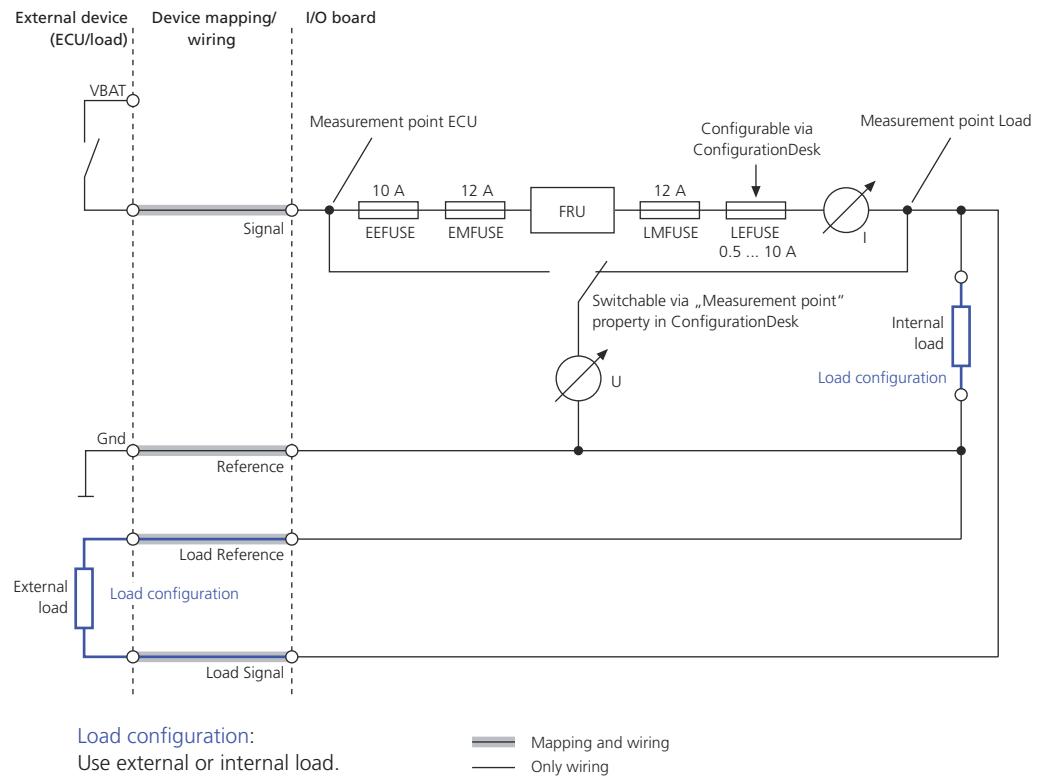


Flexible In 1

Current and voltage measurement of low-side controlled load

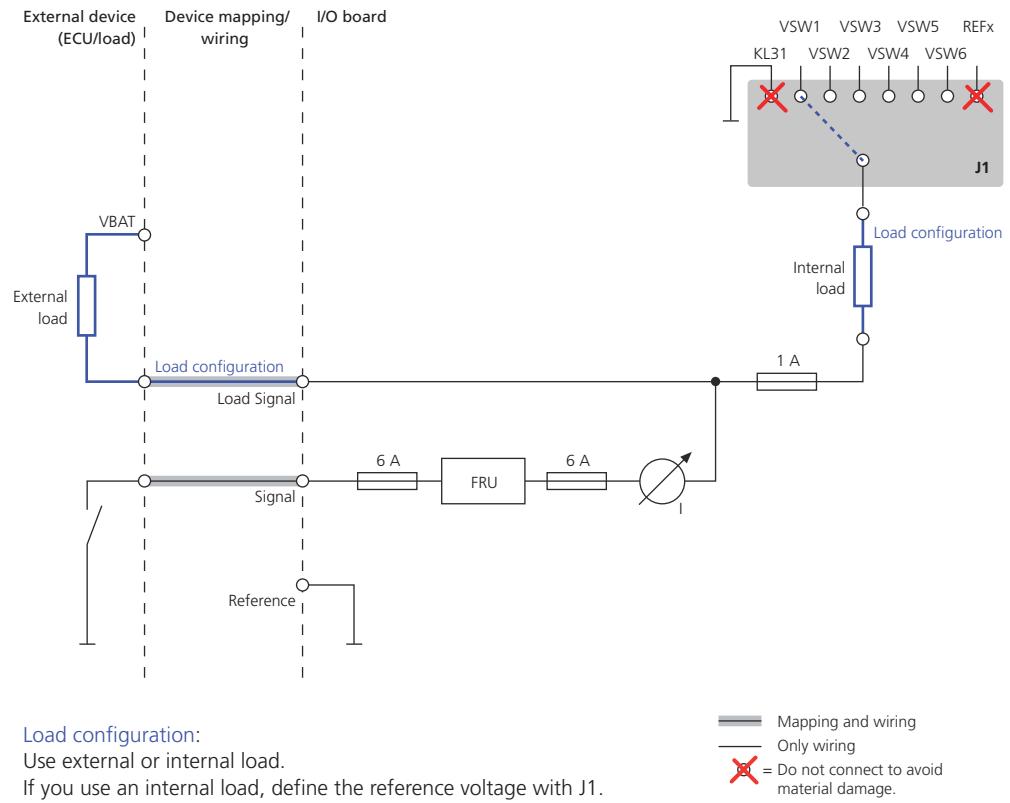


Current and voltage measurement of high-side controlled load

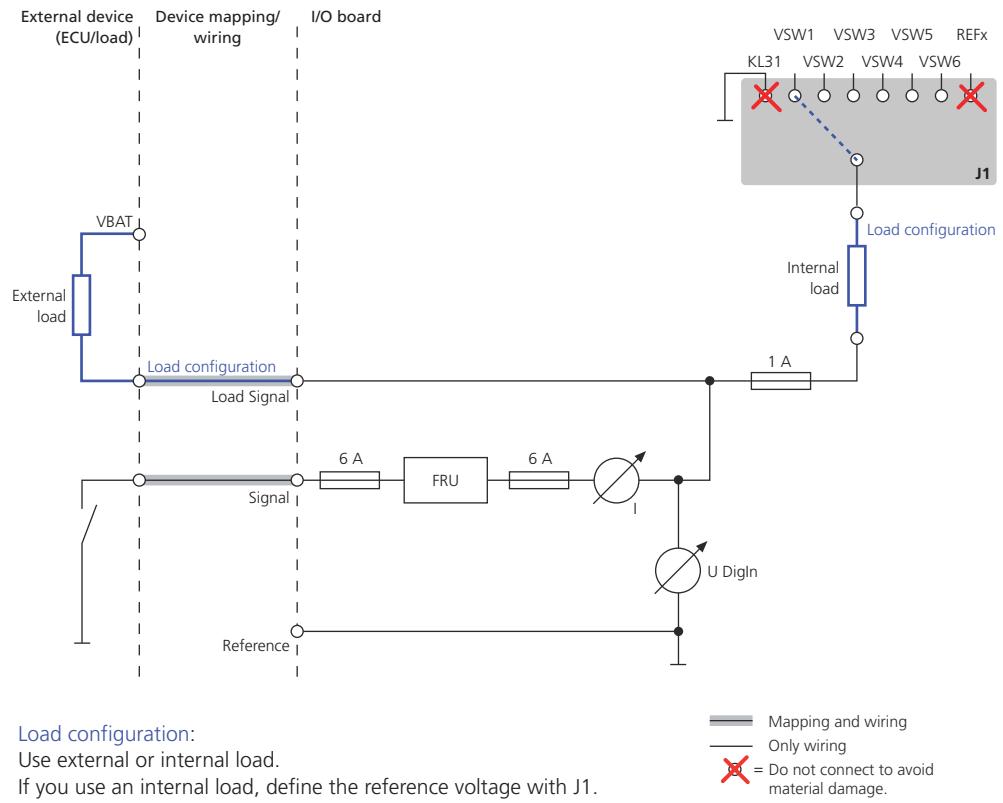


Flexible In 2

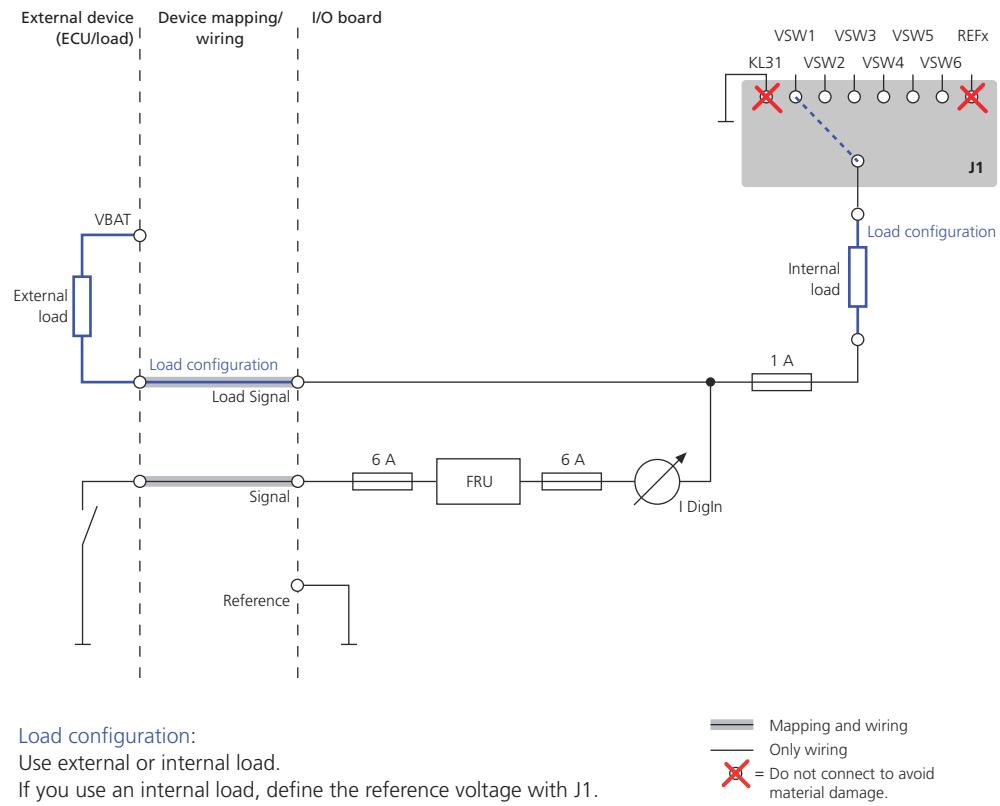
Analog current measurement of low-side controlled load



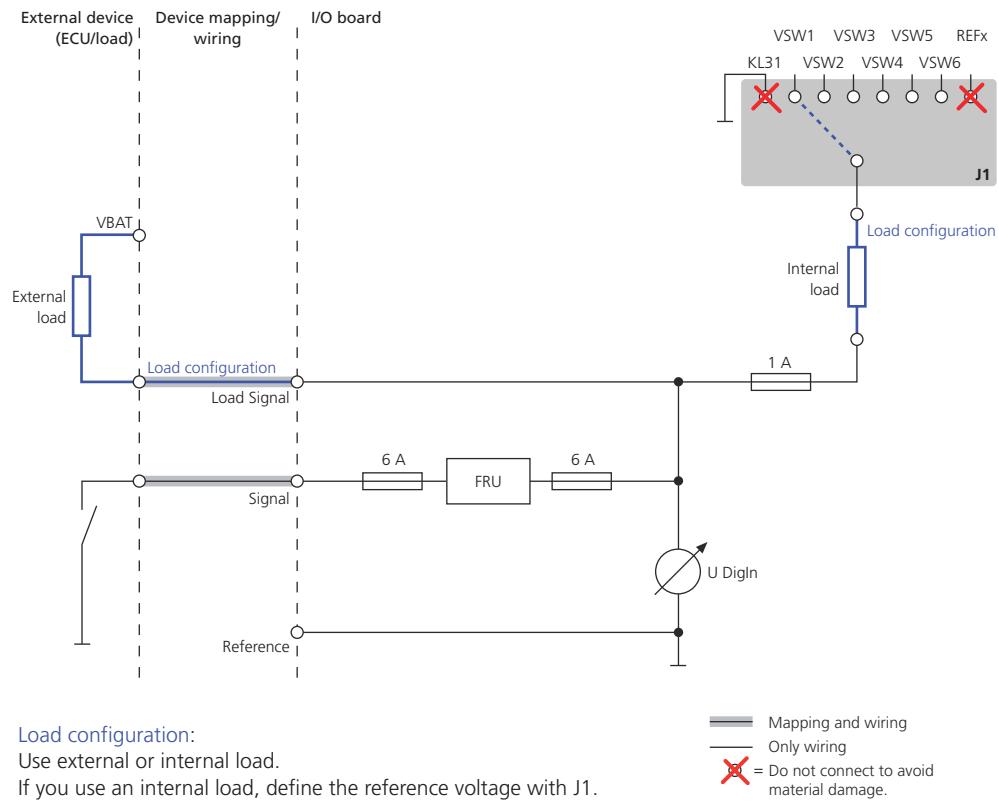
Analog current and digital voltage measurement of low-side controlled load



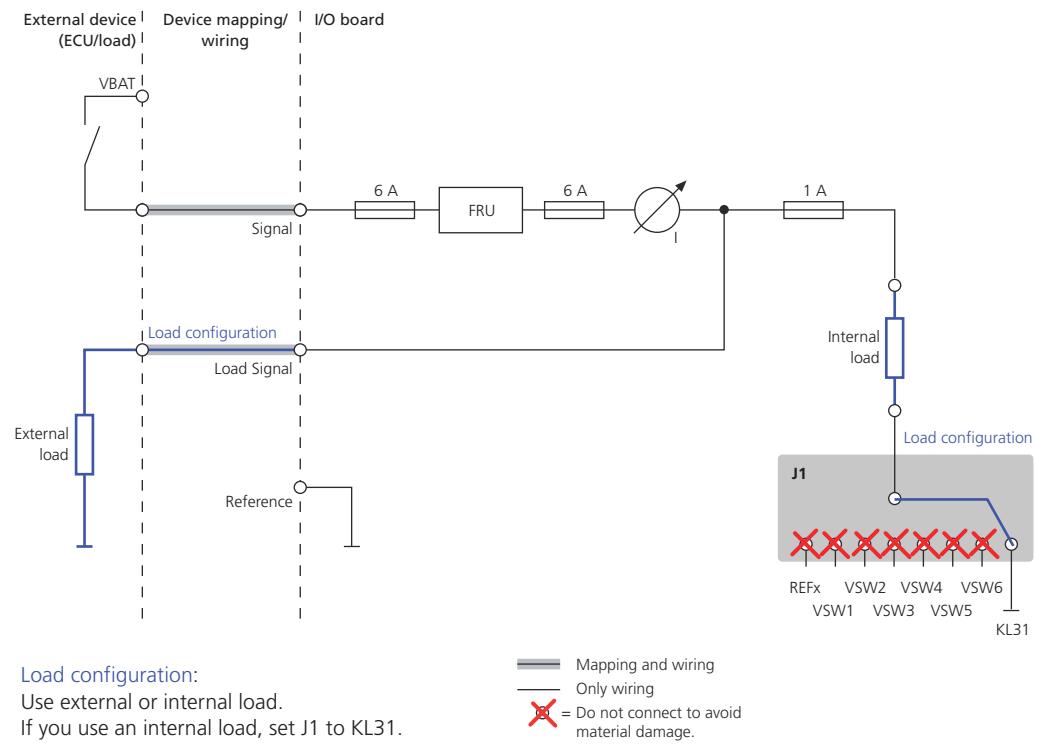
Digital current measurement of low-side controlled load



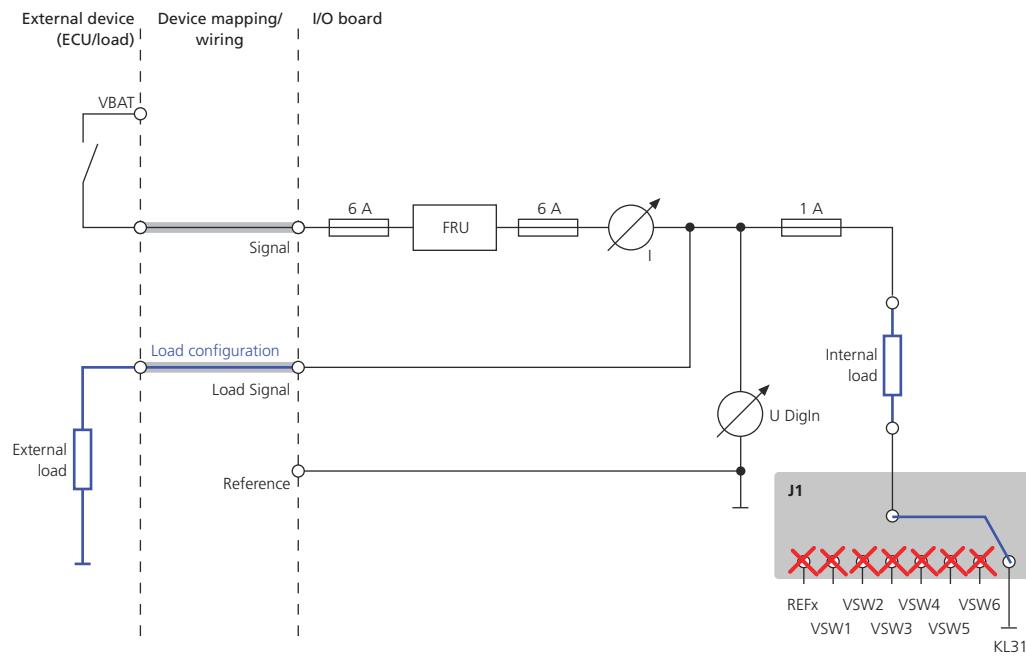
Digital voltage measurement of low-side controlled load



Analog current measurement of high-side controlled load



Analog current and digital voltage measurement of high-side controlled load

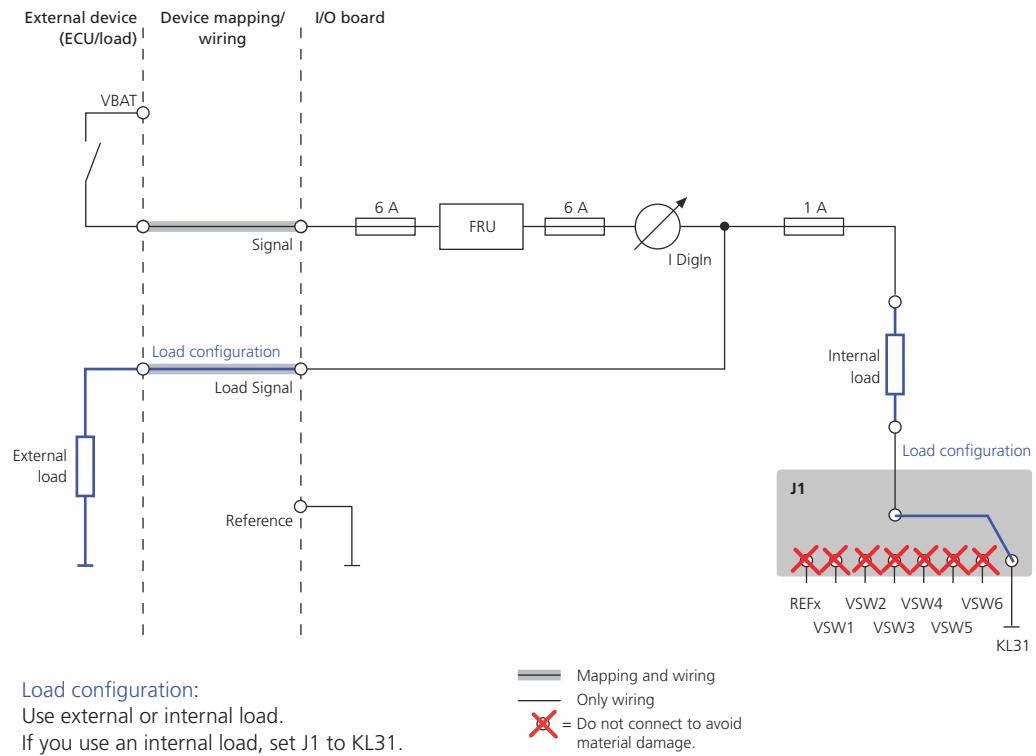


Load configuration:

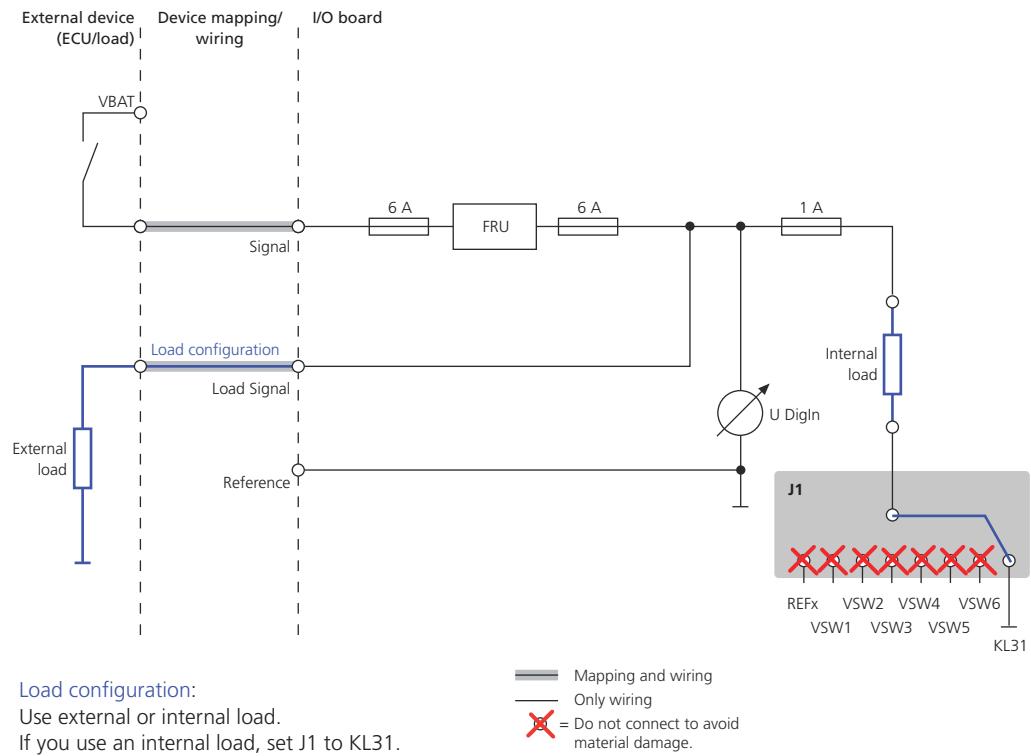
Use external or internal load.
If you use an internal load, set J1 to KL31.

- Mapping and wiring
- Only wiring
- ~~—~~ = Do not connect to avoid material damage.

Digital current measurement of high-side controlled load

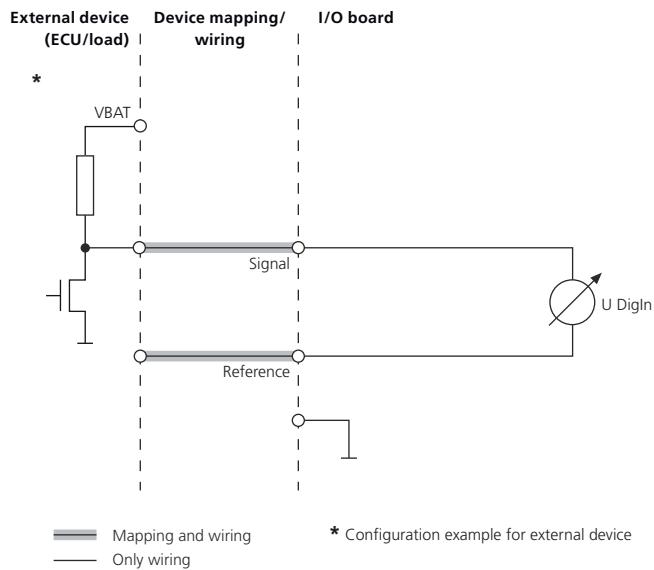


Digital voltage measurement of high-side controlled load

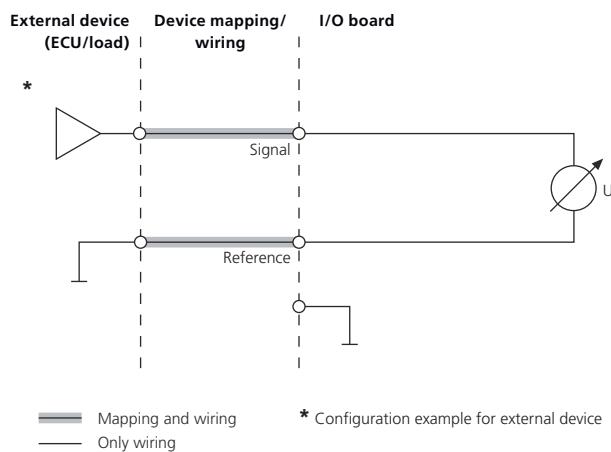


Flexible In 3

Digital voltage measurement

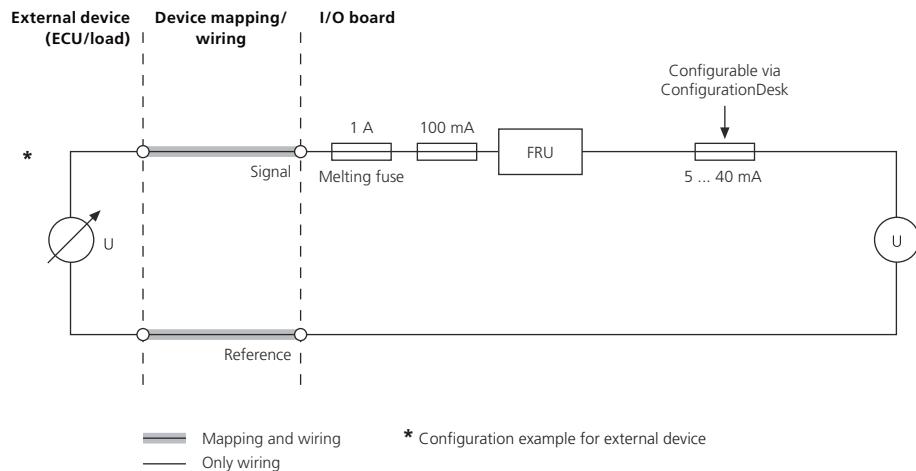


Analog voltage measurement

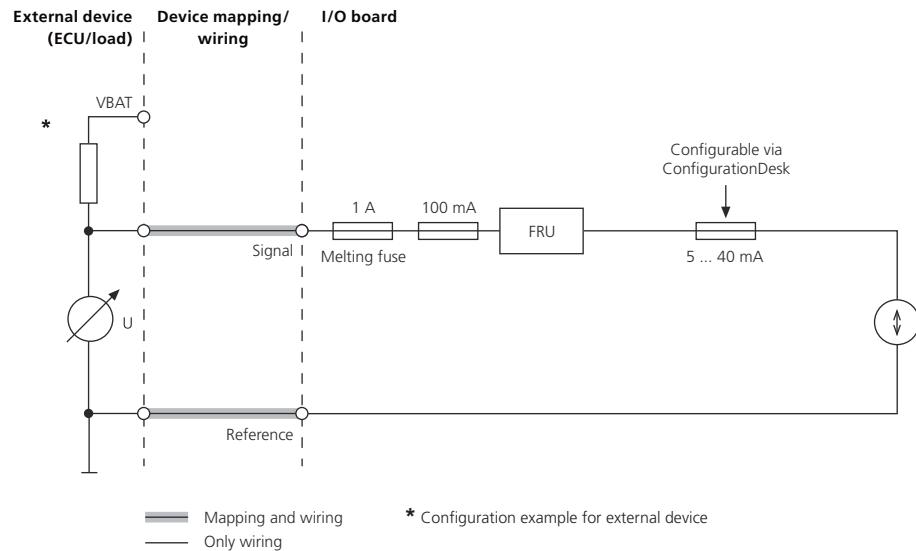


Flexible Out 1

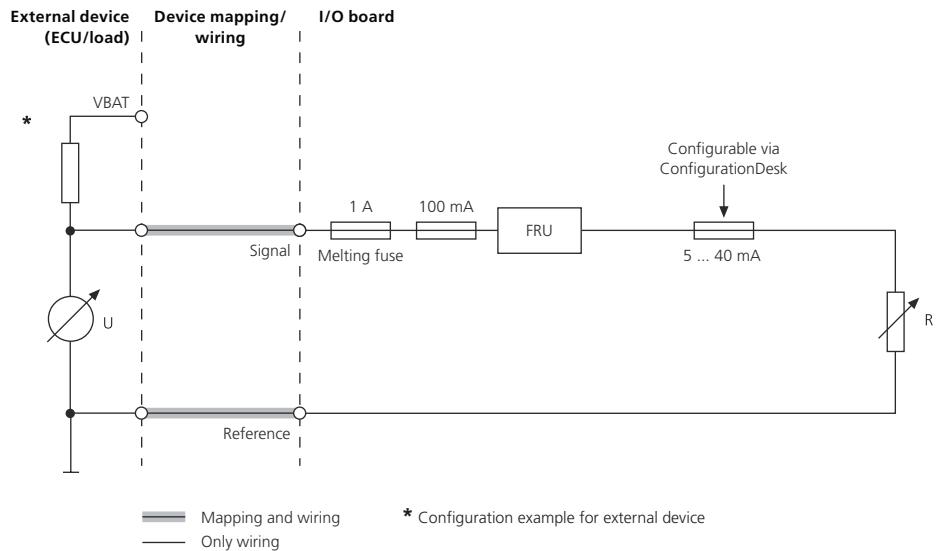
Voltage generation



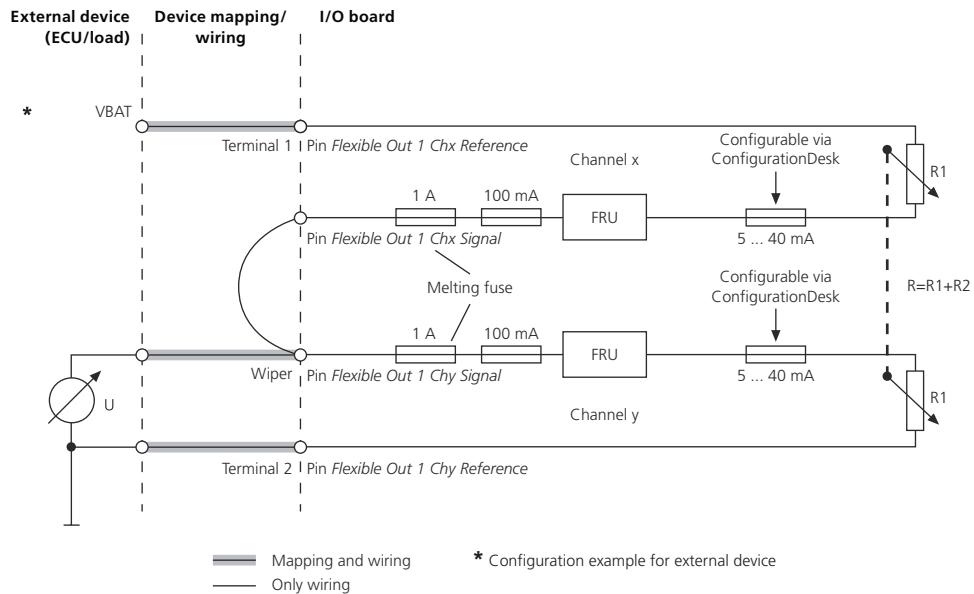
Current Sink



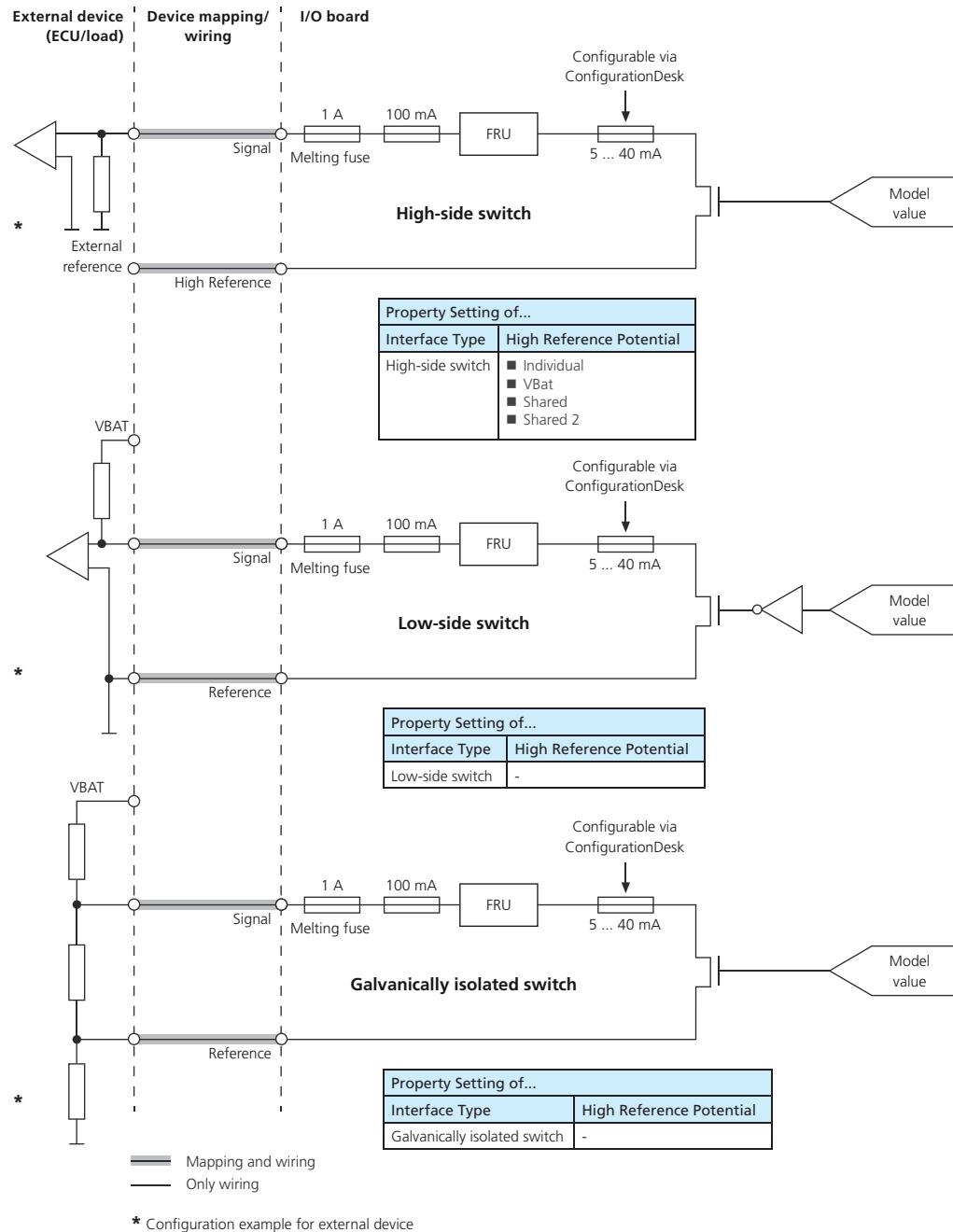
Resistor simulation



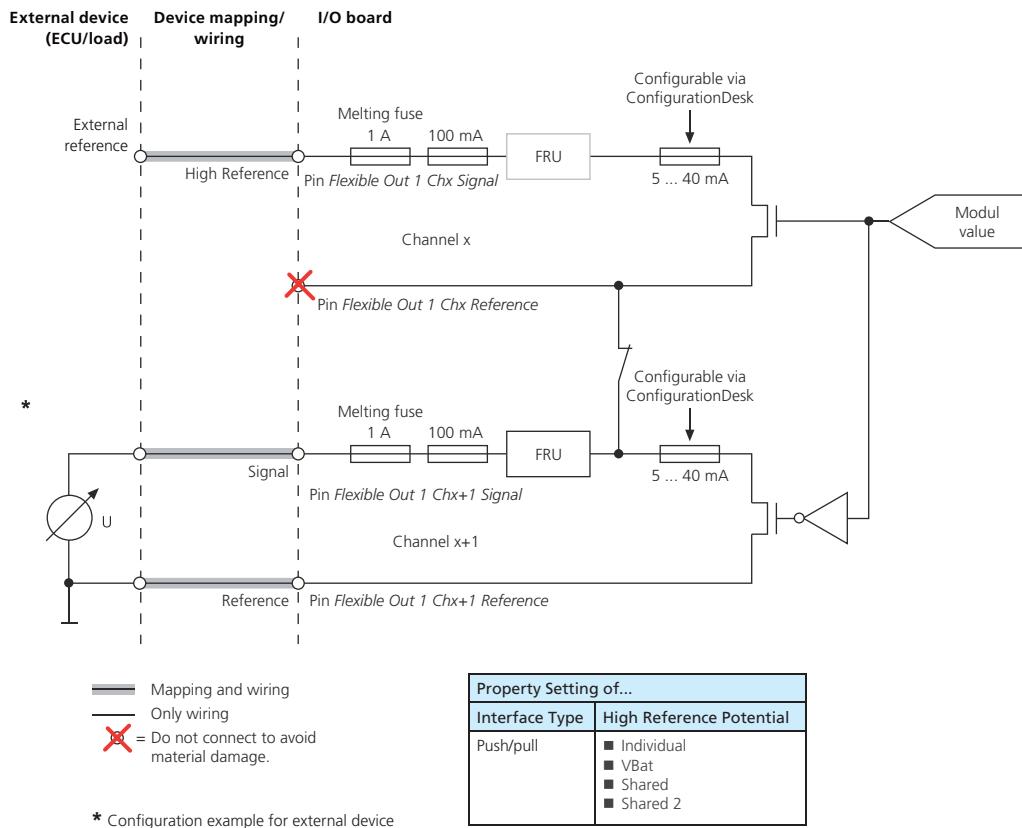
Potentiometer simulation



Digital out signal generation: switch configuration

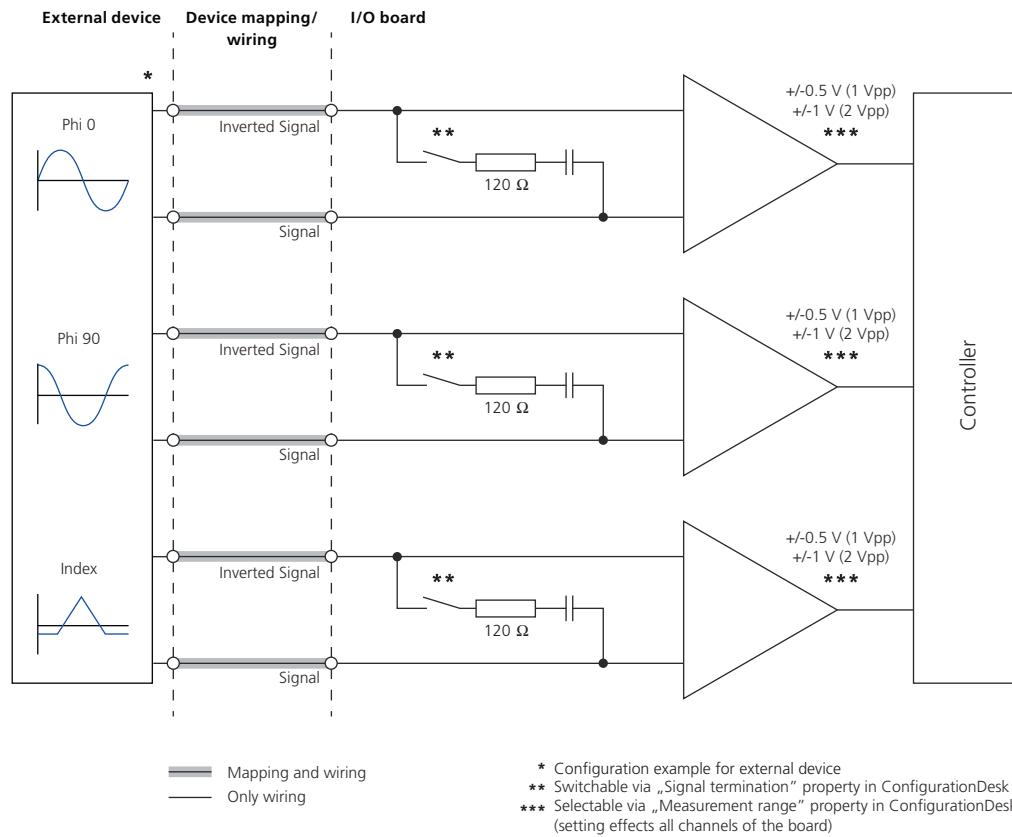


**Digital out signal generation:
push-pull configuration**

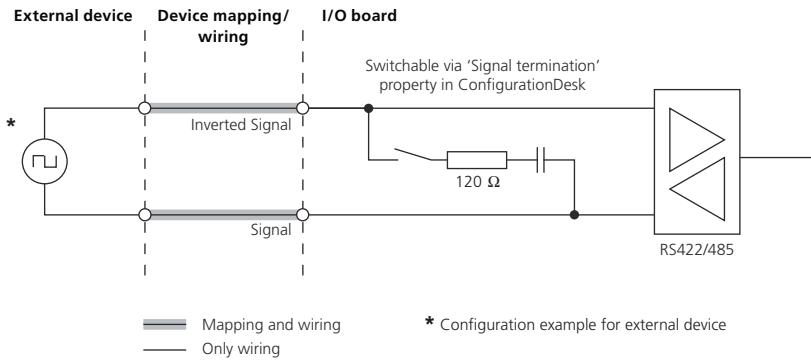


Flexible In/Out 1

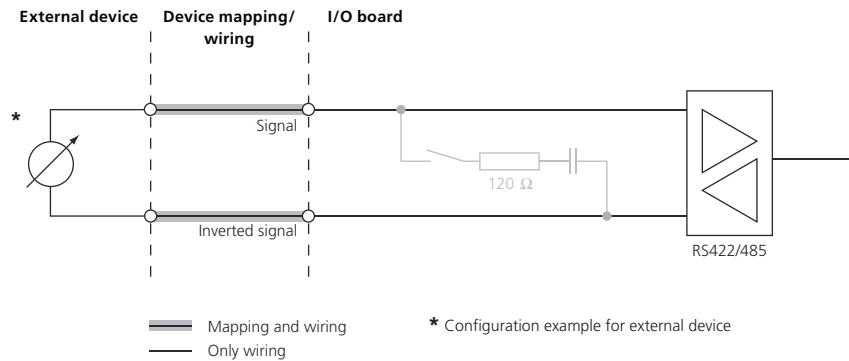
Analog encoder in



Differential digital voltage measurement

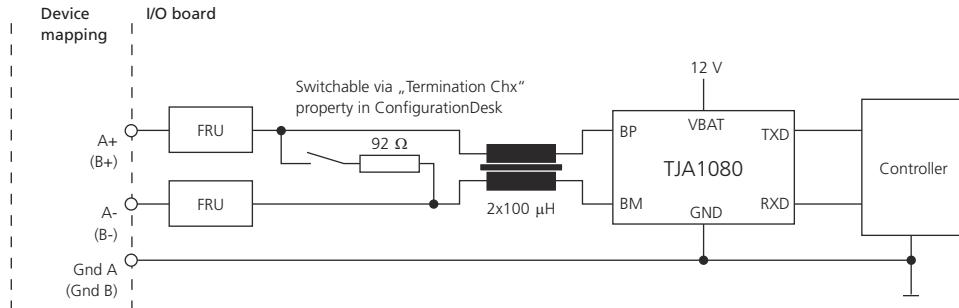


Generating differential digital output signals



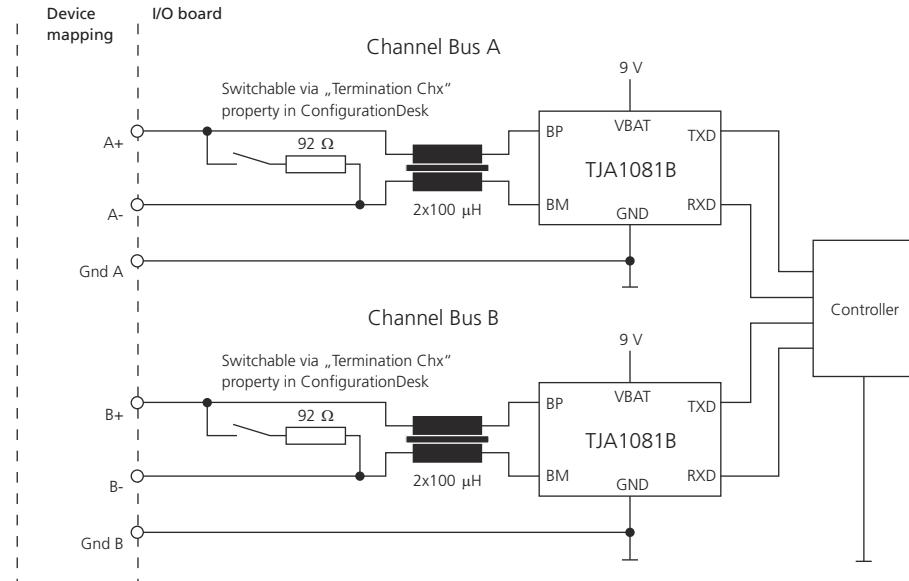
FlexRay 1

FlexRay without feedthrough

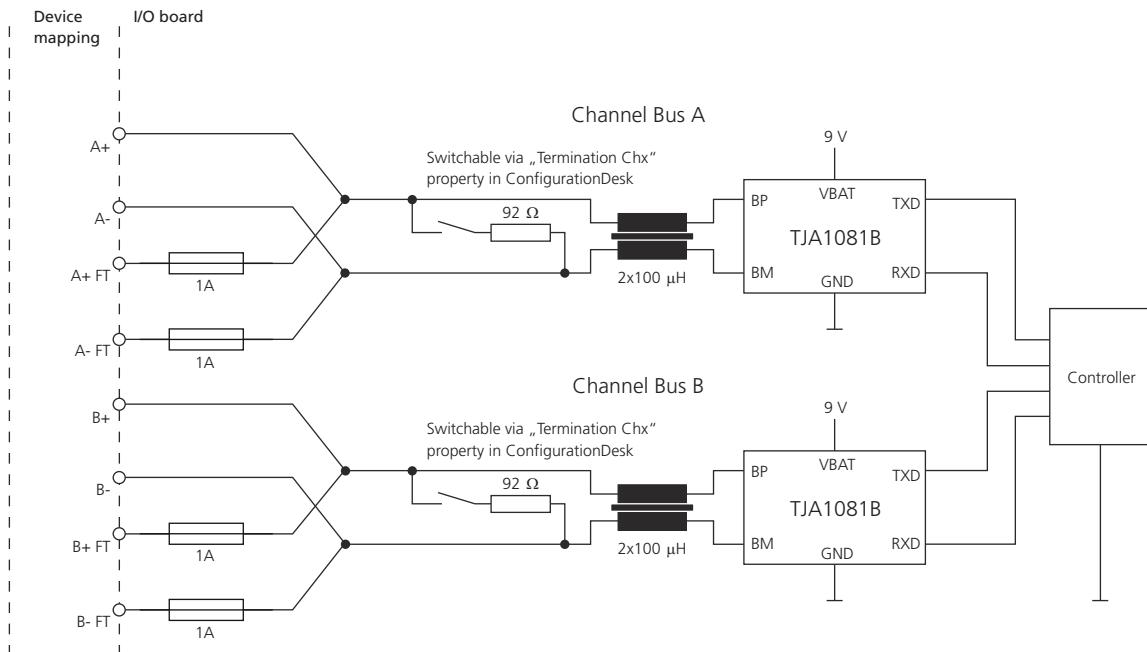


FlexRay 2

FlexRay without feedthrough

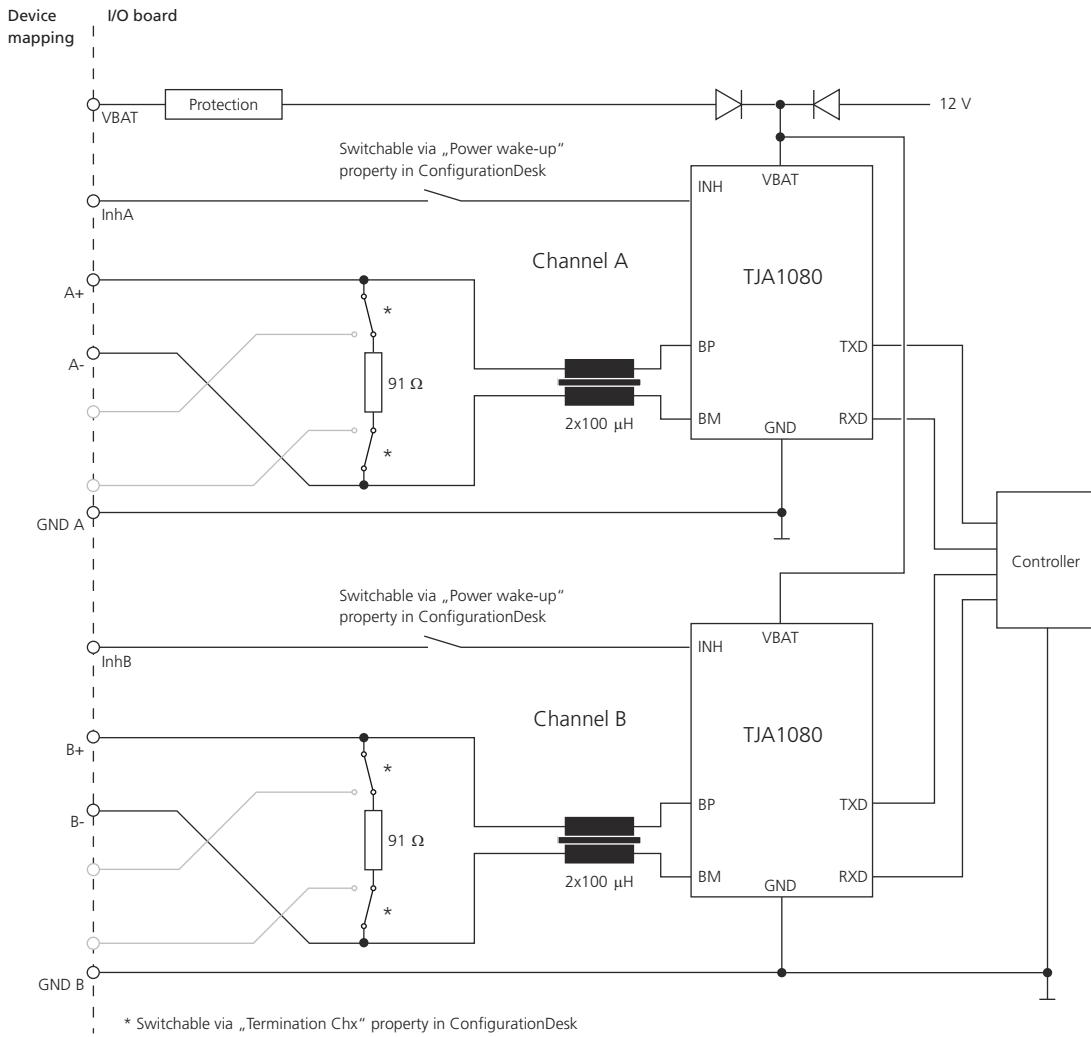


FlexRay with feedthrough

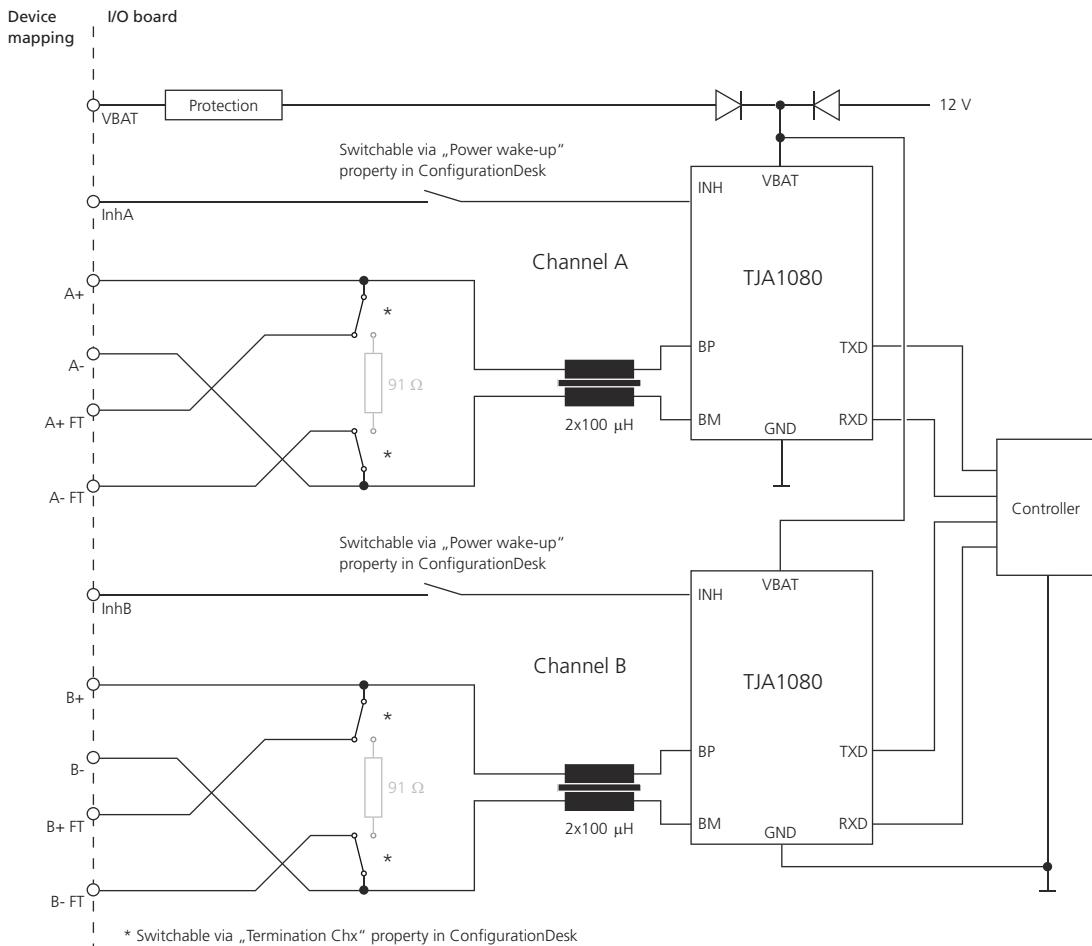


FlexRay 3

FlexRay without feedthrough

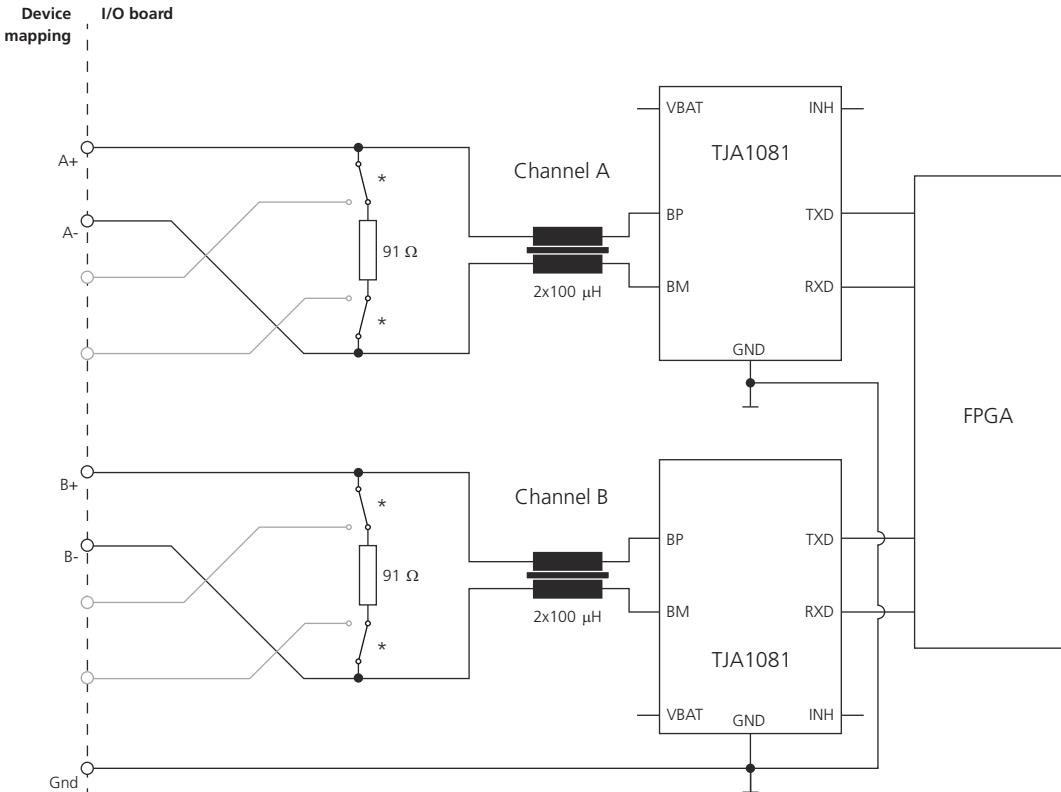


FlexRay with feedthrough



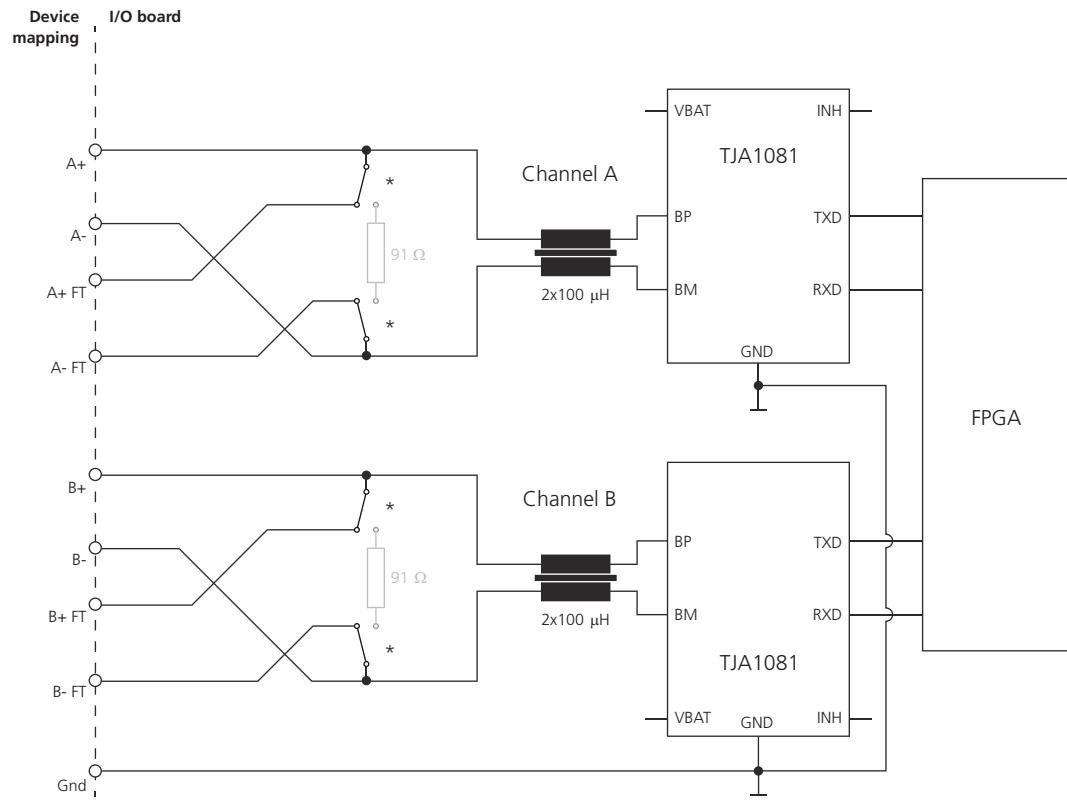
FlexRay 4

FlexRay without feedthrough



* Switchable via „Feedthrough Chx” property in ConfigurationDesk

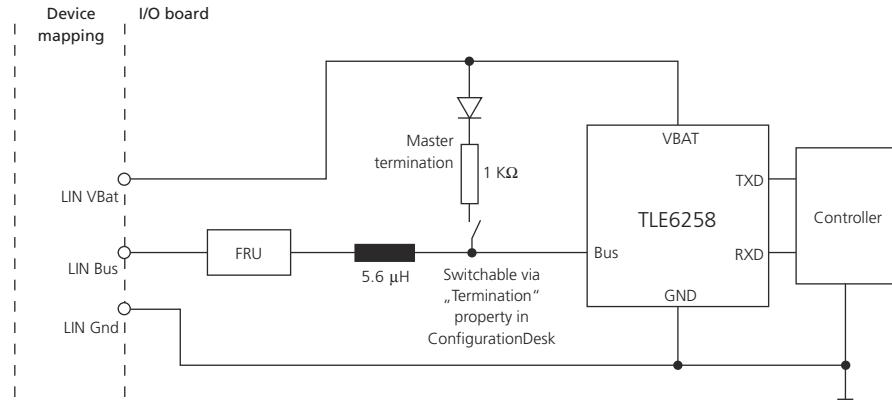
FlexRay with feedthrough



* Switchable via „Feedthrough Chx“ property in ConfigurationDesk

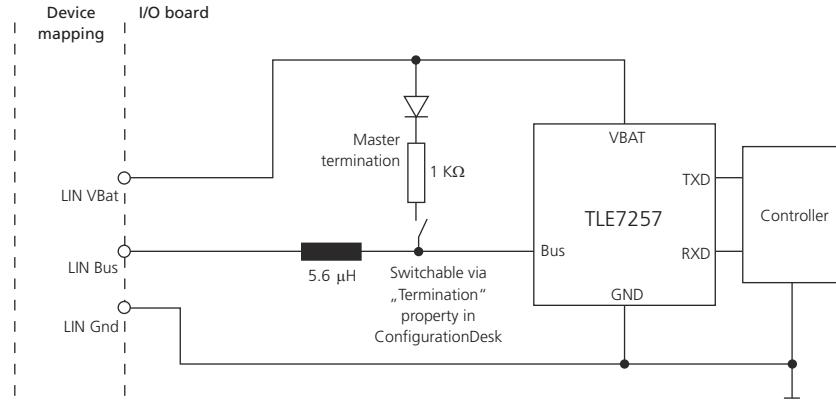
LIN 1

LIN



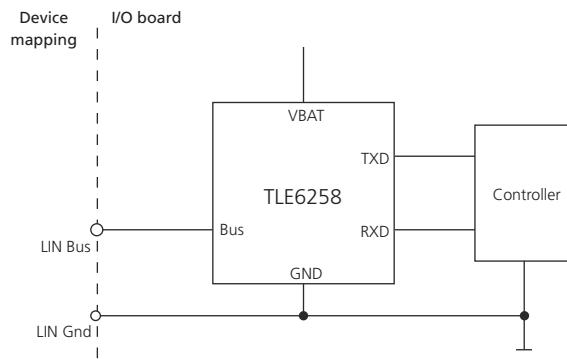
LIN 2

LIN



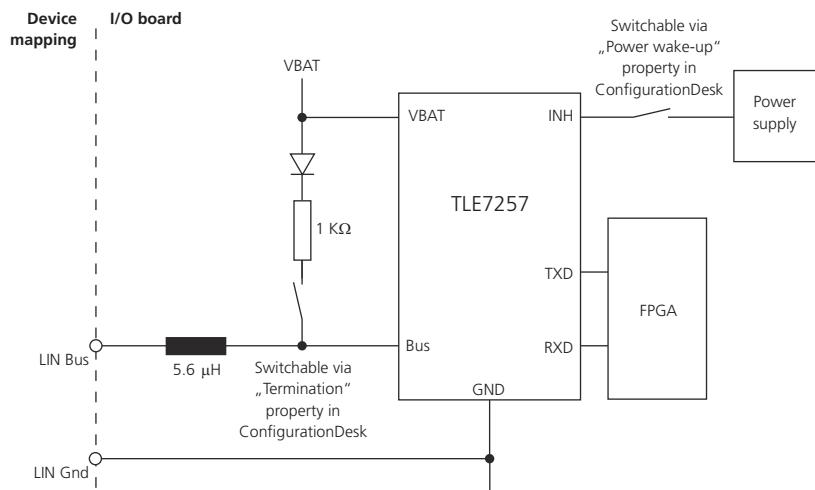
LIN 3

LIN



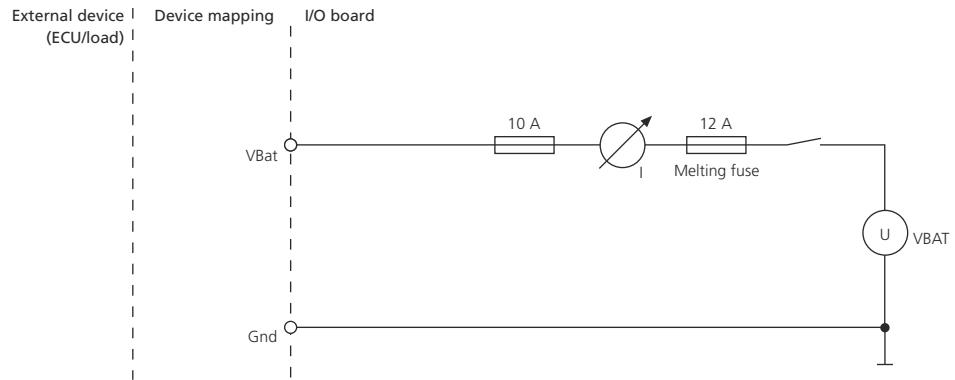
LIN 4

LIN



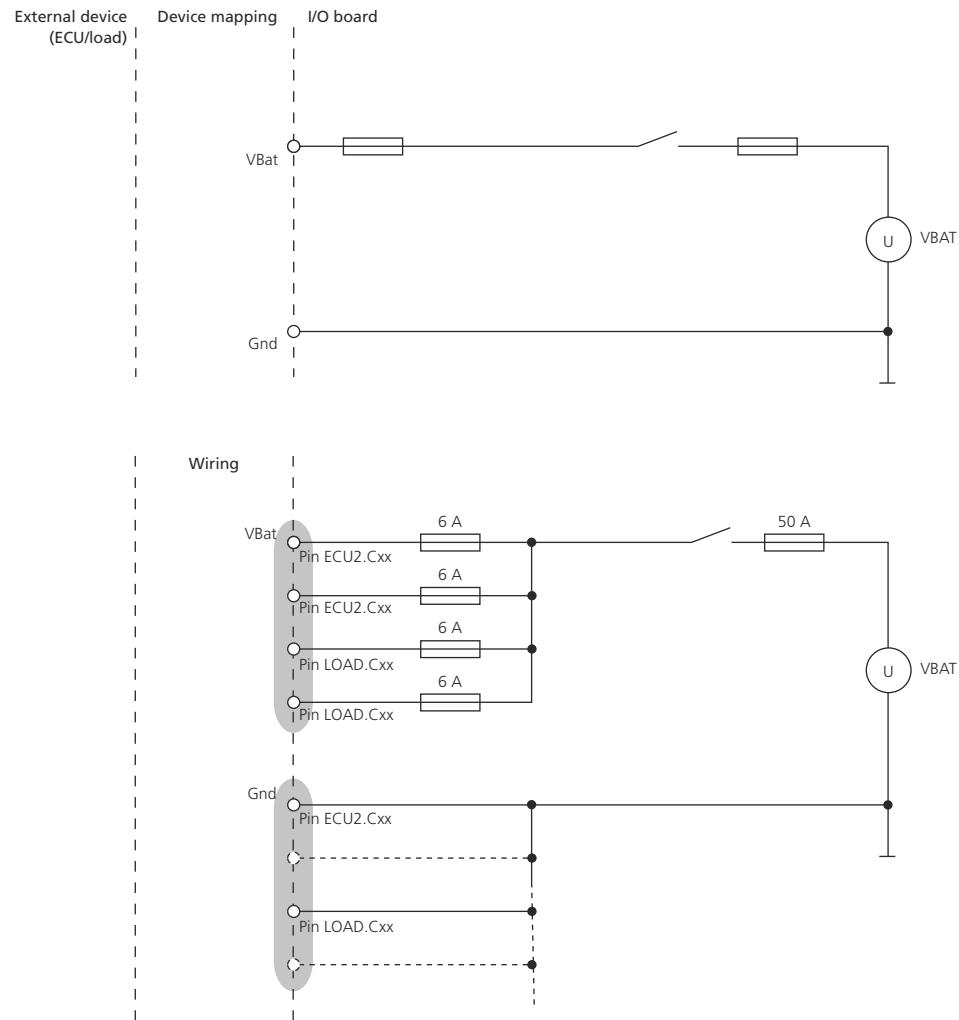
Power Switch 1

Switching supply voltage



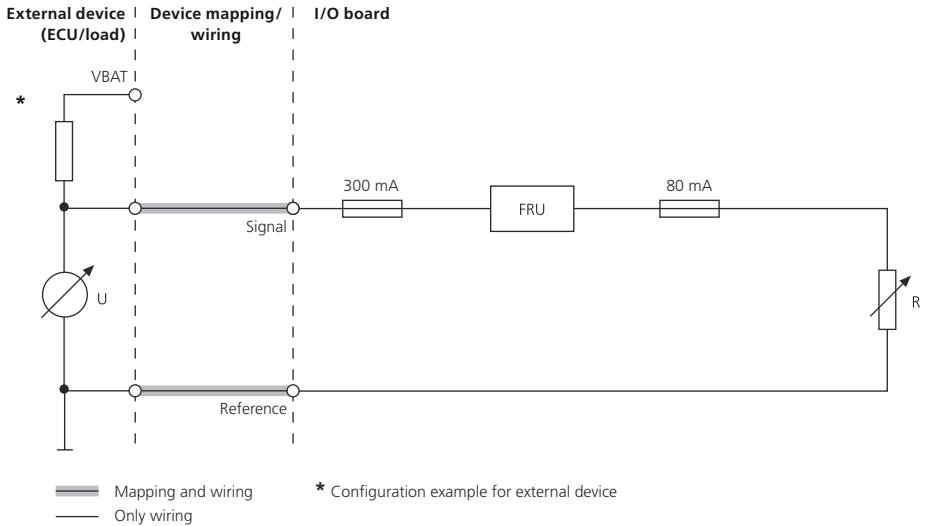
Power Switch 2

Switching supply voltage

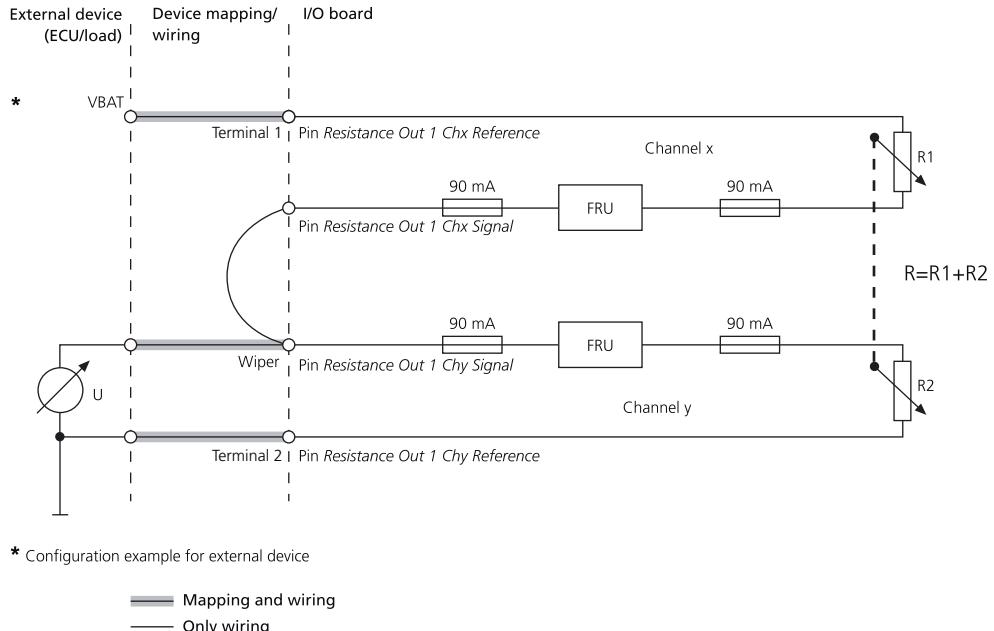


Resistance Out 1

Resistor simulation

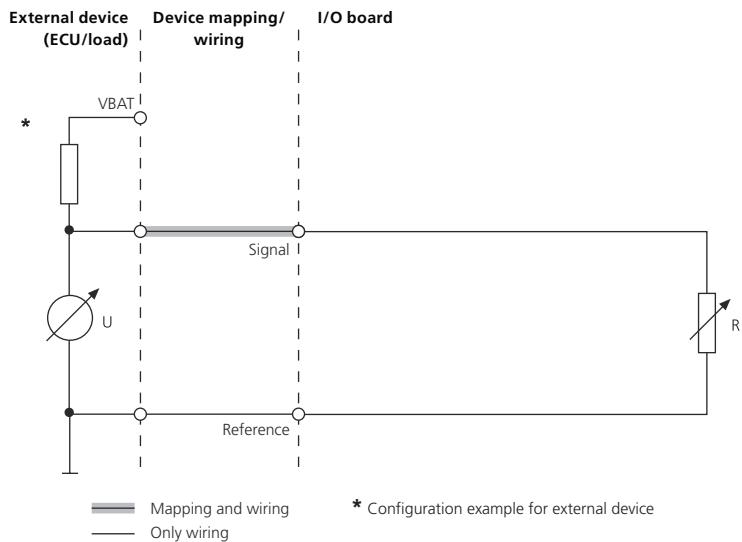


Potentiometer simulation

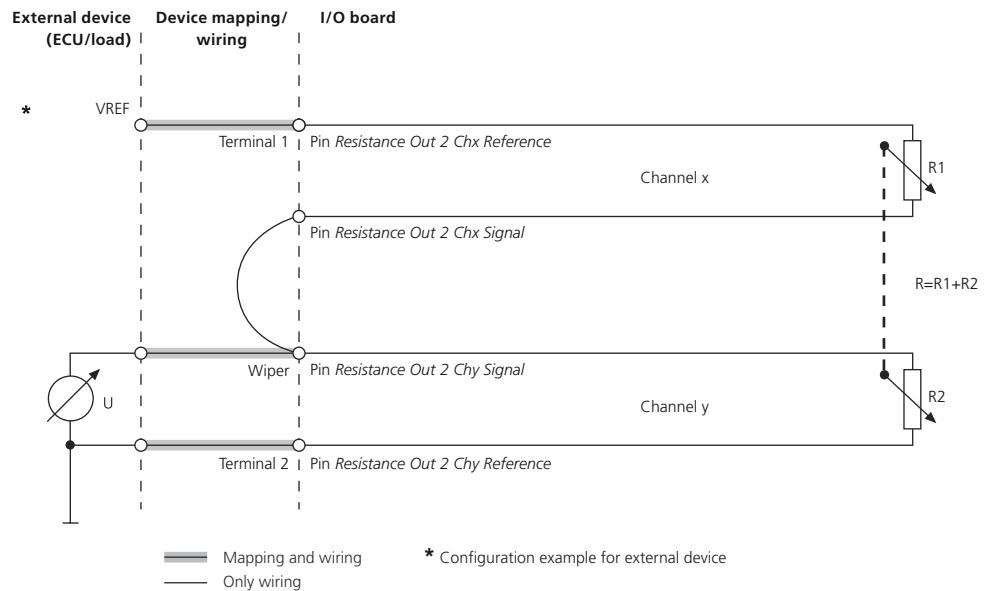


Resistance Out 2

Resistor simulation

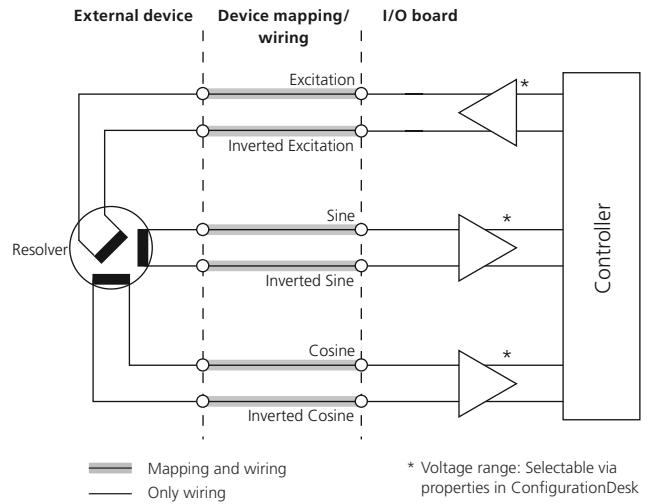


Potentiometer simulation



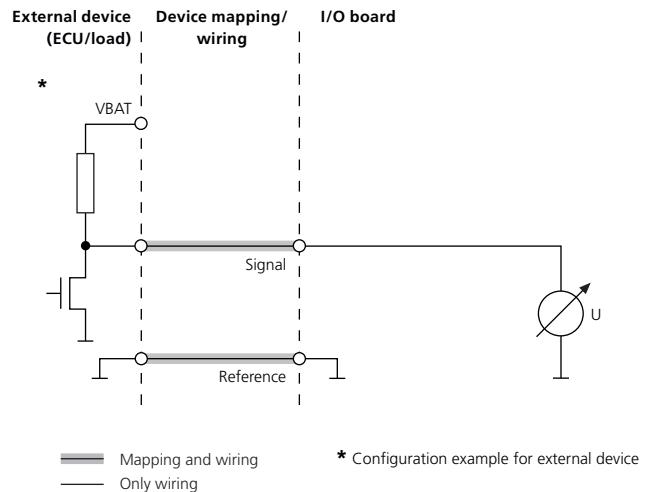
Resolver In 2

Resolver in



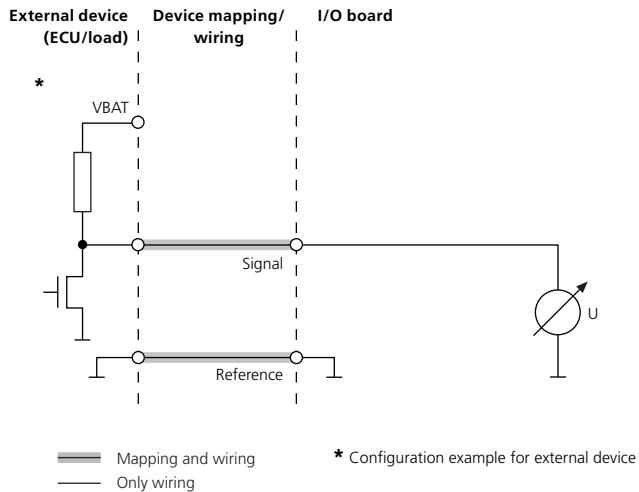
Trigger In 1

Digital voltage measurement



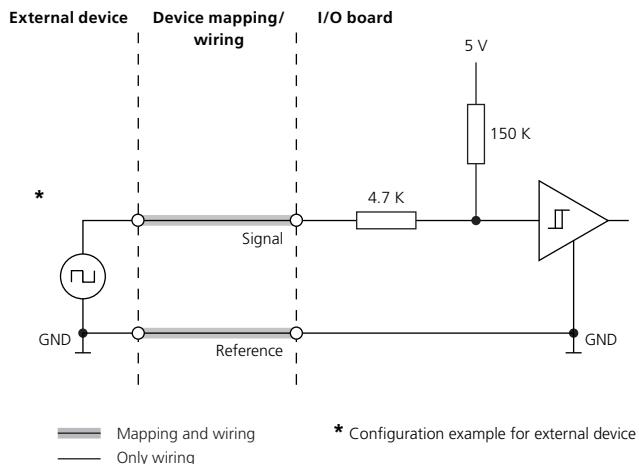
Trigger In 2

Digital voltage measurement



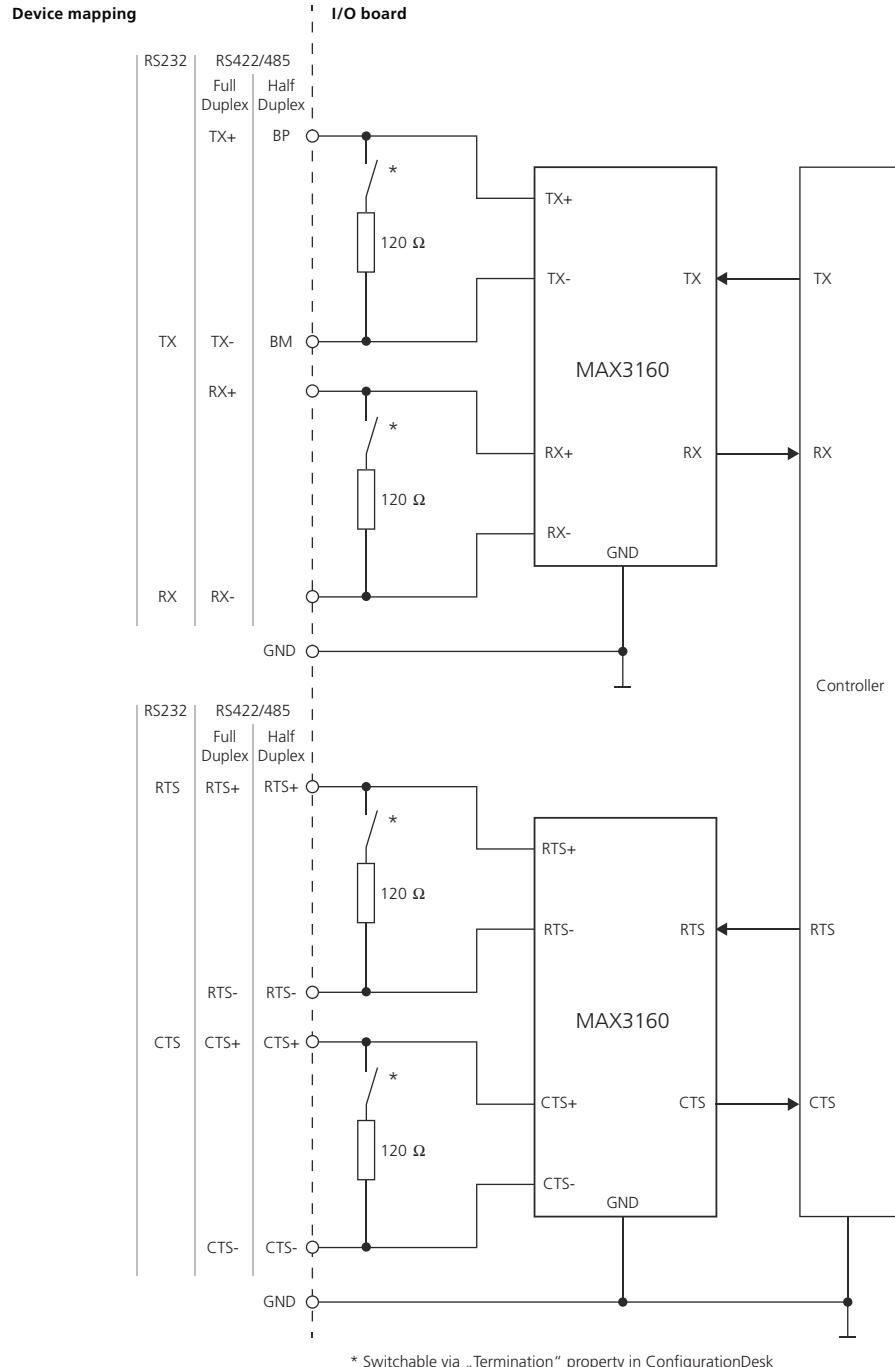
Trigger In 3

Digital voltage measurement

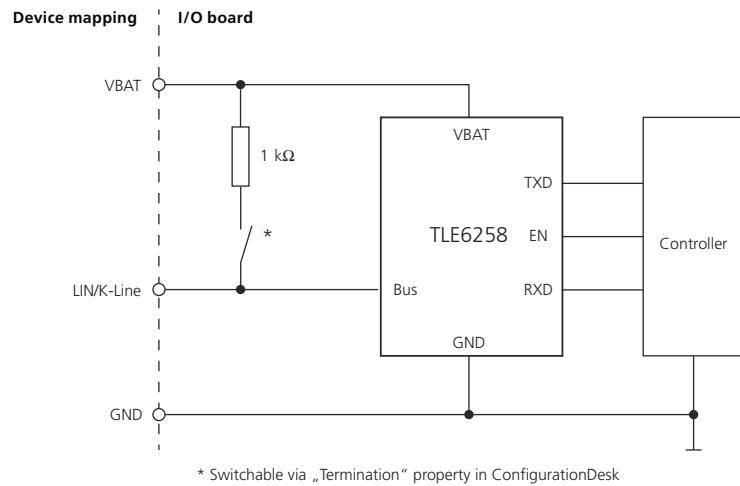


UART 1

UART

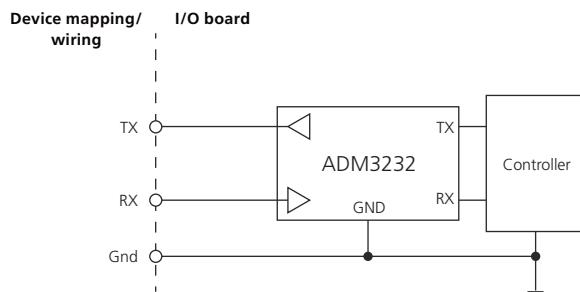


UART via K-Line



UART 2

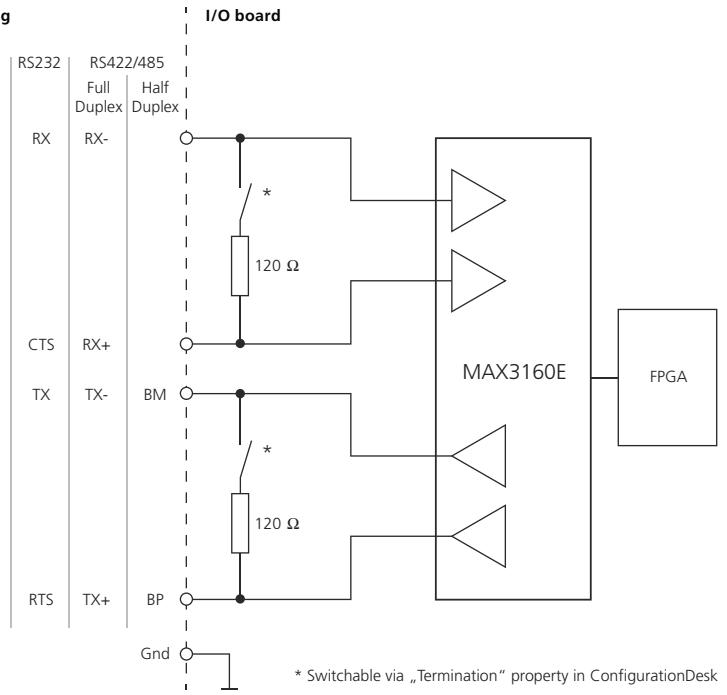
UART



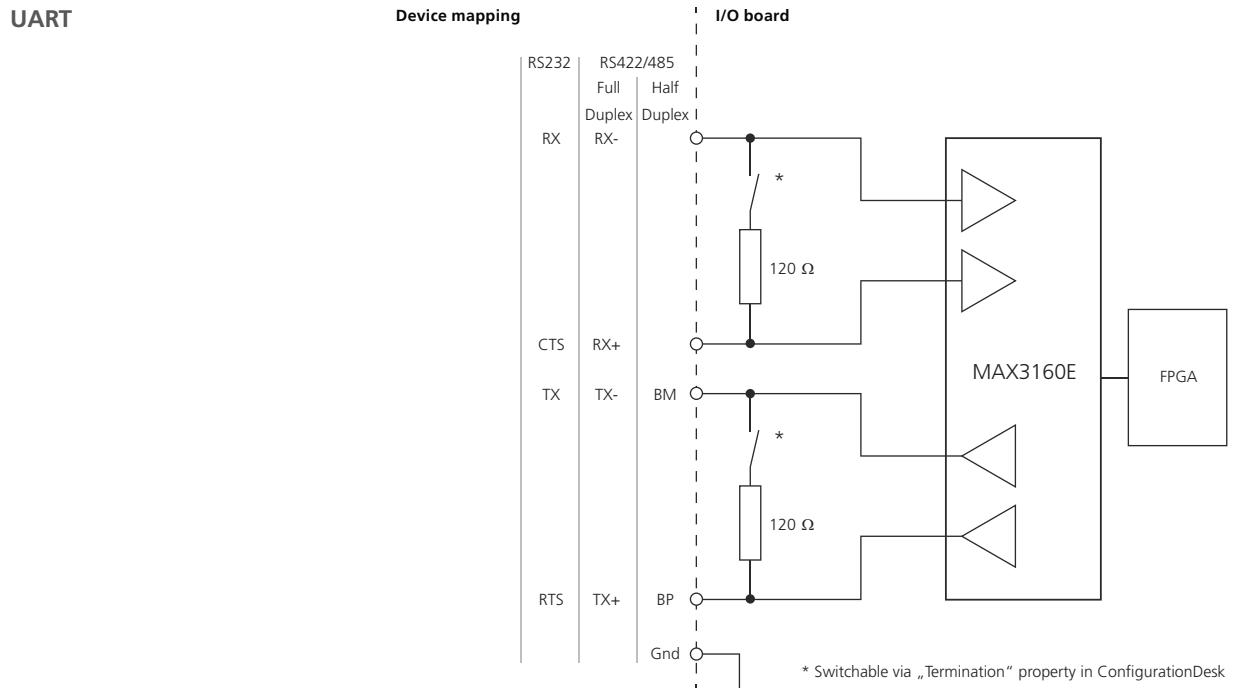
UART 3

UART

Device mapping



UART 4



A

Acceleration In
 tunable properties 1484
Angular
 Clock 369
Angular Clock Setup
 block configuration 369
Angular Wavetable
 introduction 712
Angular Wavetable Digital Out
 tunable properties 734
Angular Wavetable Voltage Out
 tunable properties 723
Atmospheric Pressure In
 tunable properties 1491

B

Block-Commutated PWM Out
 tunable properties 1032
Bus Configuration 1088
Bus Manager
 LIN bus communication 1154

C

Cam In
 configuring the function block 596
 tunable properties 596
camshaft 366, 553
camshaft phase shift 368, 555, 589
camshaft sensor 588
 active 589
 passive 589
camshaft sensor signals
 basics 588
camshaft signal
 phase shift 589
camshaft wheel 588
CAN
 tunable properties 1104
CAN bus communication
 low-power mode 1108
 partial networking 1109
cfusr.cpp 1504
cfusr.h 1504
channel multiplication 95
channel set 81
Common Program Data folder 40
configuration
 Angular Clock Setup block 369
 Engine Simulation Setup block 373
 Injection/Ignition blocks 395, 426
 Knock Signal Out block 505
 Lambda DCR block 524
 Lambda NCCR block 535
configuring standard features
 System Shutdown 1413
configuring the function block
 Cam In 596
 Crank In 578

Engine Angular Pulse Out 632
Engine Control Setup 559
Knock In 644
Crank In
 configuring the function block 578
 tunable properties 578
Crank/Cam
 basics 451
 wavetable creation 458
 wavetable selection 463
Crank/Cam Current Sink
 tunable properties 482
Crank/Cam Digital Out
 tunable properties 492
Crank/Cam Voltage Out
 tunable properties 472
crankshaft 366, 553
crankshaft sensor 564
 active 565
 passive 565
 reverse 565
crankshaft sensor signal
 basics 564
crankshaft wheel 564
 supported designs 565
Current In
 import digital filter 138
 trigger source 137
 tunable properties 136, 797
Current Sink
 tunable properties 179

D

digital filter
 Current In 138
 Voltage In 165
Digital Incremental Encoder In
 trigger threshold 866
 tunable properties 303, 862
Digital Incremental Encoder Out
 position measurement 982
 tunable properties 987
digital output signal
 enabling tristate (Digital Out 1) 105
 enabling tristate (Flexible Out 1) 102
digital output signals
 enabling tristate (Digital In/Out 1) 109
 enabling tristate (Digital In/Out 2) 109
digital output signals (Digital In/Out 3)
 enabling tristate (Digital In/Out 3) 112
digital output signals (Digital In/Out 5)
 enabling tristate (Digital In/Out 5) 115
digital output signals (Digital Out 3)
 enabling tristate (Digital Out 3) 112
Digital Pulse Capture
 trigger threshold 354
 tunable properties 351
Digital Pulse In
 tunable parameters 340
Digital Pulse Out
 trigger source 252

 tunable properties 251
Documents folder 40

E

ECU Interface Configuration 1362
EnDat Master
 tunable parameters 943
Engine Angular Pulse Out
 configuring the function block 632
 tunable properties 632
Engine Control
 Introduction 550
 Setup 556
Engine Control Setup
 configuring the function block 559
 tunable properties 559
Engine Simulation
 Crank/Cam 468, 478, 488
 Injection/Ignition 395, 425
 Knock Signal Out 498
 lambda probe simulation 509
 Setup 373
Engine Simulation Setup
 block configuration 373
 tunable properties 375
Ethernet Setup 1168
Ethernet Switch
 tunable parameters 1247
event port 75
 characteristics 75
 possible states 75

F

Field-Oriented Control In/Out
 tunable properties 1068
filter
 Current In 138
 Voltage In 165
FlexRay configuration 1142
four-stroke cycle 365
four-stroke piston engines 365, 552
FPGA applications
 exporting via FPGA framework 1513
 importing 1514
 port-specific user functions 1504
FPGA custom function
 configurable I/O signals 1529
 tunable parameters 1524
 tunable properties 1524
function block 71
 configuration 77
 deleting 67
 event port 75
 function block properties 77
 function port 73
 functions 68
 logical signals 69
 managing configuration data 79
 methods for instantiating 63
 possible states 70

- purpose 67
 - rename 83
 - signal port 71
- function block types 62
 - Function Browser 62
 - function library 62
 - function port 73
 - characteristics 74
 - possible states 74
 - FuSa Challenge-Response Monitoring
 - tunable properties 1449
 - FuSa Response Trigger
 - tunable properties 1439
 - FuSa Setup
 - tunable properties 1433
 - FuSa System Monitoring
 - tunable parameters 1459
- G**
 - galvanically isolated switch (Flexible Out 1) 101
- H**
 - Hall Encoder In
 - trigger threshold 914
 - tunable properties 909
 - hardware resource
 - definition 80
 - high-side switch (Digital In/Out 1)
 - interface type 107
 - high-side switch (Digital In/Out 3)
 - interface type 110
 - high-side switch (Digital In/Out 5)
 - interface type 113
 - high-side switch (Digital Out 2)
 - interface type 107
 - high-side switch (Digital Out 3)
 - interface type 110
 - high-side switch (Flexible Out 1) 100
 - high-side switch(Digital Out 1)
 - interface type 104
- I**
 - Ignition Out
 - tunable properties 622
 - import digital filter
 - Current In 138
 - Voltage In 165
 - importing FPGA custom function 1514
 - Injection Out
 - tunable properties 609
 - Injection/Ignition
 - block configuration 395, 426
 - SCALEXIO versus DS2211 392
 - Injection/Ignition Current In
 - tunable properties 435
 - Injection/Ignition Voltage In
 - tunable properties 405
 - inline parameter 79
 - interface type (Digital In/Out 1)
 - high-side switch 107
- Push-pull configuration 108
 - interface type (Digital In/Out 3)
 - high-side switch 110
 - low-side switch 111
 - push-pull configuration 111
 - interface type (Digital In/Out 5)
 - high-side switch 113
 - low-side switch 114
 - push-pull configuration 115
 - interface type (Digital Out 1)
 - high-side switch 104
 - low-side switch 104
 - push-pull configuration 105
 - interface type (Digital Out 2)
 - high-side switch 107
 - low-side switch 107
 - push-pull configuration 108
 - interface type (Digital Out 3)
 - high-side switch 110
 - low-side switch 111
 - Push-pull configuration 111
 - interface type (Flexible Out 1)
 - galvanically isolated switch 101
 - high-side switch 100
 - low-side switch 100
 - push-pull configuration 101
 - interface type(Digital In/Out 1)
 - low-side switch 107
 - introduction to the function block
 - Power On Signal In 1400
 - System Shutdown 1406
- K**
 - Knock In
 - configuring the function block 644
 - tunable properties 643
 - Knock Signal Out
 - block configuration 505
 - introduction 498
 - tunable properties 505
- L**
 - Lambda DCR
 - block configuration 524
 - tunable properties 526
 - Lambda NCCR
 - block configuration 535
 - tunable properties 538
 - Lambda Probe In
 - tunable properties 657
 - Lambda probe simulation
 - introduction 509
 - LED Out
 - tunable properties 1497
 - LIN bus communication
 - Bus Manager 1154
 - LIN MultiMessage 1154
 - V-ECU implementation 1154
 - LIN MultiMessage
 - LIN bus communication 1154
- Local Program Data folder 40
 - low-power mode 1108
 - low-side switch (Digital In/Out 1)
 - interface type 107
 - low-side switch (Digital In/Out 3)
 - interface type 111
 - low-side switch (Digital In/Out 5)
 - interface type 114
 - low-side switch (Digital Out 1)
 - interface type 104
 - low-side switch (Digital Out 2)
 - interface type 107
 - low-side switch (Digital Out 3)
 - interface type 111
 - low-side switch (Flexible Out 1) 100
- M**
 - MAT wavetable
 - converting to CSV 467
 - Multi Bit In
 - trigger source 208
 - trigger threshold 207
 - tunable properties 204
 - Multi Bit Out
 - tunable properties 236
- N**
 - Non-Volatile Memory Access
 - tunable parameters 1476
 - tunable properties 1476
- O**
 - overview of ports and basic properties
 - System Shutdown 1411
- P**
 - parameter categories
 - inline parameter 79
 - tunable parameter 80
 - partial networking 1109
 - Periodic signal generation
 - dynamic: waveform 741
 - static: wavetable 663
 - pitch 997
 - polarity of digital outputs 99, 103, 106, 109, 113
 - position measurement
 - Digital Incremental Encoder Out 982
 - Potentiometer Out
 - tunable properties 262
 - Power On Signal In
 - introduction 1400
 - introduction to the function block 1400
 - tunable parameters 1402
 - tunable properties 1402
 - Power Supply Control
 - tunable properties 1386
 - Power Switch
 - tunable properties 1394

- Push- pull configuration (Digital Out 1)
 interface type 105
push-pull configuration (Digital In/Out 1)
 interface type 108
Push-pull configuration (Digital In/Out 3)
 interface type 111
Push-pull configuration (Digital In/Out 5)
 interface type 115
Push-pull configuration (Digital Out 3)
 interface type 111
Push-pull configuration (Flexible Out 1) 101
push-pull mode (Digital Out 2)
 interface type 108
PWM/PFM In
 trigger threshold 325
 tunable properties 323
PWM/PFM Out
 tunable properties 282
- R**
- Resistance Out
 tunable properties 271
Resolver In
 tunable properties 927
Run-Time-Tunable parameter 80
- S**
- SCAEXIO versus DS2211
 injection/Ignition 392
SENT
 signal 1256
SENT In
 measurement point 1298
 tunable properties 1289
SENT Out
 tunable properties 1320
signal
 SENT message 1256
signal port
 characteristics 72
 possible states 72
Sine Encoder In
 tunable properties 889
speed measurement
 range 572
SPI Master
 tunable parameters 1341
 tunable properties 1341
SSI Master
 trigger 973
 tunable properties 961
Stop-Run-Tunable parameter 80
System shutdown
 introduction 1406
System Shutdown 1405
 basics on using the system shutdown
 functionality 1407
 configuring standard features 1413
 introduction to the function block 1406
 overview of ports and basic properties 1411
- tunable parameters 1412
tunable properties 1412
System Temperature Monitoring
 tunable properties 1418
- T**
- TCP
 tunable parameters 1219
TDC 367, 554
test automation support 92
threshold
 SSI Master 973
Top dead center 367, 554
trigger conditions
 Trigger In 224
Trigger In
 trigger conditions 224
 tunable properties 223
trigger source
 Current In 137
 Digital Pulse Out 252
 Multi Bit In 208
 Voltage In 164
trigger threshold
 Digital Incremental Encoder In 866
 Digital Pulse Capture 354
 Hall Encoder In 914
 Multi Bit In 207
 PWM/PFM In 325
 Triggered Current In 151
Triggered Current In
 trigger threshold 151
 tunable properties 148
tristate (Digital In/Out 1) 109
tristate (Digital In/Out 3) 112
tristate (Digital In/Out 5) 115
tristate (Digital Out 1) 105
tristate (Digital Out 2) 109
tristate (Digital Out 3) 112
Tristate (Flexible Out 1) 102
tunable parameter 80
tunable parameters
 Digital Pulse In 340
 EnDat Master 943
 Ethernet Switch 1247
 FPGA custom function 1524
 FuSa System Monitoring 1459
 Non-Volatile Memory Access 1476
 Power On Signal In 1402
 SPI Master 1341
 System Shutdown 1412
 TCP 1219
 UDP Transmit 1192, 1203
 Voltage Signal Capture (ADC Type 4) 842
tunable properties
 Acceleration In 1484
 Angular Wavetable Digital Out 734
 Angular Wavetable Voltage Out 723
 Atmospheric Pressure In 1491
 Block-Commutated PWM Out 1032
 Cam In 596
- CAN 1104
Crank In 578
Crank/Cam Current Sink 482
Crank/Cam Digital Out 492
Crank/Cam Voltage Out 472
Current In 136, 797
Current Sink 179
Digital Incremental Encoder In 303, 862
Digital Incremental Encoder Out 987
Digital Pulse Capture 351
Digital Pulse Out 251
Engine Angular Pulse Out 632
Engine Control Setup 559
Engine Simulation Setup 375
Field-Oriented Control In/Out 1068
FPGA custom function 1524
FuSa Challenge-Response Monitoring 1449
FuSa Response Trigger 1439
FuSa Setup 1433
Hall Encoder In 909
Ignition Out 622
Injection Out 609
Injection/Ignition Current In 435
Injection/Ignition Voltage In 405
Knock In 643
Knock Signal Out 505
Lambda DCR 526
Lambda NCCR 538
Lambda Probe In 657
LED Out 1497
Multi Bit In 204
Multi Bit Out 236
Non-Volatile Memory Access 1476
Potentiometer Out 262
Power On Signal In 1402
Power Supply Control 1386
Power Switch 1394
PWM/PFM In 323
PWM/PFM Out 282
Resistance Out 271
Resolver In 927
SENT In 1289
SENT Out 1320
Sine Encoder In 889
SPI Master 1341
SSI Master 961
System Shutdown 1412
System Temperature Monitoring 1418
Trigger In 223
Triggered Current In 148
USB Eject 1424
Voltage In 163
Voltage Signal Capture 821
Waveform Current Sink 765
Waveform Digital Out 778
Waveform Voltage Out 751
Wavetable Current Sink 688
Wavetable Digital Out 702
Wavetable Voltage Out 674
Wheelspeed Out 1010

U

UDP Transmit
 tunable parameters 1192, 1203
USB Eject
 tunable properties 1424

V

V-ECU implementation
 LIN bus communication 1154
viewing circuit diagrams 84
Virtual Ethernet Setup 1179
Voltage In
 import digital filter 165
 trigger source 164
 tunable properties 163
Voltage Signal Capture
 tunable properties 821
Voltage Signal Capture (ADC Type 4)
 tunable parameters 842

W

Waveform
 introduction 742
Waveform Current Sink
 tunable properties 765
Waveform Digital Out
 tunable properties 778
Waveform Voltage Out
 tunable properties 751
Wavetable
 introduction 664
Wavetable Current Sink
 tunable properties 688
Wavetable Digital Out
 tunable properties 702
Wavetable Voltage Out
 tunable properties 674
Wheelspeed Out
 tunable properties 1010