

XSG Utils Library

Version 20.1 – 06 2020

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

There are different ways to contact dSPACE Support:

- Visit our Web site at <http://www.dspace.com/goto?support>
- Send an e-mail or phone:
 - General Technical Support:
support@dspace.de
+49 5251 1638-941
 - SystemDesk Support:
support.systemdesk@dspace.de
+49 5251 1638-996
 - TargetLink Support:
support.tl@dspace.de
+49 5251 1638-700
- Use the dSPACE Installation Manager:
 - On your dSPACE DVD at *Tools\InstallationManager.exe*
 - Via Start – Programs – dSPACE Installation Manager (after installation of the dSPACE software)
 - At <http://www.dspace.com/goto?im>

You can always find the latest version of the dSPACE Installation Manager here.
dSPACE recommends that you use the dSPACE Installation Manager to contact dSPACE Support.

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.de/goto?support> for software updates and patches.

Important Notice

This document contains proprietary information that is protected by copyright. All rights are reserved. Neither the documentation nor software may 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.

© Copyright 2011 - 2018 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.
AutomationDesk, CalDesk, ConfigurationDesk, ControlDesk, SCALEXIO, SystemDesk and TargetLink 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 Guide.....	9
<i>Document Symbols and Conventions</i>	9
<i>Accessing PDF File.....</i>	10
<i>Related Documents.....</i>	11
<i>Requirements.....</i>	12
Target Hardware	13
DS5203 FPGA Board.....	13
DS1514 FPGA Board + DS1552 I/O Module.....	14
DS1202 FPGA Board (MicroLabBox)	16
DS2655/DS6601/DS6602 FPGA Board + DS2655Mx I/O Module.....	17
Simulink Model Structure.....	19
General Description.....	19
Example of Interfacing	20
The Processor Setup Block.....	23
The FPGA Setup Block.....	24
XSG Utils Library	25
Library Contents.....	25
Description / Overview.....	29
Library Structure	29
Quickstart	33
Installation Guide / Procedure.....	33
How to Generate an FPGA Model.....	33
FPGA Timing Analysis	38
FPGA Build Process.....	38
Processor Build Process.....	39
Offline Simulation.....	41
Automatic Interface Generation	41
Buffer Access Blocks	49
Goto Block	49
From Block.....	50
Automatic Interface Generation	51
Angular Processing Unit (APU).....	56
Processor Output.....	56
FPGA Main Component.....	57
Processor Input.....	58
Interface Examples	60
APU Coupling over IOCNET	61
IOCNET to XSG Utils APU.....	61
XSG Utils APU to IOCNET	64
Wave Table Encoder	68
Processor Output.....	68

FPGA Main Component.....	70
Processor Input.....	73
Interface Examples.....	73
PWM measurement.....	75
Processor Output.....	78
FPGA Main Component.....	84
Processor Input.....	86
Interface Examples.....	89
Counter PWM	91
Processor Output.....	91
FPGA Main Component.....	93
Processor Input.....	95
Interface Examples.....	96
PWM Generator	97
Processor Output.....	97
FPGA Main Component.....	101
Processor Input.....	102
Interface Examples.....	104
Protocols: UART_TX	105
Processor Output.....	105
FPGA Main Component.....	107
Processor Input.....	108
Interface Examples.....	110
Protocols: UART_RX.....	112
Processor Output.....	112
FPGA Main Component.....	116
Processor Input.....	118
Interface Examples.....	121
Data store management and feedback of Look-up Table	122
1-Dimensional Look-up Table	132
Processor Output.....	132
FPGA Main Component.....	134
Processor Input.....	138
Interface Examples.....	138
1-Dimensional Look-up Table XF	140
Processor Output.....	140
FPGA Main Component.....	142
Processor Input.....	144
Interface Examples.....	145
2-Dimensional Look-up Table	147
Processor Output.....	147
FPGA Main Component.....	149
Processor Input.....	154
Interface Examples.....	155
2-Dimensional Look-up Table XF	157
Processor Output.....	157
FPGA Main Component.....	159
Processor Input.....	162
Interface Examples.....	163
3-Dimensional Look-up Table	164
Processor Output.....	164
FPGA Main Component.....	166

Processor Input	171
Interface Examples.....	171
3-Dimensional Look-up Table XF	173
Processor Output.....	173
FPGA Main Component	175
Processor Input.....	178
Interface Examples.....	179
SCOPE	181
Processor Output.....	182
FPGA Main Component.....	185
Processor Input.....	188
Interface Examples.....	189
MULTI_SCOPE	191
Processor Output.....	192
FPGA Main Component.....	195
Processor Input.....	198
Interface Examples.....	199
Average Calculation.....	201
Processor Output.....	201
FPGA Main Component.....	202
Processor Input.....	203
Interface Examples.....	204
Sine Generator	206
Processor Output.....	206
FPGA Main Component.....	209
Interface Examples.....	211
Discrete PT 1	212
Processor Output.....	212
FPGA Main Component.....	213
Interface Examples.....	214
Integrator	215
Processor Output.....	215
FPGA Main Component.....	216
Processor Input.....	219
Interface Examples.....	220
Controller: PI-Controller.....	221
Processor Output.....	221
FPGA Main Component.....	223
Processor Input.....	225
Interface Examples.....	226
Routing: Dynamic Routing	227
Processor Output.....	227
FPGA Main Component.....	229
Interface Examples.....	233
Routing: Register 2 Buffer.....	234
FPGA Main Component.....	234
Processor Input.....	238
Interface Examples	239
Routing: Buffer 2 Register.....	241
Processor Output.....	241
FPGA Main Component.....	242

Interface Examples.....	245
Version Info.....	247
Processor Output.....	247
FPGA Main Component.....	248
Processor Input.....	249
Interface Examples.....	250
IO-Access: SCALE_DAC	251
Processor Output.....	251
FPGA Main Component	254
Processor Input	256
Interface Examples.....	257
IO-Access: SCALE_ADC	258
Processor Output.....	258
FPGA Main Component	261
Processor Input	265
Interface Examples.....	266
IO-Access: Proc_2_DAC.....	267
Processor Output.....	267
FPGA Main Component.....	268
Interface Examples.....	269
IO-Access: ADC_2_Proc.....	271
FPGA Main Component.....	271
Processor Input.....	272
Interface Examples.....	273
IO-Access: Proc_2_PWM	274
Processor Output.....	274
FPGA Main Component.....	275
Interface Examples.....	276
IO-Access: PWM_2_Proc	277
FPGA Main Component.....	277
Processor Input.....	278
Interface Examples.....	279
IO-Access: MULIT_PURPOSE_DIG_OUT.....	281
Processor Output.....	281
FPGA Main Component.....	283
Interface Examples.....	285
Watchdog	286
WATCHDOG_CENTRAL.....	286
Processor Output.....	286
FPGA Main Component.....	288
WATCHDOG_EXECUTION.....	289
Small APPS	292
EDGE DETECTION.....	292
RELAY.....	294
ABS.....	295
COMP2CONST.....	297
MOD	299
SINE / COSINE.....	301
SWITCH OFF DELAY	302
MULTIFUNCTION BLOCK.....	303
DOWNSAMPLE	305
Separate Timing and Average Functionalities.....	308

Discrete First-Order Lag Element (PT1).....	312
BACKLASH.....	314
VALID_EDGES	315
DYNAMIC_SATURATION.....	316
DYNAMIC_DELAY	318
DEFAULT_LATCH_TRIGGER_INSERTION.....	319
Park Modulator.....	321
Park Clarke Transformation	322
Star / Delta conversion.....	329
PARALLEL_2_SERIAL.....	333
SERIAL_2_PARALLEL.....	335
SERIAL_SAMPLE_HOLD	337
NATURAL_NUMBER_DIVIDER.....	339
XSG_2_SL_IF	341
VECTOR_2_BITS.....	344
BITS_2_VECTOR.....	345
RECIPROC.....	346
BUS_FCN	347
Processor based parts.....	349
Overview	349
Small APPS	350
Coding Fixpoint PROC2FPGA: Signal_2_Raw	350
Coding Fixpoint PROC2FPGA: Shift.....	351
Coding Fixpoint FPGA2PROC: Slice	352
Coding Fixpoint FPGA2PROC: Raw_2_Signal	353
Coding Floating - point PROC2FPGA: Float_2_IEEE754Codec	354
Coding Floating - point FPGA2PROC: IEEE754Codec_2_Float.....	355
Demo	357
Overview	357
Demo Installation	358
Utils Demo.....	360
Useful functions for ControlDesk.....	366
Overview	366
Custom Instruments.....	367
CI Auto Signal Connector	369
Manual signal mapping of custom instruments.....	377
SCALE_DAC	377
SCALE_ADC	377
SCOPE / MULTISCOPE.....	378
Templates	380
Overview	380
Utils Template Models.....	381
Functions	388
Overview	388
Analyze Model	389

Word Length Calculator.....	395
Automation FPGA Build Actions	397
Reserve Communication Channels for Automatic Interface Generation.....	400
Useful command line called functions.....	402
xsg_utils_fcn.....	402
xsg_utils_wordlength_calculator.....	404
xsg_utils_action_buffer_access.....	406
MATLAB Tools (shortcuts).....	407
Update Buffer Interface	408
BAF / BAG: Restore invalid.....	410
BAF / BAG: Delete invalid.....	411

About This Guide

Document Symbols and Conventions

Symbols

The following symbols may be used in this document:

	Indicates a general hazard that may cause personal injury of any kind if you do not avoid it by following the instructions given.
	Indicates the danger of electric shock which may cause death or serious injury if you do not avoid it by following the instructions given.
	Indicates a hazard that may cause material damage if you do not avoid it by following the instructions given.
	Indicates important information that should be kept in mind, for example, to avoid malfunctions.
	Indicates tips containing useful information to make your work easier.

Naming Conventions

The following abbreviations and formats are used in this document:

%name% Names enclosed in percent signs refer to environment variables for file and path names, for example, %DSpace_Python25% specifies the location of your dSPACE installation in the file system.

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

Precedes the document title in a link that refers to another document.

Indicates that a link refers to another document, which is available in dSPACE HelpDesk.

Accessing PDF File

Objective After you install your dSPACE software, the documentation for the installed products is available as Adobe® PDF file.

PDF files You can access the PDF files via the shortcut as follows:

Documentation root: <%INSTALLDIR%>\Doc\XSG_UTILS.pdf

Related Documents

Below is a list of documents that you are recommended to read when working with the XSG Utils library.

- RTI Reference
- RTI FPGA Programming Blockset Guide
- RTI FPGA Programming Blockset-Processor Interface Reference
- RTI FPGA Programming Blockset-FPGA Interface Reference

Requirements

The following tools have to be installed for using a free programmable dSPACE FPGA Board:

- MATLAB & Simulink (requires Fixed-Point Designer for bus-widths greater than 53 bits)
- Xilinx Vivado including XSG
- dSPACE Release

Below you can find the compatibility matrix for dSPACE Release, Xilinx and MATLAB for the XSG Utils library and the XSG Utils Interface library:

RTI FPGA Programming Blockset	dSPACE Release	Operating System	MATLAB	Xilinx Design Tools
3.9	2020-A	Windows 7, Windows 10	R2018b, R2019a, R2019b	Vivado 2019.2

Figure 1: Compatibility matrix for XSG Utils 20.1

dSPACE Release	Operating System	MATLAB
2020-A	Windows 7, Windows 10	R2018b, R2019a, R2019b, R2020a

Figure 2: Compatibility matrix for XSG Utils Interface 20.1

	The XSG Utils Library is available with Xilinx Design Tools
	It is extremely recommended to secure that only one XSG Utils library version is used with one Matlab!
	<p>Please note that the Library is optimized for timing, resource consumption and hardware interface for the following FPGAs (boards):</p> <ul style="list-style-type: none"> - Kintex 7 (e.g. DS2655 / DS5203 Boards)

Target Hardware

DS5203 FPGA Board

Board description

With the DS5203 FPGA board (shown in the figure below), an integration of a FPGA application into a dSPACE modular PHS-Bus based system can be performed.

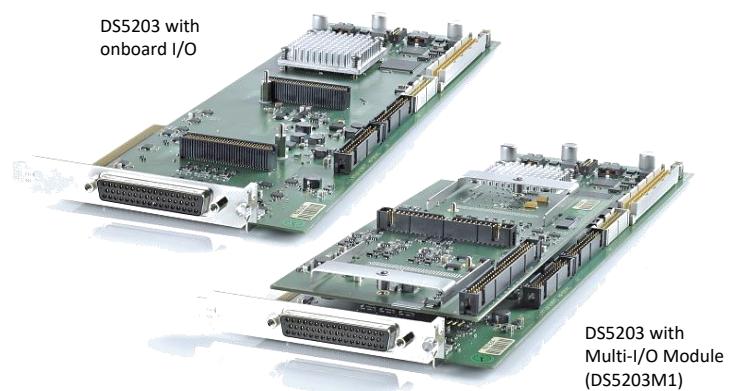


Figure 3: DS5203 Boards (PHS-based)

The board provides a Xilinx® Kintex-7 FPGA which can be programmed by the user by using the RTI FPGA Programming Blockset. The board features the following onboard I/O:

- 6 A/D channels with 10 MHz sample rate and adjustable voltage range
- 6 D/A channels with 10 MHz update rate and ± 10 V voltage range
- 16 digital single-ended input or output channels with configurable input voltage threshold and output voltage of 3.3 V or 5 V

An additional piggy back module (M1) to double the I/O can also be added, as well as a Resolver SC module (EV1099) to add an audio transformer to the DAC channels.

There are two FPGA types available:

- Kintex7 K410 (Xilinx Tool: Vivado + very huge amount of resources)
- Kintex7 K325 (Xilinx Tool: Vivado + huge amount of resources)

DS1514 FPGA Board + DS1552 I/O Module

Board description

Using the DS1514 FPGA base board for MicroAutoBox (DS1401) in combination with the DS1552 Multi-I/O piggy-back module / DS1554 Engine Control I/O Module, an integration of a FPGA application into a dSPACE MicroAutoBox can be performed.

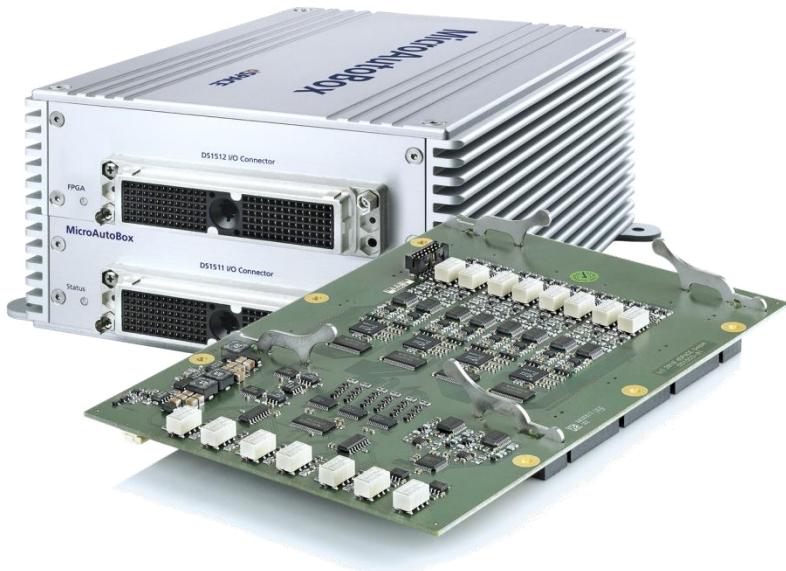


Figure 4: MicroAutoBox (DS1401) with DS1514 and DS1552

The board provides a Xilinx® Kintex-7 FPGA which can be programmed by the user, for example, by using the RTI FPGA Programming Blockset.

DS1552 I/O module:

- 8 A/D channels with a sample rate of 1 MHz and 0...5 V or ± 10 V voltage range
- 16 A/D channels with a sample rate of 200 kHz and ± 10 V voltage range
- 4 D/A channels with an update rate of 2.1 MHz and 0...5 V voltage range
- 16 digital single-ended 5 V input channels
- 16 digital single-ended output channels with configurable output voltage
- 8 digital single-ended input or output channels with configurable input voltage threshold and output voltage of 3.3 V or 5 V
- 3 crank/cam input channels with ± 40 V voltage range
- 1 Inductive zero voltage detector (for zero-crossing detection)
- 4 UART interfaces

DS1554 I/O module:

- 14 A/D channels with a sample rate of 1 MHz and ± 10 V voltage range

- 4 channels can be equipped with shunt resistors for current measurement
- 40 digital out channels for general purpose actuator control
- 8 bidirectional in- / output channels

Input:

- Threshold adjustable for each channel from 1V to 7.5V
- 4 channels can be equipped with shunt resistors for current sourcing sensors

Output:

- Push-Pull drivers
- One output voltage can be selected for all channels: 3.3V or 5V

- 2 Lambda channels, supported Lambdaprobes:
 - BOSCH LSU 4.9
 - BOSCH LSU ADV gasoline & diesel
 - BOSCH LSU 5.1
- Knock channels:
 - 4 A/D channels with a sample rate of 1 MHz and ± 5 V voltage range
- Crank / Cam channels
 - 5 digital in channels with configurable thresholds dedicated for crank/cam measurement
 - 1 Crank / Cam channel with zero crossing detection for crank / cam measurement

DS1202 FPGA Board (MicroLabBox)

Board description

Using the DS1202 FPGA board for MicroLabBox (shown in the figure below) with on-board I/O, an integration of a FPGA application into a dSPACE MicroLabBox can be performed. Three different variants are available as shown in the figure below.

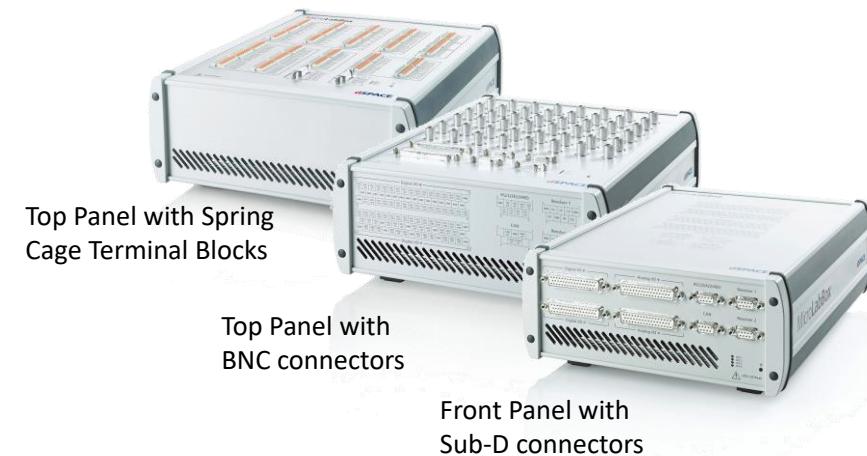


Figure 5: MicroLabBox with DS1202

The DS1202 FPGA board provides a Xilinx® Kintex-7 FPGA which can be programmed by the user by using the RTI FPGA Programming Blockset. It features the following I/O:

- 8 A/D channels with 10 MHz sample rate and ± 10 V voltage range
- 24 A/D channels 1 MHz sample rate and ± 10 V voltage range
- 16 D/A channels with 1 MHz update rate and ± 10 V voltage range
- 48 digital single-ended input or output channels with configurable input voltage threshold and output voltage of 2.5V, 3.3 V or 5 V
- 12 digital differential RS485/RS422 channels

DS2655/DS6601/DS6602 FPGA Board + DS2655Mx I/O Module

Board description

With the DS2655/DS6601/DS6602 FPGA base module and the DS2655M1 I/O module / DS2655M2 I/O module (shown in the figure below), an integration of a FPGA application into a dSPACE modular IOCNET based system (SCALEXIO) can be performed.

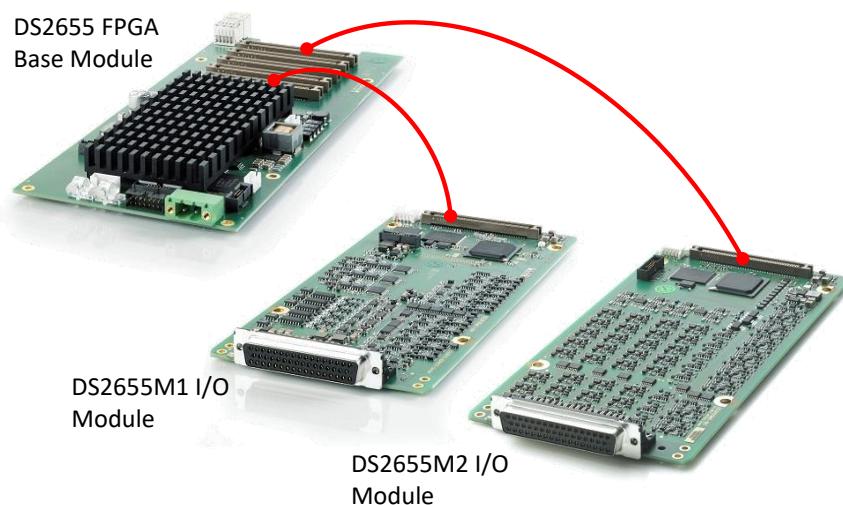


Figure 6: e.g. DS2655 FPGA base module and the DS2655M1 I/O module / DS2655M2 I/O module (SCALEXIO)

The DS2655 FPGA base module provides a Xilinx® Kintex-7 FPGA which can be programmed by the user by using the RTI FPGA Programming Blockset. Up to 5 I/O modules can be connected to the DS2655. There are two FPGA types available:

- DS6602 Kintex Ultrascale KU15P
(Xilinx Tool: Vivado + very huge amount of resources)
- DS2655 Kintex7 K410; DS6601 Kintex Ultrascale KU035
(Xilinx Tool: Vivado + huge amount of resources)
- DS2655 Kintex7 K160
(Xilinx Tool: Vivado + standard amount of resources)

DS2655M1:

- 5 A/D channels with 2 MHz sample rate and adjustable voltage range
- 5 D/A channels with 7.8125 MHz update rate and ±10 V voltage range
- 10 digital single-ended input or output channels with configurable input voltage threshold and output voltage of 3.3 V or 5 V

The I/O of the DS2655M1 enables most of the features provided by the XSG AC Motor Control library. Processing of SSI sensors, as it required RS485 or RS422 digital I/O drivers, please use the DS2655M2 I/O module. The DS2655M2 I/O module features 32 digital I/O channels with the following features:

DS2655M2:

- Push-Pull drivers
- One output voltage can be selected for all channels: 3.3V or 5V
- 16 of the channels are extended with RS485/RS232 transceivers
- RS232: max. 250 kBaud
- RS485: max. 40 MBaud
- Max. of following functions are possible:
 - 32 x digital input
 - 32 x digital output (push/pull or push or pull)
 - 16 x digital output (push/pull/tristate)
 - 8 x RS232 RX channels are free) (24 digital I/O or 8 x RS232 TX
 - 8 x RS232 TX channels are free) (24 digital I/O or 8 x RS232 RX
 - 8 x RS485 RX (16 digital I/O channels are free)
 - 8 x RS485 TX (16 digital I/O channels are free)
 - 8 x RS485 RX/TX (8 digital I/O channels are free)

Simulink Model Structure

General Description

Description

This chapter describes the user interface for accessing the FPGA Board from Simulink. In general, there are two different ways to access the boards, either via PHS bus (for DS5203) / IOCNET (for DS2655) for communication between processor and FPGA or via digital or analog in channels on the board. The FPGA output is like its input: either information can be written from the FPGA to the processor via register, or information can be sent out via hardware channels.

Two interfaces are available for handling the communication.

- **Processor interface** (read and write on the processor side)
- **FPGA interface** (read and write on the FPGA side)

The processor interface comes with the regular RTI or model port license, the FPGA interface comes with an extra RTI FPGA license. Here you can see the RTIFPGA library with both parts, the processor-based and the FPGA-based elements.

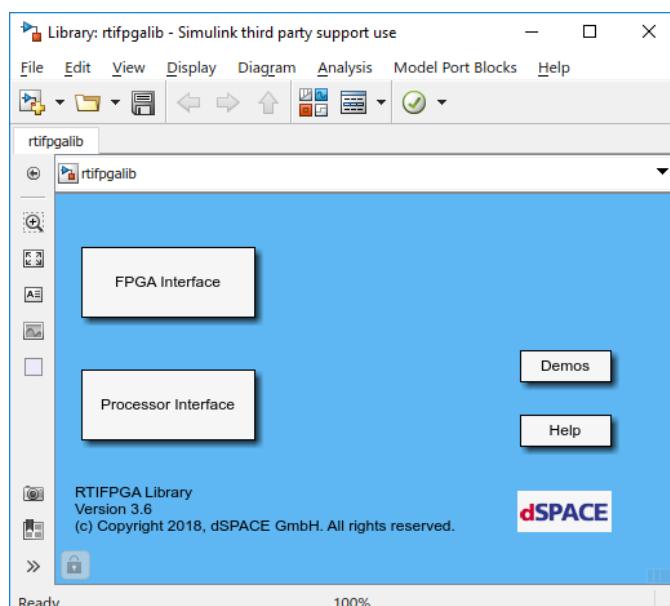


Figure 7: RTIFPGA Library



Separating these two libraries makes it possible to use precompiled FPGA applications without buying an extra license for FPGA modeling.

The XSG Electric Library contents are designed to give you easy access to create the model that you require. Each main component blockset consists of five elements:

1. **Processor out interface** (<component>_out (Processor Interface))
All the required parameters and actual values are merged into the smallest necessary number of registers or buffers.
2. **FPGA in interface** (<component>_in (FPGA Interface))
The merged signals from the processor side are decoded on the FPGA side
3. **FPGA main component** (<component>_FPGA (FPGA))
Provides the actual model functionality of the component
4. **FPGA out interface** (<component>_out (Processor Interface))
All the required signals to be returned to the processor are merged here to utilize the smallest possible amount of registers
5. **Processor in interface** (<component>_in (Processor Interface))
The merged signals from the FPGA side are decoded on the processor side

Example of Interfacing

Example: Register Access

The illustration below shows a structural example of how to implement the interfaces with usage of register access of RTI. The red blocks (1-5) are components from the dSPACE XSG Utils Library. The blue blocks (6-9) are RTI interface register blocks that define the basic communication between FPGA, processor and I/O.

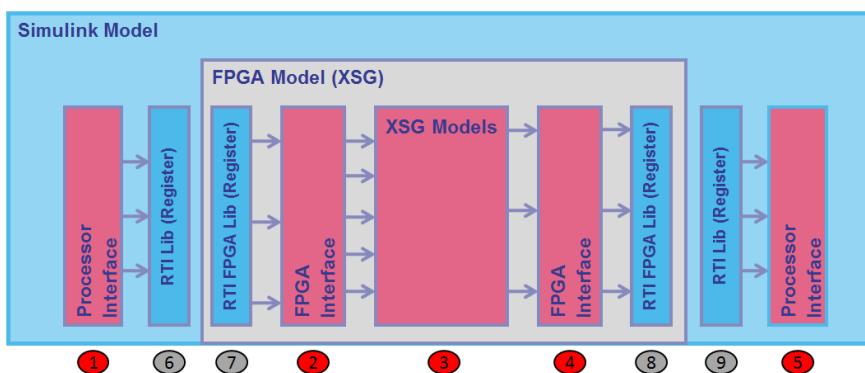


Figure 8: Using library components with register access

A detailed view of the RTI interface is shown in the next figure below. In this example, the values calculated by the processor are written via PHS bus to the FPGA (on the left-hand side) and the values calculated on the FPGA are written back to the processor (right-hand side). The I/O access is also shown (ADC1, DAC1).

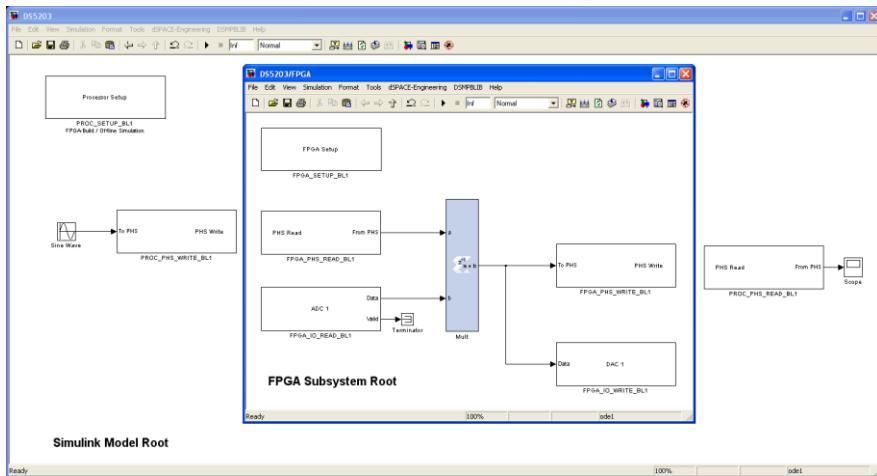


Figure 9: RTI interface usage with register access

Example: Buffer Access

Another possibility of Processor <-> FPGA interface is the realization over a buffer based access. This is useful to minimize the necessary data transfer time in IOCNET based systems between the computation node of SCALEXIO and the FPGA board. The illustration below shows a structural example of the processor and the FPGA part how to implement the interfaces with usage of buffer access with RTI Buffer.

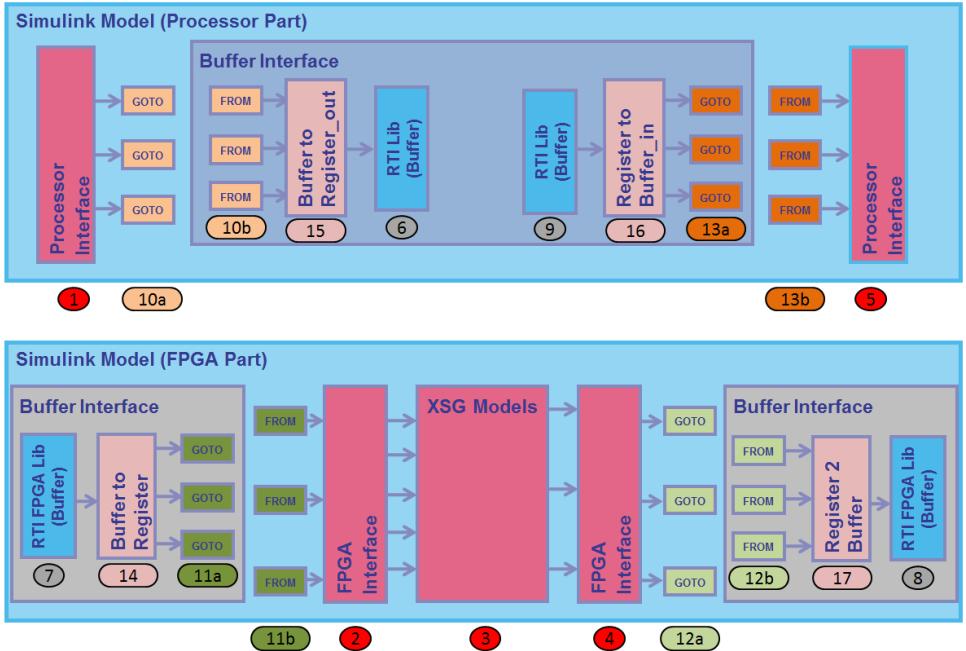


Figure 10: Using library components with buffer access

The red blocks (1-5) are components from the dSPACE XSG Utils Library. The blue blocks (6-9) are RTI interface buffer blocks that define the basic communication between FPGA and processor. To realize the mapping of the register channels over a buffer access, routing blocks (see also “Routing: Register

“2 Buffer” and “Routing: Buffer 2 Register “) of the XSG Utils library (14 – 17) are used. To get a better overview an exemplary signal route is listed below.

Exemplary signal route (Processor (1) -> FPGA (3)):

- Processor Interface (1)
 - o Output: Register signal (1 x UFix_32_0)
- Goto (10a)
 - o Type: Sink
- From (10b)
 - o Type: Source
- Buffer_2_Register_out component (15)
 - o Input: multiple Register ports (1 x UFix_32_0)
 - o Output: Buffer vector (n x UFix_32_0)
- RTI Buffer out Block (X_Data_Write block) (6)
- RTI FPGA Buffer in Block (X_Data_Read block) (7)
- Buffer_2_Register component (14)
 - o Input: Buffer input (n x UFix_32_0)
 - o Output: multiple Register ports (1 x UFix_32_0)
- Goto (11a)
 - o Type: Sink
- From (11b)
 - o Type: Source
- FPGA Interface (2)
 - o Input: Register signal (1 x UFix_32_0)
 - o Output: customized, variable data types, which are coded in the registers
- FPGA Main components block (3)

Exemplary signal route (FPGA (3) -> Processor (5)):

- FPGA Main components block (3)
- FPGA Interface (4)
 - o Input: customized, variable data types, which are collected in a SignalBus
 - o Output: Register signal (1 x UFix_32_0)
- Goto (12a)
 - o Type: Sink
- From (12b)
 - o Type: Source
- Register_2_Buffer component (17)
 - o Input: Buffer input (n x UFix_32_0)
 - o Output: multiple Register ports (1 x UFix_32_0)
- RTI FPGA Buffer out Block (X_Data_Write block) (8)
- RTI Buffer in Block (X_Data_Write block) (9)
- Register_2_Buffer_in component (16)
 - o Input: Buffer vector (n x UFix_32_0)
 - o Output: multiple Register ports (1 x UFix_32_0)
- Goto (13a)

- Type: Sink
- From (13b)
 - Type: Source
- Processor Interface (5)
 - Input: Register signal (1 x UFix_32_0)

The signal routing from the e.g. processor interface (1) to the corresponding FPGA interface (2) is n

A detailed view of the RTI interface is shown in Figure 41 and Figure 42. For a detailed description about the Goto and From blocks please refer to chapter Buffer Access Blocks.

The Processor Setup Block

Features



The Processor Setup block has to be located on the model root. All processor interface blocks can be placed anywhere inside the Simulink model except the FPGA subsystem. This block is only necessary for PHS-Bus based systems.

The number of FPGA boards which are used in the application has to be defined on the unit page, which is called double-clicking the processor setup block PROC_SETUP_BL1. Up to 16 FPGA boards can be handled. The subsystem, or precompiled binary file, has to be specified for each board. On the FPGA side the configuration can be stored in either the RAM or the flash memory.

The Interface lets you generate the interface blocks on the processor side automatically. Therefore only the interface blocks on the FPGA side have to be configured.

The Model Configuration page lets you switch between the FPGA Build and the Processor Build mode. For offline simulation, the FPGA build option must be used. Changes take effect when the Switch model mode button is clicked. If the Processor Build mode is selected, the FPGA subsystem is removed from the model file and stored inside a separate model file. When you switch back to FPGA Build mode, the FPGA subsystem is copied back to the application model.



When performing the build process on the processor side, it is mandatory to activate the Processor Build mode.

On the Advanced page, precompiled FPGA applications (*.ini) can be added. After adding them there, you can select the application on the Unit page. The FPGA application is added to the generated processor files during the processor build process.



Because of the long synthesis time and the huge resource consumption of the FPGA build process, it is recommended to use

	a separate PC. The synthesis result (*.ini file), can be linked to the application during a separate processor build on the modeling PC.
--	--

The FPGA Setup Block

Features

	<p>All blocks belonging either to the Xilinx System Generator (XSG) or to the RTI FPGA Programming Blockset have to be placed in one Simulink subsystem if they are assigned to the same FPGA board.</p> <p>The root directory of the FPGA subsystem must contain an FPGA Setup block from the RTIFPGA Library</p>
---	--

The Unit page, which is opened by double-clicking the FPGA setup block, displays the board number on which the application is stored. You must also select the framework and the piggyback.

The Parameter page shows the clock period of the FPGA. The parameters for the down-sample factor and the offline simulation period can be specified here. To start a timing analysis, an FPGA Build process or a HDL simulation, click the Execute button.

	<p>Before starting an FPGA Build process (synthesis), it is recommended to perform a timing analysis in advance to ensure that the modeled design fits the timing constraints and the resources of the target FPGA.</p> <p>It is also recommended to verify changes in the offline simulation before starting the synthesis</p>
	<p><i>If inserting interface blocks designed by dSPACE manually, ensure that all write/read registers are in an unsigned fixed format with a binary point of zero (UFix32_0).</i></p>

XSG Utils Library

Library Contents

Description / Overview	The XSG Utils Library provides general functionalities which are necessary for fast real time hardware in the loop simulation.
Quickstart	Brief description from starting the software installation process up to the download of the FPGA configuration into the real time hardware.
APU	Angular processing unit to calculate the actual position
PWM Measurement	<p>Provides the following components:</p> <ul style="list-style-type: none"> ▪ PWM measurement for one- phase systems ▪ PWM measurement for three-phase systems ▪ Counter PWM
PWM Generator	Generate a center synchronous PWM signal controlled over the processor model and provides the following components:
	<ul style="list-style-type: none"> ▪ PWM generation for one- phase systems ▪ PWM generation for three-phase systems
Wave Table Encoder	<p>Encoder with a free programmable shape over</p> <ul style="list-style-type: none"> ▪ 360 deg ▪ 720 deg
Look-Up-Table	Look up tables either with linear interpolation or input below (fixpoint / floatingpoint version available):
	<ul style="list-style-type: none"> ▪ 1D ▪ 2D ▪ 3D
Scope	<p>FPGA scope functionality to visualize FPGA signals within the FPGA clock rate or a multiple of this.</p> <ul style="list-style-type: none"> ▪ 2 FPGA signals captured (fixed) ▪ 8 out of 16 FPGA signals captured (online changeable)
Average Calculator	Calculates the average value of 3 FPGA signals, synchronously to the processor sample rate

Sine Generator	Generates sine signals for processor defined frequencies, amplitudes and offsets.
	<ul style="list-style-type: none"> ▪ Single sine wave ▪ Three phase sine wave (with online tunable phase delay)
Discrete PT 1	<ul style="list-style-type: none"> ▪ Online tunable discrete first-order lag element
Integrator	Discrete integrator block
Controller	<ul style="list-style-type: none"> ▪ Online tunable discrete PI Controller
Routing	<ul style="list-style-type: none"> ▪ Online tunable routing blocks ▪ Buffer to Register / Register to Buffer routing blocks
Version Info	<p>General information about the used XSG Library's</p> <ul style="list-style-type: none"> ▪ Version Info (stores information inside the FPGA code)
IO-Access	Analog IO access and online tunable scaling's for DACs / ADCs and DIGs <ul style="list-style-type: none"> ▪ Read ADC voltage ▪ Set DAC voltage ▪ Measurement of a simple PWM signal ▪ Generation of a simple PWM signal ▪ Scaling of DAC/ADC channels ▪ Scaling of all DAC channels (independent scaling for all channels) ▪ Multi-purpose digital output
Watchdog	Controlling of a recurrent signal.
Small Apps	<p>The library also contains elements without an interface to the processor model. This is because the elements' inputs do not need to be adjustable during run time or are of a small bit width, so that it is inefficient to use an entire 32-bit register. If a connection to the processor model is required, you have to design the interface yourself. There are different kinds of frequently used applications available:</p> <ul style="list-style-type: none"> ▪ Edge detection (rising, falling, both) ▪ Relay ▪ Absolute value of signal ▪ Compare to constant ▪ Trigonometry functions like sine and cosine ▪ Switch-off delay ▪ Multifunction block (delay and shift operations) ▪ Separate timing and average functionalities ▪ Discrete first-order lag element (PT1) ▪ Backlash ▪ Valid edge detection

- Dynamic saturation
- Dynamic delay
- Separate PWM measurement
- Clock recovery
- Default latch trigger insertion
- Scaling of a single DAC
- Clarke-Transformation
- Park-Transfromation
- Park Modulator
- Star / Delta conversion
- Bits_2_Vector
- Vector_2_Bits
- Serial_2_Parallel
- Parallel_2_Serial
- Serial_Sample_Hold
- XSG_2_SL_IF

Demos

The XSG Utils Library also contains separate demo models to illustrate an exemplary model structure of an FPGA application.

Therefore, the following model is prepared:

- DEMO_UTILS

The demonstration model contains the following main components:

- APU
- Wave Table Encoder
- Sine Generator
- Multi Scope
- Version Info

When a certain velocity is applied to the APU the actual position starts to increase and the Wave Table Encoder outputs the correct value for the actual position. The Sine Generator generates a symmetric three phase sine wave with a phase delay of 120 deg. Via the Multi Scale DAC the encoder and generator outputs are applied to DAC channels. The Multi Scope captures these outputs as well as the all ADC inputs, to be able to visualize them inside ControlDesk.

Useful functions for ControlDesk

To fasten up the layouting in ControlDesk the solution also contains special designed **Custom Instruments** (CI) for the model parts:

- SCALE_DAC
- SCALE_ADC
- MULTISCOPE
- SCOPE

To simplify the usability of these Custom Instruments a **CI Auto Signal Connector** function is also included.

Templates

To get a quick startup of programming a FPGA the XSG Utils Library also contains separate template models which supports the following FPGA Boards:

- DS5203 (7K325) with onboard I/O
 - DS5203 (7K325) with Multi-I/O Module (DS5203M1)
 - DS2655 FPGA Base Module + DS2655M1 I/O Module
-

Functions

Offline functionalities like:

- Analyze Model (several functions to analyze the actual mdl)
- Word Length Calculator (Calculation of the optimal fixpoint data type of a user value)

Description / Overview

Objective	The following section describes the general structure and features of the XSG Utils Library
------------------	---

Library Structure

General information	The XSG Utils Library is divided into:
----------------------------	--

- XSG Utils Library (FPGA-based blocks)
- XSG Utils Interface Library (processor-based blocks)

as shown in the following figure.

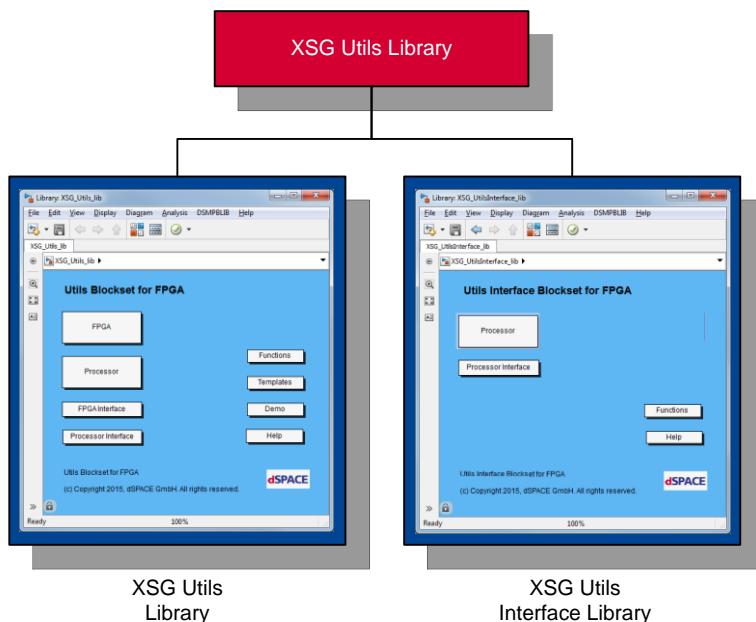


Figure 11: XSG Utils Library

XSG Utils Library

The XSG Utils Library provides easy access to FPGA based models for hardware in the loop simulation, including IO access. The library is divided into FPGA / Processor Interface and FPGA components as shown in the figure below. To open the library, type “XSG_Utils.lib” in the MATLAB Command Window.

	If the XSG Utils library will load for the first time, template models, documentation and customized ControlDesk Instruments will be installed to: My Documents\ dSPACE\ XSG_UTILS\ 20.1.
--	---

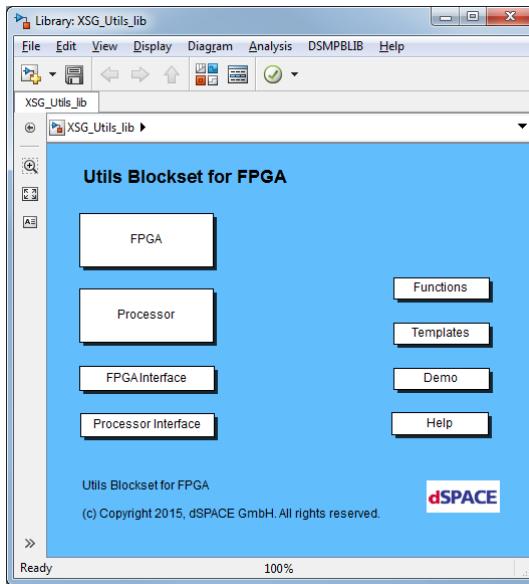


Figure 12: XSG Utils Library

The following illustration gives you an overview of the internal structure of the library.

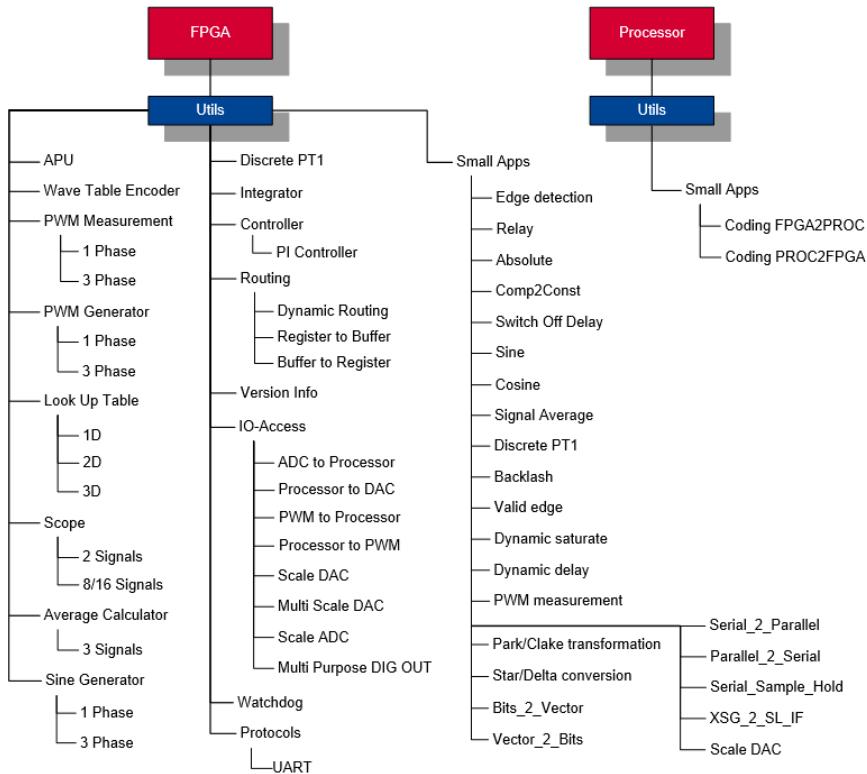


Figure 13: Overview of the XSG Utils Library

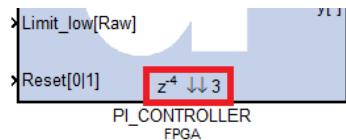
The **FPGA**-based blocks are located in the FPGA subsystem. To simplify the realization of an interface for sending processor-based parameters to the FPGA and back, optimized FPGA and processor interface blocks are located in the

subsystems FPGA Interface and Processor Interface. All the systems are organized in logical groups like “APU”, “PWM Measurement”, “LUT”, etc. All processor blocks are located in the XSG Utils Interface Library and are linked with the XSG Electric Component Library.

	The XSG Utils Library contains the XSG Utils Interface Library.
---	---

Please refer to the Demos subsystem to get an example for the general structure of a designed FPGA (Simulink) model.

To simplify the overall timing of the library blocks in a customer model, all FPGA based library blocks have a block description. This description gives information about the overall block latency and the down sampling of the internal model structure as shown in the figure below.



**Figure 14: Block information example (PI_Controller):
Latency: 4; internal model down sampling: 3**

	The fixed point blockset with Downsampling contains an average block that approximately reconstructs the original input signal.
---	---

Some of the XSG_Utils block accept Floating point arithmetic type and this can be identified with the logo ‘XF’ as shown below. Also, the input ports with color green signifies that the signal connected to this port can be of Signed, Unsigned or Floating arithmetic type; the color ‘red’ signifies that the signal should be of type Signed/Unsigned; the color blue signifies that the signal should be of Floating-point type.

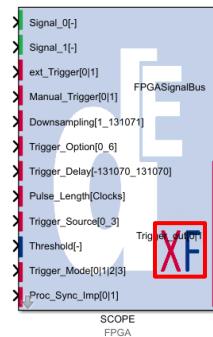


Figure 15: Block information example (Scope): Floating point

The Floating-point processor block GUI contains (?) icon, which provides detailed explanation about the corresponding parameter:

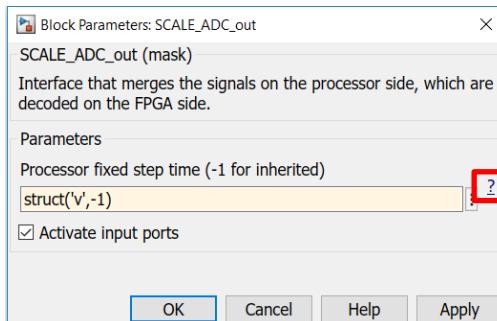


Figure 16: SCALE_ADC_out dialog (enabled input ports) with (?) icon

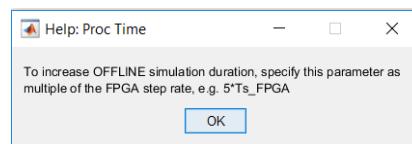


Figure 17: Help dialog for Processor fixed step time

XSG Utils Interface Library

If no user-defined adjustments have to be made in the FPGA code, for example when the FPGA configuration (*.ini file) has been delivered by a third party company or dSPACE. Then the processor interface is only required for online simulation. It is separately located inside the XSG Utils Interface Library as shown in the figure below. To open the Processor Interface Library, type "XSG_UtilsInterface_lib" in the MATLAB Command Window or navigate via the Simulink Library Browser.

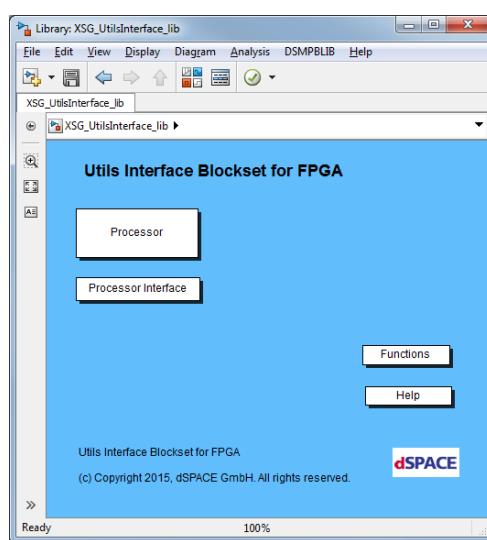


Figure 18: XSG Utils Interface Library

The internal structure of the library is the same as the structure of the XSG Utils Library, except that the interface library only contains of standard Simulink instead of XSG blocks.

Quickstart

Objective	See the following sections for a short example of building user-defined FPGA code for an <angular processor unit.
------------------	---

The chapter is subdivided into the following subchapters:

- Installation Guide / Procedure
- How to Generate an FPGA Model
- FPGA Timing Analysis
- FPGA Build Process
- Processor Build Process
- Offline Simulation
- Automatic Interface Generation

Installation Guide / Procedure

General information	To avoid errors in the installation of MATLAB, dSPACE and Xilinx software, you must install them in the following order:
----------------------------	--

- MATLAB
- dSPACE Release with RTI FPGA Programming Blockset
- Xilinx Vivado
- XSG Utils library

	For software version combinations and their compatibility, refer to Requirements chapter.
---	---

How to Generate an FPGA Model

General procedure	To generate a user-defined FPGA model, you can use the following example which explains the procedure for a PHS-Bus based model. For models which made for the DS2655 the procedure can be differs.
	<ul style="list-style-type: none"> ▪ Start MATLAB with Vivado XY.Z (the version XY.Z depends on your Vivado version) by using the System Generator token (Start\All Programs\Xilinx Design Tools\Vivado XY.Z\System Generator\System Generator) and generate a new model file (e.g. Example_MDL.mdl) ▪ Open the RTI FPGA Library by typing "rtifpgalib" in the Command Window and drag the Processor Setup block from the Processor Interface subsystem to your model. ▪ Create a subsystem for the FPGA- based model part (e.g., named FPGA)

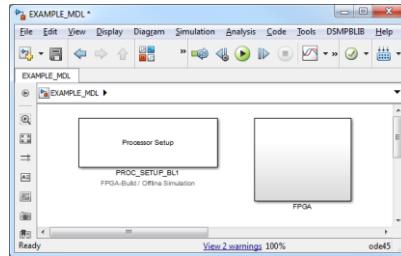


Figure 19: EXAMPLE_MDL

- Drag the FPGA Setup block from the FPGA Interface subsystem in the library to the FPGA subsystem.
- Open the Processor Setup block and on the Unit page, select the FPGA subsystem as the FPGA model for the first DS5203 board.

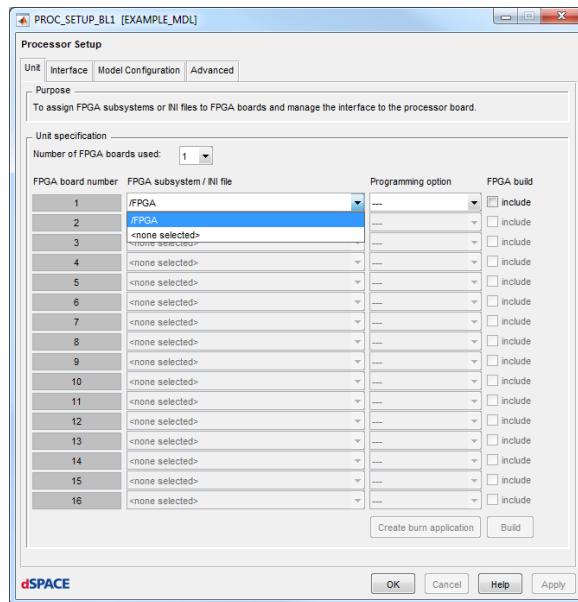


Figure 20: EXAMPLE_MDL Processor Setup block

- Open the FPGA Setup block and on the Unit page, select the FPGA chip and board for the FPGA model. In this case a DS5203 with an 7K325 FPGA is selected.

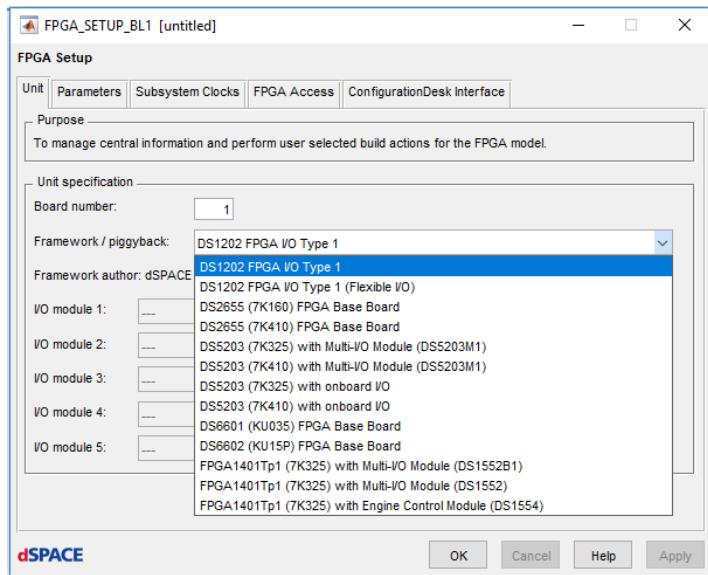


Figure 21: EXAMPLE_MDL FPGA Setup block, Page: Unit

- Configure the FPGA sample time on the Parameters page. In this case an FPGA sample rate of $10e-9$ s is parameterized.

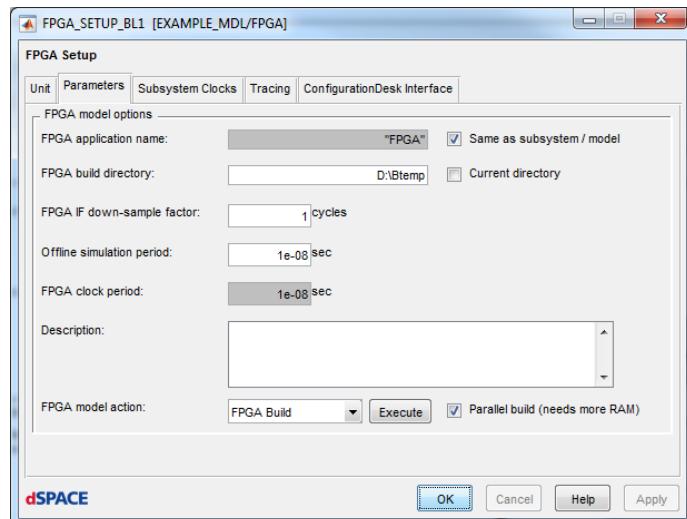


Figure 22: EXAMPLE_MDL FPGA Setup block, Page: Parameter

- After performing the changes, a system generator token is inserting in the subsystem FPGA.
- Insert the Simulink structure in the FPGA subsystem as shown in the figure below. The APU and the interface blocks are located in the **XSG Utils Library**.

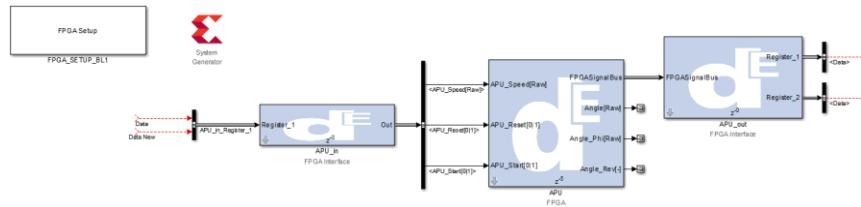


Figure 23: EXAMPLE_MDL FPGA model structure

- To generate processor-to-FPGA communication, insert an FPGA_XDATA_READ block from the RTI FPGA library in the FPGA subsystem and open the block.
- Configure the following parameters as shown below:
 - Unit Page:
 - Access type: Register
 - Channel Number: 1
 - Channel Name: APU_in_Register_1
 - Parameters:
 - Binary Point: 0
 - Format: unsigned
 - Simulation ports sample time: 10e-9
- After performing the changes, you can connect the APU_in block with the FPGA_XDATA_READ block.
- To generate FPGA-to- processor communication, insert an FPGA_XDATA_WRITE block from the RTI FPGA library in the FPGA subsystem and open the block.
- Configure the following parameters as shown below:
 - Unit Page:
 - Access type: Register
 - Channel Number: 1
 - Channel Name: APU_in_Register_1
 - Parameters:
 - Binary Point: 0
 - Format: unsigned
 - Simulation ports sample time: 10e-9
- Configure the second channel the same as the first channel with respect to a unique Channel Number (e.g. Channel Number = 2).
- After performing the changes, you can connect the APU_out block with the FPGA_XDATA_WRITE block to produce the FPGA-based model structure shown in the figure below.

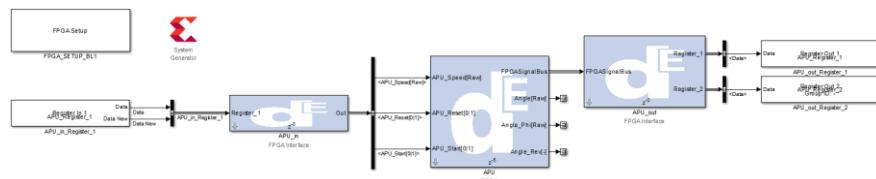


Figure 24: EXAMPLE_MDL complete FPGA model structure

- In the next step, you can insert the processor-based model part by dragging the APU_in and APU_out blocks from the **XSG Utils Interface Library**.

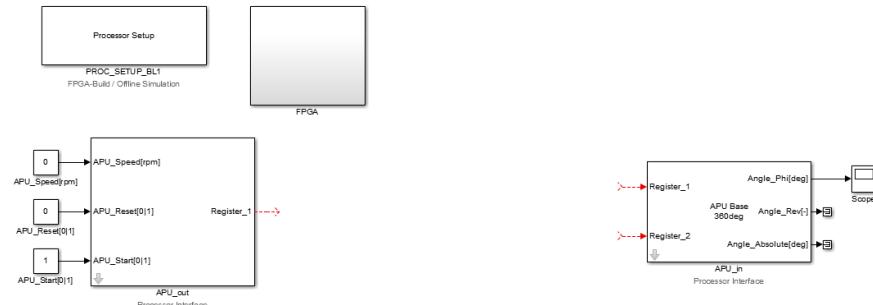


Figure 25: EXAMPLE_MDL processor model structure

- To generate processor-to-FPGA communication, insert a FPGA_XDATA_WRITE block from the RTI FPGA library and open the block.
- Configure the following parameters as shown below and connect the block to the APU_out block:
 - Unit Page:
 - Access type: Register
 - Channel Number: 1
- To generate FPGA-to- processor communication, insert a FPGA_XDATA_READ block from the RTI FPGA library and open the block.
- Configure the following parameters as shown below and connect the block to the APU_in block:
 - Unit Page:
 - Access type: Register
 - Channel Number: 1
- Configure the second channel the same as the first channel with respect to the settings of the FPGA Channel Number (e.g. Channel Number = 2).
- The complete processor model structure is shown in the figure below.

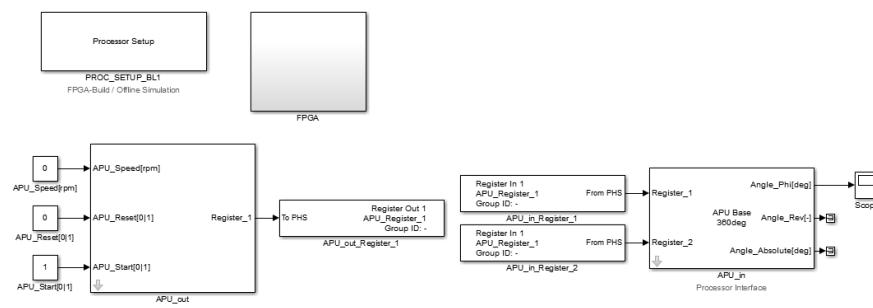
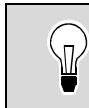


Figure 26: EXAMPLE_MDL complete processor model structure

When you have performed these steps, the model structure for angular processor unit is complete.



The processor and the FPGA-based model structure can also be built with an automatic function. For detailed information refer to the Automatic Interface Generation chapter.

FPGA Timing Analysis

General procedure

Before starting an FPGA build, it is advisable to generate a timing analysis. The following procedure can be used for this:

- Set the simulation solver options type in the Configurations Parameters to Variable-step.
- Open the FPGA Setup block which is in the FPGA subsystem and in the FPGA model, select Timing Analysis and then click Execute.
- The timing analysis starts.

After a successful timing analysis, the following window opens and the result is displayed.

	Slack (ns)	Delay (ns)	Logic Delay (ns)	Routing Delay (ns)	Levels of Logic	Source	Destination	Source Clock	Destination Clock	Path Constraints	Status
	Violation type										PASSED
1	3,292	6,723	2,867	3,856	14	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
2	4,98	4,858	2,053	2,805	10	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
3	5,138	4,877	1,627	3,25	13	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
4	5,18	3,564	3,564	0	0	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
5	6,899	2,907	1,033	1,874	8	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
6	6,942	2,683	1,033	1,652	8	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
7	7,658	1,855	0,335	1,52	1	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
8	7,665	2,337	0,399	1,938	1	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
9	7,778	2,204	1,525	0,679	13	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
10	8,198	1,768	0,322	1,446	1	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
11	8,746	1,167	0,308	0,859	0	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	
12	8,97	0,973	0,269	0,704	0	EXAMPLE_MDL..	EXAMPLE_MDL..	clk	clk	create_clock -name clk...	

Figure 27: EXAMPLE_MDL Result of the timing analysis

Each necessary delay of each path of the FPGA-based model and the current slack is displayed. If a path delay is greater than the FPGA sample time, the current path is marked in red and you can configure an additional delay on the path.

FPGA Build Process

General procedure

If an FPGA application matches the timing analysis or the timing is known, you can start a build process as follows:

- Set the simulation solver options type in the Configurations Parameters to Variable-step.

- Open the FPGA Setup block which is in the FPGA subsystem and in the FPGA model, select Timing Analysis and then click Execute.
- The FPGA build starts.

After a successful build analysis, the following text is displayed in the Command Window:

```

Starting System Generator code generation...

Starting synthesis of System Generator output files...
Packing netlists into one netlist file (EDF)...

Starting synthesis...
Starting implementation...
Starting generation of bitstream file...

WORK   DIRECTORY: E:\MartinAu\XSG_16_1_0_next_version\Templates\Example_APU
BUILD  DIRECTORY: D:\Btemp\EXAMPLE_MDL_rtiFPGA\FPGA_46B7379EE020AA
RESULT FILE:      D:\Btemp\EXAMPLE_MDL_rtiFPGA\ini\FPGA_46B7379EE020AA.ini

          Type    Used   Available Utilization [%]
Configurable Logic Block Slices (LUTs, Flip-Flops) 1038     50950      2.04
          Configurable Logic Block Slice LUTs 2837     203800     1.39
          Configurable Logic Block Slice Flip-Flops 938     407600      0.23
          Block RAM Blocks 36 Kb    1        445      0.22
          Block RAM Blocks 18 Kb    0        890      0.00
          DSP Slices            2        840      0.24

FPGA Build Done
Elapsed time is 00:11:03.
>>

```

Figure 28: EXAMPLE_MDL Result of the build process

The result of the build process in this example is the binary file FPGA_46B7379EE020AA.ini.



Because an FPGA build process needs a lot of time, it is assumed that an offline check was performed successfully before an FPGA build process is started. An additional PC is recommended for the FPGA build process.

Processor Build Process

General procedure

If the build process is successful or an INI file is available, you can start the processor build process as follows:

- Open the Processor Setup block which is located on the first layer of the model and select the INI file from the build process by clicking Add on the Advanced page.
- Select the added INI file on the Unit page and set the programming option, e.g. to "into ram", as shown in the figure below.

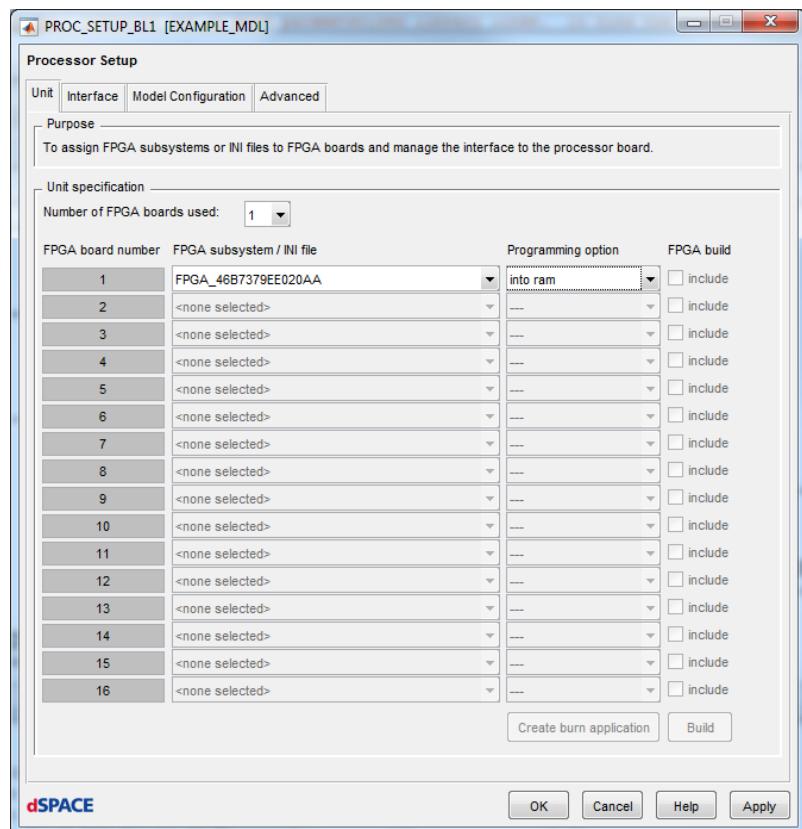


Figure 29: EXAMPLE_MDL Processor setup block

- Switch the model mode on the Model Configuration page from FPGA-Build / Offline Simulation to Processor-Build and then click the Switch model mode button.
- The model is copied to the EXAMPLE_MDL_rtiFPGASeparationFile.mdl model and the FPGA subsystem is cut off.

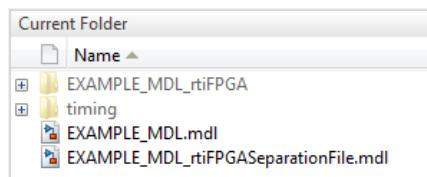


Figure 30: EXAMPLE_MDL Current directory of MATLAB after switching the model mode

- Set the simulation solver options type in the Configurations Parameters to - Fixed-step and configure the sample time of the processor, e.g., 0.001s.
- Start the processor build process by pressing Ctrl+ B.

Offline Simulation

General procedure

To generate a user offline simulation of the FPGA and the processor model, the following procedure can be used:

- Set the simulation solver options type in the Configurations Parameters to Variable-step.
- Open the Processor Setup block which is located on the first layer of the model and switch the model mode to FPGA-Build / Offline Simulation.
- Select the FPGA subsystem on the Unit page (in this case “/FPGA”)
- Configure the stop time and start the offline simulation by clicking the Run button.

Mapping of FPGA signals from / to Simulink blocks

If standard Simulink blocks are connected to Xilinx blocks, you have to insert in between the following blocks from Xilinx Blockset / Basic Elements:



Figure 31: Gateway In and Gateway Out blocks



The Gateway In block, has to be configured for the data format of the fixed point output and the offline sample period, like regular XSG blocks.

The EXAMPLE_MDL model is used gives you a better overview.

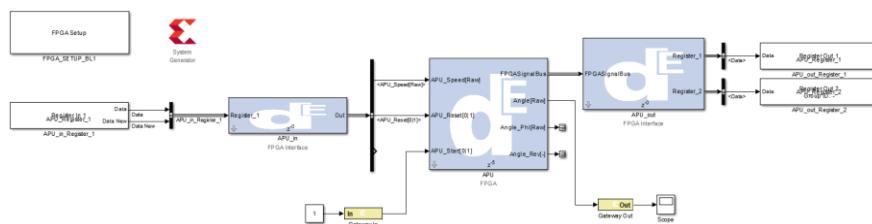


Figure 32: Gateway In and Gateway Out blocks

In the example, the APU_Start signal gets the source of a Simulink Constant block, and Angle[Raw] is connected to a Scope block.

Automatic Interface Generation

Objective

An automatic interface generation tool is available for quick, easy use of the Utils Library. The supported FPGA main components have a dialog as explained in the next section.

Example APU

The following example gives you a better overview of the functionality of automatic interface generation. Open a new model and insert a component, e.g., the FPGA main component of the angular processing unit (APU), from the library as shown in the following figure.

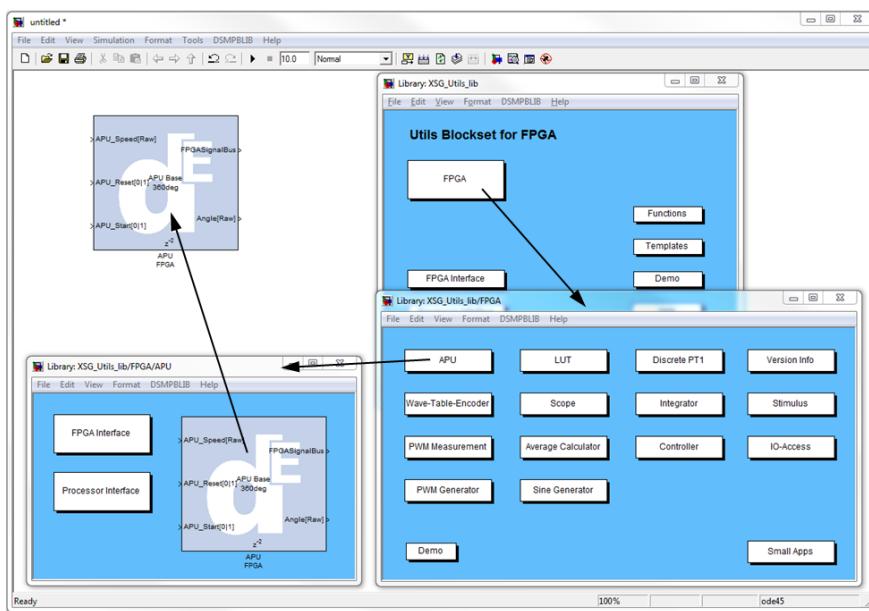


Figure 33: Tool example of automatic interface generation

After inserting the FPGA main component block of the APU, you can open the dialog by double-clicking the block. The APU's dialog parameters are shown in the figure below.

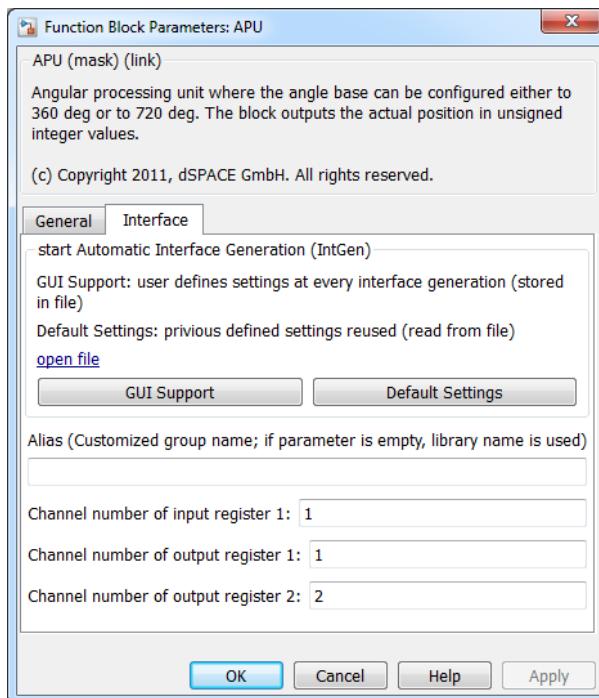


Figure 34: FPGA main component APU: Parameter dialog

Each dialog that supports automatic interface generation lets you parameterize the user-defined input and output channel numbers (Register channel numbers for the corresponding interface elements). Additional to that a customized group name (parameter Alias) can be defined. If this parameter is empty, the original library name is used.

The interface generation can start in the following ways:

- GUI Support
- Default Settings (the default settings are stored in a file which can be opened via the hyperlink in the GUI).

The following subchapters describe the both ways for the automatic generation. After that the output for different board types (Scalexio vs. PHS-Bus systems) are shown.

Interface Generation with GUI Support

At first the actual framework of the model will be analyzed. Because no FPGA framework is defined in the actual new model the following popup will open.

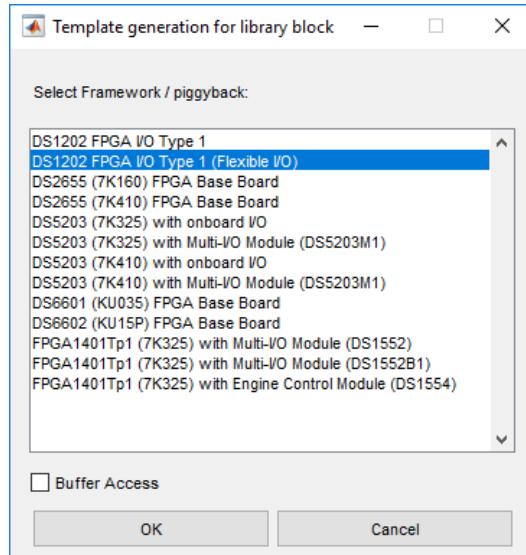


Figure 35: Selection Popup of the framework of the build process

The destination framework can be selected and the automatic interface generation continuous. If a FPGA framework is also defined in the model during the start of the automatic interface generation, the algorithm analyzed the overall model and opened a popup in which the destination submodel can be selected.

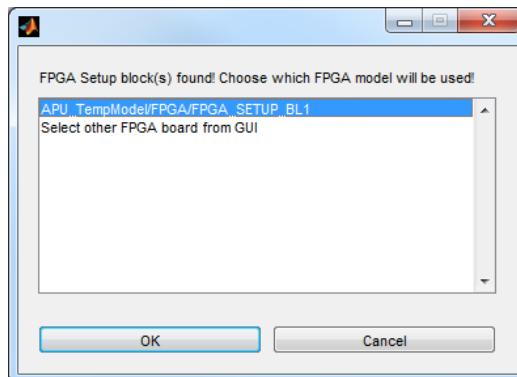


Figure 36: Selection Popup of the destination submodel of the build process

If a new framework / submodel needs to be used the Tab "Select other FPGA board from GUI" can be selected. In this case the destination framework popup from Figure 35 opens and the destination framework can be adjusted.

In case of IOCNET based FPGA boards:

If the destination framework is an IOCNET based FPGA board, it is recommended to use "Buffer Access" communication between the processor and the FPGA board. In this case an additional popup will be generated as shown in the figure below.

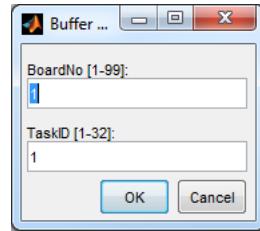


Figure 37: Selection Popup to adjust the Board and the TaskID in case of IOCNET based FPGA boards

In this popup the board number of the specific FPGA board and a TaskID can be selected. TaskID is a parameter which defines the specific task in which the overall processor and FPGA communication will be implemented.

After all necessary selections are done in the specific the automatic generation status is indicated by a progress bar, as shown in the figure below.

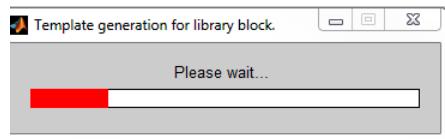


Figure 38: Status message of automatic interface generation

When an interface channel which is defined in the GUI (here, input ch.1 or output ch.1) is already used inside your source model the following message will occur:



Figure 39: Warning automatic interface generation

- Yes: The template model will be generated with the specified channel numbers defined in the GUI
- Show free Register: A dialog appears which shows the free input and output channels, so that the GUI entries can be correctly adapted
- Autoset: Replace of the double used register(s) by the first free ones, with respect to reserved channels

When the “Show free Register” or “Autoset” button is selected the actual model is also automatically checked for double used interface channels. If a channel is used more than once the following dialog appears:



Figure 40: Double used channels

When pressing the “Yes” button an html report will be generated from which related are accessible via hyperlink.

When all settings are correct the template model generation starts immediately. The result of the generation process is a template model of the APU with a standard interface configuration and a standard Simulink structure for FPGA models. The actual interface settings are stored in a file nearby the model (path: <model_path>\XSG_Sol_Folder_Default_IntGen_Settings_<modelname>.txt). This model differs, which depends on the settings on the selection popup as shown in figures below.

Interface Generation with Default Settings

If the interface generation is started with default settings, the algorithm searches the settings file which is located nearby the model. If no settings file can be found, the interface generation can be continue with **GUI Support**. If the settings file can be found the interface generation starts immediately without asking any questions. If something is not specified which is necessary for generation, the algorithm will ask the user with specific GUIs.

The actual settings can be modified via text editor. The file can be opened via the hyperlink which is located in the GUI.

Template model for SCALEXIO Hardware with Buffer Access

If a SCALEXIO framework is selected the buffer access will be default activated. If a SCALEXIO framework with buffered access is selected in the selection popup of Figure 35, the following two parts template models will be generated as show in the figure bellow.

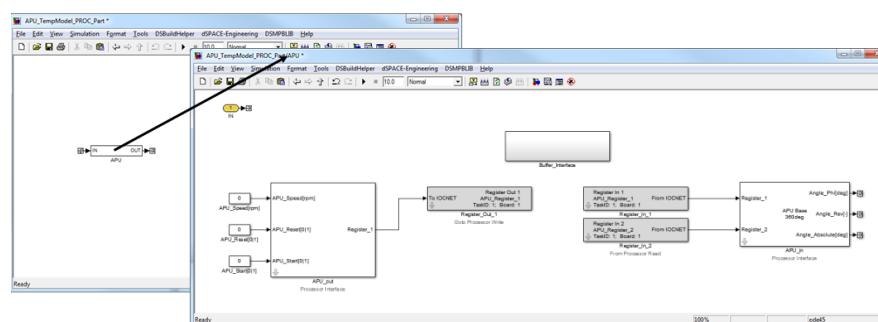


Figure 41: APU template model (processor view)

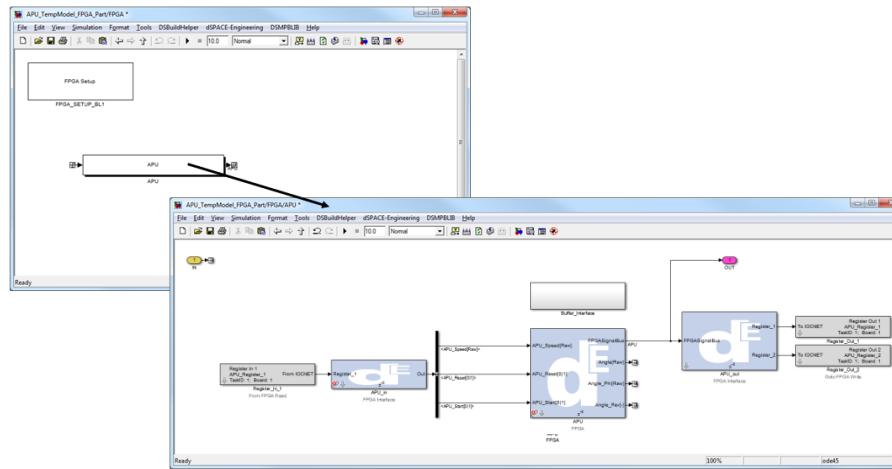


Figure 42: APU template model (FPGA view)

Template model for SCALEXIO Hardware with Register Access

If a SCALEXIO framework with register access (buffer access is not selected by the checkbox) is selected in the selection popup of Figure 35, the following two parts template models will be generated as show in the figure bellow.

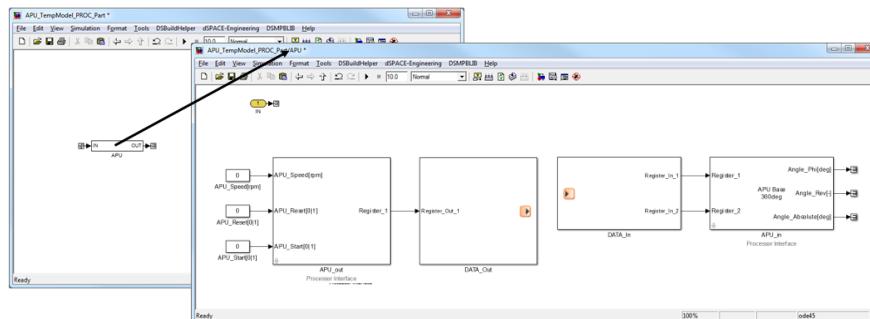


Figure 43: APU template model (processor view)

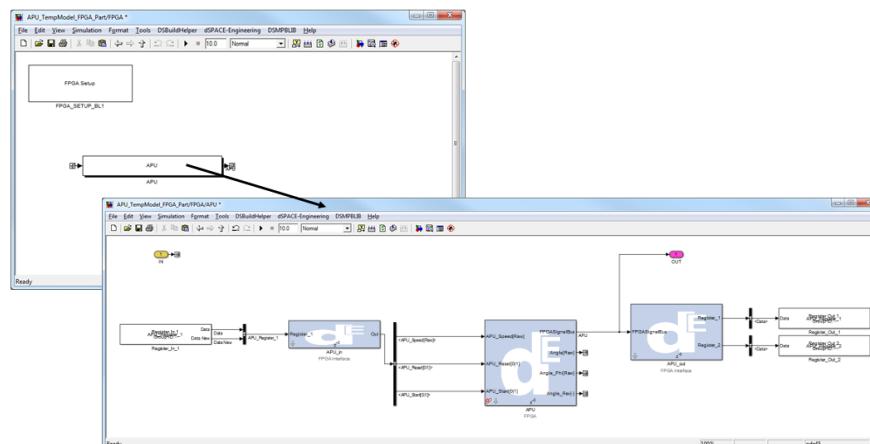


Figure 44: APU template model (FPGA view)

Template model for PHS-Bus based Hardware with Register Access

If a PHS-Bus based framework with register access (buffer access is not selected by the checkbox) is selected in the selection popup of Figure 35, the following template models will be generated as show in the figure bellow.

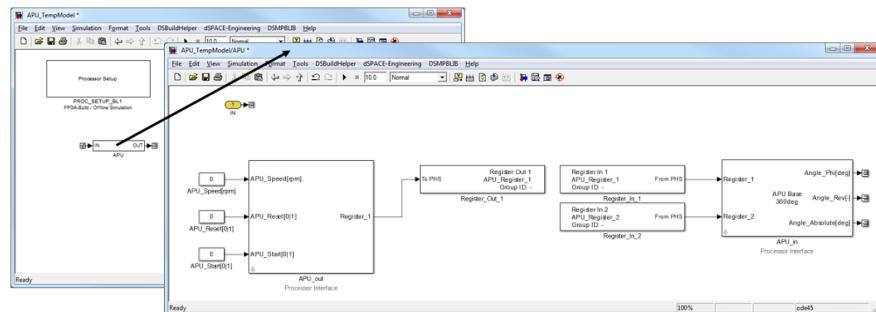


Figure 45: APU template model (processor view)

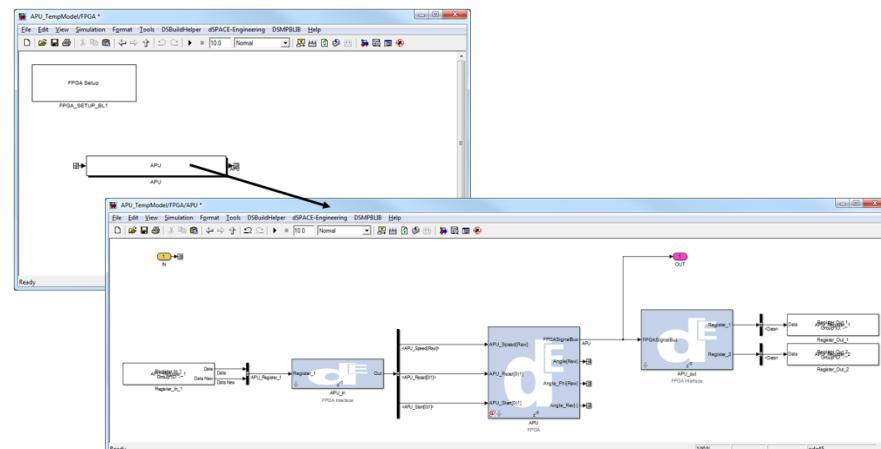


Figure 46: APU template model (FPGA view)

Buffer Access Blocks

Objective

See the following sections for information about the buffer access blocks. For each register signal routing a Goto and From block will be generated if the “Buffer Access” is enabled in the selection popup during an “Automatic Interface Generation” as shown in Figure 35.

Goto Block

Block

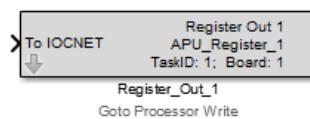


Figure 47: Goto block

Sent signals to From block that have the same parameterization as the Goto block. If the Goto or From block is not unique in the used model an error message will be generated. According to the GUI parameters of the block a unique Goto/From tag will be generated.

Block Dialog

The Goto block contains the following dialog.

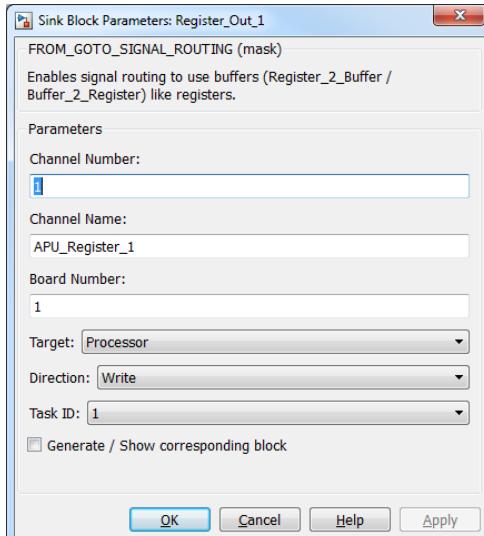


Figure 48: Goto dialog

The Goto block has the following parameters:

Name	Unit	Description	Range
Channel Number	-	Channel number of buffered access	1 - 9999
Channel Name	-	Channel name of buffered access	-
Board Number	-	Defined board number of used FPGA board	1 - 99
Target	-	Target definition; Block is located on processor side or on FPGA side of the model	-
Direction	-	Signal routing direction	-
TaskID	-	Definition of the TaskID	-
Generate / Show corresponding block	-	Generates or show the corresponding From block. If no block exists, the corresponding block will be generated next to the Goto block. If the corresponding block exists, the block will be highlighted.	-

Changes in the GUI parameters will direct change the parameters in the corresponding From block.

Input

The input of the Goto block differs from the GUI settings.

From Block

Block


Figure 49: From block

Block Dialog

The From block contains the following dialog.

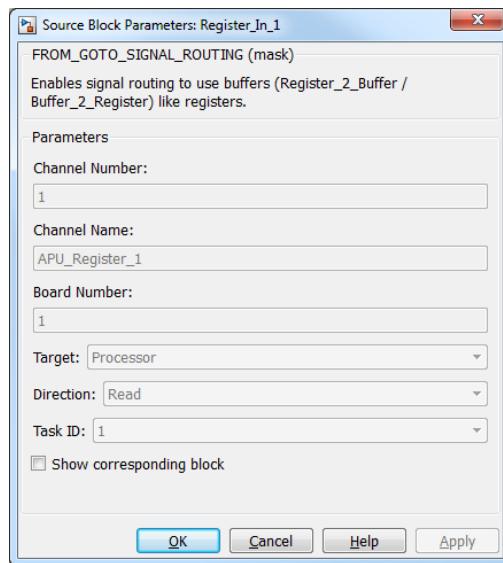


Figure 50: From dialog

The From block provides a short block description and shows the actual settings of the corresponding Goto block. These actual settings cannot be configured from the GUI of the From-Block! Only the corresponding Goto-Block can configure both settings. Additional to that the corresponding Goto-Block can be highlighted by clicking into the checkbox “Show corresponding block”.

Automatic Interface Generation

Objective

An automatic interface generation tool is available for quick, easy generation of an buffer access communication channel between the processor and the FPGA. The supported FPGA main components of the Buffer_2_Rgeister / Register_2_Buffer have a dialog tag as explained in the next section.

Communication with Buffer access

The following example gives you a better overview of the functionality of automatic interface generation. To show the specific workflow the APU example of the chapter “Quickstart” – subchapter “Automatic Interface Generation” is used.

Example: APU

During the automatic interface generation of the APU template model, the algorithm asked the definition of the Board number of the FPGA and the TaskID. These parameters are also implemented in the Goto/From blocks as marked in red in the Figure 51. The corresponding blocks are inserted in the subsystem “Buffer Interface” which is marked in blue.

In this example the communication of the TaskID 1 and the FPGA Board 1 must be realized over a buffer access.

To realize this communication, as shown in Figure 10, the additional blocks “Buffer_2_Register” and “Register_2_Buffer” blocks are necessary. Therefore insert the “Register_2_Buffer” block in the model as shown in the next figure.

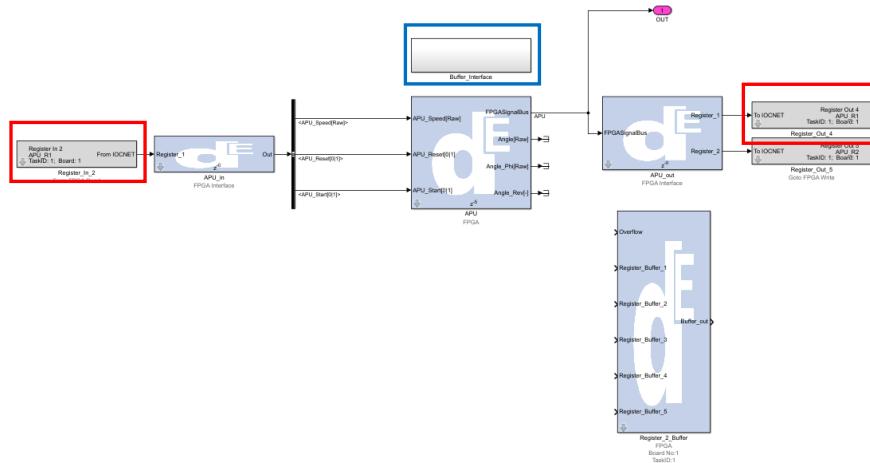


Figure 51: FPGA side: Inserted Register_2_Buffer block

Open the GUI of the Register_2_Buffer block as shown in the figure bellow.

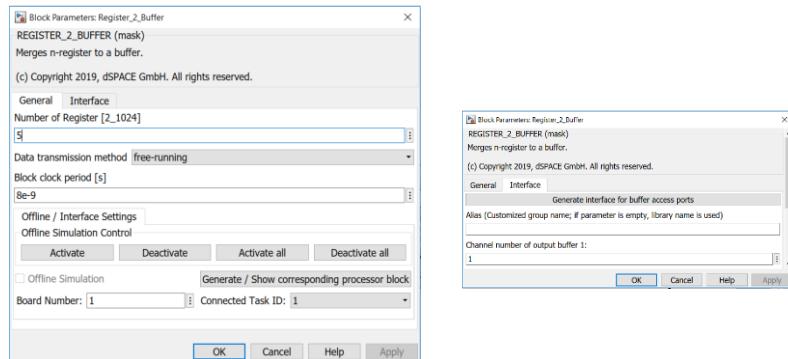


Figure 52: FPGA side: GUI of the Register_2_Buffer block

In this GUI the following settings can be defined:

Name	Unit	Description	Format
Board Number	[-]	Specify the Board number	-
Connected TaskID	[-]	Specify the TaskID	-
Generate interface for buffer access	[-]	Click to start the interface generation	-
Alias	[-]	Additional alias setting for interface generation	-
Channel number of output buffer	[-]	Specify the used buffer channel of rti	-

The automatic interface generation can be activated by clicking on the Generate Interface for buffer access button. The algorithm analyzes the model and search all specific “From” blocks with the following settings:

- Target: FPGA
- Direction: Write
- Task ID: 1
- Board Number: 1

During the interface generation, the well-known popups will be generated to control the template generation. The result will be finished with the following popup.

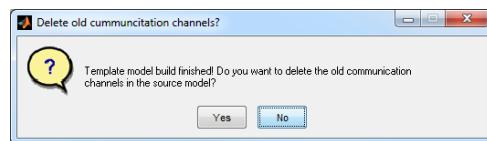


Figure 53: FPGA side: Popup for deleting the old buffer access ports

The old implemented buffer access ports can be deleted by script via clicking the yes button. A settings file (*.r2b file) is generated to control the interface generation for the processor side. At last step, the generated subsystem can be copied to the FPGA source model. The other communication direction can be generated in the same way.

To generate the processor side subsystem, drag and drop the “Register_2_Buffer” block from the library to the source model and open the GUI as shown in the figures below.

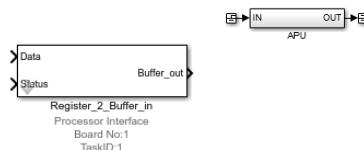


Figure 54: PROC side: Inserted Register_2_Buffer_in block

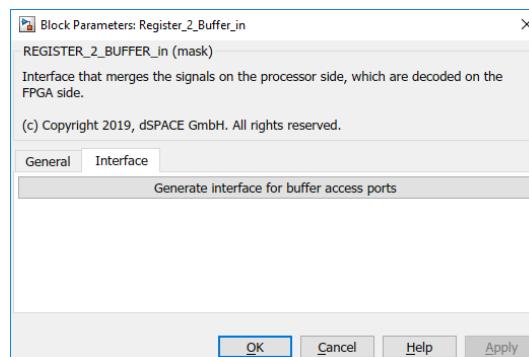


Figure 55: PROC side: GUI of the Register_2_Buffer_in block

The interface generation can be started directly by clicking the on the Generate interface for buffer access ports. After activating the generation, the algorithm asks for the description file (*.r2b file) which was generated from the corresponding FPGA block as shown in the figure bellow.

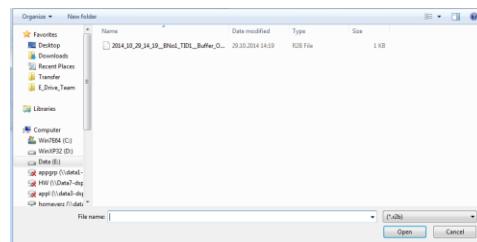


Figure 56: PROC side: GUI of the Register_2_Buffer_in block

Define the corresponding settings file and the interface generation will be finish with the popup as shown in the Figure 53. Finally copy the generated subsystem in the source model.

Switch between Online and Offline Simulation

Example: APU

The switch between Offline and Online simulation for Scalexio can be made by using “**Offline Simulation CTRL**” block in the FPGA library (Functions). Similar to PHS, FPGA and Processor blocks must be in the same model.

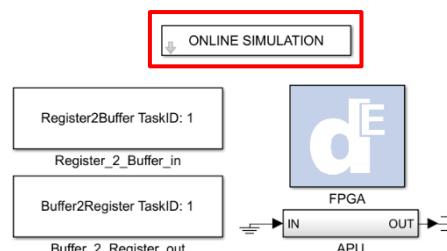


Figure 57: PROC and FPGA in the same model

Double-click the ONLINE SIMULATION block, the following GUI appears:
The GUI can be used to configure the buffer access blocks to offline mode by

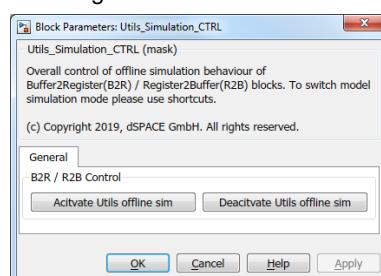


Figure 58: GUI of the Utils_Simulation_CTRL

selecting the “Activate Utils offline sim” button; to configure the buffer access blocks to online mode, select “Deactivate Utils offline sim”.

The model can be configured to online simulation with the shortcut combination Ctrl+Alt+F10 or select Prepare model for: PROC build from the Tools menu as shown below:

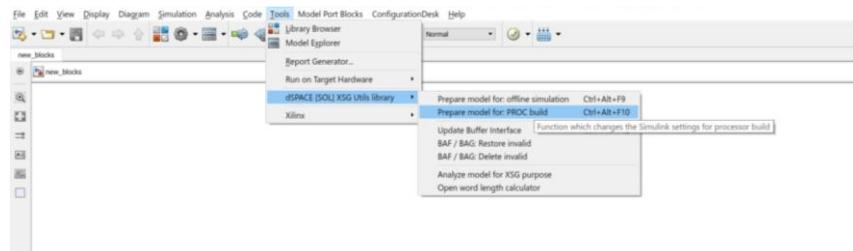


Figure 59: Menu bar option for Offline simulation in MATLAB/Simulink

To switch from Online to Offline Simulation, select the commented-out FPGA subsystem and use the shortcut combination Ctrl+Alt+F9 or select Prepare model for: offline simulation in tools menu.

Angular Processing Unit (APU)

Objective

See the following sections for information on the angular processing unit (APU), which provides angular position information which is used in core motor processing functions (such as Park-Clarke transformation or position sensor simulation).

The blockset contains the following elements:

- Processor Interface: APU_out (Processor Interface)
- FPGA Interface: APU_in (FPGA Interface)
- FPGA: APU (FPGA Main Component)
- FPGA Interface: APU_out (FPGA Interface)
- Processor Interface: APU_in (Processor Interface)

General behavior

Angular processing unit which integrates the applied speed and where the angle base can be configured either to 360 deg or to 720 deg.

Processor Output

Block

Merges the processor signals and writes them to the FPGA

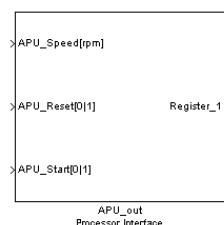


Figure 60: APU_out block

Block Dialog

The dialog provides only a short block description.

Input

The APU_out block has the following inputs:

Name	Unit	Description	Range
APU_Speed	[rpm]	Actual speed in rpm	(-732421 ... 732421) · 10E-9/ Ts_FPGA; resolution: 1.36E-3
APU_Reset	[-]	Reset on high state	0 1
APU_Start	[-]	Start on high state	0 1

Output

The Processor Out block provides one input register, whose sectioning is shown below:

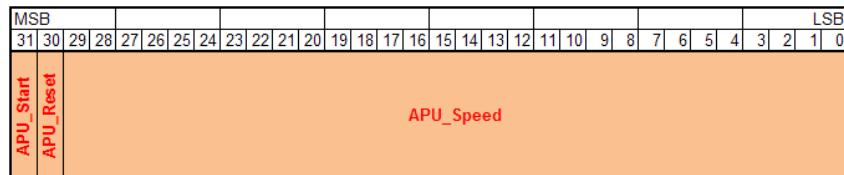
Register 1

Figure 61: APU_out Register 1

Name	Used Bits	Description	Range
APU_Speed	29 ... 0	Incremental step value based on a sample time of the FPGA	$0 \dots 2^{29} \hat{=} \text{positive speed}; 2^{29+1} \dots 2^{30} \hat{=} \text{negative speed (two's complement)}$
APU_Reset	30	Reset on high state	0 1
APU_Start	31	Start on high state	0 1

FPGA Main Component

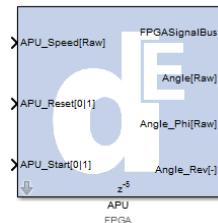
Block

Figure 62: APU FPGA Main block

Block Dialog

The FPGA main blockset contains the following dialog.

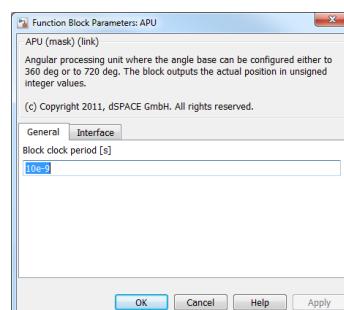


Figure 63: APU FPGA Main block dialog

	The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.
---	---

Input

The main block has the following inputs:

Name	Unit	Description	Format
APU_Speed	[Raw]	Actual speed	Fix_30_0
APU_Reset	[-]	Reset on high state	Bool
APU_Start	[-]	Start on high state	UFix_1_0

Output

The APU main block has the following outputs with a maximal latency of two:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the most significant signals of this block	-
Angle	[Raw]	Current angle 41 ... 0 Bit: Single turn bits 64 ... 42 Bit: Multi turn bits	UFix_64_0
Angle_Phi	[Raw]	Current single turn angle	UFix_42_0
Angle_Rev	[-]	Current multi turns of APU	Fix_22_0

Processor Input

Block

Adapts the FPGA signals for the processor.

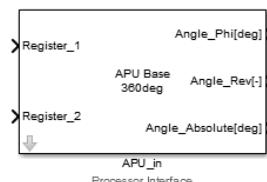


Figure 64: APU_in block

Block Dialog

The FPGA main blockset contains the following dialog.

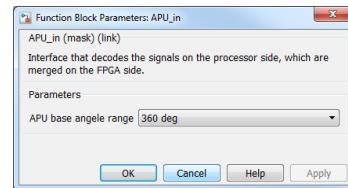


Figure 65: APU_in block dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
APU base angle range	[-]	Defines the APU base to calculates the output of the Angle Rev and Angle Phi; 360 deg (typ. electric motor): - Angle Phi 0 - 360 deg - Angle Rev counts the turns of 360 deg 720 deg (typ. combustion eng): - Angle Phi 0 - 720 deg - Angle Rev counts the turns of 720 deg	360 deg / 720 deg

Input

The processor input block set has the following inputs:

Register 1

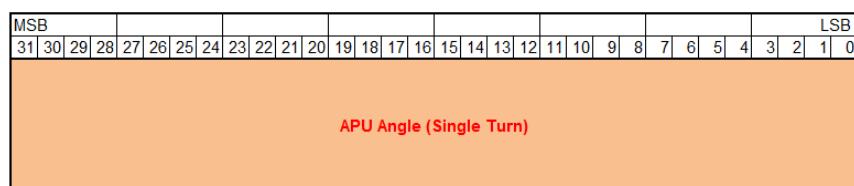


Figure 66: APU_in Register 1

Name	Used Bits	Description	Range
Angle	31 ... 0	Single turn bits of the APU	0 ... 2^32-1

Register 2

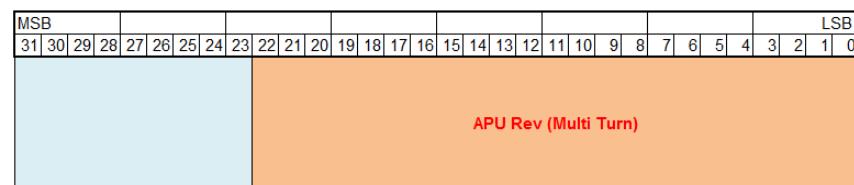


Figure 67: APU_in Register 1

Name	Used Bits	Description	Range
APU Rev	21 ... 0	Multi turn bits of the APU	0 ... 2^22-1

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Angle Phi	[deg]	actual angle (single turn)	Depends on APU Base: 360 deg:0° ... 360° 720 deg:0° ... 720°
Angle Rev	[]	Revolutions of turns	
Angle Absolute	[deg]	Absolute angle of APU	

Interface Examples

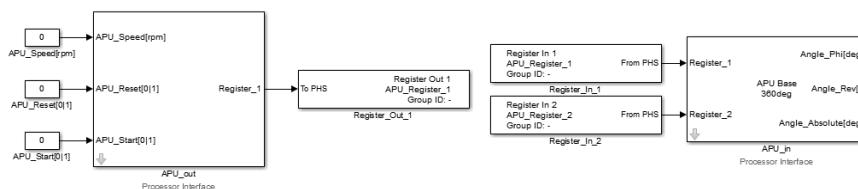
Processor blocks

Figure 68: Processor interface

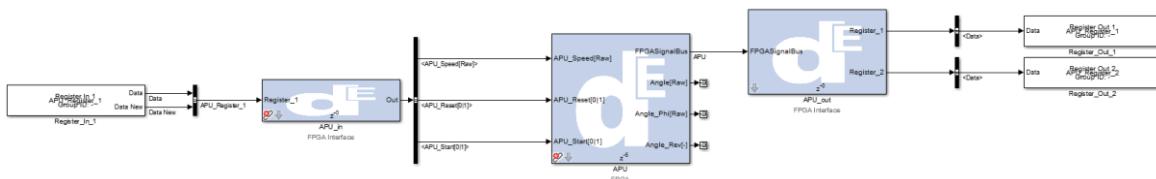
FPGA block

Figure 69: FPGA interface

APU Coupling over IOCNET

Objective	Angular Processing Units can be synchronized via IOCNET over several FPGA and IO-boards in the Scalexio System. The following two different applications are available <ul style="list-style-type: none"> ▪ IOCNET → XSG Utils APU Master APU is located on another FPGA or IO-Board. The XSG Utils APU is synchronized (Slave APU) ▪ XSG Utils APU → IOCNET Master APU is located on the actually FPGA board and realized with the XSG Utils library. This APU will be broadcast to the Scalexio System via IOCNET APU.
------------------	--

Both applications are described in detail in the following chapter.

IOCNET to XSG Utils APU

Objective	See the following sections for information if it is necessary to synchronize the APU with the IOCNET Master APU. Additional to that a gear ratio can be defined between the APU and the IOCNET Master APU. To realize an interface connection a special designed interface block can be used which is in the <ul style="list-style-type: none"> ▪ XSG_Utils_lib: XSG_Utils.lib/FPGA Interface/APU/IOCNET_APU_SLAVE_2_APU_in
------------------	--

Block

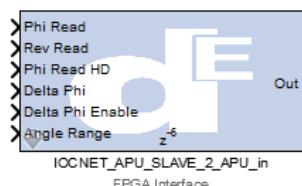


Figure 70: IOCNET_APU_SLAVE_2_APU_in block

Block Dialog	The FPGA interface block contains the following dialog.
---------------------	---

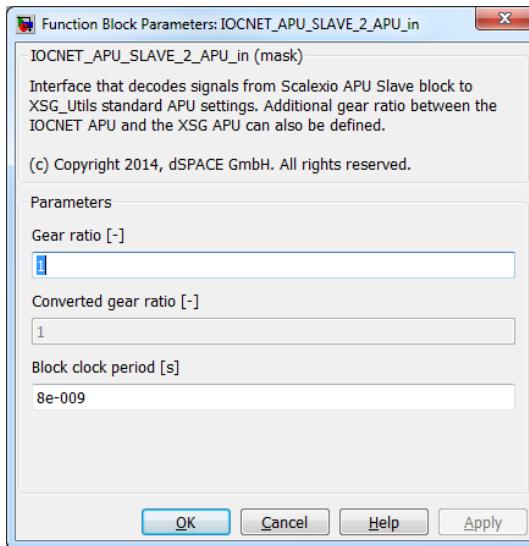


Figure 71: IOCNET_APU_SLAVE_2_APU_in block dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Format
Gear ratio	[-]	Gear ratio between the APU and the IOCNET APU	[0.03125 ... 32]
Block clock period	[s]	Block clock period of the FPGA board	-
Block clock period	[s]	Block clock period of the FPGA board	-

Input

The inputs of the IOCNET_APU_SLAVE_2_APU_in block represents the output values of the rtifpga block (IO_Read). For detailed information about the output ports settings please refer to the documentation of the rtifpga library. To ensure a right behavior of the interface block IOCNET_APU_SLAVE_2_APU_in, please configure the model as described in the next steps:

- Add a rtifpga IO_Read block to the model as shown in the figure below



Figure 72: rtifpga IO_READ block

- Double click on the added block and change on the GUI page "Unit" the settings to the used IOCNET APU (possible settings APU Slave 1 to APU Slave 6).
- Change the settings on the GUI page "Parameters" and activate the output ports by clicking in the checkbox:
 - o "Enable Phi Read HD port"

- “Enable Delta Phi port”
- “Enable Delta Phi Enable port”
- Accept all changes by clicking on the “OK” button.
- The block changes the output ports. Connect all output ports to the corresponding input port of the IOCNET_APU_SLAVE_2_APU_in block as shown in the figure bellow.

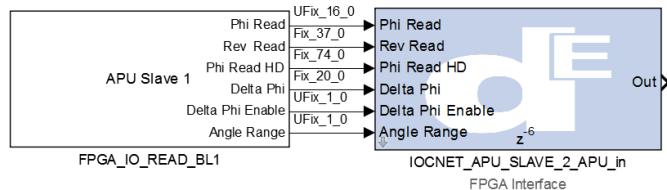
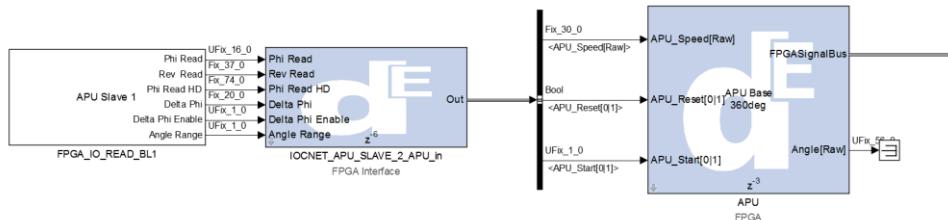


Figure 73: rtifpga IO_READ block connected with the IOCNET_APU_SLAVE_2_APU_in block

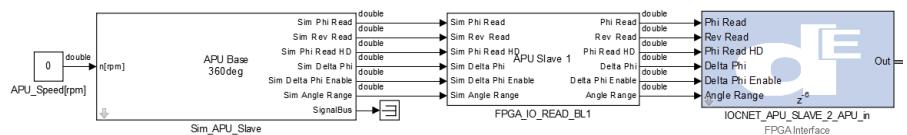
Output

The IOCNET_APU_SLAVE_2_APU_in provides the necessary input ports of the APU main block. All data is collected in a bus with the following parameters

Name	Unit	Description	Range
APU_Speed	[Raw]	Incremental step value based on a sample time of the FPGA	0...2^29 ≈ positive speed; 0 -2^29 ≈ negative speed
APU_Reset	[-]	Reset on high state	0 1
APU_Start	[-]	Start on high state	0 1
Ref_APU_FPAG_SignalBus	Bus	Settings bus of the internal reference APU which is synchronous to the IOCNET APU. The Bus can direct connected to an APU or APU_out block.	
Gear_APU_FPAG_SignalBus	Bus	Settings bus of the internal gear APU which is synchronous to the IOCNET APU multiplied with the defined gear ratio. The Bus can direct connected to an APU or APU_out block.	
IOCNET_APU_SLAVE	Bus	Parameters of the connected rtifpga block	
Debug_Port_z-2	Bus	Decoded signals to physical signals (deg / rpm) of all internal apu. Because internal decoding the output is delayed by additional two steps.	

Interface example**Figure 74: FPGA interface****Offline Simulation**

For Offline simulation of an IOCNET APU block an additional simulation block is inserted in the library which can be connected to the simulated input ports of the rtifpga block as shown in the figure below.

**Figure 75: Offline simulation of an IOCNET APU with rtifpga IO_READ block**

In the GUI of the APU simulation block the APU Base and the sample time of the FPGA and processor can be defined.

XSG Utils APU to IOCNET**Objective**

See the following sections for information if it is necessary to broadcast the XSG Utils APU with the IOCNET Slave APU. To realize an interface connection a special designed interface block can be used which is in the

- **XSG_Utils_lib:**
`XSG_Utils_lib/FPGA Interface/APU/APU_2_IOCNET_APU_MASTER_out`

Block

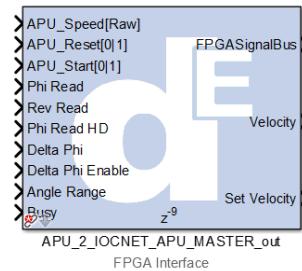


Figure 76: APU_2_IOCNET_APU_MASTER_out block

Block Dialog

The FPGA interface block contains the following dialog.

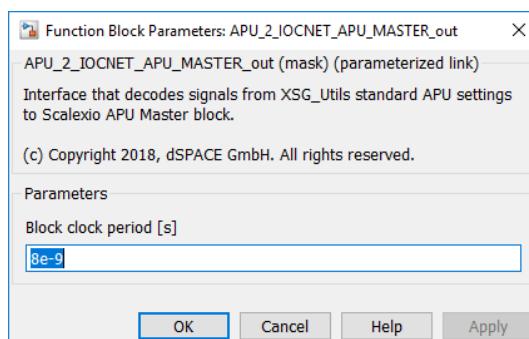


Figure 77: APU_2_IOCNET_APU_MASTER_out block dialog

Input

The APU_2_IOCNET_APU_MASTER_out provides the necessary input ports of the APU main block and the feedback signals of the IOCNET_APU_MASTER block.

Name	Unit	Description	Range
APU_Speed	[Raw]	Incremental step value based on a sample time of the FPGA	0...2^29 ≈ positive speed; 0 -2^29 ≈ negative speed
APU_Reset	[-]	Reset on high state	0 1
APU_Start	[-]	Start on high state	0 1
Phi Read	[-]	Output port of the IOCNET_APU_MASTER block	
Rev Read	[-]		
Phi Read HD	[-]		
Delta Phi	[-]		
Enable	[-]		
Angle Range	[-]		
Busy	[-]		

Output

The outputs of the APU_2_IOCNET_APU_MASTER_out block controls and synchronize the internal APU of the IOCNET_APU_MASTER block of the rtifpga

block (IO_Write). For detailed information about the output ports settings please refer to the documentation of the rtifpga library. To ensure a right behavior of the interface block APU_2_IOCNET_APU_MASTER_out, please configure the model as described in the next steps:

- Add a rtifpga IO_Write block to the model as shown in the figure below

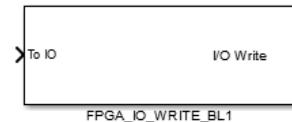


Figure 78: rtifpga IO_WRITE block

- Double click on the added block and change on the GUI page "Unit" the settings to the used IOCNET APU (possible settings APU Master 1 to APU Master 6).
- Change the settings on the GUI page "Parameters" and activate the output ports by clicking in the checkbox:
 - o "Enable advanced ports"
- Accept all changes by clicking on the "OK" button.
- The block changes the output ports. Connect all output ports to the corresponding in- and output ports of the APU_2_IOCNET_APU_MASTER_out block as shown in the figure bellow.

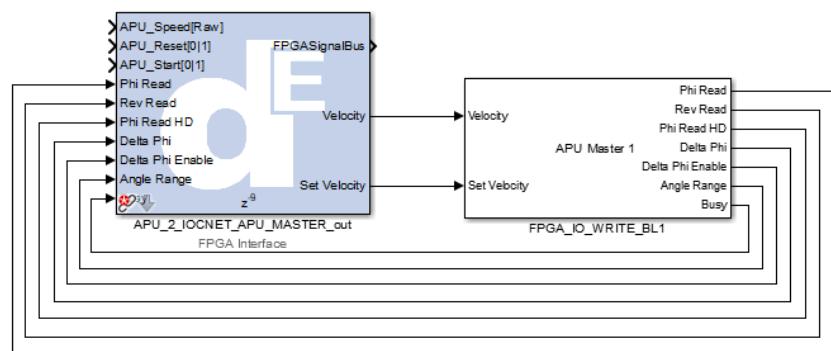


Figure 79: rtifpga IO_WRITE block connected with the APU_2_IOCNET_APU_MASTER_out block

- Connect the input ports "APU_x" to the corresponding XSG Utils APU, as shown in the "Interface example" below.

Interface example

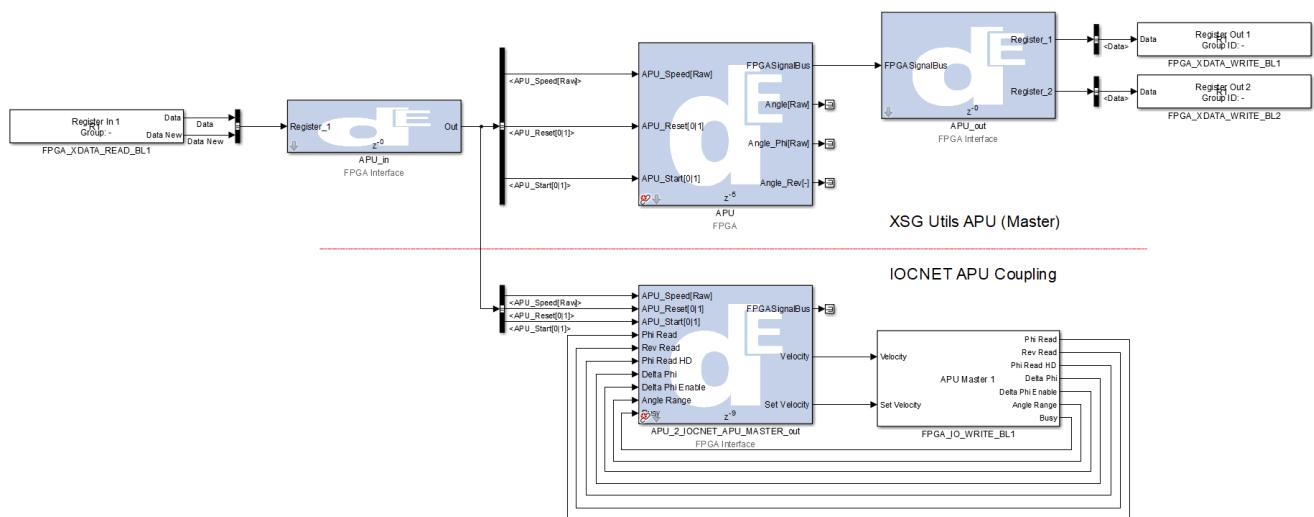


Figure 80: FPGA interface

Wave Table Encoder

Objective

Freely designable output shape format, digital or analog, over either 360 deg or 720 deg angle base format. For repeating a certain shape format n times for an angle base format, the number of lines has to be set to n.



The both processor interface blocks have to be located inside the same subsystem and the StatusBus and ENCODER_Access ports have to be connected with the corresponding one.

The blockset contains the following elements:

- Processor Interface: WAVE_TABLE_ENCODER_out (Processor Interface)
- FPGA Interface: WAVE_TABLE_ENCODER_in (FPGA Interface)
- FPGA: WAVE_TABLE_ENCODER (FPGA Main Component)
- FPGA Interface: WAVE_TABLE_ENCODER_out (FPGA Interface)
- Processor Interface: WAVE_TABLE_ENCODER_in (Processor Interface)



Please note that the Library is optimized for timing, resource consumption and hardware interface for the Virtex5 and Kintex7 FPGAs. If other FPGAs are used, relax timing constrain via the downsampling block as described in chapter DOWNSAMPLE.

Processor Output

Block

Merges the processor signals and writes them to the FPGA

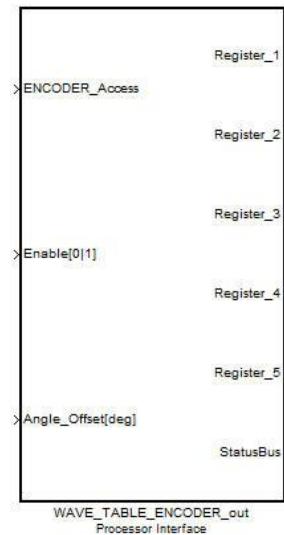


Figure 81: WAVE_TABLE_ENCODER_out block

Block Dialog

The processor output blockset contains the following dialog.

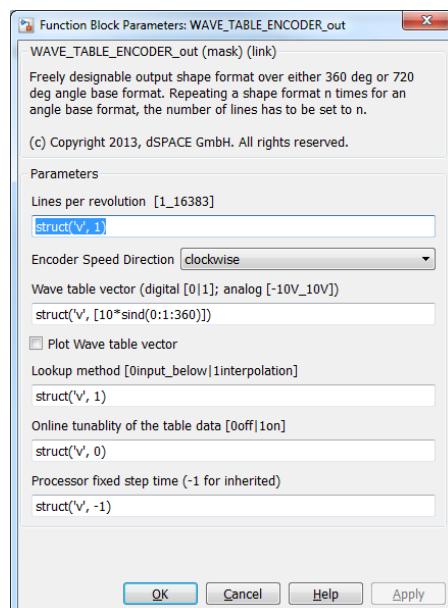


Figure 82: WAVE_TABLE_ENCODER_out GUI

Input

The WAVE_TABLE_ENCODER_out block has the following inputs:

Name	Unit	Description	Range
Encoder_Access	Bus	Feedback information of the FPGA status	-
Enable	[-]	Enables the encoder	0 1
Angle_Offset	[deg]	Mechanical position offset	0 1

Output

The processor out block has got five output registers which provide the functionality of a simple protocol, together with the processor input register. Because of the complex visualization, no register overview is shown here. For detailed information about the wavetable feedback of the port "StatusBus", please refer the chapter "Data store management and feedback of Look-up Table" in the subchapter "LUT Feedback".

FPGA Main Component

Block

Figure 83: WAVE_TABLE_ENCODER FPGA Main block

Block Dialog

The FPGA main blockset contains the following dialog.

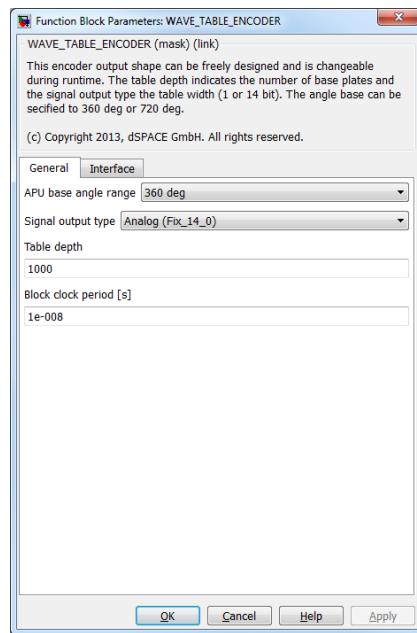


Figure 84: WAVE_TABLE_ENCODER FPGA Main block dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Format
APU base angle range	[-]	Angle range of the APU 360 and 720 degrees can be configured	-
Signal out type	[-]	Digital 1 bit Analog 14 bit	UFix_1_0 Fix_14_0
Table depth	[-]	Number of table points that which maximal can be stored in the table	UFix_16_0



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
APU_Speed	[Raw]	Actual speed	Fix_30_0
APU_Reset	[-]	Reset on high state	Bool
APU_Start	[-]	Start on high state	UFix_1_0
Enable	[-]	Enables the output of the encoder	Bool
No_Of_Lines	[-]	Number of pole pairs	UFix_16_0
Speed_Direction	[-]	Configure the speed direction (clockwise / counter-clockwise)	UFix_1_0
Angle_Offset	[Raw]	Mechanical position offset	UFix_32_0
Table_Data_A	[-]	Table data port A for parameterization of the LUT	UFix_24_0
Table_Data_B	[-]	Table data port B for parameterization of the LUT	UFix_24_0
Address	[-]	LUT table address for parameterization	Ufix_15_0
Offset Address	[-]	LUT table offset address for parameterization	UFix_15_0
New_LUT_Data	[-]	Data flag is set to 1 when new table data is available to write to the FPGA memory	UFix_1_0
Lookup_Method	[-]	0: input below 1:interpolation	0 1
Proc.Sync. Impulse	[-]	Processor-synchronous toggle bit	UFix_1_0
Inv_Step_Width_X	[-]	Inverse step width of the input vector which is configured in the processor out block	UFix_30_0
Config_Vector	[-]	Configuration of the LUT output timing	-

Output

The APU main block has the following outputs with a maximal latency of two:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the most significant signals of this block	-
Wave_Table_Enc	[-]	Digital wave table output	UFix_1_0
Wave_Table_Enc	[-]	Analog wave table output	Fix_14_0
En	[-]	Enable flag	UFix_1_0

Processor Input

Block

Adapts the FPGA signals for the processor.

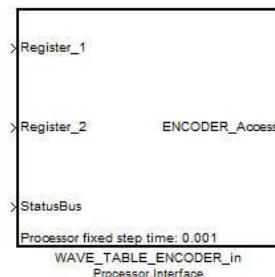


Figure 85: WAVE_TABLE_ENCODER_in block

Block Dialog

The dialog provides only a short block description.

Input

The processor in block has got two intput registers which provide the functionality of a simple protocol, together with the processor output registers. Because of the complex visualization, no register overview is shown here

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Angle	[deg]	actual angle	0° ... 360°

Interface Examples

Processor blocks

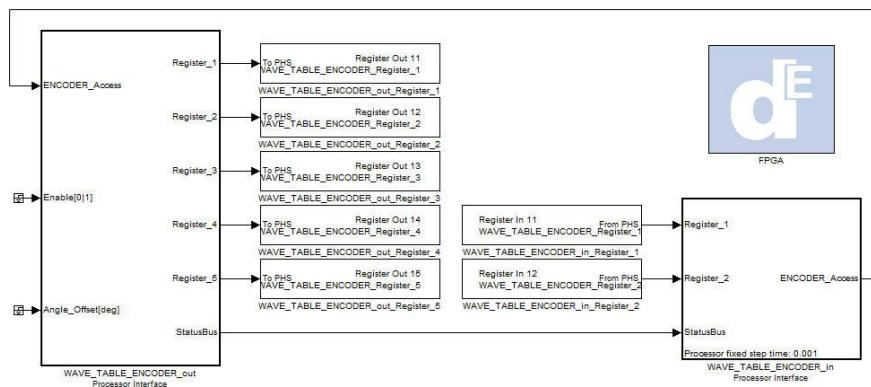


Figure 86: Processor interface

FPGA block

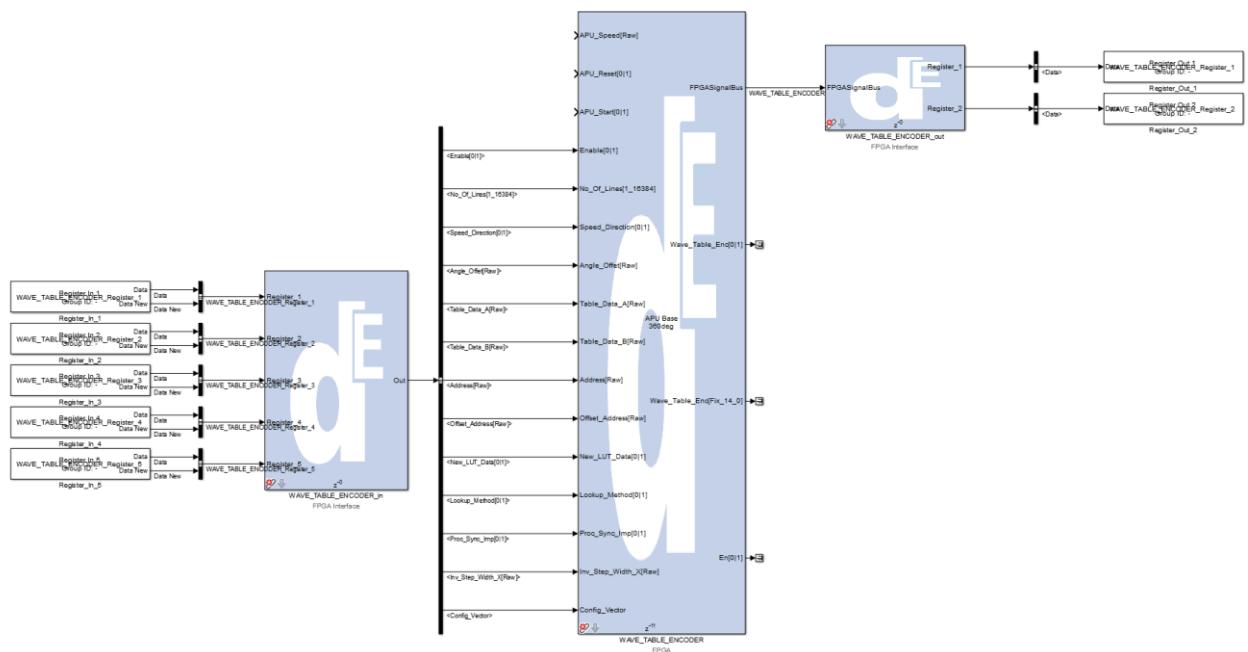


Figure 87:FPGA interface

PWM measurement

Objective

The PWM measurement blockset is able to measure the dead time (between HSD and LSD), the high time, and the period time of a single- or three-phase signal. The calculation of corresponding duty cycle(s), gate active time(s), period time(s) and dead time(s) is realized on the processor side. A pulse center-synchronous or dead time violation interrupt (one for each phase) can also be raised on the FPGA and caught on the processor.



The PWM measurement for a one phase system is similar to the three phase system. The following chapter describes only the behavior of the three-phase component.

Gate Time and Period Measurement

The XSG Utils PWM unit uses timers to calculate the high times. One of the main features is to calculate the duty cycles of the connected PWM signals. The calculated duty cycles are obtained by dividing the gate active time by the delta interrupt time. (Delta interrupt time is the time between two latch points.)

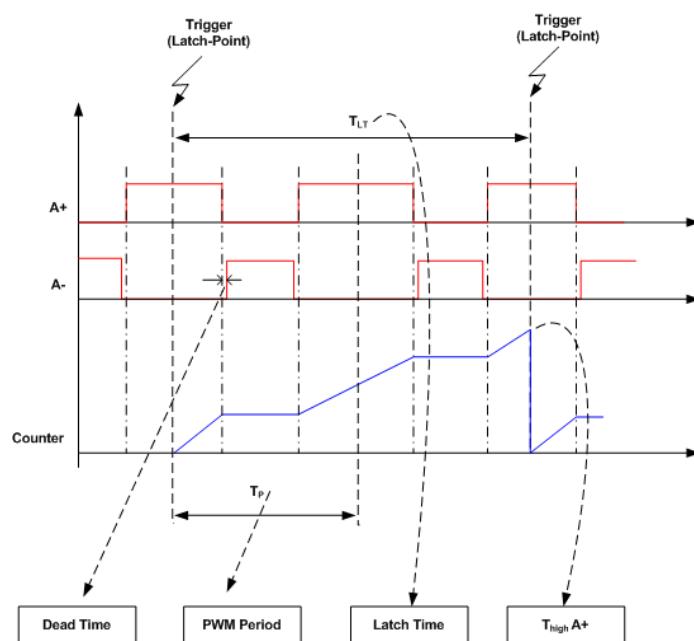


Figure 88: Measurement principle

Internal Center Impulse Generation

The XSG Utils Solution PWM unit calculates the centers of the PWM signals by evaluating the on times and off times. The FPGA is only able to calculate backwards, it has to approximate the next PWM center. You can select the

channels used to estimate the center position and accordingly generate the trigger. For maximum accuracy, use all gate signals.



For the PWM period to be calculated correctly, the maximum and minimum PWM periods / frequencies have to be defined.

Additional PWM Features

Latch Mode Selection

Three latch modes for synchronizing the gate time measurement are available: "internal center", "processor model" and "external".

In case of "internal center", the internally calculated common center (either pulse center or pause center) of the 6 gate signals is used as the latch trigger. By up- or downsampling, the measurement (latch) period can be higher or lower than the actual PWM period. In this case, the measured gate time scales this the latch period. The duty cycles are calculated as gate high times divided by latch time.

In case of "processor model", the latch trigger and period is determined by the call of the processor output block. There is no direct link between PWM period and latch period. Over- or downsampling is possible, but does not really make sense.

In case of "external", an additional digital input is used as trigger source for the latch time. This input can for example be the center impulse of another PWM measurement. If the external trigger is PWM-synchronous, over- or downsampling is applicable.

Debounce Time

The inputs are debounced in the software to avoid incorrect edge detection. This requires that the input signal is stable for the debounce time before it is detected. For this reason, gate times smaller than the debounce time cannot be detected (for on times smaller than the debounce time, the output value is the debounce time). The debounce time is the same for rising and falling edges.

Dead-Time Measurement

In power electronics applications, each inverter leg (two series switches) is controlled with signals (+) and (-). To avoid short circuits, these signals are not allowed to be high at the same time. In addition, a dead time is added between them to avoid short circuits during commutation. Dead-time measurement is performed between two adjoining gates (a+/a-, b+/b-, c+/c-).

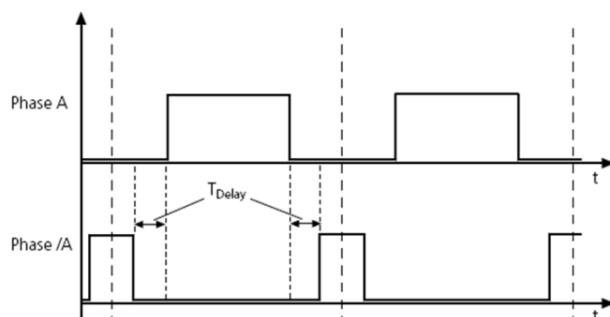


Figure 89: Dead time between the signals of two gates

Dead-Time Interrupt Generation The XSG Utils Solution is able to generate an interrupt if the measured dead time is less than a predefined value. This lets you check whether your controller generates correct control signals for your inverter and does not damage your power electronics.

Oversampling The gate time measurement can be triggered 1 to 8 times per PWM period or the external interrupt signal. In downsampling mode the interrupt triggers once each 1 to 8 PWM periods.

Default Period Using the internal latch mode, the PWM measurement generates center impulses with a pre-defined period until the measurement logic is synchronized to the PWM signal. After measuring valid PWM periods, center impulses will be generated with the last period measured in case of a measurement timeout. The default impulse generation can be disabled if desired.

The blockset contains the following elements:

- Processor Interface: THREE_PHASE_PWM_MEASURMENT_out
(Processor Interface)
- FPGA Interface: THREE_PHASE_PWM_MEASURMENT_in
(FPGA Interface)
- FPGA: THREE_PHASE_PWM_MEASURMENT
(FPGA Main Component)
- FPGA Interface: THREE_PHASE_PWM_MEASURMENT_out
(FPGA Interface)
- Processor Interface: THREE_PHASE_PWM_MEASURMENT_in
(Processor Interface)

**Import information:
PWM Pattern Symmetry** The THREE_PHASE_PWM_MEASUREMENT blockset synchronization mechanism requires a center-symmetric PWM signal to achieve best measurement results.

For measuring asymmetric PWM patterns, e.g. caused by additional duty cycle updates by the connected ECU during the pulse period, an external synchronization pulse is required for proper synchronization.
Hence, consider usage of an external signal as latch source.

	Using the internal synchronization mechanisms with an asymmetric PWM pattern will result in: <ul style="list-style-type: none"> • Latch interrupt jitter • Sporadic loss of interrupts due to resynchronization attempts of the internal synchronization mechanism
	Depending on the application, model instability and oscillations can occur.
	For asymmetric PWM patterns, usage of external signal as latch source is recommended.

Import information:
Overmodulation Behavior

During overmodulation the PWM measurement will lose synchronization due to the lack of switching events. In this case the PWM interrupts are generated with the last calculated PWM latch period. When switching events reoccur, the internal synchronization mechanism will attempt to resynchronize. During resynchronization, it is possible that interrupts are lost. Typically, a single interrupt is lost, however if you are using oversampling, the number of lost interrupts corresponds to the oversampling factor used. E.g. when you are using 2-times oversampling, you will used 2 adjacent interrupts during resynchronization.

	Depending on the application, model instability and oscillations can occur, during resynchronization.
	Consider usage of the 3-phase/ 6 channel latch option to avoid loss of synchronization as far as possible.
	In case that loss of interrupt(s) and accompanying transition effects needs to be avoided, consider usage of an external signal as latch source.

Import information:
Behavior at variable PWM Frequency

If the PWM frequency is dynamically changed, the PWM measurement will lose synchronization due to the sudden change of the PWM period. In this case the PWM interrupts are generated with the last calculated PWM latch period. The internal synchronization mechanism will also attempt to resynchronize. During resynchronization, it is possible that interrupts are lost. Typically, a single interrupt is lost, however if you are using oversampling, the number of lost interrupts corresponds to the oversampling factor used. E.g. when you are using 2-times oversampling, you will used 2 adjacent interrupts during resynchronization.

	Depending on the application, model instability and oscillations can occur, during resynchronization.
	Make sure to set the PWM Min and Max period settings of the PWM measurement setup block in accordance to the used PWM frequency range.
	In case that loss of interrupt(s) and accompanying transition effects needs to be avoided, consider usage of an external signal as latch source.

Processor Output

Block

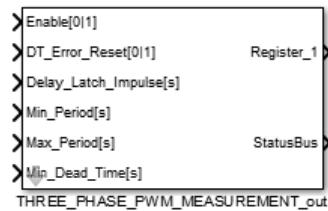


Figure 90: THREE_PHASE_PWM_MEASUREMENT_out block

Block Dialog

The processor output blockset contains the following dialog.

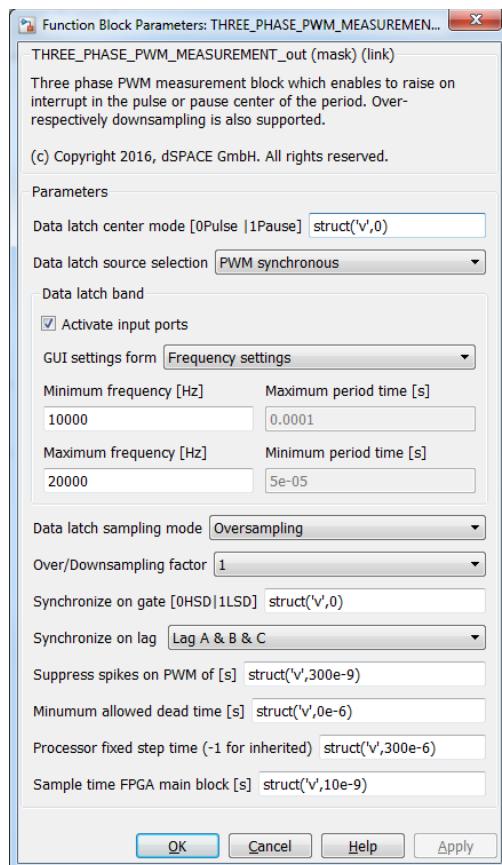


Figure 91: THREE_PHASE_PWM_MEASUREMENT_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Data latch center mode	[-]	If this option is set to 1, the center impulse will be generated at the calculated pause center of the PWM gate signals. If it is set to 0 the impulse will be generated at the calculated pulse center.	0 1
Data latch source selection	[-]	Selects the source for the data latch. The following settings are available: PWM synchronous Processor synchronous External (triggered from external input)	-
Activate input ports	[-]	Activates input ports for the maximum and minimum period and the minimum dead time. If the input ports are activated, the GUI settings are bypassed.	-
GUI setting form		Configures the GUI settings form of the maximum and minimum period time for the data latch band. If the form is set to frequency, the period fields are disabled and the actually period time is calculated corresponding to the frequency setting. If the form is set to period time, the frequency fields are disabled and the actually frequency is calculated corresponding to the period time setting.	-
Minimum Frequency	[Hz]	Minimum frequency of the data latch band	
Maximum Frequency	[Hz]	Maximum frequency of the data latch band	
Minimum Period Time	[s]	Minimum period time of the data latch band	
Maximum Period Time	[s]	Maximum period time of the data latch band	
Data latch sampling mode	[-]	The measurement of gate signals can be downsampled or oversampled	-
Over-/down-sampling factor	[-]	Factor of over- or downsampling	1 ... 8

Synchronize on gate	[-]	Set which the type of the lag which shall be used for the center calculation to generate the PWM center pulse	0 1
Synchronize on lag	[-]	Set which lags shall be used for the center calculation to generate the PWM center pulse.	-
Suppress spikes on PWM	[-]	Suppress spikes of a bouncing digital signal	
Minimum allowed dead time	[s]	The minimum valid dead time between the high side and low side switch. If the measured dead time falls below the minimum dead time, an interrupt is triggered, the dead time error flag is set and the dead time measurement is paused. The measurement resumes after the error is reset by the user. If the minimum dead time is set to 0, no error detection will be performed.	
Processor fixed step time	[s]	The sample time at which the processor blocks are to be simulated.	-
Sample time FPGA	[s]	Sample time of the FPGA	-

Input

The processor out block has the following inputs:

Name	Unit	Description	Range
Enable	[-]	Enables the PWM measurement	0 1
DT_Error_Reset	[-]	Resets error flags and resumes dead time measurement	0 1
Delay_Latch_Impulse	[s]	Additional delay of the latch impulse	
Min_Period	[s]	The minimum expected PWM input period. If a period is measured which is lower than this input, the measurement will be declared invalid and the period output will not be updated.	
Max_Period	[s]	The maximum expected PWM input period. If a period is measured which is higher than this input, the measurement will be declared invalid and the period output will not be updated.	
Min_Dead_Time	[s]	The minimum valid dead time between the high side and low side switch. If the measured dead time falls below the minimum dead time, an interrupt is triggered, the dead time error flag is set and the dead time measurement is paused. The measurement resumes after the error is reset by the user. If the minimum dead time is set to 0, no error detection will be performed.	

Output

The processor out block provides one output register which is multiplexed to transfer all input data. The sectioning is shown below:

Register 1; ID = 0

Figure 92: THREE_PHASE_PWM_MEASUREMENT_out Register 1; ID = 0

Name	Used Bits	Description	Range
Counter ID	3 ... 0	ID to select which register coding is activated via Processor	0 ... 4
Enable	4		0 1
DT_Reset_Overrun	5	Reset flag to a Deadtime overrun	0 1
Proc. Sync. Impulse	6	Processor synchronous toggle bit	0 1
Maximum Latch Period	26 ... 7		
Sampling Factor	29 ... 27		
Sampling Mode	30		
Sync on Gates	31		

Register 1; ID = 1



Figure 93: THREE_PHASE_PWM_MEASUREMENT_out Register 1; ID = 1

Name	Used Bits	Description	Range
Continuous Data	6 ... 0	Definition same as ID = 0	
Minimum Latch Period	26 ... 7		
Sync on Lag	29 ... 27		
Latch Source	31 ... 30		

Register 1; ID = 2

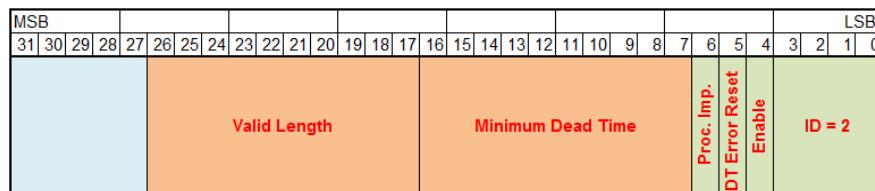


Figure 94: THREE_PHASE_PWM_MEASUREMENT_out Register 1; ID = 2

Name	Used Bits	Description	Range
Continuous Data	6 ... 0	Definition same as ID = 0	
Minimum Dead Time	16 ... 7		
Valid Length	26 ... 17		
SPARE	31 ... 27		

Register 1; ID = 3

Figure 95: THREE_PHASE_PWM_MEASUREMENT_out Register 1; ID = 3

Name	Used Bits	Description	Range
Continuous Data	6 ... 0	Definition same as ID = 0	
Delay Latch Impulse	24 ... 7		
Latch Mode	25		
SPARE	31 ... 26		

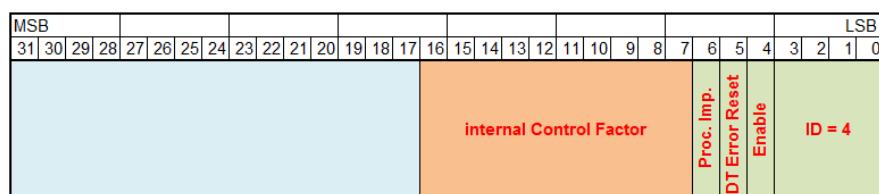
Register 1; ID = 4

Figure 96: THREE_PHASE_PWM_MEASUREMENT_out Register 1; ID = 4

Name	Used Bits	Description	Range
Continuous Data	6 ... 0	Definition same as ID = 0	
Control Factor	16 ... 7	Internal control factor	
SPARE	31 ... 17		

FPGA Main Component

Block

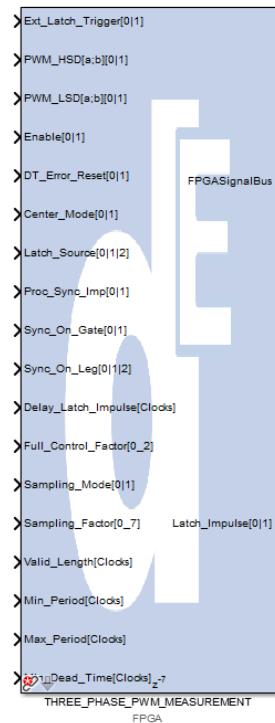


Figure 97: THREE_PHASE_PWM_MEASUREMENT Main block

Block Dialog

The FPGA main blockset contains the following dialog.

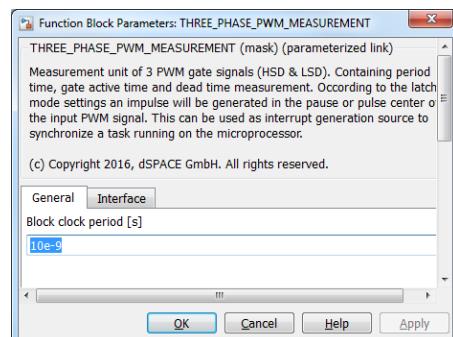


Figure 98: THREE_PHASE_PWM_MEASUREMENT FPGA Main block dialog



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Ext_Latch_Trigger	[-]	External latch trigger for gate time measurement	Bool or UFix_1_0
PWM_HSD[a;b;c]	[-]	High side gate signals	Bool or UFix_1_0
PWM_LSD[a;b;c]	[-]	Low side gate signals	Bool or UFix_1_0
Enable	[-]	Enables the PWM measurement (high-active)	Bool
DT_Error_Reset	[-]	Resets dead time errors and resumes dead time measurement.	Bool
Center_Mode	[-]		UFix_1_0
Latch_Sources	[-]		UFix_2_0
Proc_Sync_Imp	[-]		UFix_1_0
Sync_On_Gate	[-]		UFix_1_0
Sync_On_Leg	[-]		UFix_3_0
Delay_Latch_Impulse	[Clocks]		UFix_18_0
Full_Control_Factor	[-]		UFix_10_9
Sampling_Mode	[-]		UFix_1_0
Sampling_Factor	[-]		UFix_3_0
Valid_Length	[Clocks]		UFix_10_0
Min_Period	[Clocks]		UFix_20_0
Max_Period	[Clocks]		UFix_20_0
Min_Dead_Time	[Clocks]		UFix_10_0

Output

The THREE_PHASE_PWM_MEASUREMENT main block has the following outputs:

Name	Unit	Description	Format
FPGASignalBus	[Bus]	Bus containing the block's most significant signals	-
Latch_Impulse	[-]		Bool



An interrupt is raised only if an interrupt block is connected from the RTIFPGA lib to one or both interrupt signals

Processor Input**Block**

Adapts the FPGA signals for the processor side.

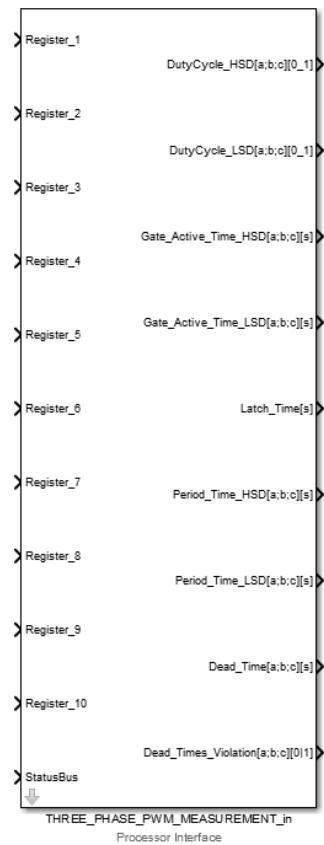


Figure 99: THREE_PHASE_PWM_MEASUREMENT_in block

Block Dialog

The dialog provides only a short block description

Input

The processor input block has the following inputs:

Register 1



Figure 100: THREE PHASE PWM MEASUREMENT in Register 1

Name	Used Bits	Description	Range
Latch Time	17 ... 0	Latch time scaled to the FPGA sample time	0 ... $2^{18}-1$
Full Control	23 ... 18		0 ... 2^6-1
Center Source	25 ... 24		0 ... 1
SPARE	31 ... 26		

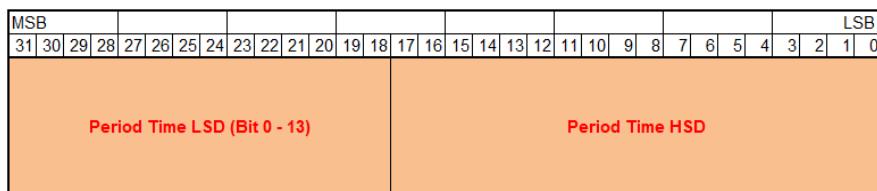
Register 2/5/8

Figure 101: THREE_PHASE_PWM_MEASUREMENT_in Register 2

Name	Used Bits	Description	Range
Period_Time_HSD_x	17 ... 0	Period time of HSD leg x scaled to the FPGA sample time	0 ... $2^{18}-1$
Period_Time_LSD_x (Bit 0-13)	31 ... 18	Period time of LSD leg x scaled to the FPGA sample time (Bit 0-13)	0 ... $2^{14}-1$

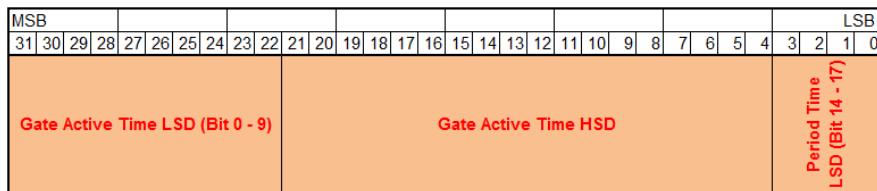
Register 3/6/9

Figure 102: THREE_PHASE_PWM_MEASUREMENT_in Register 3

Name	Used Bits	Description	Range
Period_Time_LSD_x (Bit 14-17)	3 ... 0	Period time of LSD leg x scaled to the FPGA sample time (Bit 14-17)	0 ... 2^4-1
Gate_Active_Time_HSD	21 ... 4	Gate active time HSD leg x scaled to the FPGA sample time	0 ... $2^{18}-1$
Gate_Active_Time_LSD (Bit 0-9)	31 ... 22	Gate active time LSD leg x scaled to the FPGA sample time (Bit 0-9)	0 ... $2^{10}-1$

Register 4/7/10

Figure 103: THREE_PHASE_PWM_MEASUREMENT_in Register 4

Name	Used Bits	Description	Range
Gate_Active_Time_LSD (Bit 10-17)	7 ... 0	Gate active time LSD leg x scaled to the FPGA sample time (Bit 10-17)	0 ... 2^8 -1
Dead_Time	25 ... 8	Deadtime of leg x scaled to the FPGA sample time	0 ... 2^{18} -1
Dead_Time_Violation	26	Feedback if a deadtime violation is detected	0 1
SPARE	31 ... 27		

Output

The processor out block has the following outputs:

Name	Unit	Description	Range
Duty_Cycles_HSD	[‐]	The duty cycles of a+, b+, c+	0 ... 1
Duty_Cycles_LSD	[‐]	The duty cycles of a-, b-, c-	0 ... 1
Gate_Active_Time_HSD	[s]	The gate active times of a+, b+, c+	(0 ... 2^{18} -1) * Ts_FPGA
Gate_Active_Time_LSD	[s]	The gate active times of a-, b-, c-	(0 ... 2^{18} -1) * Ts_FPGA
Latch Time	[s]	The time base for the gate time measurement	(0 ... 2^{18} -1) * Ts_FPGA
Period_Time_HSD	[s]	The period times of a+, b+, c+	(0 ... 2^{18} -1) * Ts_FPGA
Period_Time_LSD	[s]	The period times of a-, b-, c-	(0 ... 2^{18} -1) * Ts_FPGA
Dead_Times	[s]	The measures dead times of the gates a, b, c.	(0 ... 2^{18} -1) * Ts_FPGA
Dead_Time_Violation	[‐]	Feedback if the deadtime is violated of a, b, c	0 1

Interface Examples**Processor blocks**

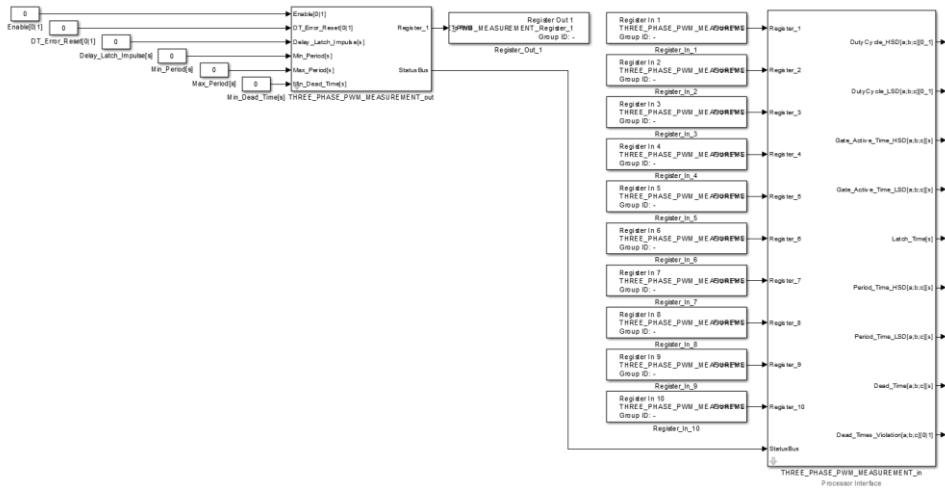


Figure 104: Processor interface

FPGA blocks

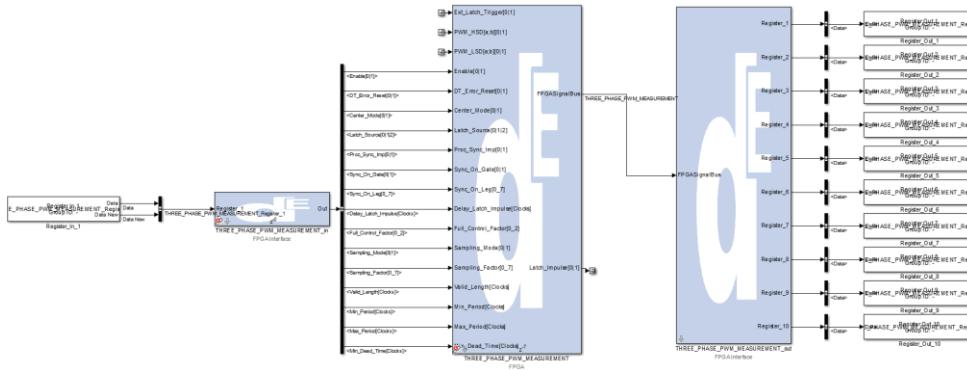


Figure 105: FPGA interface

Counter PWM

Objective

The Counter PWM blockset is able to measure the high or low time of a digital signal in FPGA clocks. The calculation of corresponding time(s) is realized on the processor side. An external or internal latch impulse can be used. Use this function, if a PWM signal with no dead time check, duty cycle calculation, e.g. is necessary to minimize the hardware resource on the FPGA.

The blockset contains the following elements:

- Processor Interface: COUNTER_PWM_out
(Processor Interface)
- FPGA Interface: COUNTER_PWM_in
(FPGA Interface)
- FPGA: COUNTER_PWM
(FPGA Main Component)
- FPGA Interface: COUNTER_PWM_out
(FPGA Interface)
- Processor Interface: COUNTER_PWM_in
(Processor Interface)

Processor Output

Block

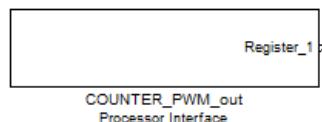


Figure 106: COUNTER_PWM_out block

Block Dialog

The processor output blockset contains the following dialog.

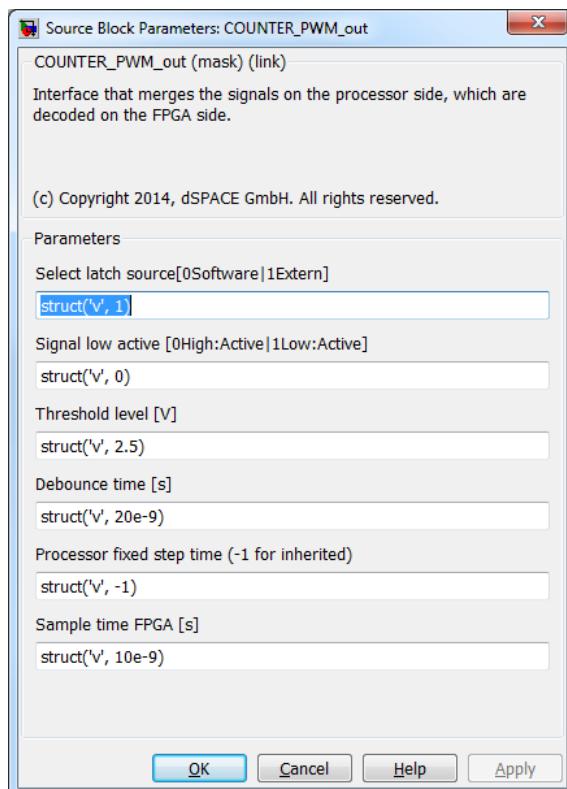


Figure 107: COUNTER_PWM_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Select latch source	[-]	Selection if the external or a software latch is used	0 1
Signal low active	[-]	Selection if the high or low level is measured from the digital signal	0 1
Threshold level	[V]	Threshold level of the digital input channel. Can direct use for digital IO blocks of rtifpga	[0_10.5V]
Processor fixed step time	[s]	The sample time at which the processor blocks are to be simulated.	-
Sample time FPGA	[s]	Sample time of the FPGA	-

Input

The COUNTER_PWM_out block has no inputs.

Output

The processor out block provides one output register which sectioning is shown below:

Register 1

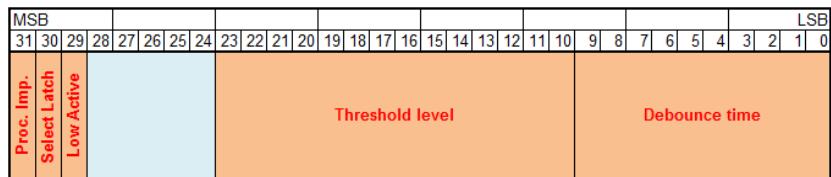


Figure 108: COUNTER_PWM_out Register 1

Name	Used Bits	Description	Range
Debounce Time	9 ... 0	Debounce Time	0 ... 1024
Threshold level	23 ... 10	Threshold voltage level in mV	0 ... 16383
SPARE	28 ... 24		
Low Active	29	Selection if the high or low level is measured from the digital signal	0 1
Select Latch Source	30	Selection if the external or a software lath is used	0 1
Proc_Sync_Imp	31	The software latch trigger from the processor model	0 1

FPGA Main Component

Block



Figure 109: COUNTER_PWM Main block

Block Dialog

The FPGA main blockset contains the following dialog.

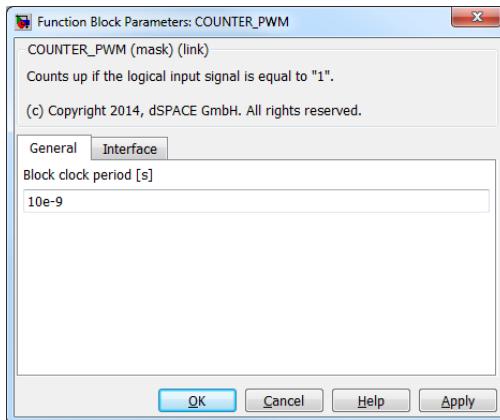


Figure 110: COUNTER_PWM FPGA Main block dialog

	The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.
--	---

Input

The main block has the following inputs:

Name	Unit	Description	Format
Signal	[0 1]	Digital input signal	Bool/UFix_1_0
External Latch	[0 1]	External latch signal	Bool/UFix_1_0
Software Latch	[0 1]	Software latch from the processor	Bool/UFix_1_0
Select Latch Source	[0 1]	Selection if the external or a software lath is used	Bool/UFix_1_0
Debounce Time	[Clocks]	Debounce Time	UFix_10_0
Low Active	[0 1]	Selection if the high or low level is measured from the digital signal	Bool/UFix_1_0
Threshold Voltage	[mV]	Threshold voltage level of the digital in block	UFix_14_0

Output

The COUNTER_PWM main block has the following outputs with a maximal latency of four:

Name	Unit	Description	Format
FPGASingalBus	[Bus]	Bus containing the block's most significant signals	-
Counter	[Clocks]	Counter value after detecting a latch signal	UFix_32_0

Processor Input

Block

Adapts the FPGA signals for the processor side.

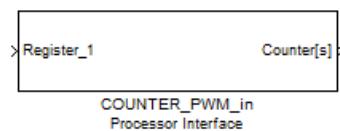


Figure 111: COUNTER_PWM_in block

Block Dialog

The processor in block contains the following dialog where the specific FPGA sample time is defined.

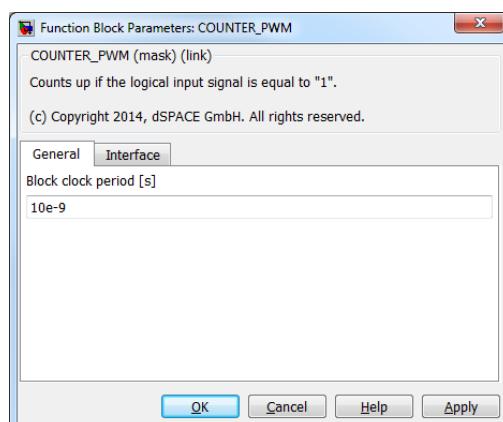


Figure 112: COUNTER_PWM_in block dialog

Input

The processor input block has the following inputs:

Register 1

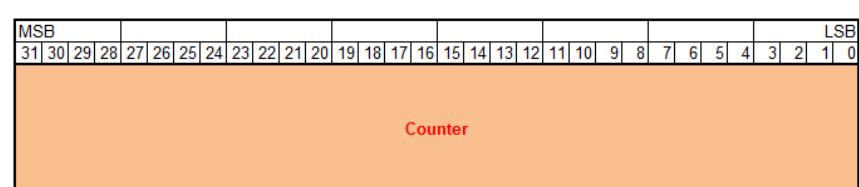


Figure 113: COUNTER_PWM_in Register 1

Name	Used Bits	Description	Range
Counter	31 ... 0	Counter value after detecting a latch signal	0 ... 2^32-1

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Counter	[s]	Measured counter time	-

Interface Examples

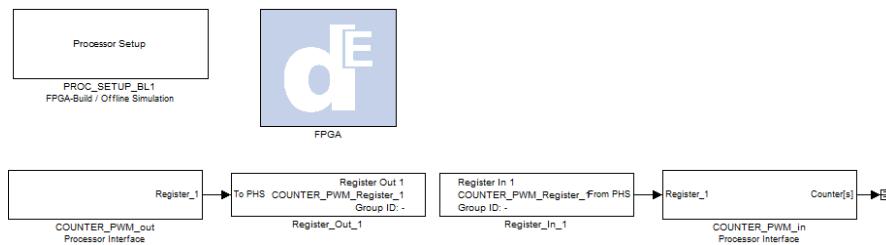
Processor blocks

Figure 114: Processor interface

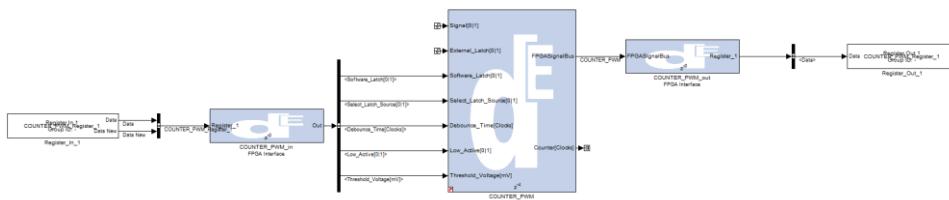
FPGA blocks

Figure 115: FPGA interface

PWM Generator

Objective

The PWM generator blockset is able to generate a pulse center synchronous PWM signal (single phase and three phase). The dead time and the duty cycle can be set on the processor side (online tunable). A pulse center-synchronous interrupt can also be raised on the FPGA and caught on the processor side.



The following chapter describes only the behavior of the single-phase component. The behavior of the three-phase component is the same but with a different number of registers.

The blockset contains the following elements:

- Processor Interface: ONE_PHASE_PWM_GENERATOR_out
(Processor Interface)
- FPGA Interface: ONE_PHASE_PWM_GENERATOR_in
(FPGA Interface)
- FPGA: ONE_PHASE_PWM_GENERATOR
(FPGA Main Component)
- FPGA Interface: ONE_PHASE_PWM_GENERATOR_out
(FPGA Interface)
- Processor Interface: ONE_PHASE_PWM_GENERATOR_in
(Processor Interface)

Processor Output

Block

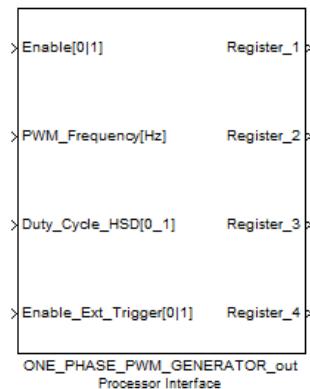


Figure 116: ONE_PHASE_PWM_GENERATOR_out block

Block Dialog

The processor output blockset contains the following dialog.

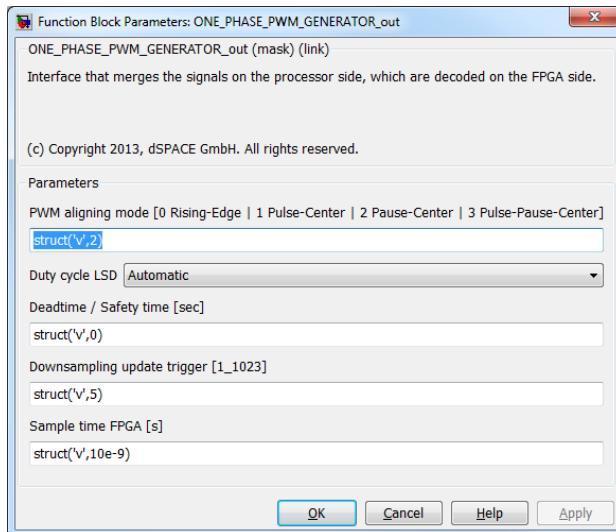


Figure 117: ONE_PHASE_PWM_GENERATOR_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
PWM aligning mode	[-]	Specify the modulation and the update algorithm of the duty cycle data from the inputs 0: rising-edge modulation and update 1: pulse-center modulation and update 2: pause-center modulation and update 3: pulse-center modulation with an update of the duty cycles every pulse- and pause center of the pwm	0 ... 3
Duty cycle LSD	[-]	Specify if the duty cycle of the LSD is generated automatically or manually.	-
Deadtime	[s]	Deadtime between the generated low and high side gate signal	0 ... 52.429 μ s resolution: 10e-9 s (FPGA Sample time: 10e-9 s)
Downsampling update trigger	[-]	If the duty cycle data is updated in the FPGA main block an update trigger flag is generated. The trigger event can be downsampled with this factor	1 ... 1023
Sample time FPGA	[s]	Sample time of the FPGA	1 ...

Input

The ONE_PHASE_PWM_GENERATOR_out block has the following inputs.

Name	Unit	Description	Range
Enable	[-]	Enable of the output	0 1
PWM Frequency	[Hz]	PWM Frequency at disabled external trigger	0 ... 5 MHz
Duty_Cycle	[-]	<p>Duty cycle (single signal or vector) of the PWM</p> <p>Automatic: Duty cycle import represents the setpoint of the duty cycle of the HSD</p> <p><i>3_Phase_PWM_Gen:</i> 1: HSD_a duty cycle 2: HSD_b duty cycle 3: HSD_c duty cycle</p> <p>Manual: Duty cycle import represents a vector with 1: HSD duty cycle 2: LSD duty cycle</p> <p><i>3_Phase_PWM_Gen:</i> 1: HSD_a duty cycle 2: HSD_b duty cycle 3: HSD_c duty cycle 4: LSD_a duty cycle 5: LSD_b duty cycle 6: LSD_c duty cycle</p>	0 ... 1
Enable_Ext_Trigger	[-]	Enable external pulse center synchronization signal	0 1

Output

The processor out block provides four output registers whose sectioning is shown below:

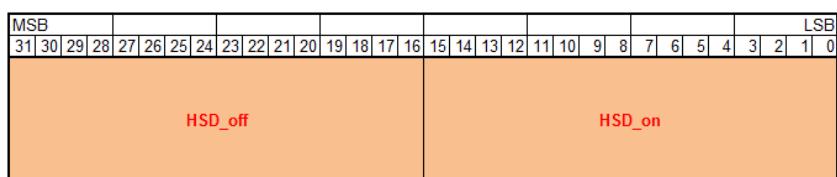
Register 1

Figure 118: ONE_PHASE_PWM_GENERATOR_out Register 1

Name	Used Bits	Description	Range
HSD_on	15 ... 0	Switch level for high side on	0 ... 1
HSD_off	31 ... 16	Switch level for high side off	0 ... 1

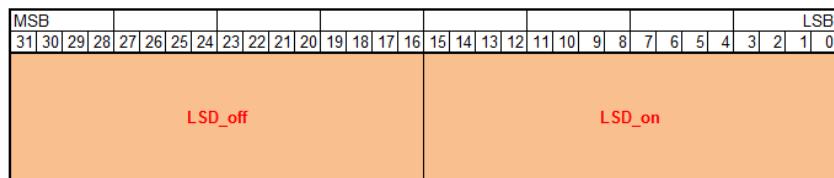
Register 2

Figure 119: ONE_PHASE_PWM_GENERATOR_out Register 2

Name	Used Bits	Description	Range
LSD_on	15 ... 0	Switch level for low side on	0 ... 1
LSD_off	31 ... 16	Switch level for low side off	0 ... 1

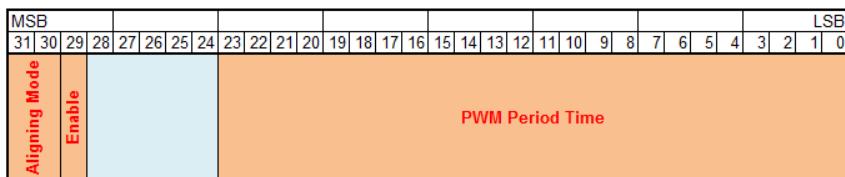
Register 3

Figure 120: ONE_PHASE_PWM_GENERATOR_out Register 3

Name	Used Bits	Description	Range
PWM Period Time	23 ... 0	Period time of the PWM	0 ... 2^24-1 represents 0 ... 0.1678 s
SPARE	28 ... 24		
Enable	29	Output Enable	0 1
Aligning Mode	31 ... 30	PWM aligning mode	0 ... 3

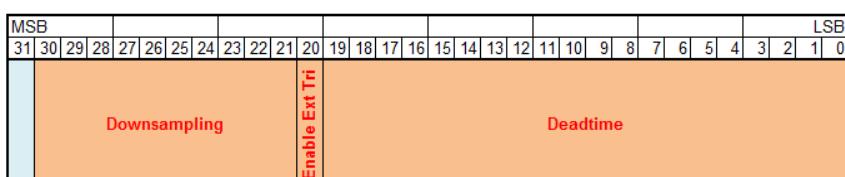
Register 4

Figure 121: ONE_PHASE_PWM_GENERATOR_out Register 4

Name	Used Bits	Description	Range
Deadtime	19 ... 0	Deadtime of the high and low side signals	0 ... 2^19-1 represents 0 ... 52.429 µs
Enable ext Trigger	20	Enable external Trigger	0 1
Downsampling	30 ... 21	Downsampling of the update trigger flag	1 ... 1023
SPARE	31		

FPGA Main Component

Block

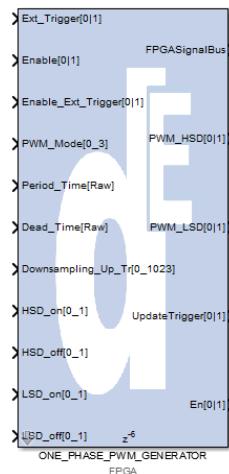


Figure 122: ONE_PHASE_PWM_GENERATION Main block

Block Dialog

The FPGA main blockset contains the following dialog.

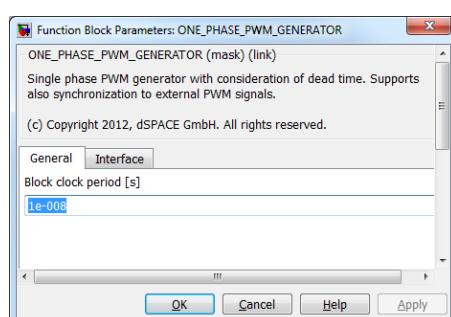


Figure 123: ONE_PHASE_PWM_GENERATION FPGA Main block dialog

	<p>The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic</p>
--	---

	interface generation, refer to Automatic Interface Generation chapter.
--	--

Input

The main block has the following inputs:

Name	Unit	Description	Format
Ext_Trigger	[-]	External pulse center trigger of an external PWM	Bool
Enable	[-]	Enable of the output	Bool
Enable_Ext_Trigger	[-]	Enable of the external trigger	Bool
PWM_Mode	[-]	PWM aligning mode	UFix_2_0
Period_Time	[Raw]	Period time of the PWM at disable external trigger	UFix_24_0
Dead_Time	[Raw]	Dead time of the PWM	UFix_20_0
Downsampling_Up_Tr	[-]	Downsampling of the update trigger flag	UFix_10_0
HSD_on	[-]	Switch level for high side on without the dead time	UFix_16_16
HSD_off	[-]	Switch level for high side off without the dead time	UFix_16_16
LSD_on	[-]	Switch level for low side on without the dead time	UFix_16_16
LSD_off	[-]	Switch level for low side off without the dead time	UFix_16_16

Output

The ONE_PHASE_PWM_GENERATOR main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
PWM_HSD	[-]	High side of gate signal	Bool
PWM_LSD	[-]	Low side of gate signal	Bool
UpdateTrigger	[-]	Update trigger flag	Bool
En	[-]	Enable Output	Bool

Processor Input

Block

Adapts the FPGA signals for the processor side.

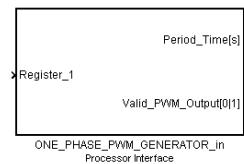


Figure 124: ONE_PHASE_PWM_GENERATOR_in block

Block Dialog

The following parameters can be configured in the processor input dialog:

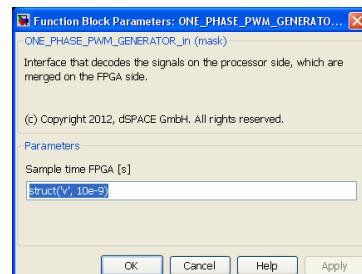


Figure 125: ONE_PHASE_PWM_GENERATOR_in dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Sample time FPGA	[s]	Sample time of the FPGA	-

Input

The processor input block has the following inputs:

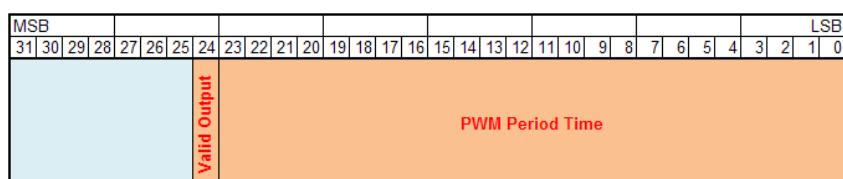
Register 1

Figure 126: ONE_PHASE_PWM_in Register 1

Name	Used Bits	Description	Range
PWM Period Time	23 ... 0	Period time of the output PWM	0 ... 2^23-1
Valid Output	25	Valid output of the PWM generator	0 1

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Period_Time	[s]	Period time of the PWM	0 ... 83,9 ms
Valid_PWM_Output	[-]	Valid flag for the PWM output	0 1

Interface Examples

Processor blocks

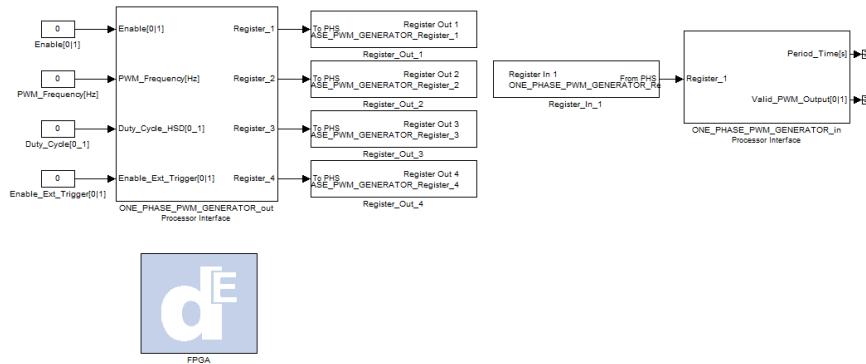


Figure 127: Processor interface

FPGA blocks

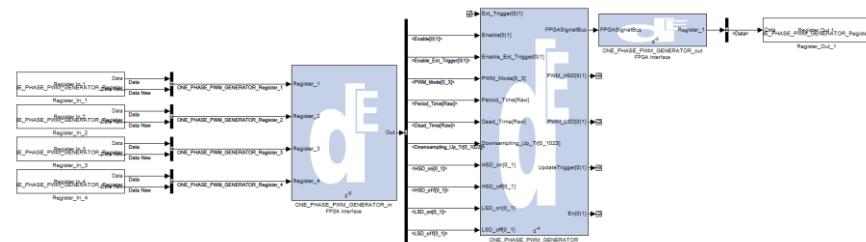


Figure 128: FPGA interface

Protocols: UART_TX

Objective

See the following sections for information on the UART transmission implementation.

The blockset contains the following elements:

- Processor Interface: UART_TX_out (Processor Interface)
- FPGA Interface: UART_TX_in (FPGA Interface)
- FPGA: UART_TX (FPGA Main Component)
- FPGA Interface: UART_TX_out (FPGA Interface)
- Processor Interface: UART_TX_in (Processor Interface)

General behavior

The universal asynchronous transmitter element allows transmitting up to 256 bytes per processor step from processor to FPGA. The FPGA stores the received data inside a FIFO memory element and transmits each byte one after the other until the FIFO is empty, which the specified baud rate. The baud rate is freely adjustable with a time resolution of the FPGA clock rate (e.g. 8ns). The amounts of data bits is adjustable, between 5 and 8 bit, the amount of stop bits between one, one dot five or two. Optional a parity bit can be activated. Since the TX_Bytes_Vector has got a fix dimension of 256, the number of bytes which shall be transmitted is defined by the Num_TX_Bytes[0..255] block input.

The unit gives feedback information, from the FPGA, about the actual FIFO settings, the last data which is transmitted and a message counter.

Processor Output

Block

Merges the processor signals and writes them to the FPGA

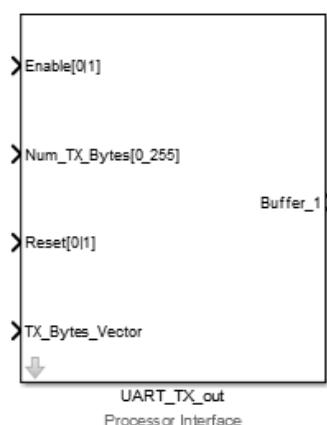


Figure 129: UART_TX_out block

Block Dialog

The dialog provides all parameter, which describe the TX frame.

Name	Unit	Description	Range
Baudrate	[Bit s]	Number of bits to be transmitted within 1s	(32 ... 25.000.000) real achieved maximum baudrate might be limited by the RS232 transceiver which is used
Data_Bits	[-]	Number of data bits per frame	5 ... 8
Stop_Bits	[-]	Number of stop bits per frame	1 1.5 2
Parity_Bit	[-]	1: No parity bit is used 2: Even parity 3: Odd parity	1 2 3
Ts_FPGA	[s]	Sample time of the corresponding main block on the FPGA side. Typical the FPGA clock rate (e.g. 8ns)	

Input

The block has the following inputs:

Name	Unit	Description	Range
Enable[0 1]	[-]	Enable on high state	0 1
Num_TX_Bytes[0_255]	[-]	Number of bytes, which are transmitted from the TX_Bytes_Vector to FPGA in one processor step.	0 1
Reset[0 1]	[-]	Reset on high state	0 1
TX_Bytes_Vector	[-]	Vector of data bytes that shall be transmitted. Internally the vector is forced to the dimension of 256. If the input vector is smaller, it is filled up with zeros.	Max length 256 Data values depend on number of data bits (2^5-1 ... 2^8-1)

Output

The block outputs all data, the configuration of the TX frame, as well as the data elements to be transmitted via a single **Buffer**! The frame configuration is located on the first two buffer elements where the data is located on the next 256 elements. This means the buffer width is always fix with a value of 258, even when less data is transmitted.

FPGA Main Component

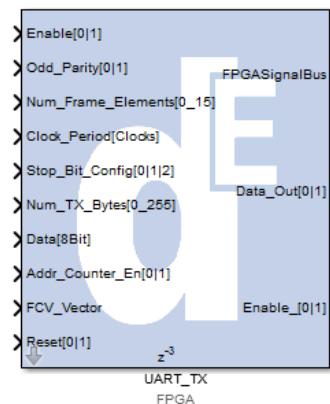
Block

Figure 130: UART_TX FPGA Main block

Block Dialog

The FPGA main blockset contains the following dialog.

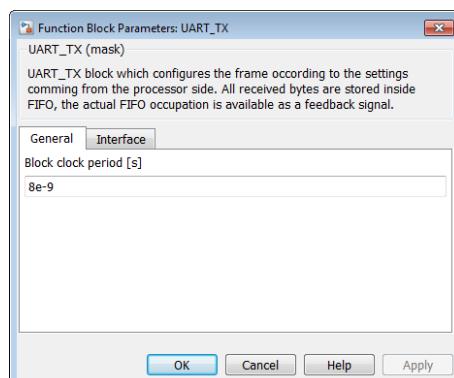


Figure 131: UART_TX FPGA Main block dialog



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Enable	[-]	Enable on high state	Bool
Odd_Parity	[-]	0: Even parity 1: Odd parity No parity is handled inside the frame configuration vector (FCV)	Bool
Num_Frame_Elements[0_15]	[-]	Number of elements within a tx frame	UFix_4_0
Clock_Period[Clocks]	[-]	Inverse baud rate in FPGA sample steps	UFix_23_0
Stop_Bit_Config[0 1 2]	[-]	Configuration of stop bits 0: 1 1: 1.5 2: 2	UFix_2_0
Num_TX_Bytes[0_255]	[-]	Number of bytes to be transmitted	UFix_8_0
Data[8Bit]	[-]	Actual data byte which is stored inside the internal FIFO memory	UFix_8_0
Addr_Counter_En[0 1]	[-]	If enabled the data is written to the FIFO	Bool
FCV_Vector	[-]	Frame configuration vector (FCV). This vector describes the position of data, parity and stop bit within a TX frame	UFix_3_0
Reset[0 1]	[-]	Reset on high state	Bool

Output

The main block has the following outputs with a maximal latency of two:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the most significant signals of this block	-
Data_Out	[-]	Actual output value	UFix_1_0
Enable	[-]	Enable on high state	Bool

Processor Input

Block

Decodes the signal from the FPGA on the processor side.

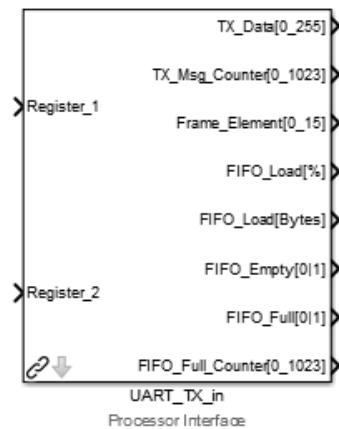


Figure 132: **UART_TX_in** block

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
TX_Data	[-]	Last data value which has been transmitted	0 ... 255
TX_Msg_Counter	[-]	Message counter, increments after a complete TX frame has been kicked out	0 ... 1023
Frame_Element	[-]	Actual frame element, starting with the start bit (0) and ending with the stop bit (depends on amount data bits and parity)	0 ... 15
FIFO_Load	[%]	Actual load of the FPGA internal FIFO	0 ... 100
FIFO_Load	[-]	Actual load of the FPGA internal FIFO	0 ... 255
FIFO_Empty	[-]	High when FIFO has no data to be transmitted	0 1
FIFO_Full	[-]	High when FIFO is full with data to be transmitted. When is at high state no data is written to the FIFO	0 1
FIFO_Full_Counter	[-]	The counter is incremented every time the FIFO_Full signal changes from zero to one	0 ... 1023

Interface Examples

Processor blocks



Figure 133: Processor interface

FPGA block

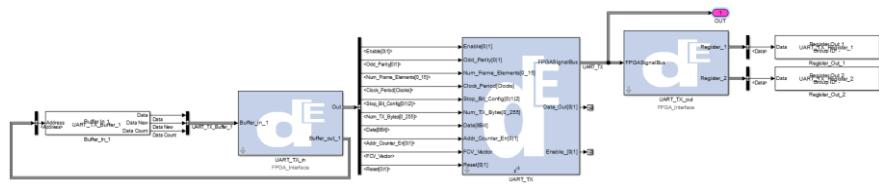


Figure 134: FPGA interface

Protocols: UART_RX

Objective

See the following sections for information on the UART reception implementation.

The blockset contains the following elements:

- Processor Interface: UART_RX_out (Processor Interface)
- FPGA Interface: UART_RX_in (FPGA Interface)
- FPGA: UART_RX (FPGA Main Component)
- FPGA Interface: UART_RX_out (FPGA Interface)
- Processor Interface: UART_RX_in (Processor Interface)



The UART implementation mainly is designed for the usage on a Scalexio HIL System. It can be used on PHS-Bus systems, when performing small modifications.

General behavior

The universal asynchronous receiving element allows to receive up to 256 bytes per processor step from processor to FPGA. The FPGA stores the received data inside a buffer memory element and transmits each byte as a vector to the processor side. A processor FIFO can be activated to store the received data, respectively to read a certain amount of received data in a single step. The baud rate is freely adjustable with a time resolution of the FPGA clock rate (e.g. 8ns). The amounts of data bits is adjustable, between 5 and 8 bit, the amount of stop bits between one, one dot five or two. Optional a parity bit can be activated.

The unit gives feedback information, from the FPGA, about the actual amount of received data, the data itself and a message counter.

Processor Output

Block

Merges the processor signals and writes them to the FPGA



Figure 135: UART_RX_out block

Block Dialog

The dialog provides all parameter, which describe the RX frame.

Name	Unit	Description	Range
Baudrate	[Bit s]	Number of bits to be transmitted within 1s	(32 ... 25.000.000) real achieved maximum baudrate might be limited by the RS232 transceiver which is used
Data_Bits	[-]	Number of data bits per frame	5 ... 8
Stop_Bits	[-]	Number of stop bits per frame	1 1.5 2
Parity_Bit	[-]	1: No parity bit is used 2: Even parity 3: Odd parity	1 2 3
RX_on_RS232	[-]	Since the high level of RS232 is inverted to the TTL logic (logical 1 on RS232 is smaller -3V), the logic on the FPGA is inverted via this switch. 0: TTL logic 1: RS232 logic	0 1
RX_Mode	[-]	RX Bytes reception mode. Either the data coming from the FPGA is directly taken or the internal FIFO is used. 0: Direct use of FPGA signals 1: internal FIFO (the FIFO stores the received data until they are read by the input Num_RX_Bytes[0_256]).The maximum FIFO width is data 256 values.	1 2
Valid_Length	[s]	Received pulses on the FPGA side are recognized as valid when its length exceeds the specified length. This parameter can be used to filter noise impact.	0s ... 81µs 0 ... 1023*Ts_FPGA

Ts_FPGA	[s]	Sample time of the corresponding main block on the FPGA side. Typical the FPGA clock rate (e.g. 8ns)	
---------	-----	---	--

Input

The block has the following inputs:

Name	Unit	Description	Range
Enable[0 1]	[-]	Enable on high state	0 1
Num_TX_Bytes[0_255]	[-]	Number of bytes, which are transmitted from the TX_Bytes_Vector to FPGA in one processor step.	0 1
Reset[0 1]	[-]	Reset on high state	0 1
TX_Bytes_Vector	[-]	Vector of data bytes that shall be transmitted. Internally the vector is forced to the dimension of 256. If the input vector is smaller, it is filled up with zeros.	Max length 256 Data values depend on number of data bits (2^5-1 ... 2^8-1)

Output

The block outputs the RX parameters via register interface. The RX frame configuration settings have to be connected to the UART_RX_in block.

FPGA Main Component

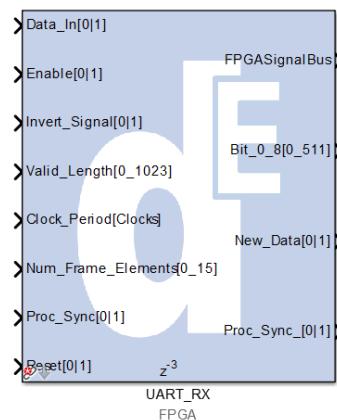
Block

Figure 136: UART_RX FPGA Main block

Block Dialog

The FPGA main blockset contains the following dialog.

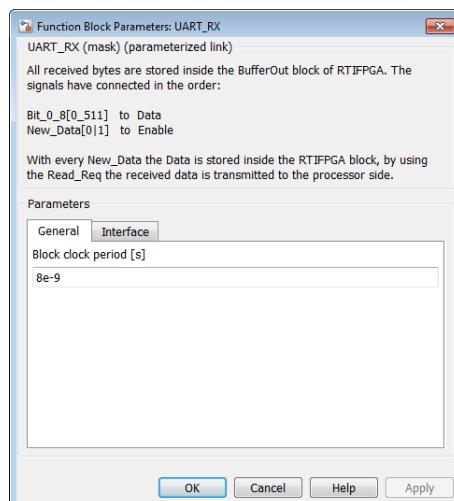


Figure 137: UART_RX FPGA Main block dialog



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Data_In[0 1]	[-]	Signal to be received. Typically comes from an digital input	UFix_1_0
Enable	[-]	Enable on high state	Bool
Invert_Signal[0 1]	[-]	Inverts the input signal, e.g. when RS 232 signal is connected to a standard digital input	UFix_1_0
Valid_Length[0_1023]	[-]	Digital filter, pulses with smaller length are ignored	UFix_10_0
Clock_Period[Clocks]	[-]	Inverse baud rate in FPGA sample steps	UFix_23_0
Num_Frame_Elments[0_15]	[-]	Number of elements within a rx frame	UFix_4_0
Proc_Sync[0 1]	[-]	Changes with every processor step	UFix_1_0
Reset[0 1]	[-]	Reset on high state	Bool

Output

The main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the most significant signals of this block	-
Bit_0_8[0_511]	[-]	RX data inclusive parity bit if activated. Includes not the start and stop bit	UFix_1_0
New_Data[0 1]	[-]	A complete new RX frame has been received	Bool
Proc_Sync_[0 1]	[-]	Route through, coming from input	UFix_1_0

The received data bytes between two processor cycles is transmitted from FPGA to processor via Buffer interface (RTIFPGA → FPGA_XDATA_WRITE). The buffer block used inside a Scalexio (e.g. DS2655) environment comes with a feature that the processor sends a read request to the FPGA, where the FPGA answers with a send acknowledge.

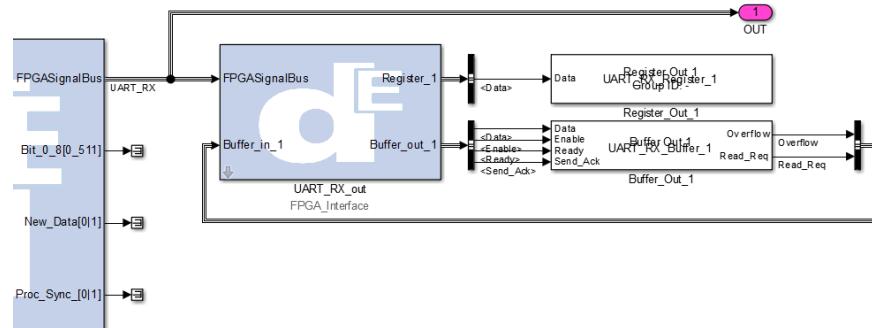


Figure 138: UART_RX FPGA Buffer_Out implementation SCLX

A PHS-Bus FPGA_XDATA_WRITE block does not come with this feature, therefore the Proc_Sync[0|1] signal is used as a workaround and a dummy Read_Req signal has to be integrated.

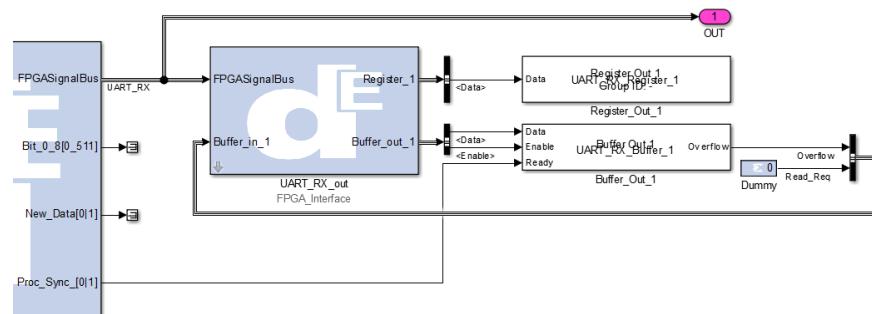


Figure 139: UART_RX FPGA Buffer_Out implementation PHS

Processor Input

Block

Decodes the signal from the FPGA on the processor side.

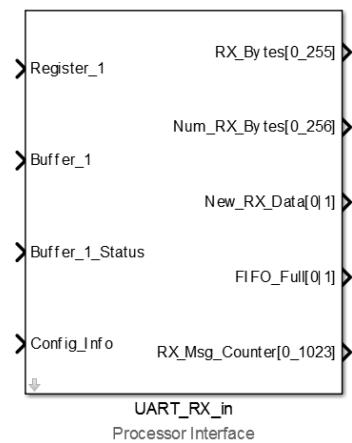


Figure 140: **UART_RX_in** block

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
RX_Bytes	[-]	<p>RX_Bytes, available as a vector with a dimension of 256.</p> <p>If the internal FIFO is deactivated, the output provides the last values received on the FPGA.</p> <p>If the internal FIFO is activated, the output provides the values which are extracted from the FIFO with the input signal Num_RX_Bytes[0_255]</p>	0 ... 255
Num_RX_Bytes	[-]	<p>If the internal FIFO is deactivated, the output provides the last number of received bytes</p> <p>If the internal FIFO is activated, the output provides the number of received bytes which are stored inside the FIFO</p>	0 ... 256
New_RX_Data	[-]	High when new data available	0 1
RX_Msg_Counter	[-]	Message counter, increments after a complete frame has been received	0 ... 1023
FIFO_Full	[-]	<p>High when FIFO is full.</p> <p>When high, no further data is stored inside the FIFO.</p> <p>The state is set to low when data is read out from the FIFO via signal Num_RX_Bytes[0_255]</p>	0 1

Interface Examples

Processor blocks

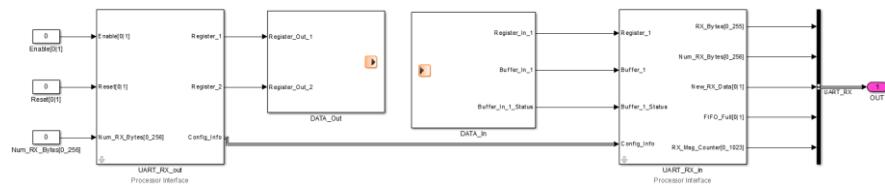


Figure 141: Processor interface

FPGA block

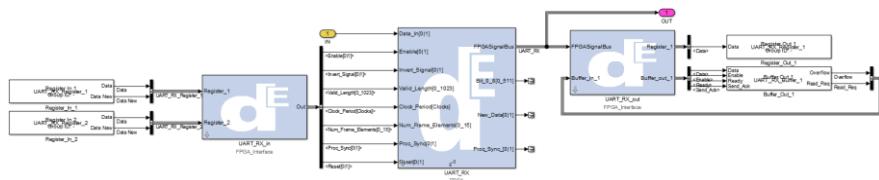


Figure 142: FPGA interface

Data store management and feedback of Look-up Table

Objective	<p>The XSG Utils library supports 1D, 2D and 3D Look-up Table (LUT) and can either be configured with linear interpolation algorithm or with the input below method. These two methods are directly comparable with the standard 1D/2D/3D-LUT from Simulink. The table content is online tunable, e.g. in ControlDesk via VariableEditor.</p> <p>For fixed point LUT, the table data has to be inserted on the processor as well as on the FPGA side! On the FPGA side the table data is stored in a normed format. For being able to control the memory consumption of the LUT the amount of bits for the normed data can be specified.</p> <p>For floating point LUT, the table data has to be inserted only on the processor side. On the FPGA side, to control the memory consumption of the LUT the amount of table data points (maybe also for future use more than the actual table requires) have to be specified in the FPGA main block GUI.</p> <p>To get a better overview of the data storage management, load and check mechanism the following chapter “Data store management” can be used. To check the overall settings of the LUT during runtime a feedback bus is also insert on the processor side. A detailed overview of the feedback the chapter “LUT Feedback” can be used.</p>
------------------	---

Data store management	The data store management is separated in a processor and a FPGA part. The overall logic is implemented in the corresponding GUI.
------------------------------	---

FPGA part (Fixed point):

The data store management in the FPGA main component block carries the analysis of the input axes and the table data which are inserted in the GUI. In this analysis the following important settings for the FPGA model will be generated:

- optimal bit settings
- necessary RAM depth
- necessary range of input axes
- optimal output bit width
- feedback for the processor part of the actual settings

These important parameters are fixed after a FPGA build process! Only the listed settings are stored in the LUT FPGA main component. No table data will be stored during build time!

Processor part (Fixed point):

The data store management of the processor part (Processor_in component) carries the load and storage logic for the table data with additional analyze

functionality. At the first step the table axes and table date will be analyzed and the following settings will be generated:

- necessary RAM depth to store the table data on the FPGA
- actual range of the input axes
- norm factors to ensure the optimal resolution of the LUT output

FPGA part (Floating point):

The data store management in the FPGA main component block carries the analysis of the number of table elements, which is inserted in the GUI. In this analysis the following important settings for the FPGA model will be generated:

- necessary RAM depth
- feedback for the processor part of the actual settings

These important parameters are fixed after a FPGA build process! Only the listed settings are stored in the LUT FPGA main component. No table data will be stored during build time!

Processor part (Floating point):

The data store management of the processor part (Processor_in component) carries the load and storage logic for the table data with additional analyze functionality. At the first step the table axes and table date will be analyzed and the following settings will be generated:

- necessary RAM depth to store the table data on the FPGA
- input value (x-address)
- output value

After this analysis the following schematic of the implemented load/store/check state flow will be used and it is same for fixed as well as floating point LUT:

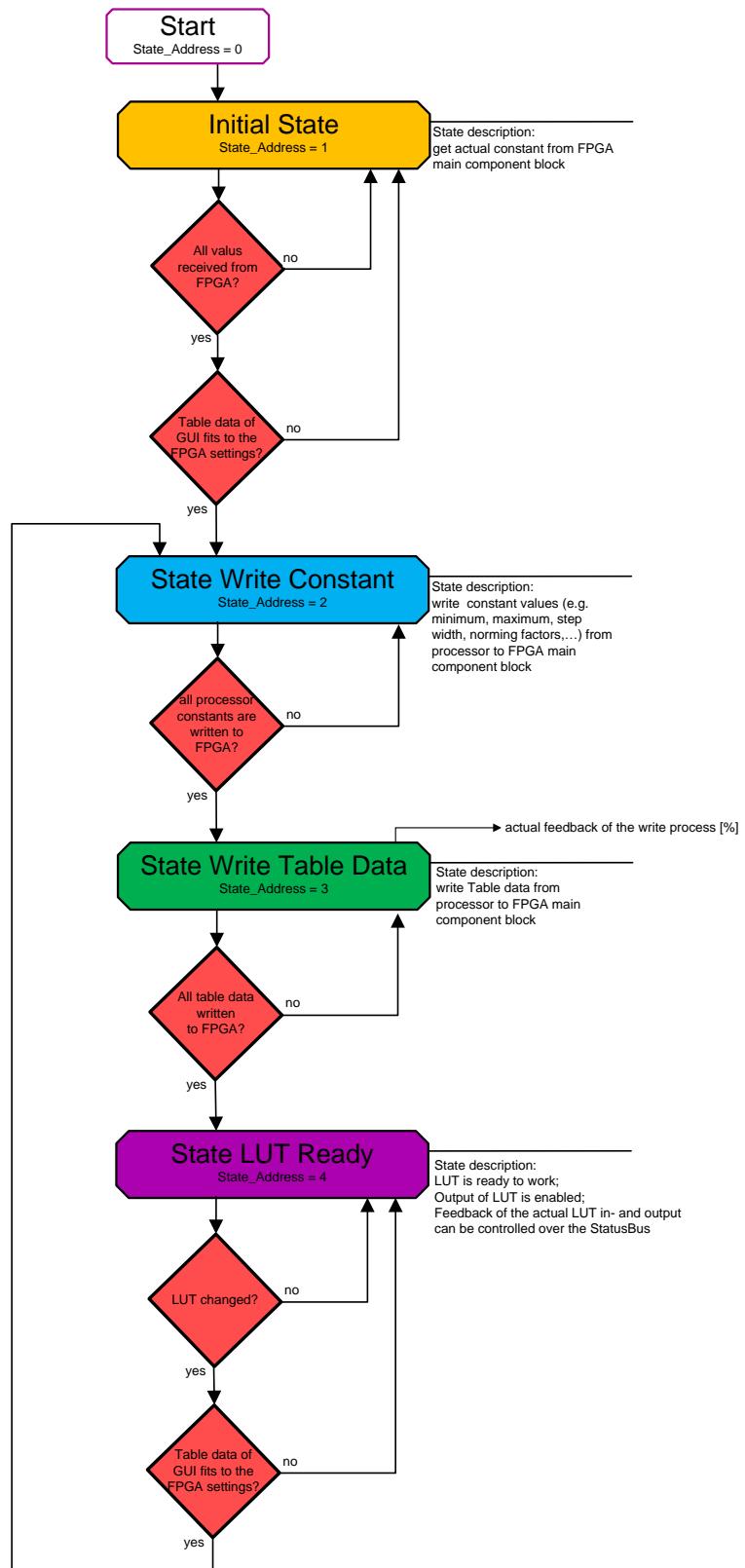


Figure 143: Implemented load and storage mechanism of LUT

The logic algorithm is implemented in the following workflow:

1. The table data of the processor part will be analyzed and interim result will be stored. These results can be controlled in the output “StatusBus/ Necessary_User_Data_Settings” of each processor output component.
2. After the analysis, the static settings of the FPGA will be loaded. This is process is marked with “Initial State”. To minimize the necessary register with respect to the performance the FPGA settings are loaded over an internal bus algorithm. After all values are received the static settings can be controlled in the ouput “StatusBus/ FPGA_Settings”.
3. If the implemented user data doesn't fits in the FPGA settings a global error is generated and the user data won't be loaded to the FPGA and the state flow is fixed in the previous step. All error messages can be controlled in the output “StatusBus/ Error_Messages”.
4. If the user data fits in the FPGA the necessary norm factors and the actual limits, step width of the axes will be sent to the FPGA model. This process is marked with “State Write Constant”.
5. If all user settings are sent to the FPGA the table date will be load to the FPGA. This process is marked with “State Write Table Date”. This and all other processes can be controlled with the output “StatusBus/ Process_Information”. Because the user data needs the longest time to load, the actual load process is can be controlled with the parameter “StatusBus/ Process_Information/ Write_Process[%]”.
6. After all data is loaded the output of the LUT is enabled. This process is marked with “State LUT Ready”.
7. If the customer change the internal LUT data during runtime, the algorithm recognize the changes and restart the state flow with step 3.

LUT Feedback

To ensure a maximum resolution of the internal table data on the FPGA, the table data is stored in a normed format, to be able to control the memory consumption of the LUT the amount of bits for the normed data can to be specified.

To check these overall settings of the LUT during runtime a feedback bus, called “StatusBus” is implemented in the processor output block of each LUT as shown in the figure below (e.g. LUT_THREE_D).

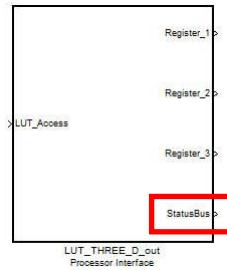


Figure 144: LUT_THREE_D_out block



The following chapter describes only the behavior of the LUT_THREE_D component. The behavior of the other LUT component is the same but with a different number of feedback signals.

The port “StatusBus” carries the following feedback which is divided in the subchapters:

- **FPGA Settings:**

Description: Gives a feedback of the current settings of the FPGA component

Signals (Fixed point):

Name	Unit	Description
RAM_Depth	[0_65536]	Implemented maximal RAM depth
Table_Bit_Length	[0_24]	Feedback of the output table bit length
Table_Lower_Limit	[·]	Lower limit of the minimal supported table value
Table_Upper_Limit	[·]	Upper limit of the maximal supported table value
X_Lower_Limit	[·]	Lower limit of the minimal supported x table
X_Upper_Limit	[·]	Upper limit of the maximal supported x table
X_Dim_max	[·]	Maximal supported dimension of the x table
Y_Lower_Limit	[·]	Lower limit of the minimal supported y table
Y_Upper_Limit	[·]	Upper limit of the maximal supported y table
Y_Dim_max	[·]	Maximal supported dimension of the y table
Z_Lower_Limit	[·]	Lower limit of the minimal supported z table
Z_Upper_Limit	[·]	Upper limit of the maximal supported z table
Z_Dim_max	[·]	Maximal supported dimension of the z table

Signals (Floating point):

Name	Unit	Description
RAM_Depth	[0_65536]	Implemented maximal RAM depth

- *Necessary_User_Data_Settings:*

Description: Gives a feedback of the necessary settings of the implemented user data on the processor side which should store in the FPGA.

Signals (Fixed point):

Name	Unit	Description
RAM_Depth	[0_65536]	Implemented maximal RAM depth
Table_Bit_Length	[0_24]	Feedback of the output table bit length
Table_Lower_Limit	[-]	Lower limit of the minimal supported table value
Table_Upper_Limit	[-]	Upper limit of the maximal supported table value
X_Lower_Limit	[-]	Lower limit of the minimal supported x table
X_Upper_Limit	[-]	Upper limit of the maximal supported x table
X_Dim_max	[-]	Maximal supported dimension of the x table
Y_Lower_Limit	[-]	Lower limit of the minimal supported y table
Y_Upper_Limit	[-]	Upper limit of the maximal supported y table
Y_Dim_max	[-]	Maximal supported dimension of the y table
Z_Lower_Limit	[-]	Lower limit of the minimal supported z table
Z_Upper_Limit	[-]	Upper limit of the maximal supported z table
Z_Dim_max	[-]	Maximal supported dimension of the z table

Signals (Floating point):

Name	Unit	Description
RAM_Depth	[0_65536]	Implemented maximal RAM depth
Table_Lower_Limit	[-]	Lower limit of the minimal supported table value
Table_Upper_Limit	[-]	Upper limit of the maximal supported table value
X_Lower_Limit	[-]	Lower limit of the minimal supported x table
X_Upper_Limit	[-]	Upper limit of the maximal supported x table
X_Dim_max	[-]	Maximal supported dimension of the x table
Y_Lower_Limit	[-]	Lower limit of the minimal supported y table
Y_Upper_Limit	[-]	Upper limit of the maximal supported y table
Y_Dim_max	[-]	Maximal supported dimension of the y table
Z_Lower_Limit	[-]	Lower limit of the minimal supported z table
Z_Upper_Limit	[-]	Upper limit of the maximal supported z table
Z_Dim_max	[-]	Maximal supported dimension of the z table

- *Actual_FPGA_Values (Only for fixed point):*

Description: Gives a feedback of the actual input and output data of the LUT on the FPGA.

Signals:

Name	Unit	Description
Output_Value	[-]	Actual output value
X_Value	[-]	Actual input value of the x axis
Y_Value	[-]	Actual input value of the y axis
Z_Value	[-]	Actual input value of the z axis

- *Online_Tumble (Fixed and Floating point):*

Description: gives a feedback if the online tune ability is activated or not.

- *Error_Messages:*

Description: Gives a feedback of the actual error message class.

Signals (Fixed point):

Name	Unit	Description
Global_Error	[0 1]	Feedback if a global error is detected
RAM_Depth_not_supported	[0 1]	If the necessary RAM depth of the processor LUT is not sufficient with the implemented RAM depth on the FPGA
Lowest_Table_Value_not_supported	[0 1]	Lowest table value of the processor not fit to the settings of the FPGA
Highest_Table_Value_not_supported	[0 1]	Highest table value of the processor not fit to the settings of the FPGA
Lowest_X_Value_not_supported	[0 1]	Lowest table value of the x axis of the processor not fit to the settings of the FPGA
Highest_X_Value_not_supported	[0 1]	Highest table value of the x axis of the processor not fit to the settings of the FPGA
Dimension_X_not_supported	[0 1]	Dimension of the table value of the x axis of the processor not fit to the settings of the FPGA
Lowest_Y_Value_not_supported	[0 1]	Lowest table value of the y axis of the processor not fit to the settings of the FPGA
Highest_Y_Value_not_supported	[0 1]	Highest table value of the y axis of the processor not fit to the settings of the FPGA
Dimension_Y_not_supported	[0 1]	Dimension of the table value of the y axis of the processor not fit to the settings of the FPGA
Lowest_Z_Value_not_supported	[0 1]	Lowest table value of the z axis of the processor not fit to the settings of the FPGA
Highest_Z_Value_not_supported	[0 1]	Highest table value of the z axis of the processor not fit to the settings of the FPGA
Dimension_Z_not_supported	[0 1]	Dimension of the table value of the z axis of the processor not fit to the settings of the FPGA

Signals (Floating point):

Name	Unit	Description
Global_Error	[0 1]	Feedback if a global error is detected
RAM_Depth_not_supported	[0 1]	If the necessary RAM depth of the processor LUT is not sufficient with the implemented RAM depth on the FPGA

- *Process_Information:*

Description: Gives a feedback of the actual process of the internal data store management.

Signals (Fixed and Floating point):

Name	Unit	Description
Write_Process	[%]	Actual percent of the write process of the processor table data
Initial_State	[0 1]	Indicator for the phase "Initial_State" of the data store management
State_Write_Const	[0 1]	Indicator for the phase "State Write Constant" of the data store management
State_Write_TableData	[0 1]	Indicator for the phase "State Write Table Data" of the data store management
State_LUT_Ready	[0 1]	Indicator for the phase "State LUT Ready" of the data store management; feedback if the LUT is ready for use
State_Address	[0_4]	Feedback ID of the actual state
State_FPGA_Address	[0_3]	Feedback FPGA ID of the actual state

1-Dimensional Look-up Table

Objective

The 1D LUT can either be configured with linear interpolation algorithm or with the input below method. These two methods are directly comparable with the standard 1D-LUT from Simulink. The table content is online tunable. The table input value vector (x-axis) and the table data has to be inserted on the **processor as well as on the FPGA side!** On the FPGA side the table data is stored in a normed format, to be able to control the memory consumption of the LUT the amount of bits for the normed data can to be specified.

Another workflow is to specify the accuracy of the normed table, the minimum and maximum data value which shall be covered and let the amount of bits be calculated automatically. The added min. and max. data values are only relevant when after the FPGA build process the table data on the processor side is changed, so that these extreme values can be downloaded to the FPGA. All these settings can be set when the Enlarge table data checkbox is selected at the FPGA main components GUI.



The both processor interface blocks have to be located inside the same subsystem and the StatusBus and LUT_Access ports have to be connected with the corresponding one.

The blockset contains the following elements:

- Processor Interface: LUT_ONE_D_out (Processor Interface)
- FPGA Interface: LUT_ONE_D_in (FPGA Interface)
- FPGA: LUT_ONE_D (FPGA Main Component)
- FPGA Interface: LUT_ONE_D_out (FPGA Interface)
- Processor Interface: LUT_ONE_D_in (Processor Interface)

Processor Output

Block

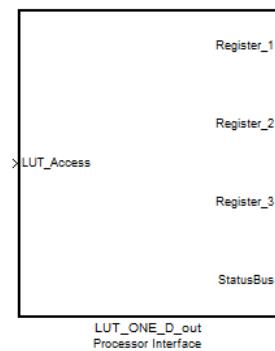


Figure 145: LUT_ONE_D_out block

Block Dialog

The processor output blockset contains the following dialog.

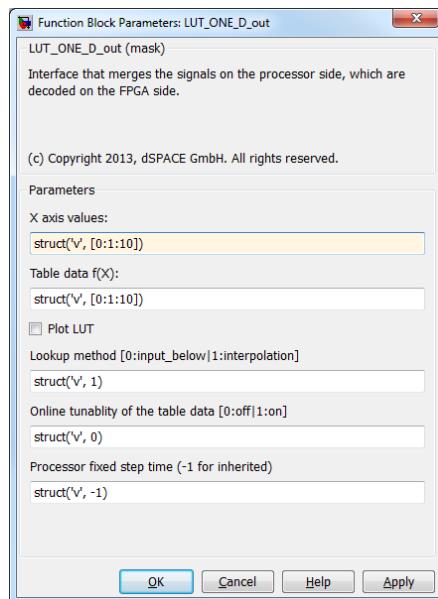


Figure 146: LUT_ONE_D_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
X axis values	[-]	Vector of the input values of the table, must be increasing in an equidistant format	-
Table Data f(X)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Lookup method	[-]	0: use input below 1: linear interpolation	0 1
Online tunability of the table data	[-]	0: not online tunable 1: online tunable (can cause high turnaround time by huge tables)	0 1
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a smaller sample time can be set to speed up the programming of the table, e.g. 20e-9.	-



Ensure that the input vector is monotonically increasing, the step width of the input vector is the same and the input vector has the same length as the table vector. **Infractions of this requirement produce an error message at configuration the dialog parameters.**

Input

The LUT_ONE_D_out block has the following inputs:

Name	Unit	Description	Range
LUT_Acces	[-]	Handshaking information bus between the FPGA-based and the processor-based blocks. Must be connected to LUT_ONE_D_in block.	-

Output

The processor out block has got three output registers which provide the functionality of a simple protocol, together with the processor input register. Because of the complex visualization, no register overview is shown here. For detailed information about the LUT feedback of the port "StatusBus", please refer the chapter "Data store management and feedback of Look-up Table" in the subchapter "LUT Feedback".

FPGA Main Component

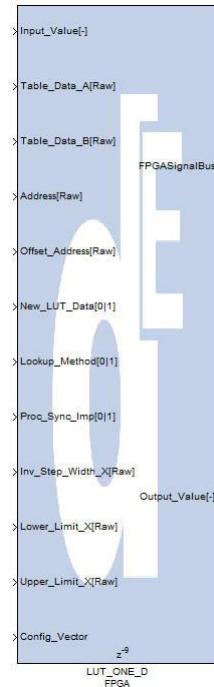
Block

Figure 147: LUT_ONE_D Main block

Block Dialog

The FPGA main blockset contains the following regular dialog.

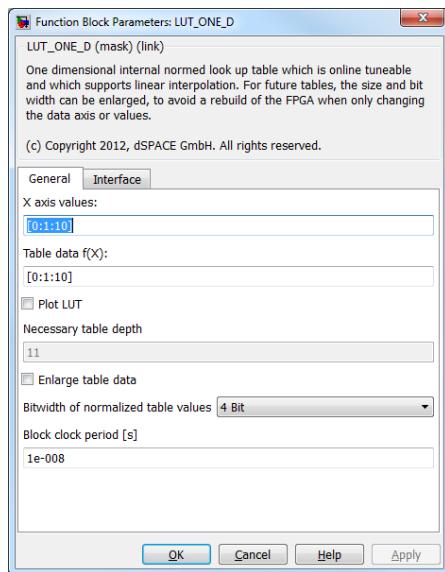


Figure 148: LUT_ONE_D regular dialog

The dialog blockset has the following regular parameters:

Name	Unit	Description	Format
X axis values	[-]	Vector of the input values of the table, must be increasing in an equidistant format	-
Table Data f(X)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Necessary table depth	[-]	Displays the actual amount of data points. Read only	-
Enlarge table data	[-]	Provides more features for data settings	-
Bitwidth of normalized table values	[Bit]	Amount of bits for the normed table data	1 ... 24
Block clock period	[s]	Sample time of the block	



The parameters of the input and output registers can be defined for automatic interface generation. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation. The block clock period can be configured too.

When the Enlarge table data check box is selected the GUI provides these additional settings:

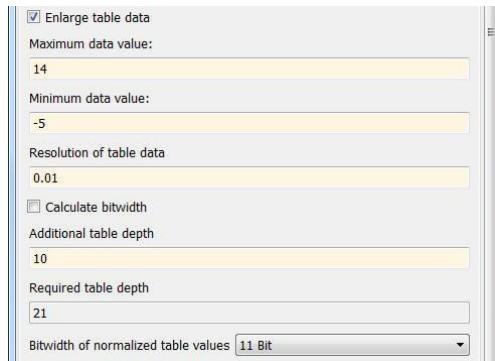


Figure 149: LUT GUI with enlarged table data

The dialog blockset has the following enlarge parameters:

Name	Unit	Description	Format
Maximum data value	[-]	Allows to download bigger values of data than the maximum value of the table data	-
Minimum data value	[-]	Allows to download smaller values of data than the minimum value of the table data	-
Resolution of table data	[-]	Adapts the amount of bits to achieve the specified resolution	-
Calculate bitwidth	[-]	Takes the minimum, maximum, the resolution into account and calculates the required amount of bits	-
Additional table depth	[-]	Allows to increase the allocated memory of the table, to be able to download a bigger table (more data point) after the FPGA build	-
Required table depth	[-]	Displays the total amount of data points. Table data points + additional depth. Read only	-
Bitwidth of normalized table values	[Bit]	Amount of bits for the normed table data	1 ... 24

Input

The main block has the following inputs:

Name	Unit	Description	Format
Input_Value	[-]	FPGA model input signal	User-defined
Table_Data A	[-]	Table data port A for parameterization of the LUT	UFix_24_0
Table_Data B	[-]	Table data port B for parameterization of the LUT	UFix_24_0
Address	[-]	LUT table address for parameterization	Ufix_15_0
Offset Address	[-]	LUT table offset address for parameterization	UFix_15_0
New_LUT_Data	[-]	Data flag is set to 1 when new table data is available to write to the FPGA memory	UFix_1_0
Lookup_Method	[-]	0: input below 1:interpolation	0 1
Proc.Sync. Impulse	[-]	Processor-synchronous toggle bit	UFix_1_0
Inv_Step_Width_X	[-]	Inverse step width of the input vector which is configured in the processor out block	UFix_30_0
Lower_Limit_X	[-]	Minimum of the input vector which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Upper_Limit_X	[-]	Maximum of the input vector which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Config_Vector	[-]	Configuration of the LUT output timing	-

Output

The LUT_ONE_D main block has the following outputs with a maximal latency of six:

Name	Unit	Description	Format
LUT_Acces	Bus	Handshaking information bus between the FPGA-based and the processor-based blocks. Must be connected to LUT_ONE_D_in block.	-

Processor Input

Block	Adapts the FPGA signals for the processor side. The LUT_Access output port has to be connected to the LUT_Access input port of the LUT_ONE_D_out block. Both blocks must be located in the same subsystem.
--------------	--

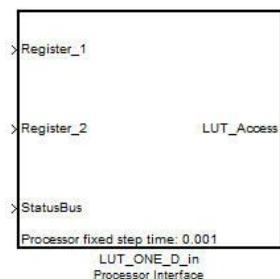


Figure 150: LUT_ONE_D_in block

Block Dialog	The dialog only provides a short block description.
---------------------	---

Input	The processor in block has got two input registers which provide the functionality of a simple protocol, together with the processor output registers. Because of the complex visualization, no register overview is shown here.
--------------	--

Output	The processor input block has the following outputs:
---------------	--

Name	Unit	Description	Range
Read_Enable	[-]	Look-up table is ready to read.	0 1
Output_Value	[-]	Output value of the look-up table	User-defined

Interface Examples

Processor blocks

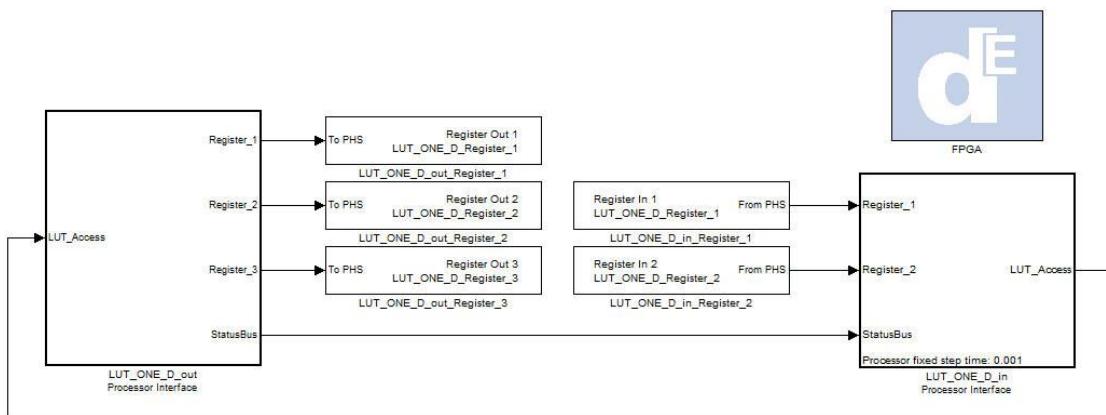


Figure 151: Processor interface

FPGA blocks

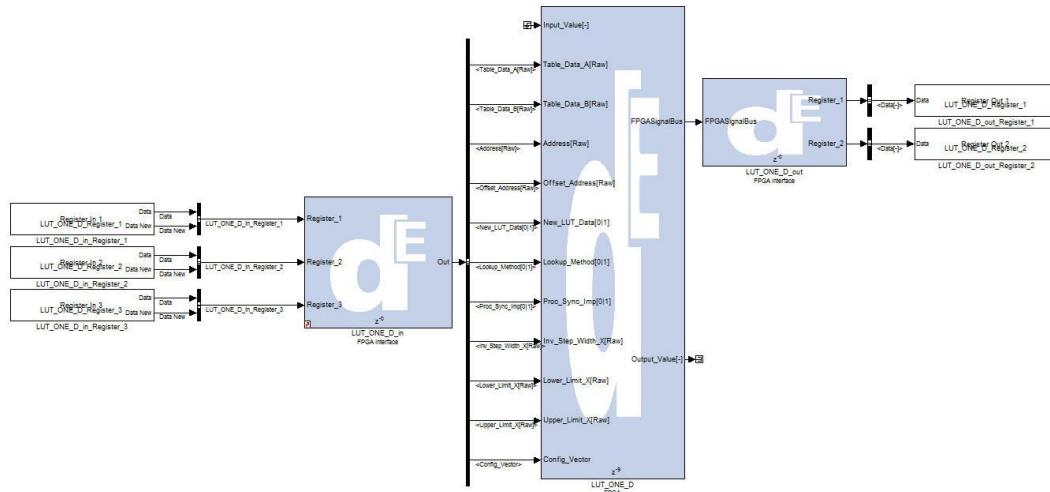


Figure 152: FPGA interface

1-Dimensional Look-up Table XF

Objective

The 1D LUT can either be configured with linear interpolation algorithm or with the input below method. These two methods are directly comparable with the standard 1D-LUT from Simulink. The table content is online tunable. The table input value vector (x-axis) and the table data has to be inserted on the **processor side**. On the FPGA side, the number of table elements is specified in the GUI, based on the length of table data on the processor side.



The both processor interface blocks have to be located inside the same subsystem and the StatusBus and LUT_Access ports have to be connected with the corresponding one.

The blockset contains the following elements:

- Processor Interface: LUT_ONE_D_XF_out (Processor Interface)
- FPGA Interface: LUT_ONE_D_XF_in (FPGA Interface)
- FPGA: LUT_ONE_D_XF (FPGA Main Component)
- FPGA Interface: LUT_ONE_D_XF_out (FPGA Interface)
- Processor Interface: LUT_ONE_D_XF_in (Processor Interface)

Processor Output

Block

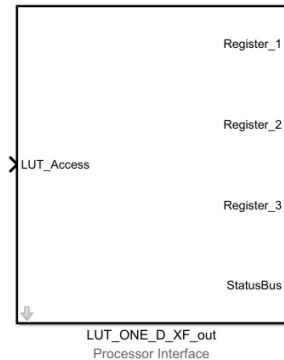


Figure 153: LUT_ONE_D_XF_out block

Block Dialog

The processor output blockset contains the following dialog.

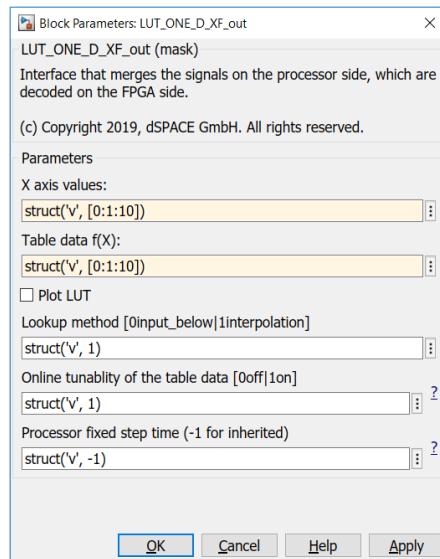


Figure 154: LUT_ONE_D_XF_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
X axis values	[-]	Vector of the input values of the table, must be increasing in an equidistant format	-
Table Data f(X)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Lookup method	[-]	0: use input below 1: linear interpolation	0 1
Online tunability of the table data	[-]	0: not online tunable 1: online tunable (can cause high turnaround time by huge tables)	0 1
Processor fixed step time	[s]	Step time of the processor task (e.g. 0.001s). For offline simulation a smaller sample time can be set to speed up the programming of the table, e.g. 5*Ts_FPGA.	-

	<p>Ensure that the input vector is monotonically increasing, the step width of the input vector is the same and the input vector has the same length as the table vector. Infractions of this requirement</p>
--	--

	produce an error message at configuration the dialog parameters.
--	---

Input

The LUT_ONE_D_XF_out block has the following inputs:

Name	Unit	Description	Format
LUT_Access	[-]	Bus containing the feedback signals from the FPGA coming from LUT_ONE_D_XF_in	-

Output

Via output register the processor out provides the functionality of a simple protocol, together with the processor inputinterface. For detailed information about the LUT feedback of the port "StatusBus", please refer the chapter "Data store management and feedback of Look-up Table" in the subchapter "LUT Feedback".

FPGA Main Component

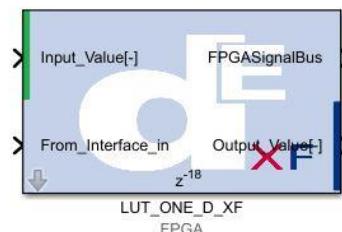
Block

Figure 155: LUT_ONE_D_XF Main block

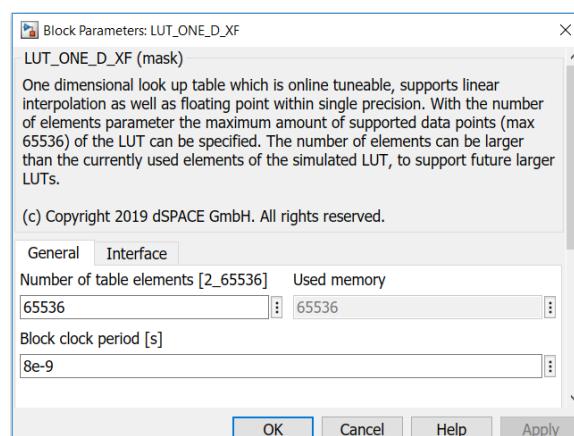


Figure 156: LUT_ONE_D_XF Main block GUI

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Number of table elements	[-]	Amount of table data points	2..65536
Block clock period	[s]	Sample time of the block	-



The parameters of the input and output registers can be defined for automatic interface generation. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation. The block clock period can be configured too.

Input

Following inputs are given to the main block as bus signals:

Name	Unit	Description	Format
Input_Value	[-]	FPGA model input signal	User-defined
Table_Data A	[-]	Table data port A for parameterization of the LUT	XFloat_8_24
Table_Data B	[-]	Table data port B for parameterization of the LUT	XFloat_8_24
Address	[-]	LUT table address for parameterization	Ufix_15_0
Offset Address	[-]	LUT table offset address for parameterization	UFix_16_0
New_LUT_Data	[-]	Data flag is set to 1 when new table data is available to write to the FPGA memory	UFix_1_0
Lookup_Method	[-]	0: input below 1:interpolation	0 1
Proc.Sync. Impulse	[-]	Processor-synchronous toggle bit	UFix_1_0
Inv_Step_Width_X	[-]	Inverse step width of the input vector which is configured in the processor out block	XFloat_8_24
Lower_Limit_X	[-]	Minimum of the input vector which is configured in the processor out block	XFloat_8_24
Upper_Limit_X	[-]	Maximum of the input vector which is configured in the processor out block	XFloat_8_24
State_Address[0_3]	[-]	Configuration of the LUT output timing	UFix_2_0

Output

The LUT_ONE_D_XF main block has the following outputs:

Name	Unit	Description	Format
FPGASignalBus	Bus	Bus containing the block's most significant signals	-
Output_Value	[-]	Table output value	XFloat_8_24

Processor Input

Block

Adapts the signals coming from the FPGA on the processor side. The LUT_Access output port has to be connected to the LUT_Access input port of the LUT_ONE_D_XF_out block. Both blocks must be located in the same subsystem.

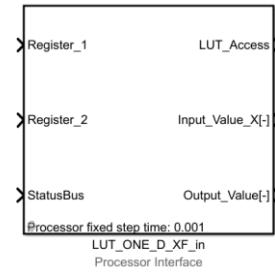


Figure 157: LUT_ONE_D_XF_in block

Block Dialog

The dialog only provides a short block description.

Input

The processor in block provides the functionality of a simple protocol, together with the processor output interface.

Output

The processor input block has the following outputs:

Name	Unit	Description	Format
LUT Access	Bus	Bus containing the feedback signals coming from the FPGA to LUT_ONE_D_XF_out	-
Input_Value_X	[-]	Input value of the FPGA LUT	XFloat_8_24
Output_Value	[-]	Output value of the FPGA LUT	XFloat_8_24

Interface Examples

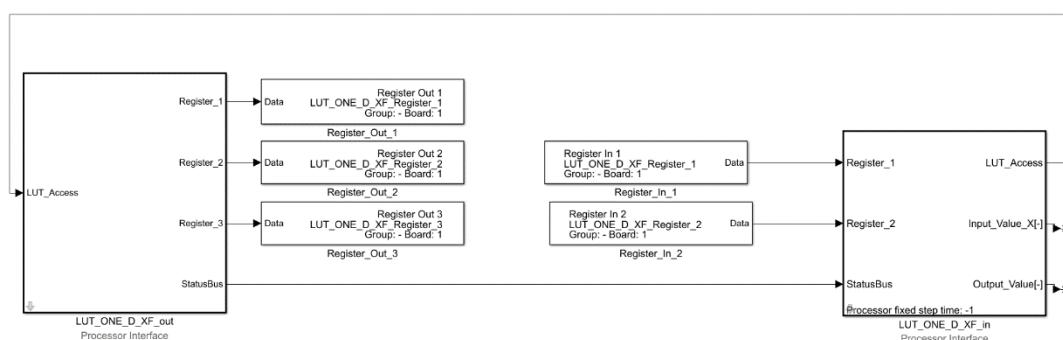
Processor blocks

Figure 158: Processor interface

FPGA blocks

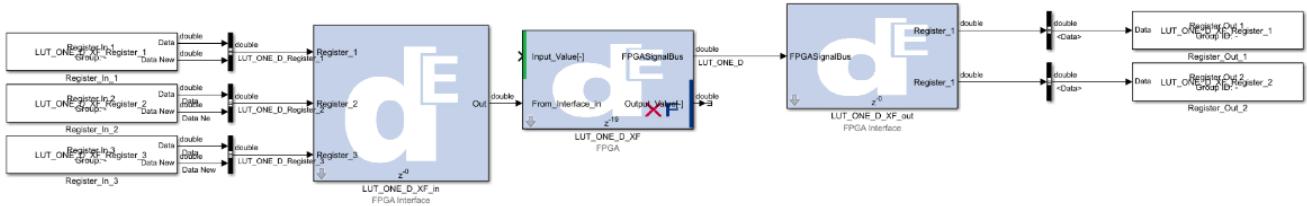


Figure 159: FPGA interface

2-Dimensional Look-up Table

Objective

The 2D LUT can either be configured with linear interpolation algorithm or with the input below method. These two methods are directly comparable with the standard 2D-LUT from Simulink. The table content is online tunable. The table axis value for x (row index) and y (column index) dimension and the table data has to be inserted on the **processor as well as on the FPGA side!** On the FPGA side the table data is stored in a normed format. For being able to control the memory consumption of the LUT the amount of bits for the normed data can be specified.

Another workflow is to specify the accuracy of the normed table, the minimum and maximum data value which shall be covered and let the amount of bits be calculated automatically. The added min. and max. data values are only relevant when after the FPGA build process the table data on the processor side is changed, so that these extreme values can be downloaded to the FPGA. All these settings can be set when the Enlarge table data checkbox is selected at the FPGA main components GUI.



The both processor interface blocks have to be located inside the same subsystem and the StatusBus and LUT_Access ports have to be connected with the corresponding one.

The blockset contains the following elements:

- Processor Interface: LUT_TWO_D_out (Processor Interface)
- FPGA Interface: LUT_TWO_D_in (FPGA Interface)
- FPGA: LUT_TWO_D (FPGA Main Component)
- FPGA Interface: LUT_TWO_D_out (FPGA Interface)
- Processor Interface: LUT_TWO_D_in (Processor Interface)

Processor Output

Block

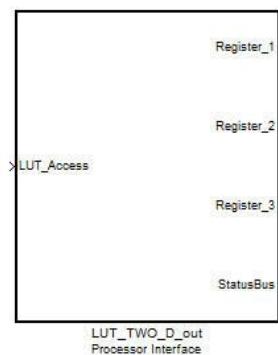


Figure 160: LUT_TWO_D_out block

Block Dialog

The processor output blockset contains the following dialog.

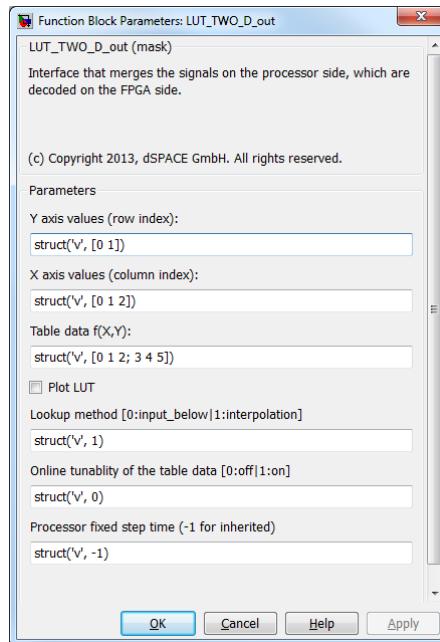


Figure 161: LUT_TWO_D_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Y axis values (row index)	[-]	Y vector of the input values of the table	-
X axis values (column index)	[-]	X vector of the input values of the table	
Table Data f(X,Y)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Lookup method	[-]	0: input below 1: linear interpolation	0 1
Online tunability of the table data	[-]	0: not online tunable 1: online tunable (can cause high turnaround time by huge tables)	0 1
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-



Ensure that the input vector is monotonically increasing, the step width of the input vector is the same and the input vector has the same length as the table vector. **Infractions of this requirement produce an error message at configuration the dialog parameters.**

Input

The LUT_TWO_D_out block has the following inputs:

Name	Unit	Description	Range
LUT_Acces	Bus	Handshaking information bus between the FPGA-based and the processor-based blocks. Must be connected to LUT_TWO_D_in block.	-

Output

The processor out block has got three output registers which provide the functionality of a simple protocol, together with the processor input registers. Because of the complex visualization, no register overview is shown here. For detailed information about the LUT feedback of the port "StatusBus", please refer the chapter "Data store management and feedback of Look-up Table" in the subchapter "LUT Feedback".

FPGA Main Component

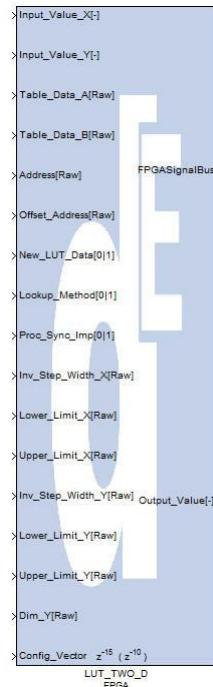
Block

Figure 162: LUT_TWO_D Main block

Block Dialog

The FPGA main blockset contains the following dialog.

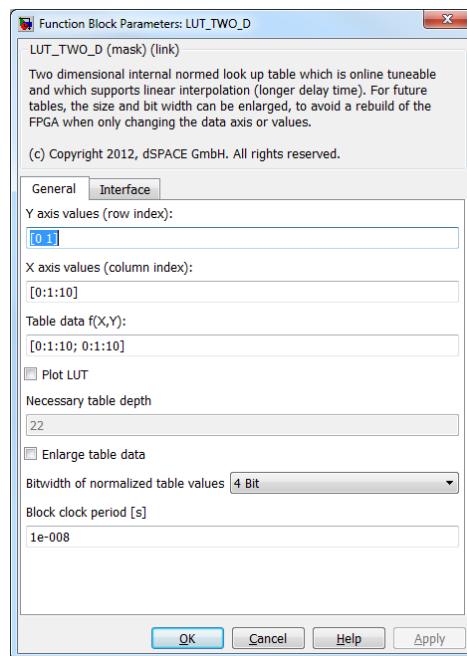


Figure 163: LUT_TWO_D dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Format
Y axis values	[-]	y vector of the input values of the table, must be increasing in an equidistant format	-
X axis values	[-]	X vector of the input values of the table, must be increasing in an equidistant format	-
Table Data f(x)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Necessary table depth	[-]	Displays the actual amount of data points. Read only	-
Enlarge table data	[-]	Provides more features for data settings	-
Bitwidth of normalized table values	[Bit]	Amount of bits for the normed table data	1 ... 24
Block clock period	[s]	Sample time of the block	-



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

When the Enlarge table data check box is selected the GUI provides these additional settings:

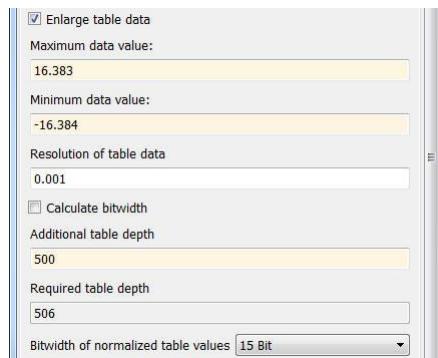


Figure 164: LUT GUI with enlarged table data

The dialog blockset has the following enlarge parameters:

Name	Unit	Description	Format
Maximum data value	[-]	Allows to download bigger values of data than the maximum value of the table data	-
Minimum data value	[-]	Allows to download smaller values of data than the minimum value of the table data	-
Resolution of table data	[-]	Adapts the amount of bits to achieve the specified resolution	-
Calculate bitwidth	[-]	Takes the minimum, maximum, the resolution into account and calculates the required amount of bits	-
Additional table depth	[-]	Allows to increase the allocated memory of the table, to be able to download a bigger table (more data point) after the FPGA build	-
Required table depth	[-]	Displays the total amount of data points. Table data points + additional depth. Read only	-
Bitwidth of normalized table values	[Bit]	Amount of bits for the normed table data	1 ... 24



The parameters of the input and output registers can be defined for automatic interface generation. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation. The block clock period can be configured too.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Input_Value_X	[-]	FPGA model input signal for X axis	User-defined
Input_Value_Y	[-]	FPGA model input signal for Y axis	User-defined
Table_Data A	[-]	Table data port A for parameterization of the LUT	UFix_24_0
Table_Data B	[-]	Table data port B for parameterization of the LUT	UFix_24_0
Address	[-]	LUT table address for parameterization	Ufix_15_0
Offset Address	[-]	LUT table offset address for parameterization	UFix_15_0
New_LUT_Data	[-]	Data flag is set to 1 when new table data is available to write to the FPGA memory	UFix_1_0
Lookup_Method	[-]	0: input below 1:interpolation	0 1
Proc.Sync. Impulse	[-]	Processor-synchronous toggle bit	UFix_1_0
Inv_Step_Width_X	[-]	Inverse step width of the input vector X which is configured in the processor out block	UFix_30_0
Lower_Limit_X	[-]	Minimum of the input vector X which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Upper_Limit_X	[-]	Maximum of the input vector X which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Inv_Step_Width_Y	[-]	Inverse step width of the input vector Y which is configured in the processor out block	UFix_30_0

Lower_Limit_Y	[-]	Minimum of the input vector Y which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Upper_Limit_Y	[-]	Maximum of the input vector Y which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Config_Vector	[-]	Configuration of the LUT output timing	-

Output

The LUT_TWO_D main block has the following outputs with a maximal latency of 10 for input below and 15 for linear interpolation method:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Output_Value	[-]	Actual output of the look-up table	User-defined

Processor Input

Block

Adapts the FPGA signals for the processor side.

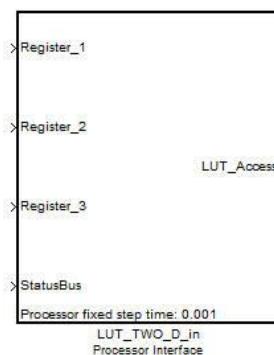


Figure 165: LUT_TWO_D_in block

Block Dialog

The dialog only provides a short block description.

Input

The processor out block has got three input registers which provide the functionality of a simple protocol, together with the processor output registers. Because of the complex visualization, no register overview is shown here.

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
LUT_Acces	Bus	Handshaking information bus between the FPGA-based and the processor-based blocks. Must be connected to LUT_TWO_D_in block.	-

Interface Examples

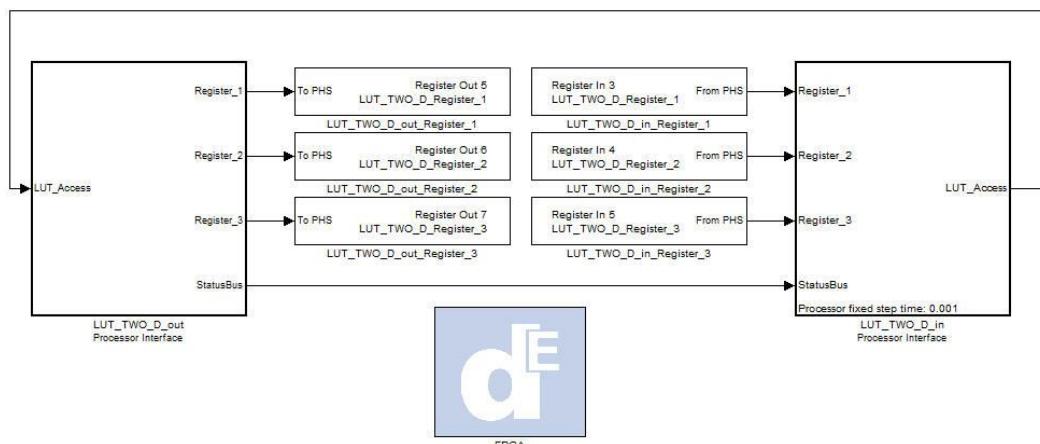
Processor blocks

Figure 166: Processor interface

FPGA blocks

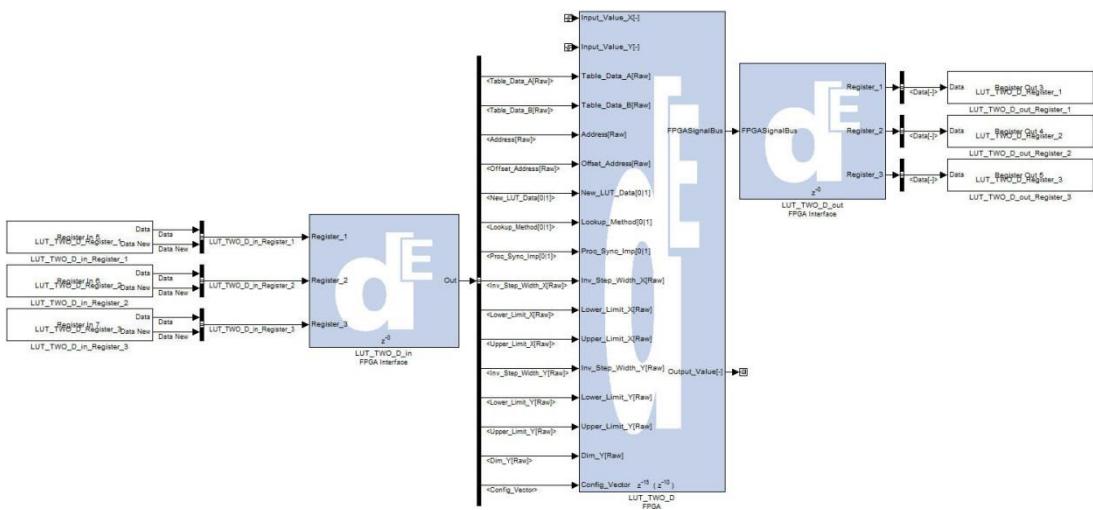


Figure 167: FPGA interface

2-Dimensional Look-up Table XF

Objective

The 2D LUT can either be configured with linear interpolation algorithm or with the input below method. These two methods are directly comparable with the standard 2D-LUT from Simulink. The table content is online tunable. The table axis value for x (row index) and y (column index) dimension and the table data has to be inserted on the **processor side**. On the FPGA side, the number of table elements is specified in the GUI, based on the length of table data on the processor side.



The both processor interface blocks have to be located inside the same subsystem and the StatusBus and LUT_Access ports have to be connected with the corresponding one.

The blockset contains the following elements:

- Processor Interface: LUT_TWO_D_XF_out (Processor Interface)
- FPGA Interface: LUT_TWO_D_XF_in (FPGA Interface)
- FPGA: LUT_TWO_D_XF (FPGA Main Component)
- FPGA Interface: LUT_TWO_D_XF_out (FPGA Interface)
- Processor Interface: LUT_TWO_D_XF_in (Processor Interface)

Processor Output

Block

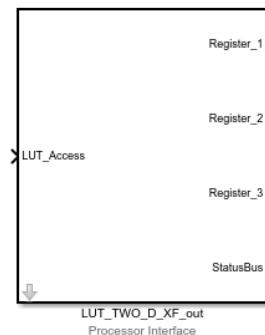


Figure 168: LUT_TWO_D_XF_out block

Block Dialog

The processor output blockset contains the following dialog.

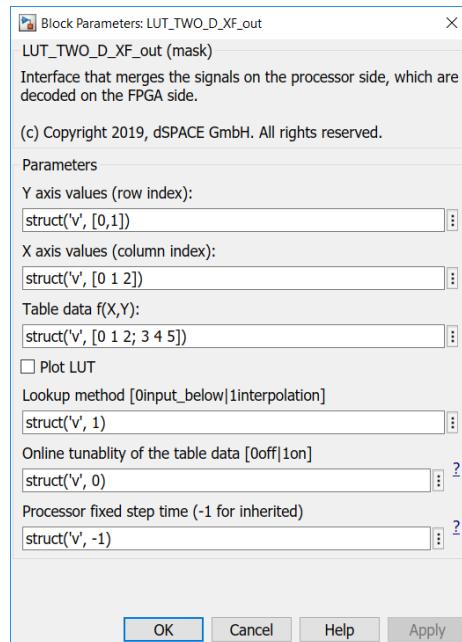


Figure 169: LUT_TWO_D_XF_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Y axis values (row index)	[-]	Y vector of the input values of the table	-
X axis values (column index)	[-]	X vector of the input values of the table	
Table Data f(X,Y)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Lookup method	[-]	0: input below 1: linear interpolation	0 1
Online tenability of the table data	[-]	0: not online tunable 1: online tunable (can cause high turnaround time by huge tables)	0 1
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-



Ensure that the input vector is monotonically increasing, the step width of the input vector is the same and the input vector has the same length as the table vector. **Infractions of this requirement**

produce an error message at configuration the dialog parameters.

Input

The LUT_TWO_D_XF_out block has the following inputs:

Name	Unit	Description	Range
LUT_Acces	Bus	Bus containing the feedback signals from the FPGA coming from LUT_TWO_D_XF_in	-

Output

Via output register the processor out provides the functionality of a simple protocol, together with the processor input interface. For detailed information about the LUT feedback of the port "StatusBus", please refer the chapter "Data store management and feedback of Look-up Table" in the subchapter "LUT Feedback".

FPGA Main Component

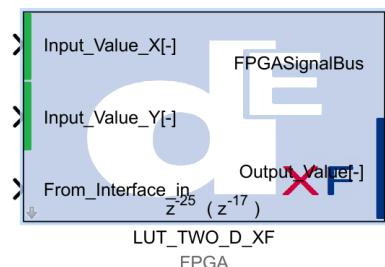
Block

Figure 170: LUT_TWO_D_XF Main block

Block Dialog

The FPGA main blockset contains the following dialog.

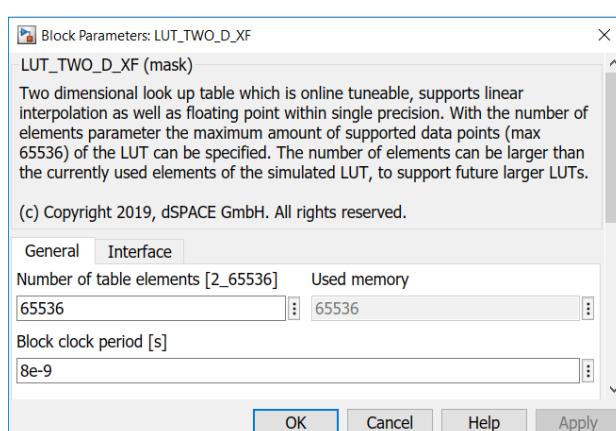


Figure 171: LUT_TWO_D_XF dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Number of table elements	[-]	Amount of table data points	2..65536
Block clock period	[s]	Sample time of the block	-



The parameters of the input and output registers can be defined for automatic interface generation. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation. The block clock period can be configured too.

Input

Following inputs are given to the main block as bus signals:

Name	Unit	Description	Format
Input_Value_X	[-]	FPGA model input X signal	User-defined
Input_Value_Y	[-]	FPGA model input Y signal	User-defined
Table_Data A	[-]	Table data port A for parameterization of the LUT	XFloat_8_24
Table_Data B	[-]	Table data port B for parameterization of the LUT	XFloat_8_24
Address	[-]	LUT table address for parameterization	Ufix_15_0
Offset Address	[-]	LUT table offset address for parameterization	UFix_16_0
New_LUT_Data	[-]	Data flag is set to 1 when new table data is available to write to the FPGA memory	UFix_1_0
Lookup_Method	[-]	0: input below 1:interpolation	0 1
Proc.Sync. Impulse	[-]	Processor-synchronous toggle bit	UFix_1_0
Inv_Step_Width_X	[-]	Inverse step width of the input vector which is configured in the processor out block	XFloat_8_24
Lower_Limit_X	[-]	Minimum of the input vector which is configured in the processor out block	XFloat_8_24
Upper_Limit_X	[-]	Maximum of the input vector which is configured in the processor out block	XFloat_8_24
Inv_Step_Width_Y	[-]	Inverse step width of the input vector which is configured in the processor out block	XFloat_8_24
Lower_Limit_Y	[-]	Minimum of the input vector which is configured in the processor out block	XFloat_8_24
Upper_Limit_Y	[-]	Maximum of the input vector which is configured in the processor out block	XFloat_8_24
Dim_Y	[-]	Dimension or length of the Y-axis values	UFix_16_0
State_Address[0_3]	[-]	Configuration of the LUT output timing	UFix_2_0

Output

The LUT_TWO_D_XF main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Output_Value	[-]	Table output value	XFloat_8_24

Processor Input

Block

Adapts the signals coming from the FPGA on the processor side. The LUT_Access output port has to be connected to the LUT_Access input port of the LUT_TWO_D_XF_out block. Both blocks must be located in the same subsystem.

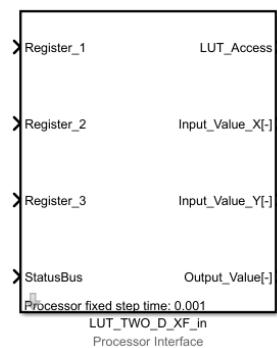


Figure 172: LUT_TWO_D_XF_in block

Block Dialog

The dialog only provides a short block description.

Input

The processor in block provides the functionality of a simple protocol, together with the processor output interface.

Output

The processor input block has the following outputs:

Name	Unit	Description	Format
LUT Access	Bus	Bus containing the feedback signals coming from the FPGA to LUT_TWO_D_XF_out	-
Input_Value_X	[-]	Input value for X-axis of the FPGA LUT	XFloat_8_24
Input_Value_Y	[-]	Input value for Y-axis of the FPGA LUT	XFloat_8_24
Output_Value	[-]	Output value of the FPGA LUT	XFloat_8_24

Interface Examples

Processor blocks

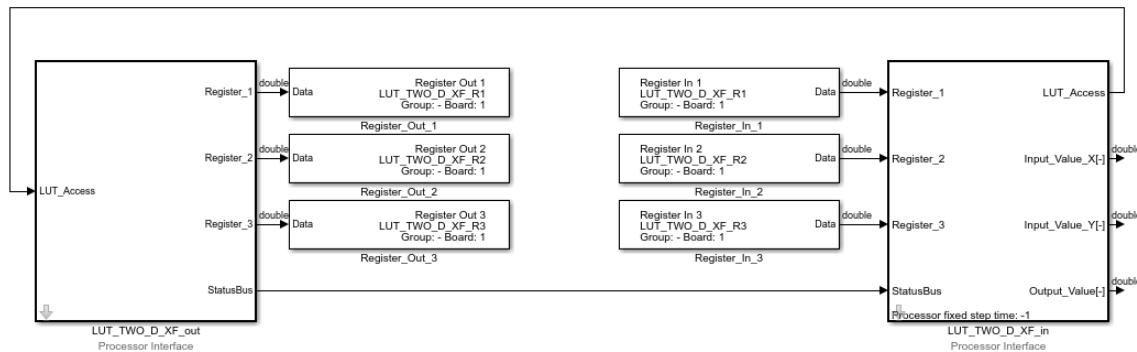


Figure 173: Processor interface

FPGA blocks

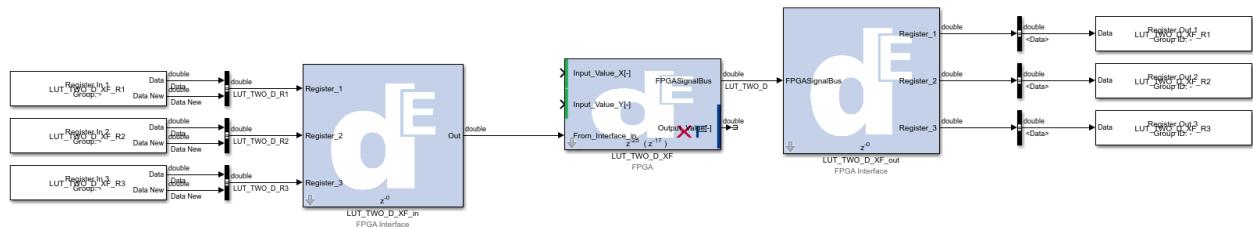


Figure 174: FPGA interface

3-Dimensional Look-up Table

Objective

The 3D LUT can either be configured with linear interpolation algorithm or with the input below method. These two methods are directly comparable with the standard 3D-LUT from Simulink. The table content is online tunable. The table axis value for x (row index) and y (column index) dimension and the table data has to be inserted on the **processor as well as on the FPGA side!** On the FPGA side the table data is stored in a normed format. For being able to control the memory consumption of the LUT the amount of bits for the normed data can to be specified.

Another workflow is to specify the accuracy of the normed table, the minimum and maximum data value which shall be covered and let the amount of bits be calculated automatically. The added min. and max. data values are only relevant when after the FPGA build process the table data on the processor side is changed, so that these extreme values can be downloaded to the FPGA. All these settings can be set when the Enlarge table data checkbox is selected at the FPGA main components GUI.



The both processor interface blocks have to be located inside the same subsystem and the StatusBus and LUT_Access ports have to be connected with the corresponding one.

The blockset contains the following elements:

- | | |
|------------------------|---------------------------------------|
| ▪ Processor Interface: | LUT_THREE_D_out (Processor Interface) |
| ▪ FPGA Interface: | LUT_THREE_D_in (FPGA Interface) |
| ▪ FPGA: | LUT_THREE_D (FPGA Main Component) |
| ▪ FPGA Interface: | LUT_THREE_D_out (FPGA Interface) |
| ▪ Processor Interface: | LUT_THREE_D_in (Processor Interface) |

Processor Output

Block

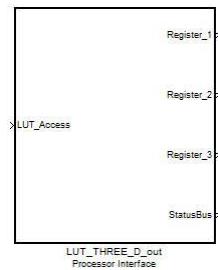


Figure 175: LUT_THREE_D_out block

Block Dialog

The processor output blockset contains the following dialog.

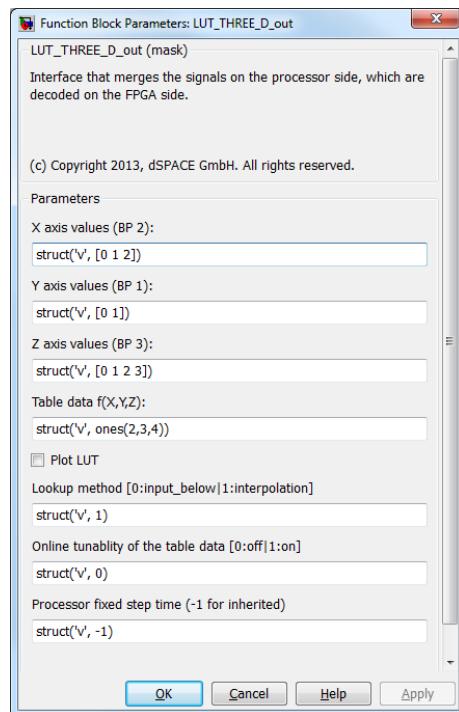


Figure 176: LUT_THREE_D_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
X axis values (BP 2)	[-]	X vector of the input values of the table	-
Y axis values (BP 1)	[-]	Y vector of the input values of the table	-
Z axis values (BP 3)	[-]	Z vector of the input values of the table	-
Table Data f(X,Y,Z)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Lookup method	[-]	0: input below 1: linear interpolation	0 1
Online tunability of the table data	[-]	0: not online tunable 1: online tunable (can cause high turnaround time by huge tables)	0 1
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-



Ensure that the input vector is monotonically increasing, the step width of the input vector is the same and the input vector has the same length as the table vector. **Infractions of this requirement produce an error message at configuration the dialog parameters.**

Input

The LUT_TWO_D_out block has the following inputs:

Name	Unit	Description	Range
LUT_Acces	[-]	Handshaking information bus between the FPGA-based and the processor-based blocks. Must be connected to LUT_THREE_D_in block.	-

Output

The processor out block has got three output registers which provide the functionality of a simple protocol, together with the processor input registers. Because of the complex visualization, no register overview is shown here. For detailed information about the LUT feedback of the port "StatusBus", please refer the chapter "Data store management and feedback of Look-up Table" in the subchapter "LUT Feedback".

FPGA Main Component

Block

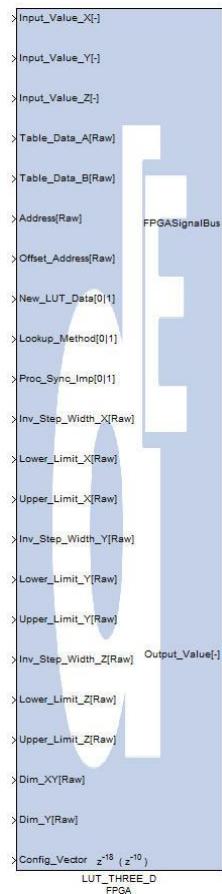


Figure 177: LUT_THREE_D Main block

Block Dialog

The FPGA main blockset contains the following dialog.

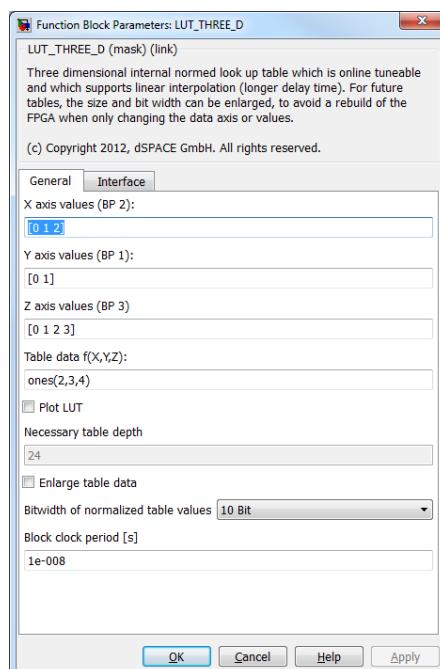


Figure 178: LUT_THREE_D dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Format
XYZ axis values	[-]	y vector of the input values of the table, must be increasing in an equidistant format	-
Table Data f(X,Y,Z)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Necessary table depth	[-]	Displays the actual amount of data points. Read only	-
Enlarge table data	[-]	Provides more features for data settings	-
Bitwidth of normalized table values	[Bit]	Amount of bits for the normed table data	1 ... 24
Block clock period	[s]	Sample time of the block	



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

When the Enlarge table data check box is selected the GUI provides these additional settings:

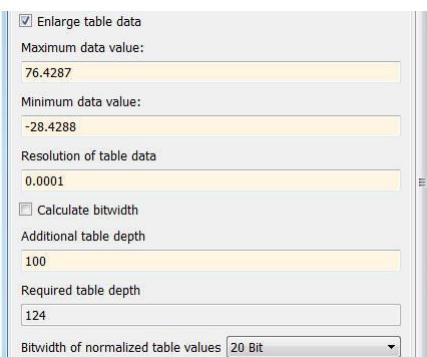


Figure 179: LUT GUI with enlarged table data

The dialog blockset has the following enlarge parameters:

Name	Unit	Description	Format
Maximum data value	[-]	Allows to download bigger values of data than the maximum value of the table data	-
Minimum data value	[-]	Allows to download smaller values of data than the minimum value of the table data	-
Resolution of table data	[-]	Adapts the amount of bits to achieve the specified resolution	-
Calculate bitwidth	[-]	Takes the minimum, maximum, the resolution into account and calculates the required amount of bits	-
Additional table depth	[-]	Allows to increase the allocated memory of the table, to be able to download a bigger table (more data point) after the FPGA build	-
Required table depth	[-]	Displays the total amount of data points. Table data points + additional depth. Read only	-
Bitwidth of normalized table values	[Bit]	Amount of bits for the normed table data	1 ... 24



The parameters of the input and output registers can be defined for automatic interface generation. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation. The block clock period can be configured too.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Input_Value_XYZ	[-]	FPGA model input signal for XYZ axis	User-defined
Table_Data A	[-]	Table data port A for parameterization of the LUT	UFix_24_0
Table_Data B	[-]	Table data port B for parameterization of the LUT	UFix_24_0
Address	[-]	LUT table address for parameterization	Ufix_15_0
Offset Address	[-]	LUT table offset address for parameterization	UFix_15_0
New_LUT_Data	[-]	Data flag is set to 1 when new table data is available to write to the FPGA memory	UFix_1_0
Lookup_Method	[-]	0: input below 1:interpolation	0 1
Proc.Sync. Impulse	[-]	Processor-synchronous toggle bit	UFix_1_0
Inv_Step_Width_XYZ	[-]	Inverse step width of the input vectors which is configured in the processor out block	UFix_30_0
Lower_Limit_XYZ	[-]	Minimum of the input vectors which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Upper_Limit_XYZ	[-]	Maximum of the input vectors which is configured in the processor out block; coded with the binary point which is configured in the dialog	UFix_25_0
Dim_XY	[-]	Dimension of the XY direction	UFix_15_0
Dim_Y	[-]	Dimension of Y direction	UFix_15_0
Config_Vector	[-]	Configuration of the LUT output timing	-

Output

The LUT_THREE_D main block has the following outputs with a maximal latency of 10 for input below and 18 for linear interpolation method:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Output_Value	[-]	Actual output of the look-up table	User-defined

Processor Input

Block Adapts the FPGA signals for the processor side.

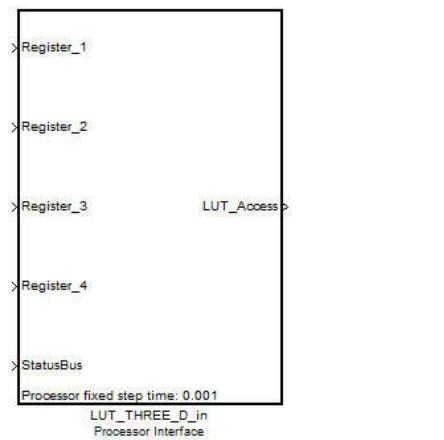


Figure 180: LUT_THREE_D_in block

Block Dialog The dialog only provides a short block description.

Input The processor in block has got four input registers which provide the functionality of a simple protocol, together with the processor output registers. Because of the complex visualization, no register overview is shown here:

Interface Examples

Processor blocks

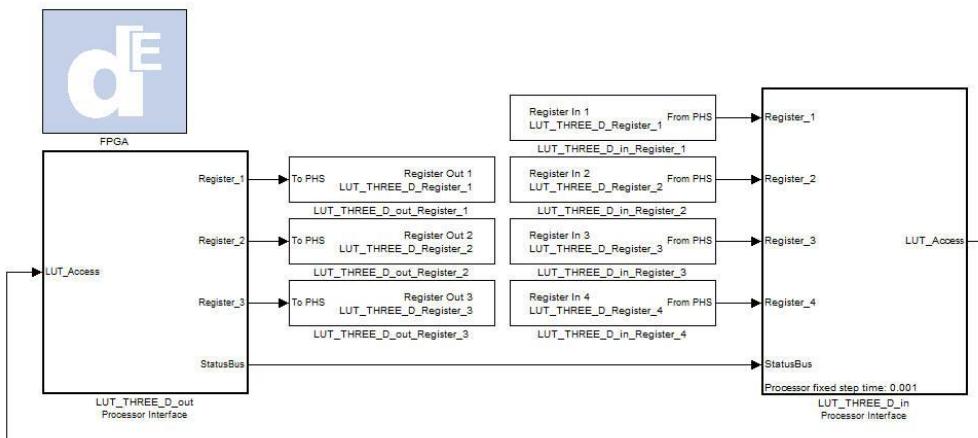


Figure 181: Processor interface

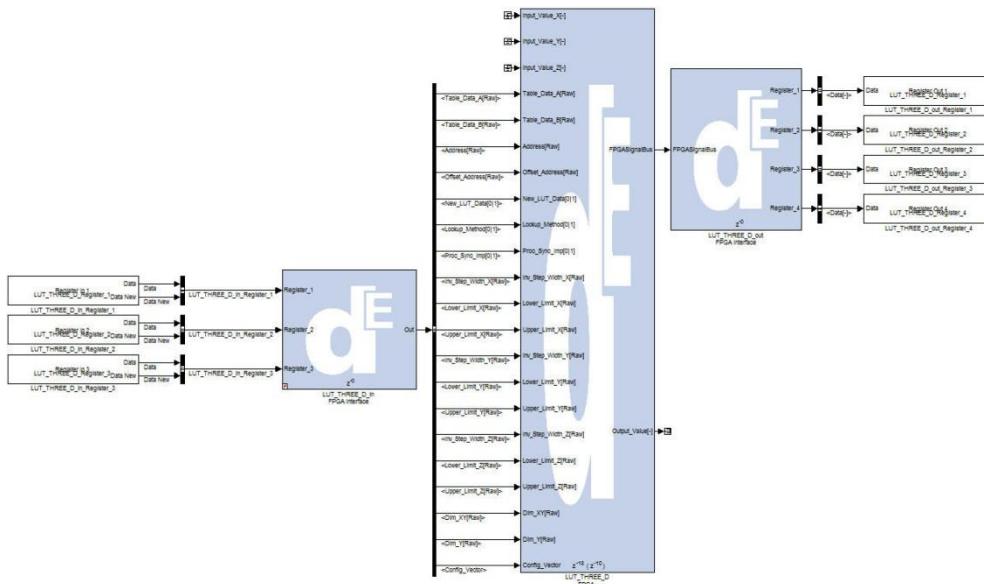
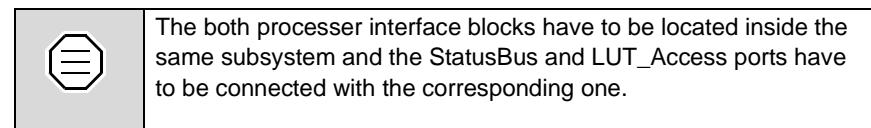
FPGA blocks

Figure 182: FPGA interface

3-Dimensional Look-up Table XF

Objective

The 3D LUT can either be configured with linear interpolation algorithm or with the input below method. These two methods are directly comparable with the standard 3D-LUT from Simulink. The table content is online tunable. The table axis value for x (row index) and y (column index) dimension and the table data has to be inserted on the **processor side**. On the FPGA side, the number of table elements is specified in the GUI, based on the length of table data on the processor side.



The blockset contains the following elements:

- Processor Interface: LUT_THREE_D_XF_out (Processor Interface)
- FPGA Interface: LUT_THREE_D_XF_in (FPGA Interface)
- FPGA: LUT_THREE_D_XF (FPGA Main Component)
- FPGA Interface: LUT_THREE_D_XF_out (FPGA Interface)
- Processor Interface: LUT_THREE_D_XF_in (Processor Interface)

Processor Output

Block

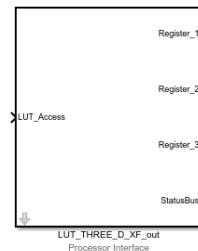


Figure 183: LUT_THREE_D_XF_out block

Block Dialog

The processor output blockset contains the following dialog.

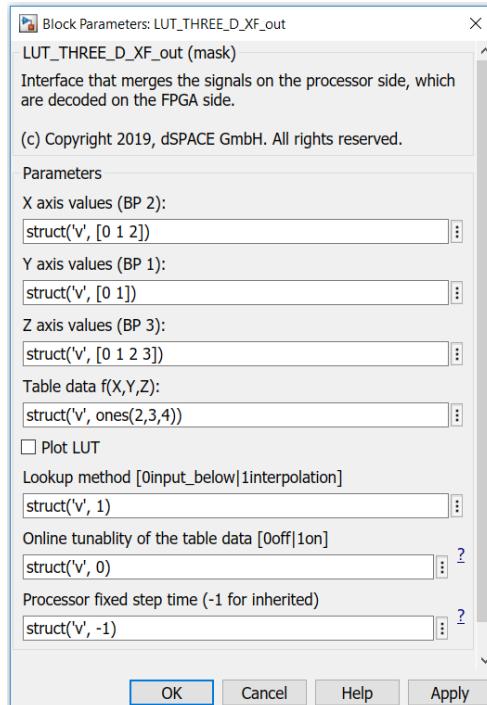
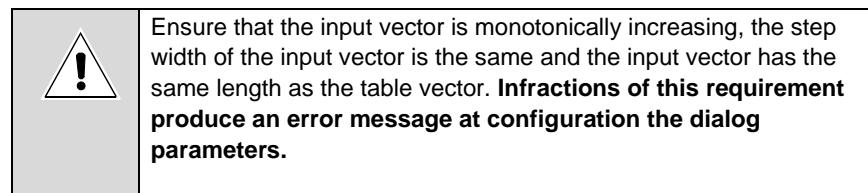


Figure 184: LUT_THREE_D_XF_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
X axis values (BP 2)	[-]	X vector of the input values of the table	-
Y axis values (BP 1)	[-]	Y vector of the input values of the table	-
Z axis values (BP 3)	[-]	Z vector of the input values of the table	-
Table Data f(X,Y,Z)	[-]	Table data of the look-up table	-
Plot LUT	[-]	Function to plot the actual LUT data	-
Lookup method	[-]	0: input below 1: linear interpolation	0 1
Online tunability of the table data	[-]	0: not online tunable 1: online tunable (can cause high turnaround time by huge tables)	0 1
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-

**Input**

The LUT_THREE_D_out block has the following inputs:

Name	Unit	Description	Range
LUT_Acces	[-]	Bus containing the feedback signals from the FPGA coming from LUT_THREE_D_XF_in	-

Output

Via output register the processor out provides the functionality of a simple protocol, together with the processor input interface. For detailed information about the LUT feedback of the port "StatusBus", please refer the chapter "Data store management and feedback of Look-up Table" in the subchapter "LUT Feedback".

FPGA Main Component

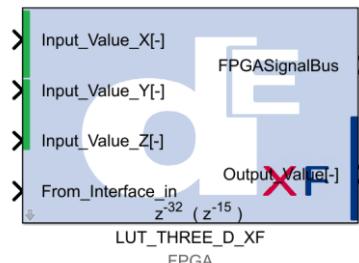
Block

Figure 185: LUT_THREE_D_XF Main block

Block Dialog

The FPGA main blockset contains the following dialog.

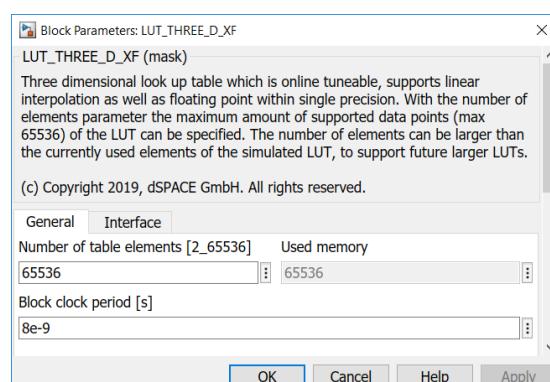


Figure 186: LUT_THREE_D_XF dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Number of table elements	[-]	Amount of table data points	2..65536
Block clock period	[s]	Sample time of the block	-



The parameters of the input and output registers can be defined for automatic interface generation. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation. The block clock period can be configured too.

Input

Following inputs are given to the main block as bus signals:

Name	Unit	Description	Format
Input_Value_X	[-]	FPGA model input X signal	User-defined
Input_Value_Y	[-]	FPGA model input Y signal	User-defined
Input_Value_Z	[-]	FPGA model input Z signal	User-defined
Table_Data A	[-]	Table data port A for parameterization of the LUT	XFloat_8_24
Table_Data B	[-]	Table data port B for parameterization of the LUT	XFloat_8_24
Address	[-]	LUT table address for parameterization	Ufix_15_0
Offset Address	[-]	LUT table offset address for parameterization	UFix_16_0
New_LUT_Data	[-]	Data flag is set to 1 when new table data is available to write to the FPGA memory	UFix_1_0
Lookup_Method	[-]	0: input below 1:interpolation	0 1
Proc.Sync. Impulse	[-]	Processor-synchronous toggle bit	UFix_1_0
Inv_Step_Width_X	[-]	Inverse step width of the input vector which is configured in the processor out block	XFloat_8_24
Lower_Limit_X	[-]	Minimum of the input vector which is configured in the processor out block	XFloat_8_24
Upper_Limit_X	[-]	Maximum of the input vector which is configured in the processor out block	XFloat_8_24
Inv_Step_Width_Y	[-]	Inverse step width of the input vector which is configured in the processor out block	XFloat_8_24
Lower_Limit_Y	[-]	Minimum of the input vector which is configured in the processor out block	XFloat_8_24
Upper_Limit_Y	[-]	Maximum of the input vector which is configured in the processor out block	XFloat_8_24

Inv_Step_Width_Z	[-]	Inverse step width of the input vector which is configured in the processor out block	XFloat_8_24
Lower_Limit_Z	[-]	Minimum of the input vector which is configured in the processor out block	XFloat_8_24
Upper_Limit_Z	[-]	Maximum of the input vector which is configured in the processor out block	XFloat_8_24
Dim_XY	[-]	Dimension or length of the XY-axis values	UFix_16_0
Dim_Y	[-]	Dimension or length of the Y-axis values	UFix_16_0
State_Address[0_3]	[-]	Configuration of the LUT output timing	UFix_2_0

Output

The LUT_THREE_D main block has the following outputs with a maximal latency of 10 for input below and 18 for linear interpolation method:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Output_Value	[-]	Table output value	XFloat_8_24

Processor Input

Block

Adapts the FPGA signals for the processor side. The LUT_Access output port has to be connected to the LUT_Access input port of the LUT_THREE_D_XF_out block. Both blocks must be located in the same subsystem.

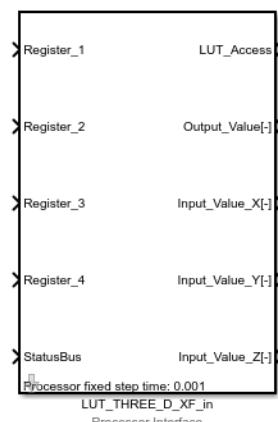


Figure 187: LUT_THREE_D_XF_in block

Block Dialog The dialog only provides a short block description.

Input The processor in block provides the functionality of a simple protocol, together with the processor output interface.

Output The processor input block has the following outputs:

Name	Unit	Description	Format
LUT Access	Bus	Bus containing the feedback signals coming from the FPGA to LUT_THREE_D_XF_out	-
Input_Value_X	[-]	Input value for X-axis of the FPGA LUT	XFloat_8_24
Input_Value_Y	[-]	Input value for Y-axis of the FPGA LUT	XFloat_8_24
Input_Value_Z	[-]	Input value for Z-axis of the FPGA LUT	XFloat_8_24
Output_Value	[-]	Output value of the FPGA LUT	XFloat_8_24

Interface Examples

Processor blocks

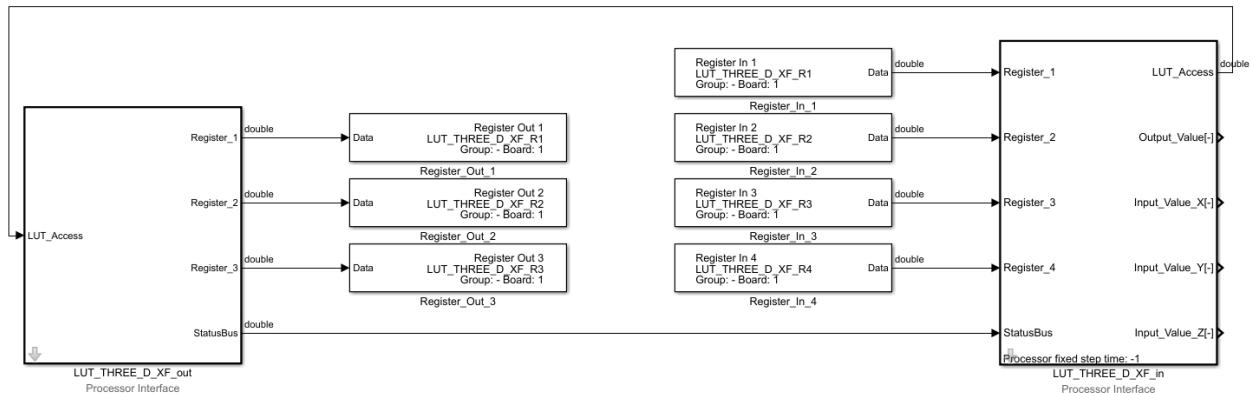


Figure 188: Processor interface

FPGA blocks

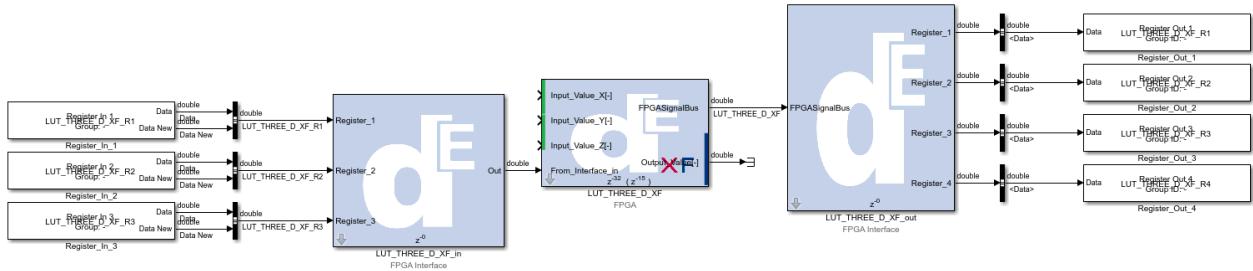


Figure 189: FPGA interface

SCOPE

Objective

The Scope is able to capture two FPGA signals with a time resolution of one FPGA sample step. When longer sequences shall be captured the downsampling factor can be increased, which leads to a loss of time accuracy. The length of the record (Scope depth) is user specific and has to be set inside the FPGA main component GUI. The recorded time length is calculated after the equation:

$$t_{rec_FPGA} = Scope\ depth \cdot \text{downsampling} \cdot t_{sample_FPGA}$$

The scope starts recording on a rising edge for a fulfilled trigger condition. The trigger condition can be directly related to one of the input signals, to an external trigger or to manual trigger which can be set from the processor side. When a pre- or post-trigger delay is defined, the capturing of signals starts with the defined delay before/after the trigger condition becomes true. After the recording is finished, the recorded data and time stamp (with trigger event as $t = 0$) is sent within the processor clock rate to the processor. From here the "time" and "Y_Axis_0/1" signals can be put to a XY-Plotter inside ControlDesk to visualize the recorded data. A Host Service trigger signal must be mapped to the Trigger output port with a reference value > 0 and a trigger condition on the positive edge, like shown below.

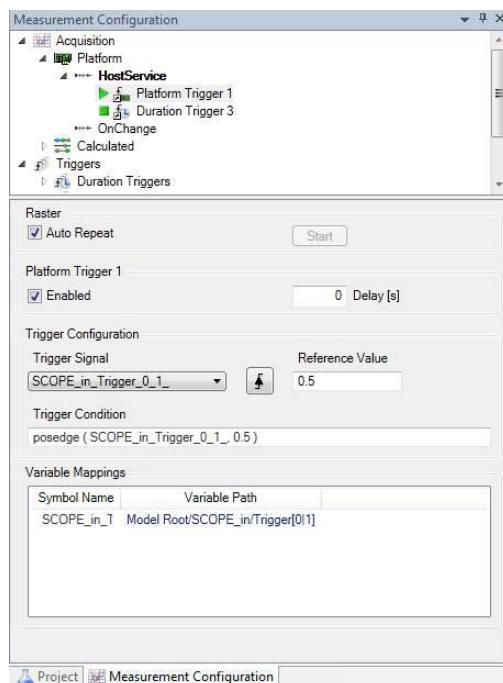


Figure 190: Scope Platform Trigger settings in Control Desk

The Duration Trigger (in ControlDesk) has to be set to the value of Duration_XY_Plotter which you can find inside the MeasureSettingsBus. It follows the equation

$$t_{Duration_Proc} = Scope\ depth \cdot t_{sample_Proc}$$

	When using the scope functionality on an analog input signal, you can set the downsampling factor up to 10 without losing information.
	If the standard interface of the XSG Utils Library is used, make sure that the SCOPE_in and SCOPE_out block of the processor interface is located in the same subsystem, otherwise an error will raise.
	Custom library blocks for ControlDesk are available. For detailed information please refer to the ControlDesk documentation.

The blockset contains the following elements:

- Processor Interface: SCOPE_out (Processor Interface)
- FPGA Interface: SCOPE_in (FPGA Interface)
- FPGA: SCOPE (FPGA Main Component)
- FPGA Interface: SCOPE_out (FPGA Interface)
- Processor Interface: SCOPE_in (Processor Interface)

Processor Output

Block

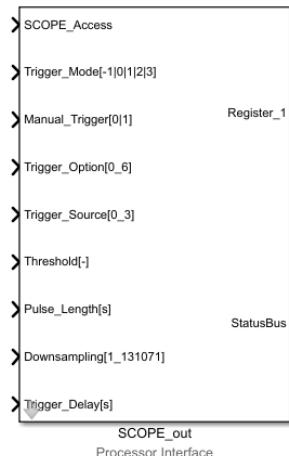


Figure 191: SCOPE_out block

Block Dialog

The processor output blockset contains the following dialog.

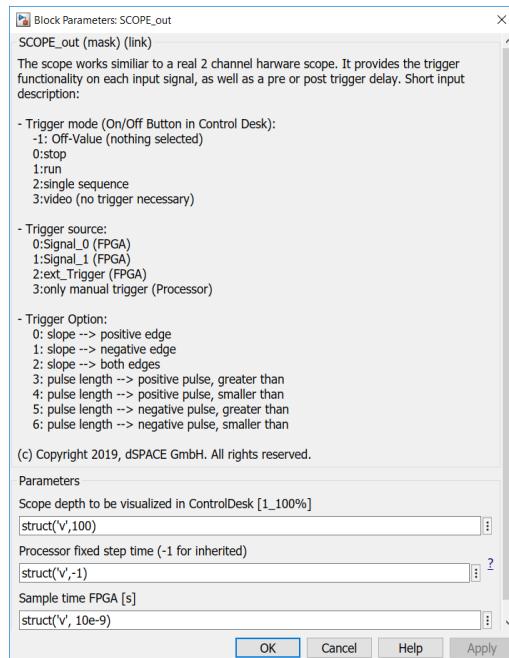


Figure 192: SCOPE_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Scope depth to be visualized in ControlDesk	[%]	Duration of the XY Plotter. The amount of data to be visualized in ControlDesk can be specified here.	1...100
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-
Sample time FPGA	[s]	Sample time of the FPGA framework	-

Input

The SCOPE_out block has the following inputs:

Name	Unit	Description	Range
SCOPE_Access	[-]	Feedback bus from SCOPE_in	
Trigger_Mode	[-]	Different trigger modes are support, like on real oscilloscope -1: default value 0: stop 1: run 2: single sequence 3: video	-1 0 1 2 3
Manual_Trigger	[-]	Rising edge on this port starts the data record, when scope is in armed mode	0 1
Trigger_Option	[-]	Configure the trigger behavior of the scope 0: slope --> positive edge 1: slope --> negative edge 2: slope --> both edges 3: pulse length --> positive pulse, greater than 4: pulse length --> positive pulse, smaller than 5: pulse length --> negative pulse, greater than 6: pulse length --> negative pulse, smaller than	0 ... 6
Trigger_Source	[-]	Trigger sources can be input ports from FPGA or Manual_Trigger. 0: Signal_0 (on FPGA) 1: Signal_1 (on FPGA) 2: ext_Trigger (on FPGA) 3: only Manual_Trigger is supported (on Proc)	0 ... 3
Threshold	[-]	Threshold for the trigger level	Float_11_53
Pulse_Length	[s]	Defined pulse length for detecting a trigger event	$(2^{20}-1) * T_{FPGA}$

Downsampling	[-]	Downsampling factor of the scope	1 ... 131071
Trigger_Delay	[s]	Pre- or post-trigger time for the record with t = 0 @ trigger event	

Output

The processor out block has got one output register which provides the functionality of a simple protocol, together with the processor input registers. Because of the complex visualization, no register overview is shown here.

FPGA Main Component

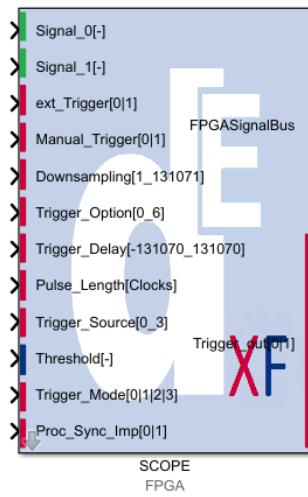
Block

Figure 193: SCOPE Main block

Block Dialog

The FPGA main blockset contains the following dialog.

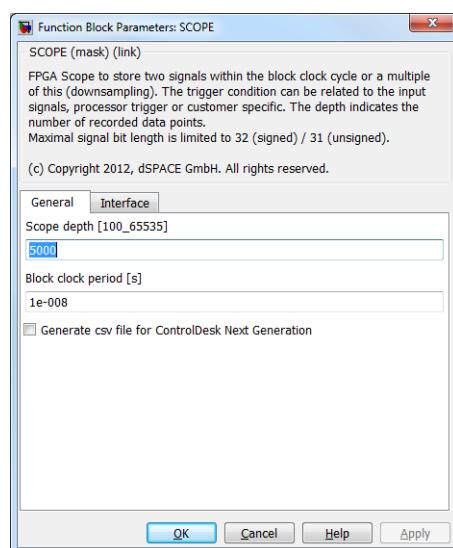


Figure 194: SCOPE dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Format
Scope Depth	[-]	Depth of the FPGA memory reserved for capturing data	UINT 100 ... 65536
Block clock period	[s]	Clock period of the block	User-defined
Generate csv file for ControlDesk	[-]	Button to start a generation of a csv settings file for easily setup the SelectionBox in ConrolDesk. For further information, please refer to the ControlDesk documentation.	



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Signal_0	[-]	Measured data 0	User-defined Max. data format: Fix_32_x; UFix_31_x; Float_8_24
Signal_1	[-]	Measured data 1	User-defined Max. data format: Fix_32_x; UFix_31_x; Float_8_24
ext_Trigger	[-]	External trigger, e.g. for sync with other scopes	UFix_1_0
Manual_Trigger	[-]	Trigger value from proc.	UFix_1_0
Downsampling	[-]	Downsampling factor	UFix_17_0
Trigger_Option	[-]	Trigger option to select trigger event	UFix_3_0
Trigger_Delay	[-]	Pre- or post-trigger delay in sample steps	Fix_28_0
Pulse_Length	[Clocks]	Pulse length to define trigger event	UFix_20_0
Trigger_Source	[-]	Trigger sources can be input ports from FPGA or Manual_Trigger. 0: Signal_0 (on FPGA) 1: Signal_1 (on FPGA) 2: ext_Trigger (on FPGA) 3: Manual_Trigger (on Proc)	UFix_2_0
Threshold	[-]	Trigger level for the selected Trigger_Source	Float_11_53
Trigger_Mode	[-]	Different trigger modes are support, like on real oscilloscope 0: stop 1: run 2: single sequence 3: video	UFix_2_0
Proc_Sync_Imp	[-]	Processor-synchronous impulse	UFix_1_0



The muxed data format of Signal_0 and Signal_1 with precision "full" must not be greater than 32 bits, otherwise an error occurs

Output

The SCOPE main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Trigger_out	[-]	Start trigger event of this Scope, can be used to trigger other Scopes (sync.)	Bool

Processor Input

Block

Adapts the FPGA signals for the processor side and provides the signals for the ControlDesk XY-Plotter.

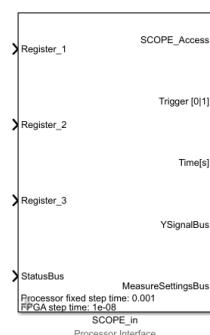


Figure 195: SCOPE_in block

Block Dialog

The following parameters can be configured in the processor input dialog:

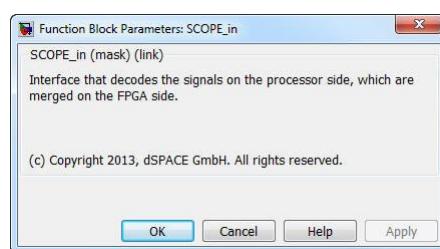


Figure 196: SCOPE_in dialog

Input

The processor in block has got three input registers which provide the functionality of a simple protocol, together with the processor input register. Because of the complex visualization, no register overview is shown here:

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
SCOPE_Access	Bus	Has to be connected to SCOPE_out block, contains parameterization information	-
Trigger	[·]	Trigger event for ControlDesk XY-Plotter Host-Service	0 1
Time	[s]	XY-Plotter time base (x-axis)	-
YSignalBus	Bus	Contains the Y-values for Signal_0/1. XY-Plotter y-axis values	-
MeasureSettings Bus	Bus	Scope_Depth = measured data points Actual_Trigger_Delay Duration → for FPGA record Duration_XY_Plotter → Has to be connected to the XY-Plotter Duration Trigger Host Service Read_Process = actual read process for this measurement Scope_Process_Feedback = actual Scope mode	-

Interface Examples

Processor blocks

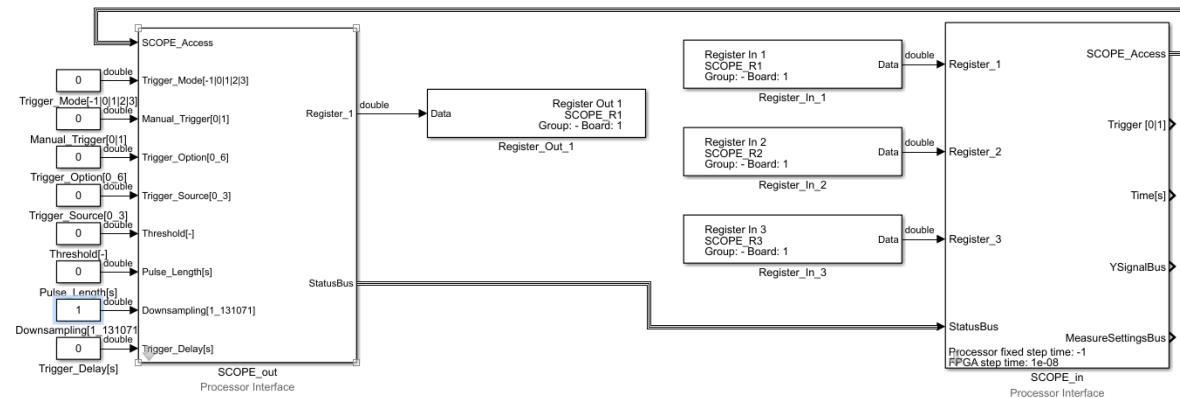


Figure 197: Processor interface

FPGA blocks

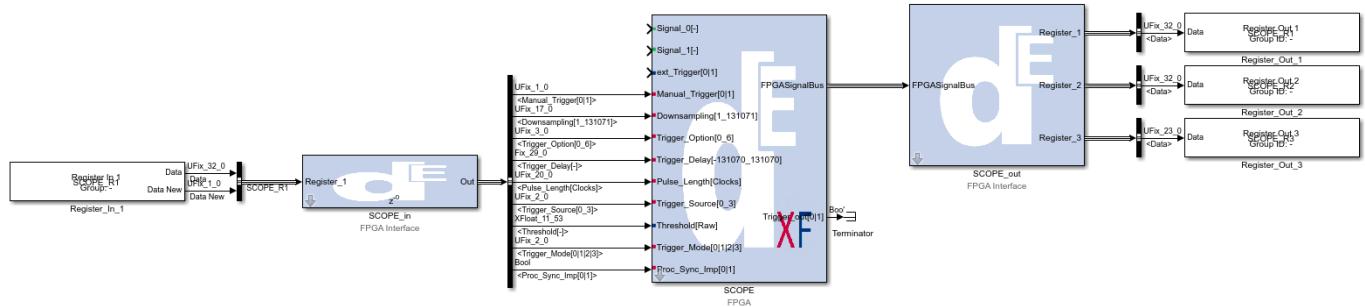


Figure 198: FPGA interface

MULTI_SCOPE

Objective

The [MULTI_SCOPE](#) works similar to the SCOPE, for detailed information and information about the XY-Plotter in ControlDesk please refer to the chapter SCOPE. Here only the differences are explained. The MULTI_SCOPE provides **8 channels** to capture data fully parallel. All channels are triggered synchronously, either by processor (Manual Trigger) or by FPGA caused trigger (signal0 ... 15, or external). The MULTI_SCOPE provides a selection of **8 signals out 16** which can be captured in parallel. The signal to capture channel mapping is changeable during runtime. Below you can find the picture which shows the switch ability of the captured signals, here as an example for the capture channel 0.

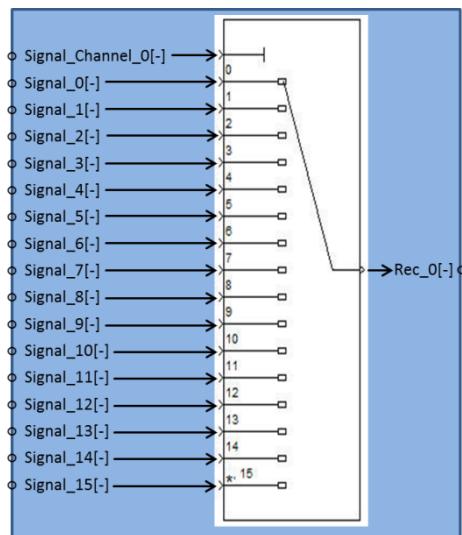


Figure 199: Signal to recorder mapping

The necessary internal used BLOCK_RAM depends of the maximum overall used bit length of all input signals of the switch.

	When using the scope functionality on an analog input signal, you can set the downsampling factor up to 10 without losing information
	If the standard interface of the XSG Utils Library is used, make sure that the MULTI_SCOPE_in and MULTI_SCOPE_out block of the processor interface is located in the same subsystem and are connected, otherwise an error will raise
	Custom library blocks for ControlDesk are available. For detailed information please refer to the ControlDesk documentation.

The blockset contains the following elements:

- Processor Interface: MULTI_SCOPE_out (Processor Interface)
- FPGA Interface: MULTI_SCOPE_in (FPGA Interface)
- FPGA: MULTI_SCOPE (FPGA Main Component)

- FPGA Interface: MULTI_SCOPE_out (FPGA Interface)
- Processor Interface: MULTI_SCOPE_in (Processor Interface)

Processor Output

Block

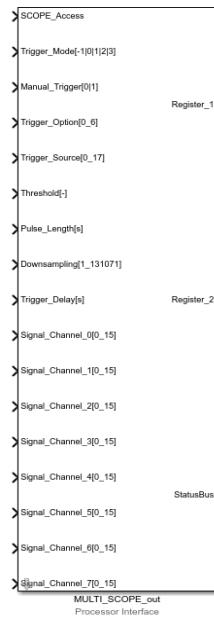


Figure 200: MULTI_SCOPE_out block

Block Dialog

The processor output blockset contains the following dialog.

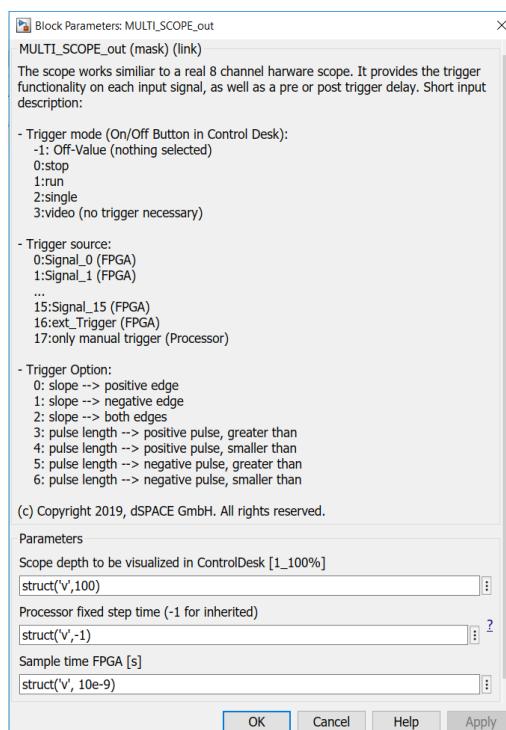


Figure 201: MULTI_SCOPE_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Scope depth to be visualized in ControlDesk	[%]	Duration of the XY Plotter. The amount of data to be visualized in ControlDesk can be specified here.	1...100
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-
Sample time FPGA	[s]	Sample time of the FPGA framework	-

Input

The MULTI_SCOPE_out block has the following inputs:

Name	Unit	Description	Range
SCOPE_Access	[-]	Feedback bus from MULTI_SCOPE_in	
Trigger_Mode	[-]	Different trigger modes are support, like on real hw oscilloscope -1: default value 0: stop 1: run 2: single sequence 3: video	-1 0 1 2 3
Manual_Trigger	[-]	Rising edge on this port starts the data record, when scope is in armed mode	0 1
Trigger_Option	[-]	Configure the trigger behavior of the scope 0: slope --> positive edge 1: slope --> negative edge 2: slope --> both edges 3: pulse length --> positive pulse, greater than 4: pulse length --> positive pulse, smaller than 5: pulse length --> negative pulse, greater than 6: pulse length --> negative pulse, smaller than	0 ... 6
Trigger_Source	[-]	Trigger sources can be input ports from FPGA or Manual_Trigger. 0: Signal_0 (on FPGA) 1: Signal_1 (on FPGA) ... 15: Signal_15 (on FPGA) 16: ext_Trigger (on FPGA) 17: Manual_Trigger (on Proc)	0 ... 17
Threshold	[-]	Threshold for the trigger level	Float_11_53

Pulse_Length	[s]	Defined pulse length for detecting a trigger event	$(2^{20}-1) * T_{FPGA}$
Downsampling	[-]	Downsampling factor of the scope	1 ... 131071
Trigger_Delay	[s]	Pre- or post-trigger time for the record with $t = 0$ @ trigger event	
Signal_Channel_0 ... 7	[-]	The signal number which shall be mapped to the internal scope record channels.	0 ... 15

Output

The processor out block has got two output registers which provide the functionality of a simple protocol, together with the processor input registers. Because of the complex visualization, no register overview is shown here:

FPGA Main Component

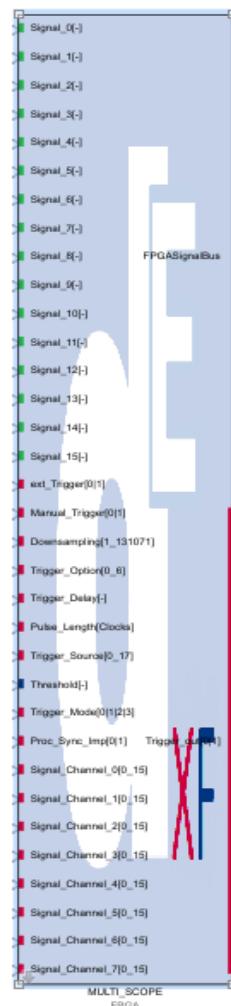
Block

Figure 202: MULTI_SCOPE Main block

Block Dialog

The FPGA main blockset contains the following dialog.

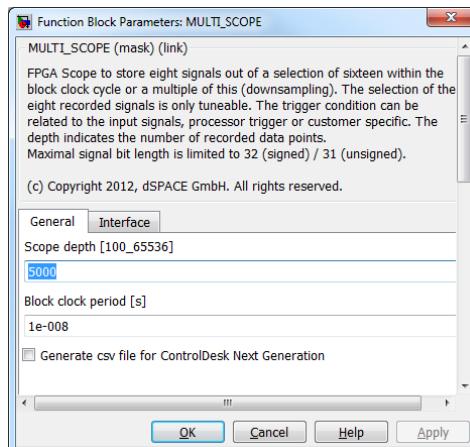


Figure 203: MULTI_SCOPE dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Format
Scope Depth	[-]	Depth of the FPGA memory reserved for capturing data	UINT 100 ... 65536
Block clock period	[s]	Clock period of the block	User-defined
Generate csv file for ControlDesk	[-]	Button to start a generation of a csv settings file for easily setup the SelectionBox in ConrolDesk. For further information, please refer to the ControlDesk documentation.	



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Signal_0 ... Signal_15	[-]	Signal input	User-defined Max. data format: Fix_32_x; UFix_31_x; Float_8_24;
ext_Trigger	[-]	External trigger, e.g. for sync with other scopes	UFix_1_0
Manual_Trigger	[-]	Trigger value from proc.	UFix_1_0
Downsampling	[-]	Downsampling factor	UFix_17_0
Trigger_Option	[-]	Trigger option to select trigger event	UFix_3_0
Trigger_Delay	[-]	Pre- or post-trigger delay in sample steps	Fix_28_0
Pulse_Length	[Clocks]	Pulse length to define trigger event	UFix_20_0
Trigger_Source	[-]	Trigger sources can be input ports from FPGA or Manual_Trigger. 0: Signal_0 (on FPGA) 1: Signal_1 (on FPGA) ... 15: Signal_15 (on FPGA) 16: ext_Trigger (on FPGA) 17: Manual_Trigger (on Proc)	UFix_5_0
Threshold	[-]	Trigger level for the selected Trigger_Source	Float_11_53
Trigger_Mode	[-]	Different trigger modes are support, like on real oscilloscope 0: stop 1: run 2: single sequence 3: video	UFix_2_0
Proc_Sync_Imp	[-]	Processor-synchronous impulse	UFix_1_0
Signal_Channel_0 ...7	[-]	Signal on record channel 0 ... 7	UFix_4_0



The muxed data format of Signal_0 and Signal_1 with precision "full" must not be greater than 32 bits, otherwise an error occurs.

Output

The MULTI_SCOPE main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Trigger_out	[-]	Start trigger event of this Scope, can be used to trigger other Scopes (sync.)	0 1

Processor Input

Block

Adapts the FPGA signals for the processor side and provides the signals for the ControlDesk XY-Plotter.

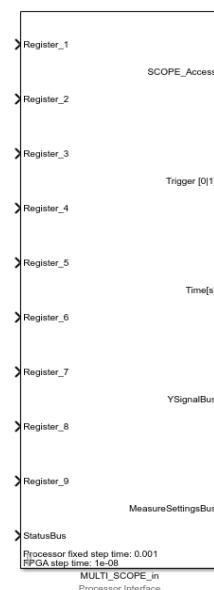


Figure 204: MULTI_SCOPE_in block

Block Dialog

The following parameters can be configured in the processor input dialog:

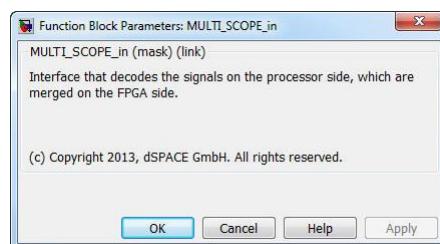


Figure 205: MULTI_SCOPE_in dialog

Input

The processor in block has got nine input registers which provide the functionality of a simple protocol, together with the processor output registers. Because of the complex visualization, no register overview is shown here:

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Output_Channel_0 ... Output_Channel_7	Bus	Value output bus of each channel; e.g. channel 0: X_Axis_0: time axis of the measured data Y_Axis_0: value axis of the measured data Trigger_0: Trigger signal of the downloading process	-

Interface Examples

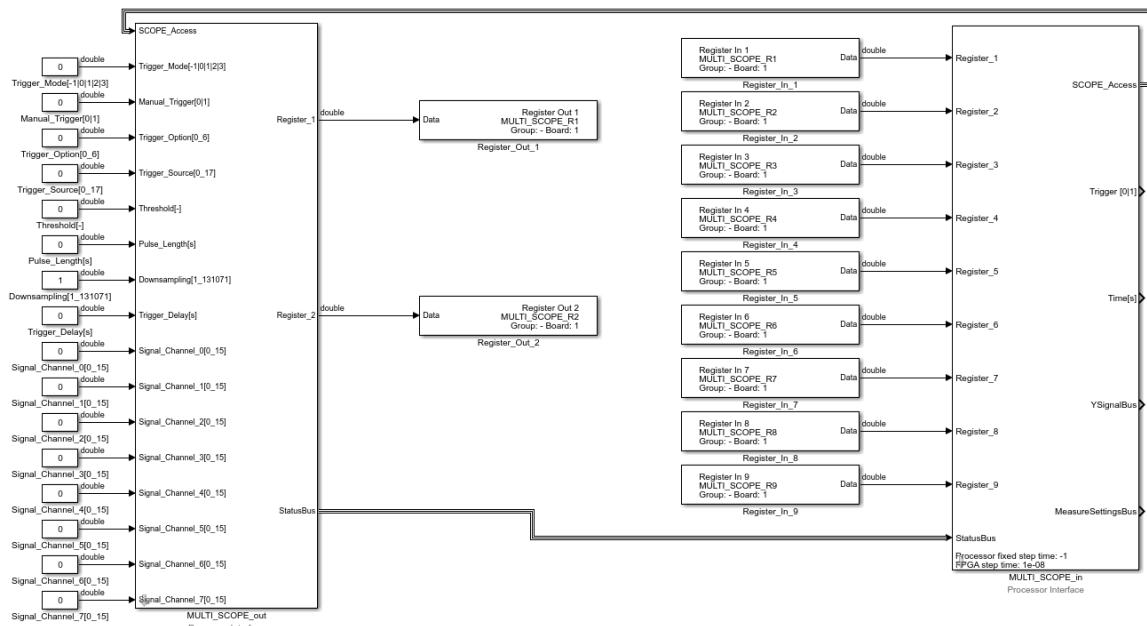
Processor blocks

Figure 206: Processor interface

FPGA blocks

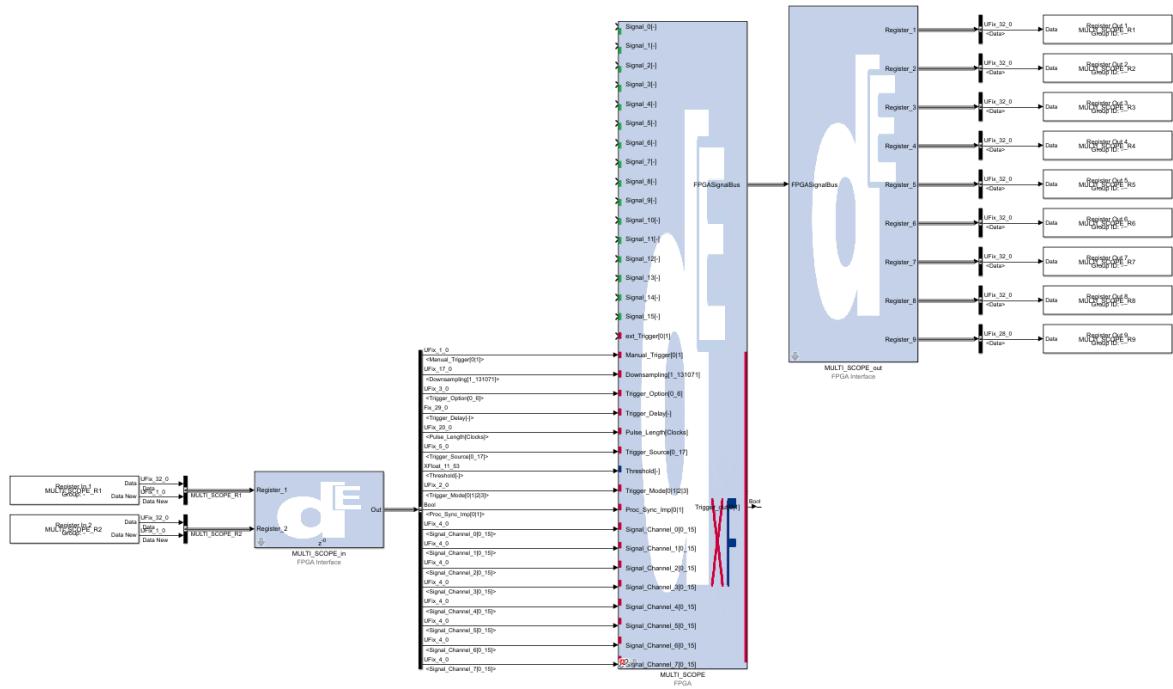


Figure 207: FPGA interface

Average Calculation

Objective

The average calculation blockset provides a processor-synchronous average value of three user-defined FPGA signals. On the processor side a bit toggles with every sample step (processor-synchronous impulse (Proc_Sync_Imp)), and on the FPGA side, the time is measured and the signal values are accumulated between two edges of the Proc_Sync_Imp. Downsampling is also possible to enlarge the capture time, but keep in mind that the resolution might decrease.



When averaging only analog input signals, you can set the downsampling factor up to ten without losing information.

The blockset contains the following elements:

- Processor Interface: THREE_SIGNAL_SYNCRONOUS_AVERAGE_out (Processor Interface)
- FPGA Interface: THREE_SIGNAL_SYNCRONOUS_AVERAGE_in (FPGA Interface)
- FPGA: THREE_SIGNAL_SYNCRONOUS_AVERAGE (FPGA Main Component)
- FPGA Interface: THREE_SIGNAL_SYNCRONOUS_AVERAGE_out (FPGA Interface)
- Processor Interface: THREE_SIGNAL_SYNCRONOUS_AVERAGE_in (Processor Interface)

Processor Output

Block

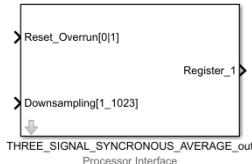


Figure 208: THREE_SIGNAL_SYNCRONOUS_AVERAGE_out block

Block Dialog

The processor output blockset contains the following dialog.

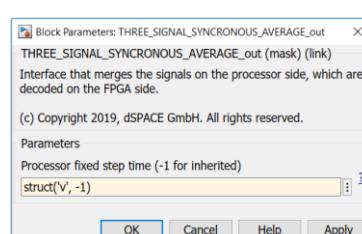


Figure 209: THREE_SIGNAL_SYNCRONOUS_AVERAGE_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-

Input

The THREE_SIGNAL_SYNCRONOUS_AVERAGE_out block has the following inputs:

Name	Unit	Description	Range
Reset_OVERRUN	[-]	Reset of the accumulated values on the FPGA	0 1
Downsampling	[-]	Downsampling of the accumulated values	1 ... 1023

FPGA Main Component

Block

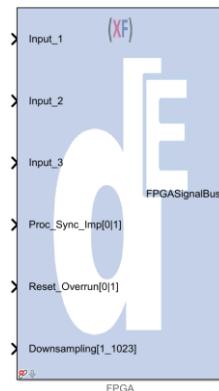


Figure 210: THREE_SIGNAL_SYNCRONOUS_AVERAGE Main block for fixed point

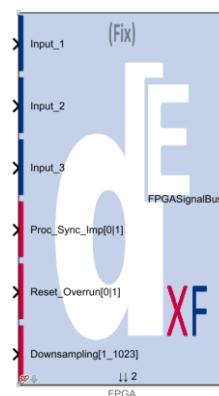


Figure 211: THREE_SIGNAL_SYNCRONOUS_AVERAGE Main block for floating point

Block Dialog

The FPGA main blockset contains the following dialog.

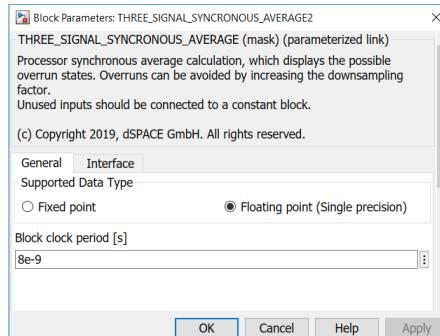


Figure 212: THREE_SIGNAL_SYNCHRONOUS_AVERAGE dialog



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Input_1	[-]	Input signal channel 1	User-defined
Input_2	[-]	Input signal channel 2	User-defined
Input_3	[-]	Input signal channel 3	User-defined
Proc_Sync_Imp	[-]	Processor-synchronous impulse	UFix_1_0
Reset_OVERRUN	[-]	Reset of the accumulator of the input signals	Bool
Downsampling	[-]	Downsampling factor of the accumulator of the input signals	UFix_10_0

Output

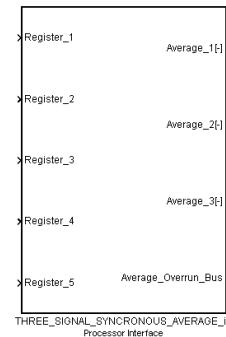
The THREE_SIGNAL_SYNCHRONOUS_AVERAGE main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-

Processor Input

Block

Adapts the FPGA signals for the processor side

**Figure 213: THREE_SIGNAL_SYNCHRONOUS_AVERAGE_in block****Block Dialog**

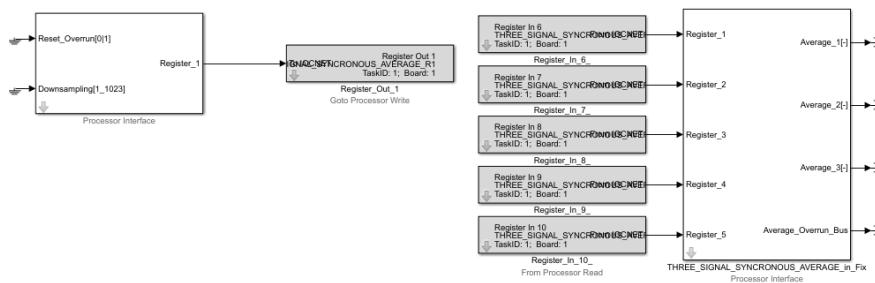
The dialog only provides a short block description.

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Average_1	[-]	Average value of the input 1	-
Average_2	[-]	Average value of the input 2	-
Average_3	[-]	Average value of the input 3	-
Average_OVERRUN_Bus	Bus	Overrun feedback Average_OVERRUN_channel1 Average_OVERRUN_channel2 Average_OVERRUN_channel3	0 1

Interface Examples

Processor blocks**Figure 214: Processor interface****FPGA blocks**

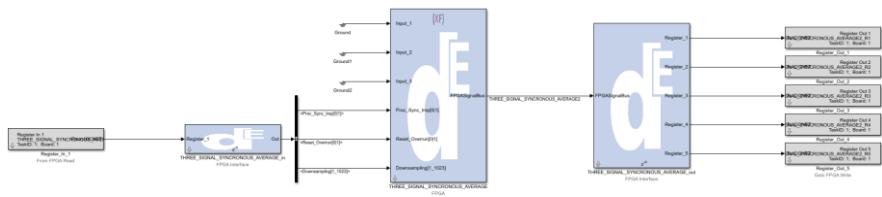


Figure 215: FPGA interface for fixed point

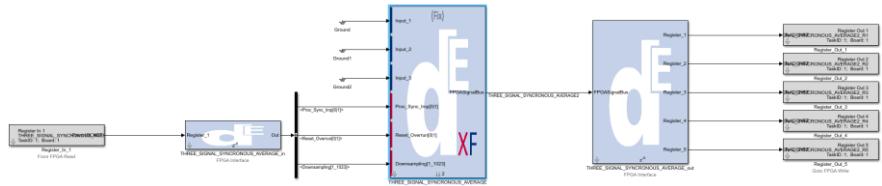


Figure 216: FPGA interface for floating point

Sine Generator

Objective

The sine generator blockset lets you generate a user-defined one- or three-phase fast sine signal over a processor-based user interface. Next to the frequency, the gain, offset and phase shift can be also be configured during runtime. The FPGA main component outputs the three sine signals in an normed format which is optimal for the DAC channels, only a Reinterpret block has to be added with a forced binary point of 0.



The following chapter describes only the behavior of the three-phase component. The behavior of the one-phase component is the same but with a different number of registers.

The blockset contains the following elements:

- Processor Interface: THREE_SINE_GENERATOR_out (Processor Interface)
- FPGA Interface: THREE_SINE_GENERATOR_in (FPGA Interface)
- FPGA: THREE_SINE_GENERATOR (FPGA Main Component)

Processor Output

Block

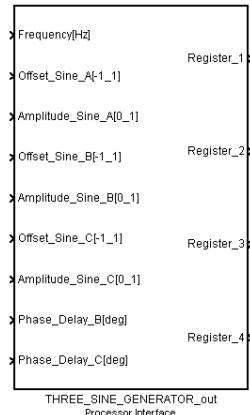


Figure 217: THREE_SINE_GENERATOR_out block

Block Dialog

The processor output blockset contains the following dialog.

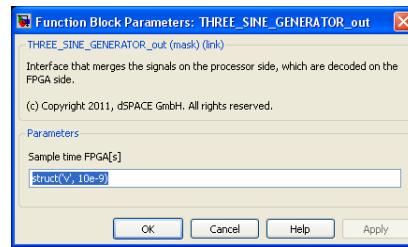


Figure 218: THREE_SINE_GENERATOR_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Sample time of the FPGA	[s]	Step time of the FPGA	-

Input

The THREE_SINE_GENERATOR_out block has the following inputs:

Name	Unit	Description	Range
Frequency	[Hz]	Frequency of the sine	390.5 ... 0 kHz with a minimal step size of 95.3 Hz
Offset_Sine_A	[-]	Normalized offset of sine channel A	[-1 ... 1] with a minimal step size of 0.002
Gain_Sine_A	[-]	Gain of sine channel A	[0 ... 1] with a minimal step size of 0.001
Offset_Sine_B	[-]	Normalized offset of sine channel B	[-1 ... 1] with a minimal step size of 0.002
Gain_Sine_B	[-]	Gain of sine channel B	[0 ... 1] with a minimal step size of 0.001
Offset_Sine_C	[-]	Normalized offset of sine channel C	[-1 ... 1] with a minimal step size of 0.002
Gain_Sine_C	[-]	Gain of sine channel C	[0 ... 1] with a minimal step size of 0.001

Output

The processor out block provides four output registers whose sectioning is shown below:

Register 1



Figure 219: THREE_SINE_GENERATOR_out Register 1

Name	Used Bits	Description	Range
Increment	0 ... 11	Increment of the weighted frequency	0 ... 4096
Offset Sine Channel A	12 ... 21	Offset of the sine of channel A	0 ... 1023
Gain Sine Channel A	31 ... 22	Gain of the sine of channel A	0 ... 1023

Register 2

Figure 220: THREE_SINE_GENERATOR_out Register 2

Name	Used Bits	Description	Range
Offset Sine Channel B	9 ... 0	Offset of the sine of channel A	0 ... 1023
Gain Sine Channel B	19 ... 10	Gain of the sine of channel A	0 ... 1023
SPARE	31 ... 20		

Register 3

Figure 221: THREE_SINE_GENERATOR_out Register 3

Name	Used Bits	Description	Range
Offset Sine Channel C	9 ... 0	Offset of the sine of channel A	0 ... 1023
Gain Sine Channel C	19 ... 10	Gain of the sine of channel A	0 ... 1023
SPARE	31 ... 20		

Register 4

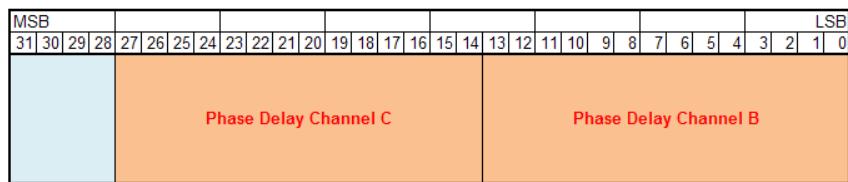


Figure 222: THREE_SINE_GENERATOR_out Register 4

Name	Used Bits	Description	Range
Phase Delay Channel B	13 ... 0	Phase delay of the sine of channel B to channel A	0 ... 4095
Phase Delay Channel C	27 ... 14	Phase delay of the sine of channel C to channel A	0 ... 4095
SPARE	31 ... 28		

FPGA Main Component

Block

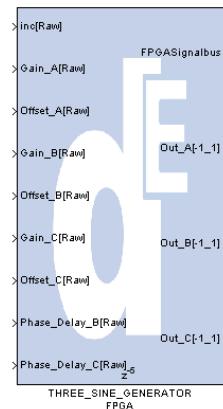


Figure 223: THREE_SINE_GENERATOR Main block

Block Dialog

The FPGA main blockset contains the following dialog.

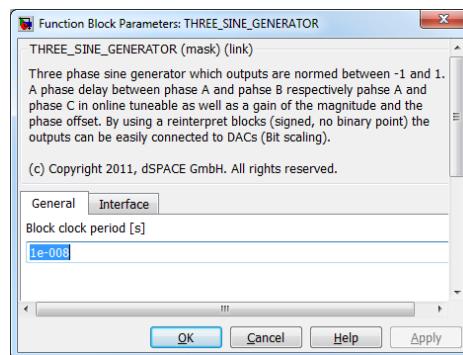


Figure 224: THREE_SINE_GENERATOR dialog



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
inc	[-]	Increment of the weighted frequency of the sine	UFix_12_0
Gain_A	[-]	Gain of the sine of channel A	UFix_10_0
Offset_A	[-]	Offset of the sine of channel A	UFix_10_0
Gain_B	[-]	Gain of the sine of channel A	UFix_10_0
Offset_B	[-]	Offset of the sine of channel A	UFix_10_0
Gain_C	[-]	Gain of the sine of channel C	UFix_10_0
Offset_C	[-]	Offset of the sine of channel C	UFix_10_0
Phase_Delay_B	[-]	Phase delay of the sine of channel B to channel A	UFix_14_0
Phase_Delay_C	[-]	Phase delay of the sine of channel C to channel A	UFix_14_0

Output

The THREE_SINE_GENERATOR main block has the following outputs:

Name	Unit	Description	Range
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Out_A	[·]	Normalized sine output of channel A	Fix_14_13
Out_B	[·]	Normalized sine output of channel B	Fix_14_13
Out_C	[·]	Normalized sine output of channel C	Fix_14_13

Interface Examples

Processor blocks

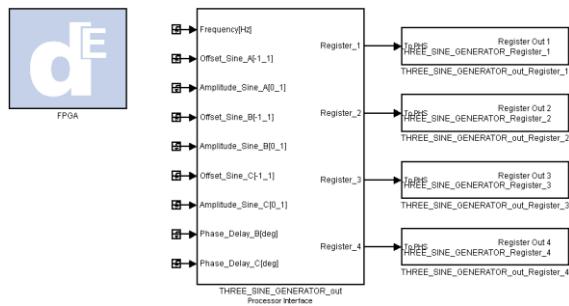


Figure 225: Processor interface

FPGA blocks

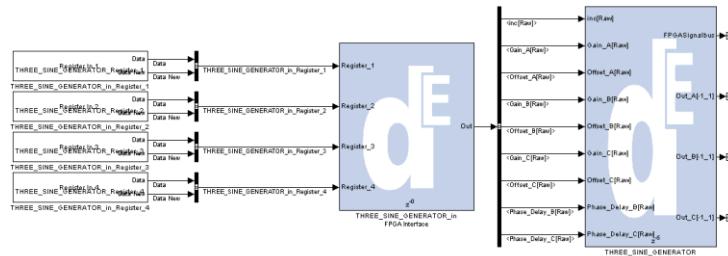


Figure 226: FPGA interface

Discrete PT 1

Objective

The discrete PT 1 blockset lets you configure the filter time of a discrete first-order lag element via the processor interface.

The blockset contains the following elements:

- Processor Interface: DISCRET_PT_ONE_out
(Processor Interface)
- FPGA Interface: DISCRET_PT_ONE_in
(FPGA Interface)
- FPGA: DISCRET_PT_ONE
(FPGA Main Component)

Processor Output

Block

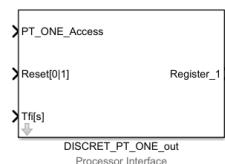


Figure 227: DISCRET_PT_ONE_out block

Block Dialog

The processor output blockset contains the following dialog.

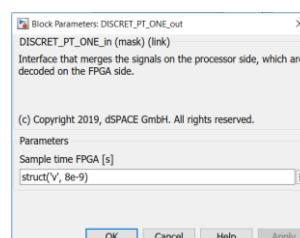


Figure 228: DISCRET_PT_ONE_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Sample time of the FPGA	[s]	Step time of the FPGA	-

Input

The DISCRET_PT_ONE_out block has the following inputs:

Name	Unit	Description	Range
Reset	[-]	Reset of the FPGA integrator used	0 1
Tf	[s]	Filter time of the first-order lag	[0 ... ~10] s (depends on FPGA sample time)

FPGA Main Component

Block

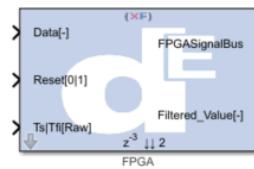


Figure 229: DISCRET_PT_ONE Main block for fixed point

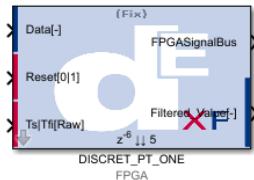


Figure 230: DISCRET_PT_ONE Main block for floating point

Block Dialog

The FPGA main blockset contains the following dialog.

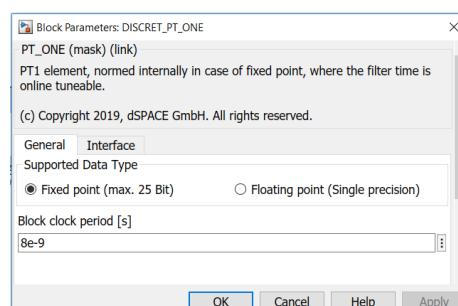


Figure 231: DISCRETE_PT_ONE dialog

You can define the parameters of the input register for automatic interface generation. The dialog also contains a button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Data	[·]	Data input of the first-order lag	User-defined Max. Dataformat: Fix_25_x/XFloat_8_24
Reset	[·]	Reset of the internal integrator	Bool
Tf	[·]	Weighted filter time of the first-order lag	User-defined

Output

The DISRET_PT_ONE main block has the following outputs:

Name	Unit	Description	Range
FilteredValue	[·]	Filtered value of the input data	User-defined; same data format as the input data

Interface Examples

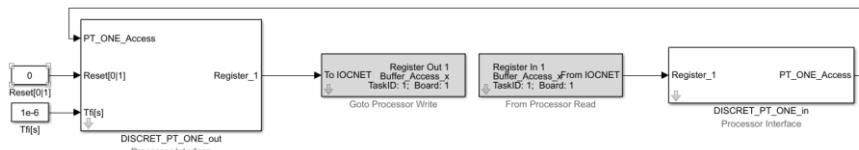
Processor blocks

Figure 232: Processor interface

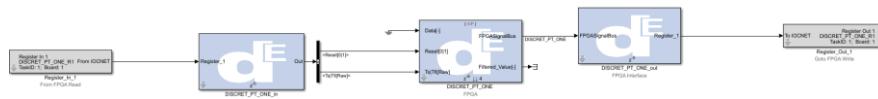
FPGA blocks

Figure 233: FPGA interface for fixed point

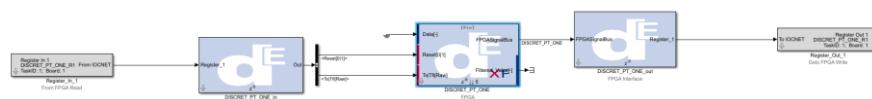


Figure 234: FPGA interface for floating point

Integrator

Objective

The integrator blockset provides a discrete integrator with an internal automatically bitwidth optimization which depends on the output signal. The output bitwidth and the binary point can be easily adjusted over the block dialog.

The blockset contains the following elements:

- Processor Interface: DISCRETE_INTEGRATOR_out
(Processor Interface)
- FPGA Interface: DISCRETE_INTEGRATOR_in
(FPGA Interface)
- FPGA: DISCRETE_INTEGRATOR
(FPGA Main Component)
- FPGA Interface: DISCRETE_INTEGRATOR_out
(FPGA Interface)
- Processor Interface: DISCRETE_INTEGRATOR_in
(Processor Interface)

Processor Output

Block

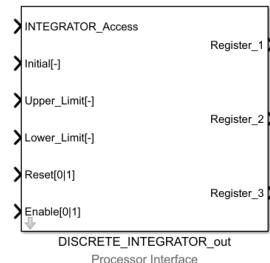


Figure 235: DISCRETE_INTEGRATOR_out block



When the parameter upper limit is lower than the lower limit, the parameters are automatically switched in the DISCRETE_INTEGRATOR_out block. This means that the maximum of upper and lower limit is used as Upper limit and minimum of upper and lower limit is used as Lower limit.

Block Dialog

The processor output blockset contains the following dialog.

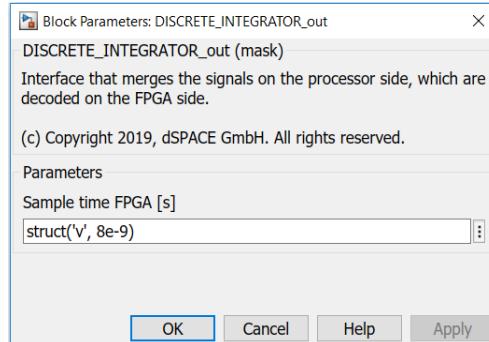


Figure 236: DISCRETE_INTEGRATOR_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Sample time FPGA	[s]	Sample time of the FPGA	-

Input

The DISCRETE_INTEGRATOR_out block has the following inputs:

Name	Unit	Description	Range
INTEGRATOR_Access	BUS	Feedback bus from DISCRETE_INTEGRATOR_in	
Reset	[-]	Reset of the integrator	0 1
Initial	[-]	Initial value of the integrator	Depends on the output bitwidth and binarypoint of the FPGA component
Upper Limit	[-]	Upper limit of the integrator	Depends on the output bitwidth and binarypoint of the FPGA component
Lower Limit	[-]	Lower limit of the integrator	Depends on the output bitwidth and binarypoint of the FPGA component
Enable	[-]	Enables the Integrator	0 1

FPGA Main Component

Block

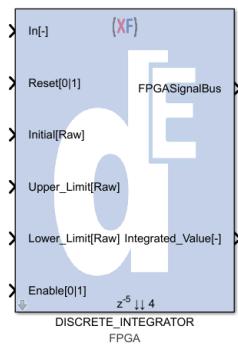


Figure 237: DISCRETE_INTEGRATOR Main block for fixed point

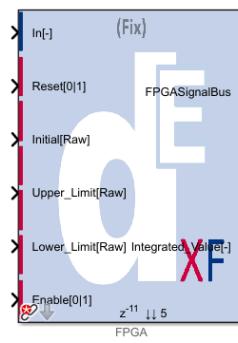


Figure 238: DISCRETE_INTEGRATOR Main block for floating point

Block Dialog

The FPGA main blockset contains the following dialog.

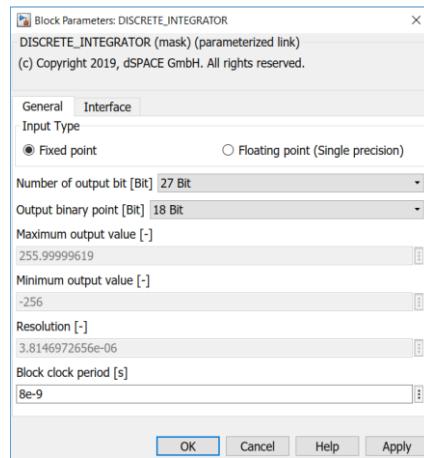


Figure 239: DISCRETE_INTEGRATOR dialog for fixed point

The dialog blockset has the following parameters for fixed point data type:

Name	Unit	Description	Format
Number of output bit	[Bit]	Specify the number of output bits	1 ... 32 Bit
Output binary point	[Bit]	Specify the output binary point	0 ... 31 Bit
Maximum output value	[\cdot]	Passiv output value; Calculate the actual maximum value of the output depending on the Bit settings	-
Minimum output value	[\cdot]	Passiv output value; Calculate the actual minimum value of the output depending on the Bit settings	-
Resolution	[\cdot]	Passiv output value; Calculate the actual resolution of the output depending on the Bit settings	-
Block clock period	[s]	Block clock period	-

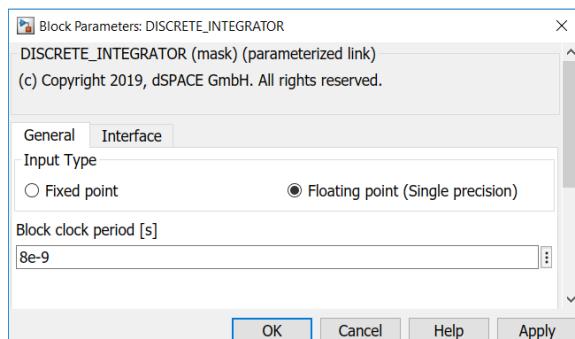


Figure 240: DISCRETE_INTEGRATOR dialog for floating point

The dialog blockset has the following parameters for floating point data type:

Name	Unit	Description	Format
Block clock period	[s]	Block clock period	-

	<p>The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.</p>
--	--

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input signal channel 1	User-defined
Reset	[-]	Reset of the integrator	Bool
Initial	[-]	Initial value of the integrator	User-defined
Upper Limit	[-]	Upper limit of the integrator	User-defined
Lower Limit	[-]	Lower limit of the integrator	User-defined

Output

The DISCRETE_INTEGRATOR main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Integrated Value	[-]	Output of the integrator	(depend on dialog settings)

Processor Input

Block

Adapts the FPGA signals for the processor side



Figure 241: DISCRETE_INTEGRATOR_in block

Block Dialog

The dialog only provides a short block description.

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
INTEGRATOR_Access	Bus	Has to be connected to DISCRETE_INTEGRATOR_out block	-
StatusBus	Bus	Status Bus of the Integrator; Includes the error counter	-

Interface Examples

Processor blocks

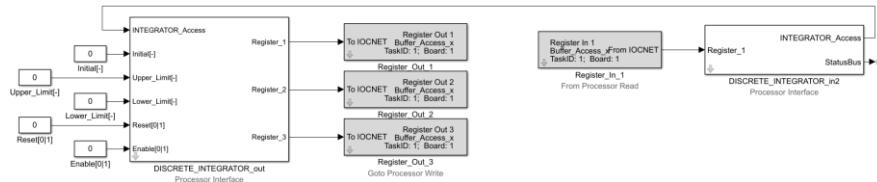


Figure 242: Processor interface

FPGA blocks

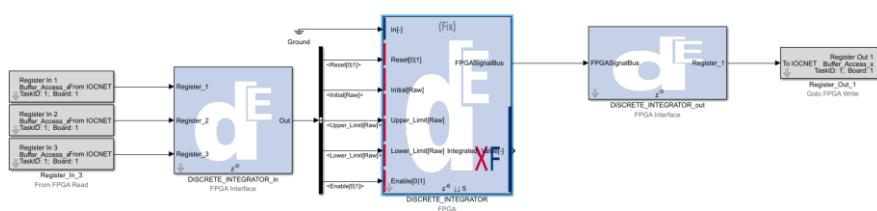


Figure 243: FPGA interface for floating point

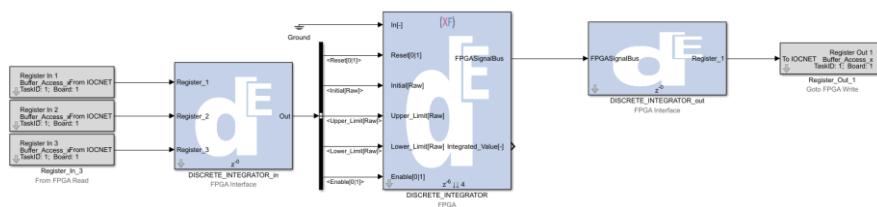


Figure 244: FPGA interface for fixed point

Controller: PI-Controller

Objective

The PI-Controller blockset provides a discrete controller with an internal automatically bitwidth optimization which depends on the output signal. The output bitwidth and the binary point can be easily adjusted over the block dialog. Controller settings like the factors Kp, Ki and the upper and lower limit can be set over the processor interface.

The blockset contains the following elements:

- Processor Interface: PI_CONTROLLER_out
(Processor Interface)
- FPGA Interface: PI_CONTROLLER_in
(FPGA Interface)
- FPGA: PI_CONTROLLER
(FPGA Main Component)
- FPGA Interface: PI_CONTROLLER_out
(FPGA Interface)
- Processor Interface: PI_CONTROLLER_in
(Processor Interface)

Processor Output

Block

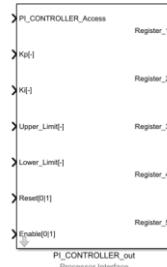


Figure 245: PI_CONTROLLER_out block

Block Dialog

The processor output blockset contains the following dialog.

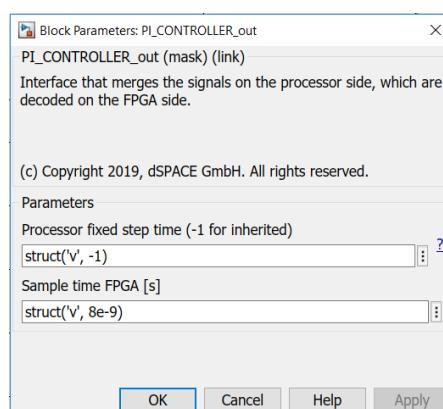


Figure 246: PI_CONTROLLER_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Processor fixed step time	[s]	Step time of the processor task. For offline simulation a different sample time can be set to simulate the processor interface communication behavior.	-
Sample time FPGA	[s]	Sample time of the FPGA	-

Input

The PI_CONTROLLER_out block has the following inputs:

Name	Unit	Description	Range
PI_CONTROLLER_Access	BUS	Feedback bus from PI_CONTROLLER_in	
Kp	[-]	Proportional controller factor Kp	0 ... 2^15-1 resolution: 30,5E-6 for fixed point.
Ki	[-]	Integral controller factor Ki	Depends on the actual factor of Kp and the sample time of the FPGA
Limit_up	[-]	Upper output limit of the Controller	Depends on the output bitwidth and binarypoint of the FPGA component
Limit_low	[-]	Lower output limit of the Controller	Depends on the output bitwidth and binarypoint of the FPGA component
Reset	[-]	Reset of the controller	0 1
Enable	[-]	Enables the controller	0 1

Name	Used Bits	Description	Range
(Ki * T_FPGA * 3) + Kp	29 ... 0	Controller factor (Ki * T_FPGA * 3) + Kp	Depends on the actual factor of Kp and the sample time of the FPGA
SPARE	30		
Reset	31	Reset of the Controller	0 1

FPGA Main Component

Block

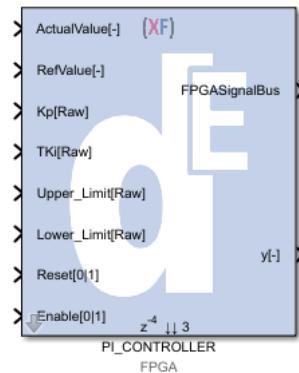


Figure 247: PI_CONTROLLER Main block for fixed point

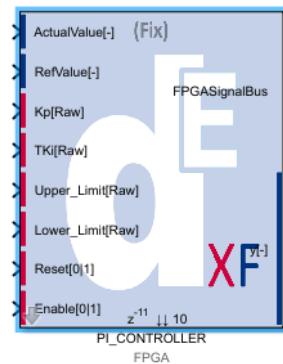


Figure 248: PI_CONTROLLER Main block for floating point

Block Dialog

The FPGA main blockset contains the following dialog.

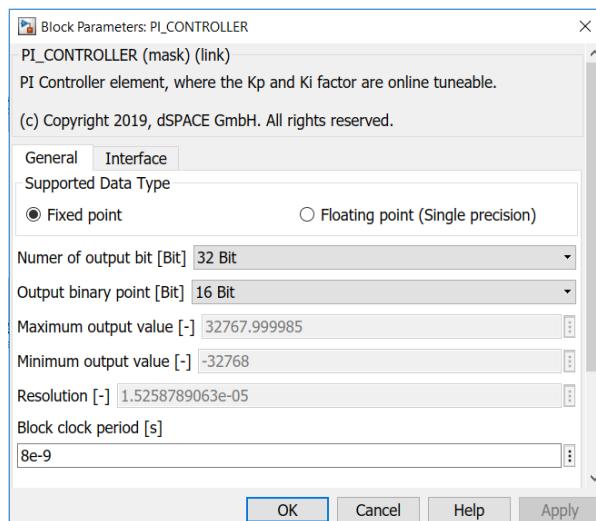


Figure 249: PI_CONTROLLER dialog for fixed point

The dialog blockset has the following parameters:

Name	Unit	Description	Format
Number of output bit	[Bit]	Specify the number of output bits	1 ... 32 Bit
Output binary point	[Bit]	Specify the output binary point	0 ... 31 Bit
Maximum output value	[-]	Passiv output value; Calculate the actual maximum value of the output depending on the Bit settings	-
Minimum output value	[-]	Passiv output value; Calculate the actual minimum value of the output depending on the Bit settings	-
Resolution	[-]	Passiv output value; Calculate the actual resolution of the output depending on the Bit settings	-

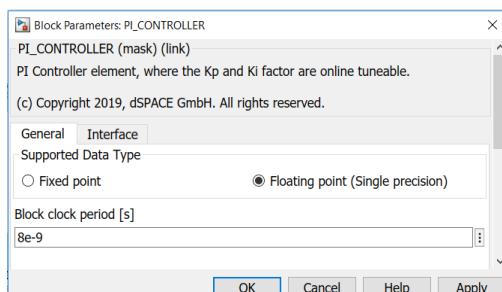


Figure 250: PI_CONTROLLER dialog for floating point

	<p>The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.</p>
--	--

Input

The main block has the following inputs:

Name	Unit	Description	Format
ActualValue	[-]	Actual value	Fix_x_y Max. data format: Fix_32_yor XFloat_8_24
RefValue	[Raw]	Reference value (Reference and actual value must have the same dataformat!)	Fix_x_y Max. data format: Fix_32_yor XFloat_8_24
Kp	[Raw]	Kp factor of the Controller	UFix_32_0
Ki	[Raw]	Controller factor (Ki * T_FPGA * 3) + Kp	UFix_32_0
Limit_up	[Raw]	Upper limit of the controller	UFix_32_0
Limit_low	[Raw]	Lower limit of the controller	UFix_32_0
Reset	[-]	Reset of the integrator	Bool
Enable	[-]	Enables the integrator	Bool

Output

The PI_CONTROLLER main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Out	[-]	Output of the PI-Controller	Fix_x_y/XFloat_8_24 (depend on dialog settings)

Processor Input

Block

Adapts the FPGA signals for the processor side

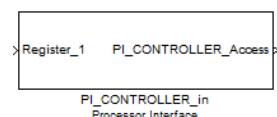


Figure 251: PI_CONTROLLER_in block

Block Dialog

The dialog only provides a short block description.

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
PI_CONTROLLER_Access	Bus	Has to be connected to PI_CONTROLLER_out block	-

Interface Examples

Processor blocks

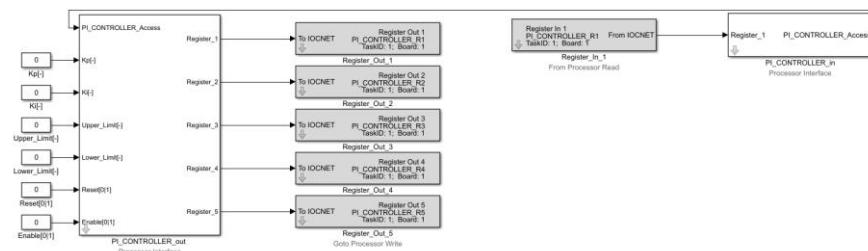


Figure 252: Processor interface

FPGA blocks

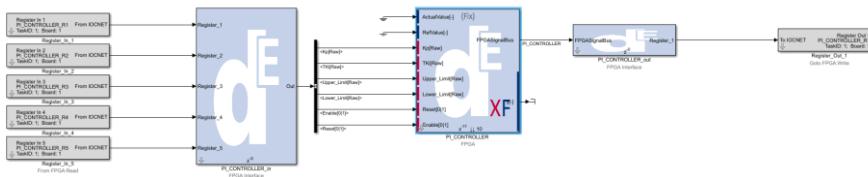


Figure 253: FPGA interface for floating point

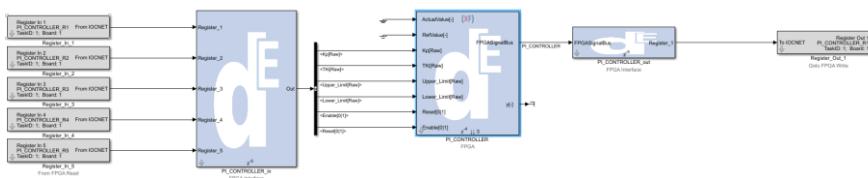


Figure 254: FPGA interface for fixed point

Routing: Dynamic Routing

Objective	The Dynamic Routing blockset provides to change the routing of the input to the output port during runtime. The routing matrix can be set over the processor interface.
------------------	---

The blockset contains the following elements:

- Processor Interface: DYNAMIC_ROUTING_out
(Processor Interface)
- FPGA Interface: DYNAMIC_ROUTING_in
(FPGA Interface)
- FPGA: DYNAMIC_ROUTING
(FPGA Main Component)

Processor Output

Block

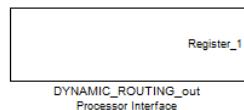


Figure 255: DYNAMIC_ROUTING_out block

Block Dialog	The processor output blockset contains the following dialog.
---------------------	--

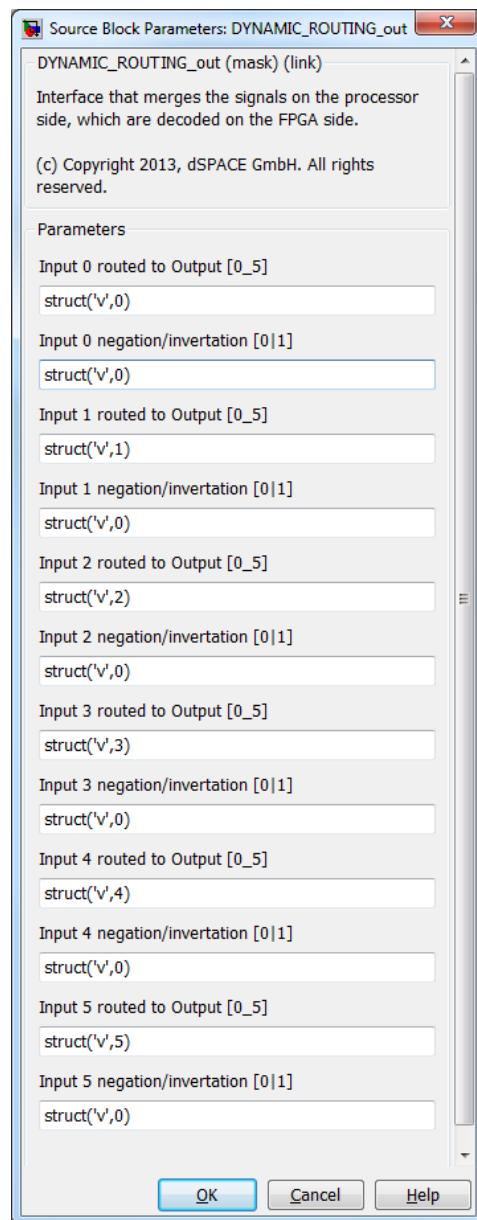


Figure 256: DYNAMIC_ROUTING_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Routing Parameter Input 0	[-]	Routing parameter for Input 0	0 ... 5
Negation/Invertation Input 0	[-]	Negation (analog values) / Invertation (digital values) flag of input 0	0 1
Routing Parameter Input 1	[-]	Routing parameter for Input 1	0 ... 5
Negation/Invertation Input 1	[-]	Negation (analog values) / Invertation (digital values) flag of input 1	0 1
Routing Parameter Input 2	[-]	Routing parameter for Input 2	0 ... 5
Negation/Invertation Input 2	[-]	Negation (analog values) / Invertation (digital values) flag of input 2	0 1
Routing Parameter Input 3	[-]	Routing parameter for Input 3	0 ... 5
Negation/Invertation Input 3	[-]	Negation (analog values) / Invertation (digital values) flag of input 3	0 1
Routing Parameter Input 4	[-]	Routing parameter for Input 4	0 ... 5
Negation/Invertation Input 4	[-]	Negation (analog values) / Invertation (digital values) flag of input 4	0 1
Routing Parameter Input 5	[-]	Routing parameter for Input 5	0 ... 5
Negation/Invertation Input 5	[-]	Negation (analog values) / Invertation (digital values) flag of input 5	0 1

Input

The DYNAMIC_ROUTING_out block has no inputs.

FPGA Main Component

Block

The dynamic routing for fixed and floating point is shown below:

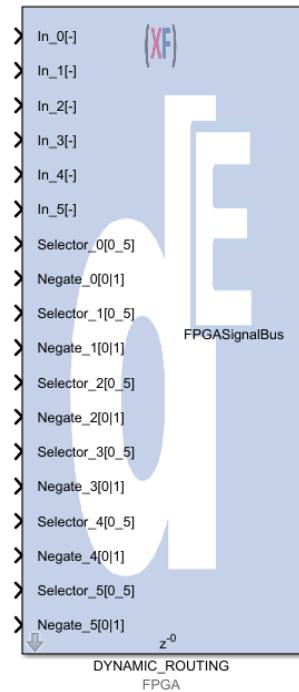


Figure 257: DYNAMIC_ROUTING Main block for fixed point

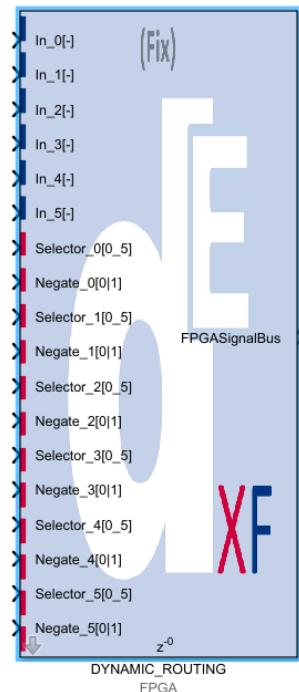


Figure 258: DYNAMIC_ROUTING Main block for floating point

Block Dialog

The FPGA main blockset contains the following dialog.

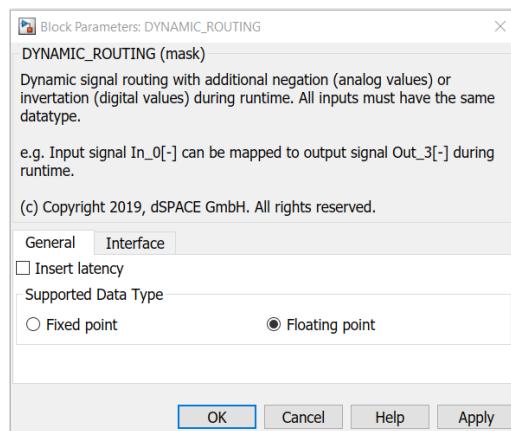


Figure 259: DYNAMIC_ROUTING dialog

An internal latency can be insert with the checkbox Insert latency.



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
In_0	[-]	Input 0	User-defined
In_1	[-]	Input 1	User-defined
In_2	[-]	Input 2	User-defined
In_3	[-]	Input 3	User-defined
In_4	[-]	Input 4	User-defined
In_5	[-]	Input 5	User-defined
Selector_0	[-]	Routing_Selector input 0	UFix_3_0
Negate_0	[-]	Negation (analog values) / Invertation (digital values) flag of input 0	UFix_1_0
Selector_1	[-]	Routing_Selector input 1	UFix_3_0
Negate_1	[-]	Negation (analog values) / Invertation (digital values) flag of input 1	UFix_1_0
Selector_2	[-]	Routing_Selector input 2	UFix_3_0
Negate_2	[-]	Negation (analog values) / Invertation (digital values) flag of input 2	UFix_1_0
Selector_3	[-]	Routing_Selector input 3	UFix_3_0
Negate_3	[-]	Negation (analog values) / Invertation (digital values) flag of input 3	UFix_1_0
Selector_4	[-]	Routing_Selector input 4	UFix_3_0
Negate_4	[-]	Negation (analog values) / Invertation (digital values) flag of input 4	UFix_1_0
Selector_5	[-]	Routing_Selector input 5	UFix_3_0
Negate_5	[-]	Negation (analog values) / Invertation (digital values) flag of input 5	UFix_1_0

Output

The DYNAMIC_ROUTING main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
Out_0	[-]	Output 0	User-defined
Out_1	[-]	Output 1	User-defined
Out_2	[-]	Output 2	User-defined
Out_3	[-]	Output 3	User-defined
Out_4	[-]	Output 4	User-defined
Out_5	[-]	Output 5	User-defined

Interface Examples

Processor blocks

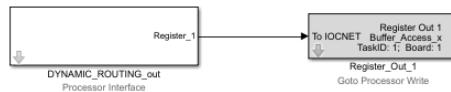


Figure 260: Processor interface

FPGA blocks

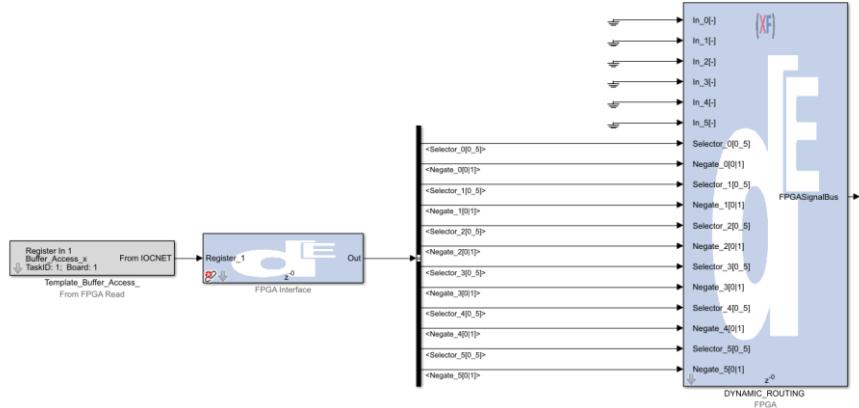


Figure 261: FPGA interface for fixed point

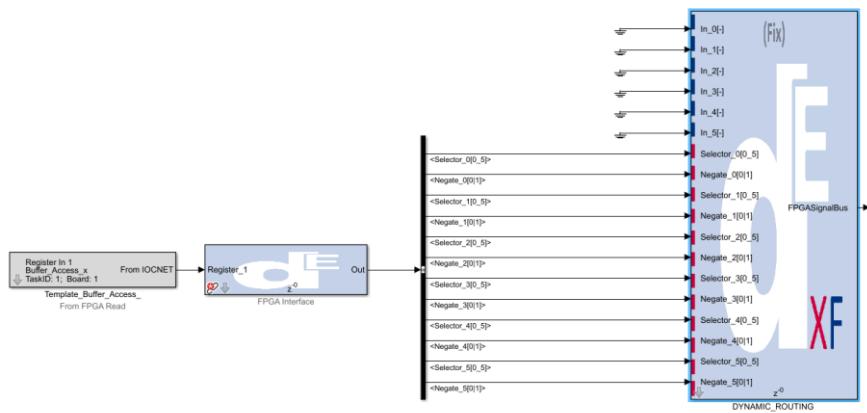


Figure 262: FPGA interface for floating point

Routing: Register 2 Buffer

Objective

The Register_2_Buffer blockset provides the possibility to use the buffer of the rtifpga library as register storage management. Therefore, the user data is stored from the FPGA step by step in a buffer and the processor reads the whole buffer data in one step. A simplified schematic is shown below:

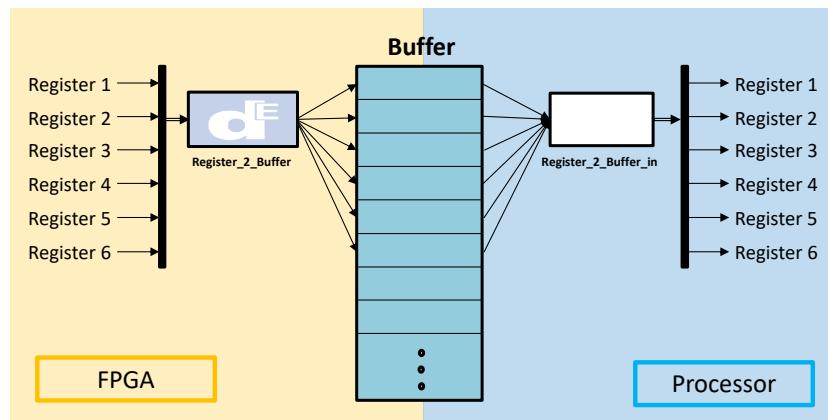


Figure 263: Schematic of the FPGA-Processor communication with the Register_2_Buffer blockset

The blockset contains the following elements:

- **FPGA:** Register_2_Buffer (FPGA Main Component)
- **Processor Interface:** Register_2_Buffer_in (Processor Interface)

FPGA Main Component

Block

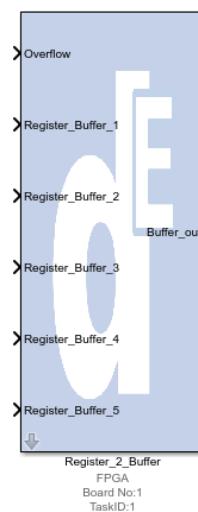


Figure 264: Register_2_Buffer Main block for 5 register (Board: 1; TaskID: 1)

Block Dialog

The FPGA main blockset contains the following dialog.

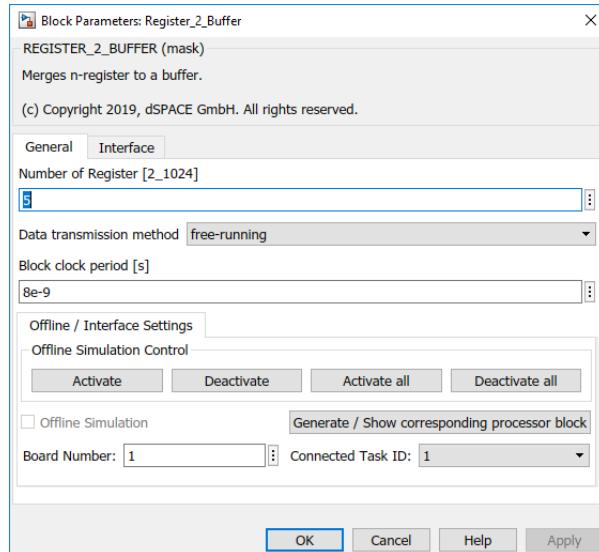


Figure 265: Register_2_Buffer main block dialog; Tab: General

The dialog blockset has the following parameters:

Name	Unit	Description	Format
Number of Register	[·]	Specification of the number of registers which are sent over a buffer. The maximum size of supported registers is 1024	UINT
Data transmission method	[·]	Specification of the data transmission method to the FPGA output buffer Free-running: The input data of the registers will be continuously updated to the connected FPGA buffer Synchronous to Read_Req: The input data of the registers will be updated to the connected buffer after a flag to the block input "Read_Req". After the update the block sent a acknowledge flag "Send_Ack"	-
Block clock period	[s]	Block clock period	-
Subtab: Offline simulation control			
Button: Activate		Activates the offline simulation for this and the corresponding block; if a corresponding block is not available the offline simulation is also not available	
Button: Deactivate		Deactivate the offline simulation of this and the corresponding block; if a corresponding block I not available this mode is default	
Button: Activate All		Activates the offline simulation for all Register2Buffer and Buffer2register blocks in the model; only paring blocks are enabled	

Button: Deactivate All		Deactivate the offline simulation for all Register2Buffer and Buffer2register blocks in the model	
Board Number	[-]	Specify the unique number of FPGA board	UINT [1-99]
TaskID	[-]	Used TaskID of used buffer	UINT [1-32]



For details information about the automatic interface generation for buffer access, which can be controlled over the tab "Interface", please refer to the chapter Automatic Interface Generation in the subchapter Buffer Access Blocks.

Input

The FPGA main block has the following inputs (Data transmission method: free-running):

Name	Unit	Description	Format
Overflow	[-]	Feedback of the Buffer (not necessary in actual version)	UFix_1_0
Buffer_Register_x	[-]	Buffered register channel x; Please ensure that the input signal is a bus with an internal singal called "Data"	UFix_31_0

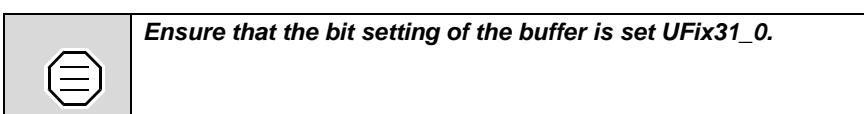
The FPGA main block has the following inputs (Data transmission method: Synchronous to Read_Req):

Name	Unit	Description	Format
Overflow	[-]	Feedback of the Buffer (not necessary in actual version)	UFix_1_0
Read_Req	[-]	Input flag which will be start the update algorithm of the register data to the connected FPGA buffer. This port can be connected to the corresponding output port of the FPGA buffer	UFix_1_0
Buffer_Register_x	[-]	Buffered register channel x	UFix_31_0

Output

The FPGA main block has the following outputs:

Name	Unit	Description	Format
Buffer_out	Bus	Buffer bus; all necessary input ports of the corresponding FPGA buffer are inserted in this bus	
Sim_Data	Vector	If the checkbox in the GUI is selected, the output port Sim_Data is active. This port is for offline simulation and can be directly connected (or over From-Goto connections) to the Sim_Data port of the Register_2_Buffer_in block	



Processor Input

Block

Adapts the FPGA signals for the processor side.

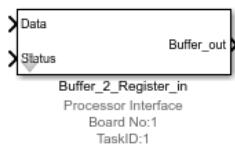


Figure 266: Register_2_Buffer_in block (Board: 1; TaskID: 1)

Block Dialog

The FPGA main blockset contains the following dialog.

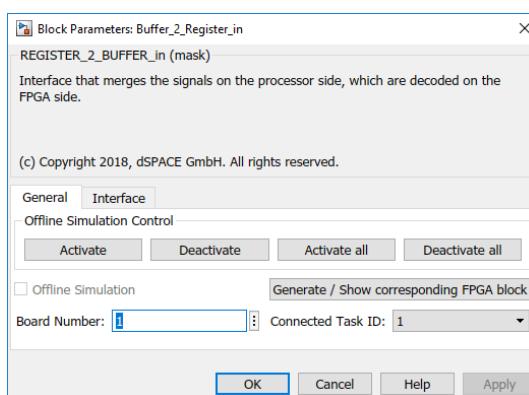


Figure 267: Register_2_Buffer_in block dialog

The dialog blockset have the similar inputs as the FGA main component. For further information please refer to the FPGA main component.



If one parameter will be changed in the GUI, the block will be updated automatically to fit the new parameterization.



For details information about the automatic interface generation for buffer access, which can be controlled over the tab “Interface”, please refer to the chapter Automatic Interface Generation in the subchapter Buffer Access Blocks.

Input

The processor input block has the following inputs:

Name	Unit	Description	Range
Data	[-]	Data vector of the Buffer	

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Buffer_Out	[-]	Data vector of the Buffer	-

Interface Examples

Processor blocks

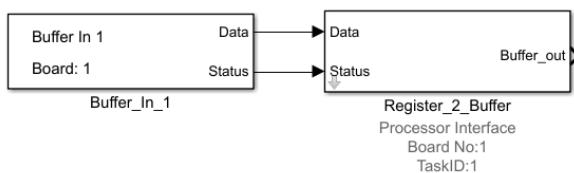


Figure 268: Processor interface

FPGA blocks

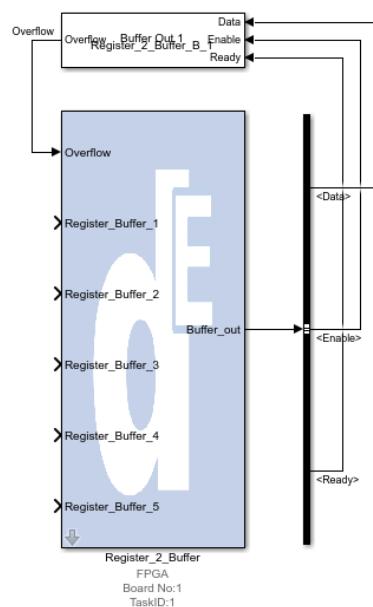


Figure 269: FPGA interface

Routing: Buffer 2 Register

Objective

The Buffer_2_Register blockset provides the possibility to use the buffer of the rtifpga library as register storage management. Therefore, the user data is stored from the processor in one step in a buffer and the FPGA reads this buffer data step by step. A simplified schematic is shown below:

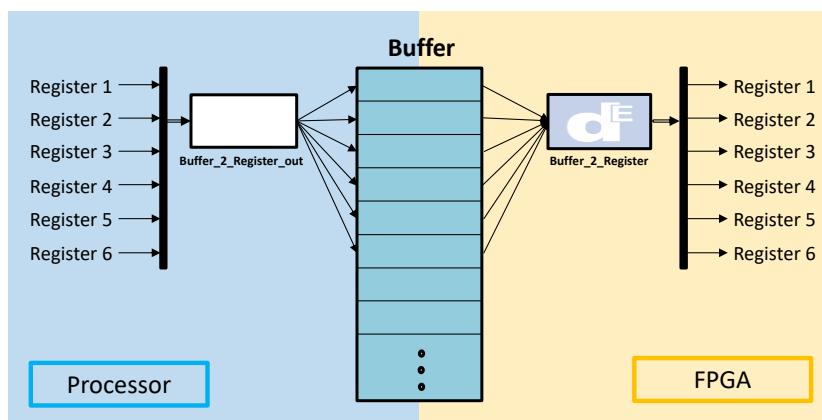


Figure 270: Schematic of the FPGA-Processor communication with the Buffer_2_Register blockset

The blockset contains the following elements:

- Processor Interface: Buffer_2_Register_out (Processor Interface)
- FPGA: Buffer_2_Register (FPGA Main Component)

Processor Output

Block

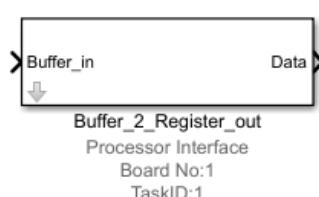


Figure 271: Buffer_2_Register_out block (Board: 1; TaskID: 1)

Block Dialog

The processor output blockset contains the following dialog:

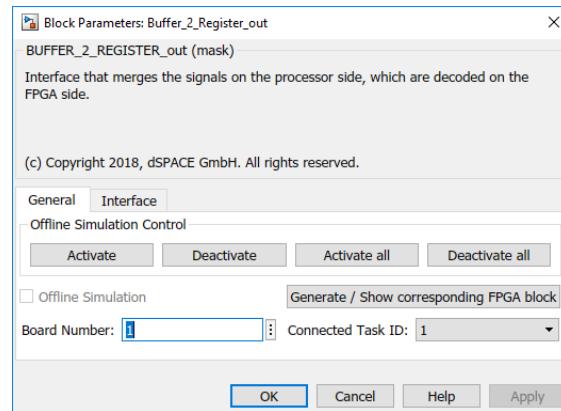


Figure 272: Buffer_2_Register_out dialog

The dialog blockset have the similar inputs as the FGA main component. For further information please refer to the FPGA main component.

	<p>For details information about the automatic interface generation for buffer access, which can be controlled over the tab “Interface”, please refer to the chapter Automatic Interface Generation in the subchapter Buffer Access Blocks.</p>
---	---

Input

The processor output block contains the following inputs:

Name	Unit	Description	Range
Buffer_in	[-]	Input vector with register signals	-

Output

The processor out block provides the following output:

Name	Unit	Description	Range
Data	[-]	Data vector of the Buffer	

FPGA Main Component

Block

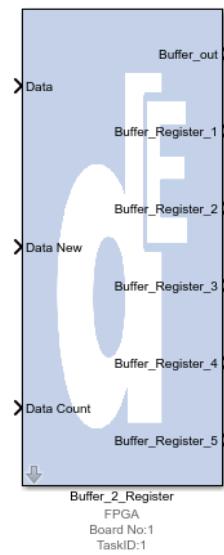


Figure 273: Buffer_2_Register Main block (Board: 1; TaskID: 1)

Block Dialog

The main blockset contains the following dialog.

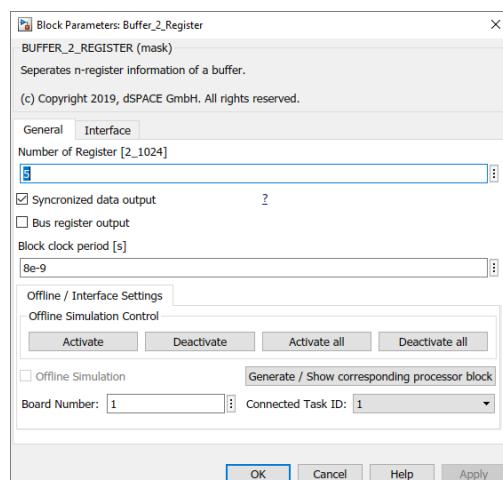


Figure 274: Buffer_2_Register main block dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Format
Number of Register	[-]	Specification of the number of registers which are sent over a buffer. The maximum size of supported registers is 1024	UINT
Synchronized data output		Output of the block is synchronized. After all data is read the output is updated	
Bus register output		Checkbox to change the output from separate ports to bus output	
Block clock period	[s]	Block clock period	-
Subtab: Offline simulation control			
Button: Activate		Activates the offline simulation for this and the corresponding block; if a corresponding block is not available the offline simulation is also not available	
Button: Deactivate		Deactivate the offline simulation of this and the corresponding block; if a corresponding block is not available this mode is default	
Button: Activate All		Activates the offline simulation for all Register2Buffer and Buffer2register blocks in the model; only paring blocks are enabled	
Button: Deactivate All		Deactivate the offline simulation for all Register2Buffer and Buffer2register blocks in the model	
Board Number	[-]	Specify the unique number of FPGA board	UINT [1-99]
TaskID	[-]	Used TaskID of used buffer	UINT [1-32]



For details information about the automatic interface generation for buffer access, which can be controlled over the tab "Interface",

please refer to the chapter Automatic Interface Generation in the subchapter Buffer Access Blocks.

Input

The FPGA main block has the following inputs:

Name	Unit	Description	Format
Data	Bus	Buffer signals: Data	
Data New	Bus	Buffer signals: Data New	
Data Count	Bus	Buffer signals: Data Count	

Output

The FPGA main block has the following outputs:

Name	Unit	Description	Format
Buffer_out	Bus	Address feedback of the Buffer	
Buffer_Register_x	[-]	Buffered register channel x	UFix_31_0



Ensure that the bit setting of the buffer is set UFix31_0.

Interface Examples

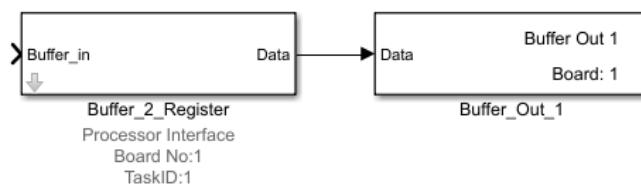
Processor blocks

Figure 275: Processor interface

FPGA blocks

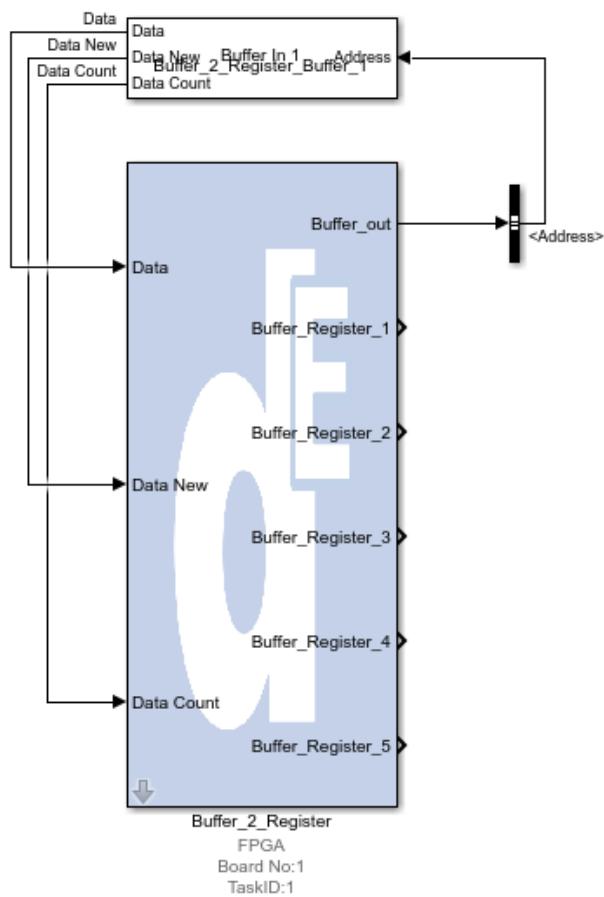


Figure 276: FPGA interface

Version Info

Objective

The version info blockset sends the on the FPGA used XSG_Utils/ElectricComponents version information to the processor side. It is mandatory to use this block to ensure that the components of the FPGA fit to the components on the processor side!



To simplify later support it is useful to integrate this version info block in the FPGA model.

The blockset contains the following elements:

- Processor Interface: LIBRARY_VERSION_INFO_out
(Processor Interface)
- FPGA Interface: LIBRARY_VERSION_INFO_in
(FPGA Interface)
- FPGA: LIBRARY_VERSION_INFO
(FPGA Main Component)
- FPGA Interface: LIBRARY_VERSION_INFO_out
(FPGA Interface)
- Processor Interface: LIBRARY_VERSION_INFO_in
(Processor Interface)

Processor Output

Block

Sends synchronization information to the FPGA.

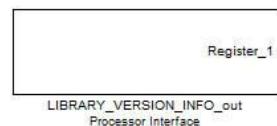


Figure 277: LIBRARY_VERSION_INFO_in block

Block Dialog

The dialog only provides a short block description.

Output

The processor input block has the following outputs:

Register 1

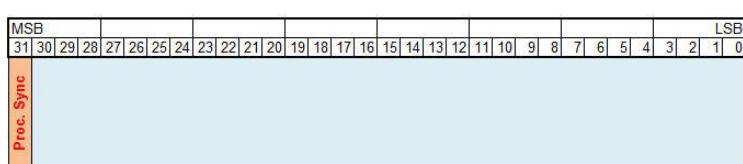


Figure 278: LIBRARY_VERSION_INFO_out Register 1

Name	Used Bits	Description	Range
SPARE	30 ... 0		
Proc.Sync. Impulse	31	Processor synchronous toggle bit	0 1

FPGA Main Component

Block



Figure 279: LIBRARY_VERSION_INFO Main block

Block Dialog

The FPGA main blockset contains the following dialog.

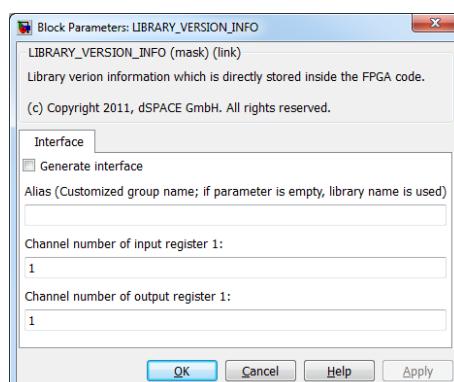


Figure 280: Verion Info main block dialog

The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter. After generating the interface, you can delete the FPGA main block in the model.



It is recommended to use the last interface register (e.g. 128) of the PHS bus communication for the version information.

Input

The FPGA main component has no inputs.

Output

The FPGA main component has no outputs.

Processor Input

Block	Adapts the FPGA signals for the processor side.
--------------	---

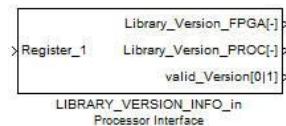


Figure 281: LIBRARY_VERSION_INFO_in block

Block Dialog	The dialog only provides a short block description.
---------------------	---

Input	The processor input block has the following inputs, where the content changes on every processor clock related to the used libraries (ElectricComponents, Utils ...):
--------------	---

Register 1

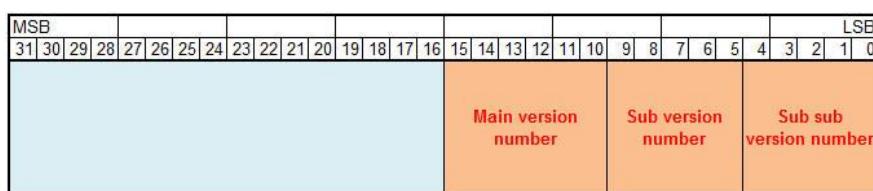


Figure 282: LIBRARY_VERSION_INFO_in Register 1

Name	Used Bits	Description	Range
Sub sub version number	4 ... 0	Last version number	0 ... 31
Sub version number	9 ... 5	Middle version number	0 ... 31
Main version number	5 ... 10	First version number	0 ... 63
SPARE	31 ... 16		

Output	The processor input block has the following outputs:
---------------	--

Name	Unit	Description	Range
Library Version FPGA	[-]	Vector of FPGA library version information Example: 7.11 represents library version 7.1.1	-
Library Version Proc	[-]	Vector of processor library version information Example: 7.11 represents library version 7.1.1	-
valid Version	[-]	Checks if all processor library versions fit to its FPGA library versions. 0: invalid 1: valid	0 1

Interface Examples

Processor blocks

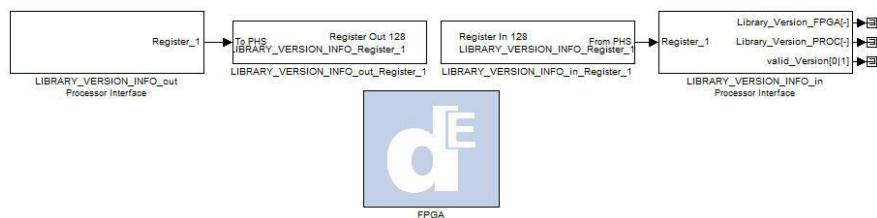


Figure 283: Processor interface

FPGA blocks

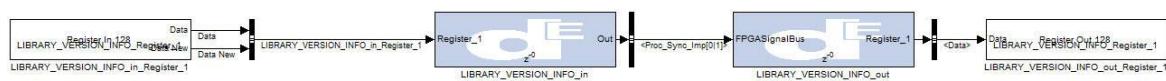


Figure 284: FPGA interface

IO-Access: SCALE_DAC

Description / Overview	Provides the opportunity to change the ratio, offset, minimum and maximum value of dynamic FPGA signals where its electrical output is connected to an analog output (DAC). Additionally this block provides stimulus functionality which allows to tune the scaled value within the processor step rate during runtime. The block automatically outputs directly the scaled value in Bit format, to be able to connect its output to a DAC block. Example: the FPGA model value is a current [A], then it calculates the voltage value [V] by using the specified processor parameters and finally the voltage is transferred into binary format [Bit] for the DAC. In the case that the processor-FPGA communication breaks down, an internal watchdog algorithm is integrated which sets the output to zero. To realize the possibility of debug functionality a feedback of the output is also implemented. A simplified equivalent circuit is shown below:
-------------------------------	--

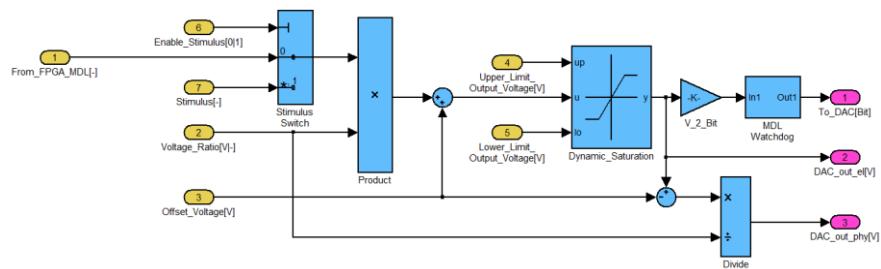


Figure 285: Schematic circuit Scale_DAC

	Custom library blocks for ControlDesk are available. For detailed information please refer to the Custom Instruments documentation.
---	---

The blockset contains the following elements:

- Processor Interface: SCALE_DAC_out (Processor Interface)
- FPGA Interface: SCALE_DAC_in (FPGA Interface)
- FPGA: SCALE_DAC (FPGA Main Component)
- FPGA Interface: SCALE_DAC_out (FPGA Interface)
- Processor Interface: SCALE_DAC_in (Processor Interface)

Processor Output

Block

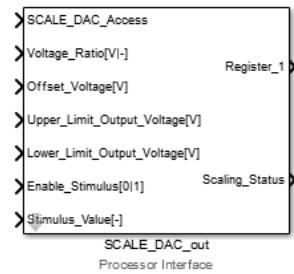


Figure 286: SCALE_DAC_out block (enabled input ports)

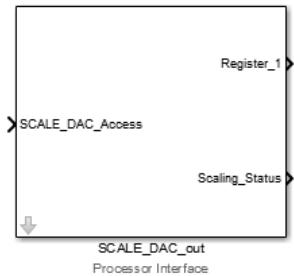


Figure 287: SCALE_DAC_out block (disabled input ports)

Block Dialog

The processor output block contains several dialogs. This depends on the parameterization if the input ports are enabled or disabled. If the input ports are disabled the block contains the following dialog:

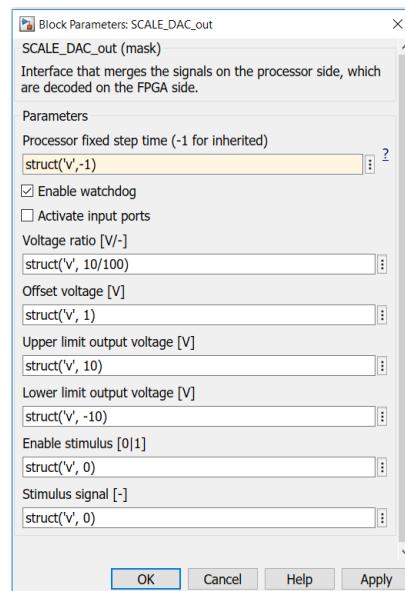


Figure 288: SCALE_DAC_out dialog (disabled input ports)

If the input ports are disabled the dialog blockset has the following parameters:

Name	Unit	Description	Range
Processor fixed step time	[s]	Processor fixed step time	-
Enable Watchdog	[-]	Enables the Watchdog function on the FPGA	0 1
Active input ports	[-]	Checkbox which parameterize the input ports; if checkbox isn't selected all input ports are disabled and the GUI parameter settings are enabled	0 1
Voltage ratio	[V/-]	Voltage ratio	-
Offset voltage	[V]	Offset voltage	-10 ... 10V
Upper limit output voltage	[V]	Upper limit of the output voltage of the output	-10 ... 10V
Lower limit output voltage	[V]	Lower limit of the output voltage of the output	-10 ... 10V
Enable Stimulus	[0 1]	Enables the stimulus output	0 1
Stimulus signal	[V]	Stimulus signal	-10 ... 10V

If the input ports are enabled the block contains the following dialog:

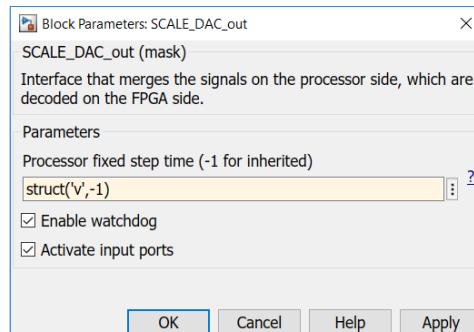


Figure 289: SCALE_DAC_out dialog (enabled input ports)

If the input ports are disabled the dialog blockset has the following parameters:

Name	Unit	Description	Range
Processor fixed step time	[s]	Processor fixed step time	[-]
Enable Watchdog	[-]	Enables the Watchdog function on the FPGA	[0 1]
Active input ports	[-]	Checkbox which parameterize the input ports; if checkbox isn't selected all input ports are disabled and the GUI parameter settings are enabled	[0 1]

Input

The processor output block contains several states of inputs. This depends on the parameterization if the input ports are enabled or disabled. If the input ports are enabled the block contains the ports:

Name	Unit	Description	Format
SCALE_DAC_Access	Bus	Bus feedback containing the SCALE_DAC_in block's most significant signals	-
Voltage ratio	[V/-]	Voltage ratio	-
Offset voltage	[V]	Offset voltage	-10 ... 10V
Upper limit output voltage	[V]	Upper limit of the output voltage of the output	-10 ... 10V
Lower limit output voltage	[V]	Lower limit of the output voltage of the output	-10 ... 10V
Enable Stimulus	[0 1]	Enables the stimulus output	0 1
Stimulus signal	[V]	Stimulus signal	-10 ... 10V

If the input ports are disabled the SCALE_DAC block doesn't supports any inputs.

Output

The processor out block provides one output register which is multiplexed to transfer all input data. The parameter transfer is based on Enable_stimulus value. If the Enable_stimulus is 0, the parameters, such as voltage ratio, offset, upper and lower limit are transferred continuously. If the Enable_stimulus is 1, the parameters are transferred once and the stimulus value is transferred continuously. When there is a change in parameter (for instance if voltage ratio is changed), the new parameters are transferred once and then the stimulus value is transferred continuously.

FPGA Main Component

Block

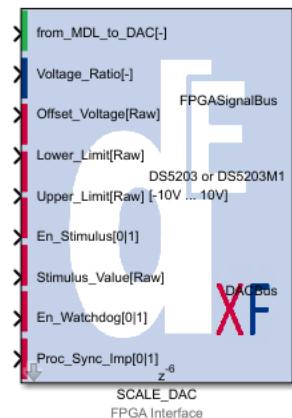


Figure 290: SCALE_DAC Main block

Block Dialog

The FPGA main blockset contains the following dialog.

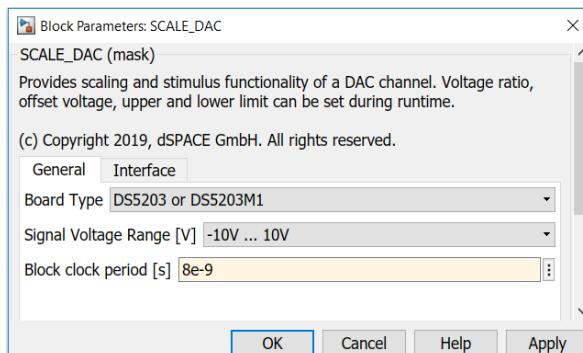


Figure 291: Scale_DAC dialog

Name	Unit	Description	Range
Board Type	[-]	Specify the FPGA Board	0..5
Signal Voltage Range	[V]	Specify the output range of the connected DAC	-

	The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.
--	---

Input

The main block has the following inputs:

Name	Unit	Description	Format
From_MDL_to_DAC	[-]	User-defined original output (not scaled) which must be sent via a DAC channel.	User-defined
Voltage ratio	[-]	Factor of the voltage ratio. Represents the factor x of the mathematical description $x * 2^y(-y)$	XFloat_8_24
Offset voltage	[Raw]	Offset voltage	Fix_16_0
Upper Limit Output Voltage	[Raw]	Upper limit of the voltage of the DAC used	Fix_16_0
Lower Limit Output Voltage	[Raw]	Upper limit of the voltage of the DAC used	Fix_16_0
En_Stimulus	[0 1]	Enable stimulus	UFix_1_0
Stimulus_Value	[Raw]	Stimulus signal	Fix_16_0
En_Watchdog	[0 1]	Enable internal watchdog	UFix_1_0
Proc_Sync_Imp	[0 1]	Processor synchronous impulse for watchdog functionality	UFix_1_0

Output

The SCALE_DAC main block has the following outputs:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the block's most significant signals	-
DACBus	Bus	Scaled output for a DAC block of RTI	-

Processor Input

Block

Adapts the FPGA signals for the processor side

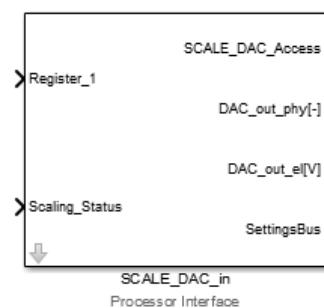


Figure 292: SCALE_DAC_in block

Block Dialog

The dialog only provides a short block description.

Input

One Register is used to get the relevant data from the FPGA. These include, the feedback voltage, Flag for Board/ADC ID and Board/ADC ID feedback.

Output

The processor input block has the following outputs:

Name	Unit	Description	Format
SCALE_DAC_Access	Bus	Bus feedback containing the SCALE_DAC_in block's most significant signals	-
DAC_out_phy	[-]	Actual physical output of the DAC which depends on the scaling	-
DAC_out_el	[V]	Actual electrical output voltage of the DAC	-
SettingsBus	Bus	Feedback for the customer which settings are used in the FPGA block	-

Interface Examples

Processor blocks

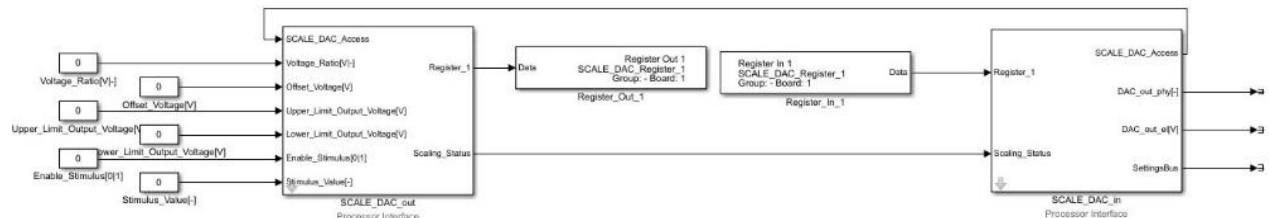


Figure 293: Processor interface

FPGA blocks

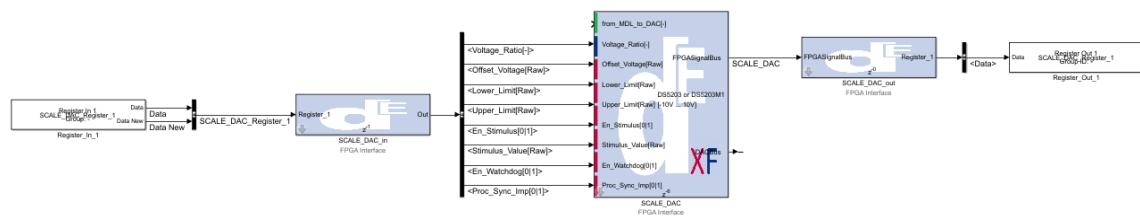


Figure 294: FPGA interface

IO-Access: SCALE_ADC

Description / Overview Scales a dynamic analog input signals (ADC) according to ratio, offset, minimum and maximum into a physical value (e.g. a voltage to a pressure). Additionally, this block provides stimulus functionality which allows to tune the scaled value within the processor step rate during runtime. The ADC block connected to this block must be configured in the scaling mode “Bit”. The inverted workflow is similar to the SCALE_DAC block. A simplified equivalent circuit is shown below:

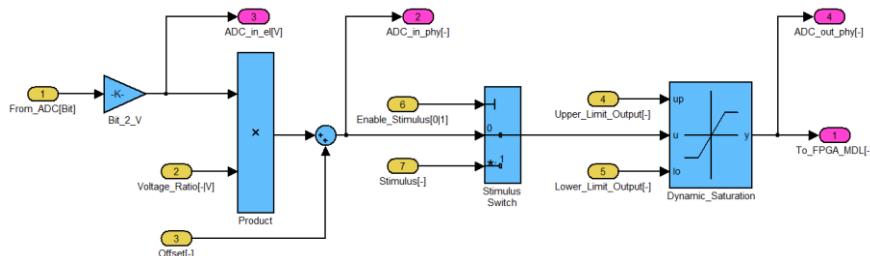


Figure 295: Schematic circuit Scale_ADC

	Custom library blocks for ControlDesk are available. For detailed information, please refer to the Custom Instruments documentation.
--	--

The blockset contains the following elements:

- | | |
|---|---|
| <ul style="list-style-type: none"> ▪ Processor Interface: ▪ FPGA Interface: ▪ FPGA: ▪ FPGA Interface: ▪ Processor Interface: | <ul style="list-style-type: none"> SCALE_ADC_out (Processor Interface) SCALE_ADC_in (FPGA Interface) SCALE_ADC (FPGA Main Component) SCALE_ADC_out (FPGA Interface) SCALE_ADC_in (Processor Interface) |
|---|---|

Processor Output

Block

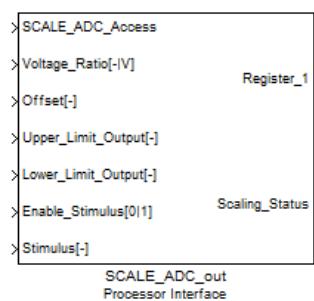


Figure 296: SCALE_ADC_out block (enabled input ports)

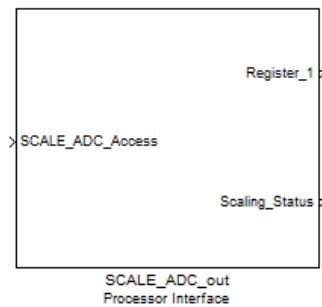


Figure 297: SCALE_ADC_out block (disabled input ports)

Block Dialog

The processor output block contains several dialogs. This depends on the parameterization if the input ports are enabled or disabled. If the input ports are disabled, the block contains the following dialog:

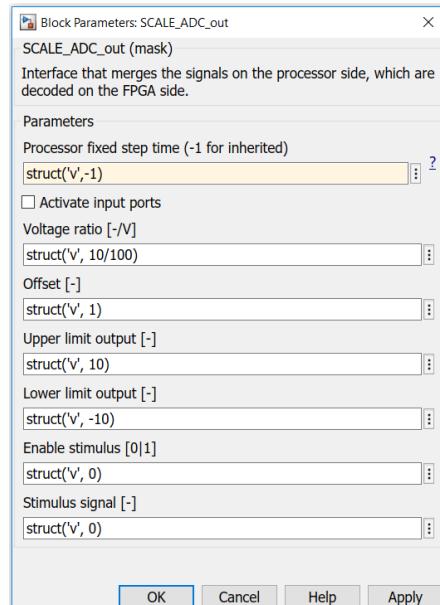


Figure 298: SCALE_ADC_out dialog (disabled input ports)

If the input ports are disabled, the dialog blockset has the following parameters:

Name	Unit	Description	Range
Processor fixed step time	[s]	Processor fixed step time	-
Active input ports	[-]	Checkbox which parameterize the input ports; if checkbox isn't selected all input ports are disabled and the GUI parameter settings are enabled	0 1
Voltage ratio	[- / V]	Voltage ratio	-
Offset voltage	[-]	Offset level	-
Upper limit output	[-]	Upper limit of the output	-
Lower limit output	[-]	Lower limit of the output	-
Enable Stimulus	[0 1]	Enables the stimulus output	0 1
Stimulus signal	[-]	Stimulus signal	-

If the input ports are enabled the block contains the following dialog:

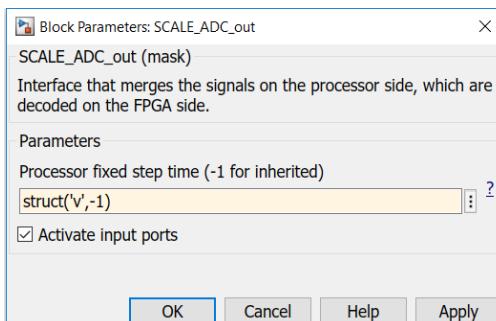


Figure 299: SCALE_ADC_out dialog (enabled input ports)

If the input ports are disabled the dialog blockset has the following parameters:

Name	Unit	Description	Range
Processor fixed step time	[s]	Processor fixed step time	-
Active input ports	[-]	Checkbox which parameterize the input ports; if checkbox isn't selected all input ports are disabled and the GUI parameter settings are enabled	0 1

Input

The processor output block contains several states of inputs. This depends on the parameterization if the input ports are enabled or disabled. If the input ports are enabled the block contains the ports:

Name	Unit	Description	Range
Voltage ratio	[-/V]	Ratio to convert from [V] into another physical unit (e.g. [Pa])	-
Offset voltage	[-]	Offset voltage in physical unit	-
Upper limit output	[-]	Upper limit of the output in physical unit	-
Lower limit output	[-]	Lower limit of the output in physical unit	-
Enable Stimulus	[0 1]	Enables the stimulus output	[0 1]
Stimulus signal	[-]	Stimulus signal in physical unit	-

If the input ports are disabled the SCALE_DAC block only supports the SCALE_ADC_Access Bus.

Output

The processor out block provides one output register which is multiplexed to transfer all input data. The parameter transfer is based on Enable_Stimulus and change in the parameter value. If the Enable_Stimulus is 0, the parameters, such as voltage ratio, offset, upper and lower limit are transferred continuously. If the Enable_stimulus is 1, the parameters are transferred once and the stimulus value is transferred continuously. When there is a change in parameter (for instance if voltage ratio is changed), the new parameters are transferred once and then the stimulus value is transferred continuously.

FPGA Main Component

Block

The output type can be selected in the Main component GUI. If the output type is fixed point, the output port (from_ADC_to_MDL[-]) is colored red, whereas if the output type is floating point, the output port is colored blue as shown in [Figure 300](#) and in [Figure 302](#)

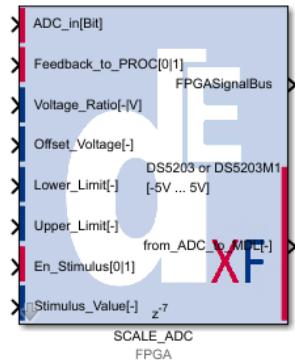


Figure 300: SCALE_ADC Main block (Fixed point output)

Block Dialog

The FPGA main blockset contains the following dialog.

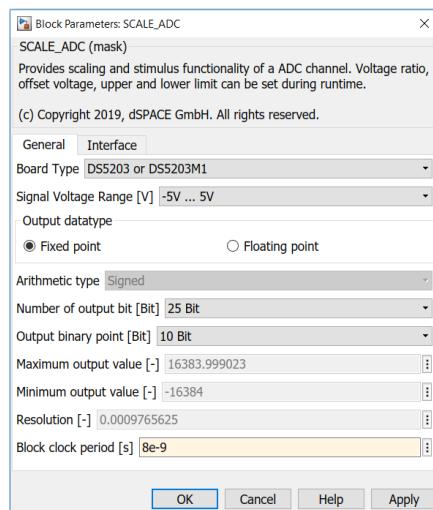


Figure 301: SCALE_ADC dialog for fixed point output type

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Board Type	[-]	Specify the FPGA Board	0...7
Signal Voltage Range	[V]	Specify the output range of the connected ADC	-
Output datatype	[-]	Specify the output datatype (Fixed or Floating point)	0 1
Arithmetic type	[-]	Signed for fixed point	-
Number of output bit	[Bit]	Specify the number of output bits	14 ... 25 Bit
Output binary point	[Bit]	Specify the output binary point	0 ... 25 Bit
Maximum output value	[-]	Passive output value; Calculate the actual maximum value of the output depending on the Bit settings	-
Minimum output value	[-]	Passive output value; Calculate the actual minimum value of the output depending on the Bit settings	-
Resolution	[-]	Passive output value; Calculate the actual resolution of the output depending on the Bit settings	-

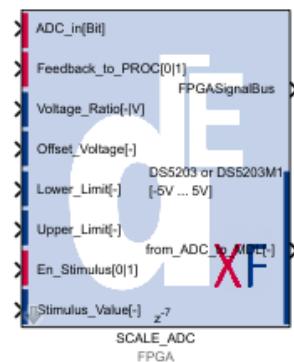


Figure 302: SCALE_ADC Main block (Floating point output)

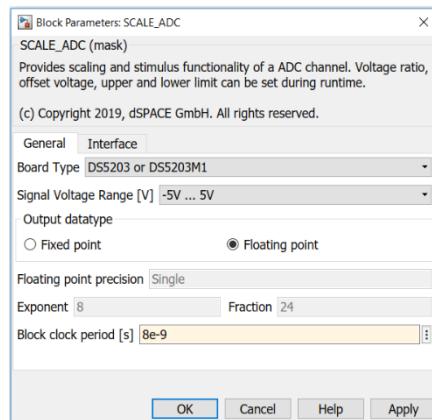


Figure 303: SCALE_ADC dialog for floating point output type

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Board Type	[-]	Specify the FPGA Board	0...6
Signal Voltage Range	[V]	Specify the output range of the connected ADC	-
Output datatype	[-]	Specify the output type (Fixed or Floating point)	0 1
Floating point precision	[-]	For floating point output type, single [XFloat_8_24] is the default precision	Single
Exponent	[-]	For single precision exponent has 8 bits	8
Fraction	[-]	For single precision fraction width is 24 bits	24
Block clock period	[s]	Sample time of the FPGA	-



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

The main block has the following inputs:

Name	Unit	Description	Format
ADC_in	[Bit]	Output signal of an ADC channel	Fix_16_0
Voltage ratio	[-/V]	Voltage ratio	XFloat_8_24
Offset	[·]	Offset	XFloat_8_24
Upper Limit Output	[·]	Upper limit of the output of the ADC used	XFloat_8_24
Lower Limit Output	[·]	Upper limit of the output of the ADC used	XFloat_8_24
En_Stimulus	[0 1]	Enable stimulus	UFix_1_0
Stimulus	[·]	Stimulus signal	XFloat_8_24

Output The SCALE_ADC main block has the following outputs:

Name	Unit	Description	Format
FPGASignalBus	Bus	Bus containing the block's most significant signals	-
From ADC to MDL	[-]	Scaled output in user defined bit format and in physical unit	Depends on GUI settings

Processor Input

Block Adapts the FPGA signals for the processor side

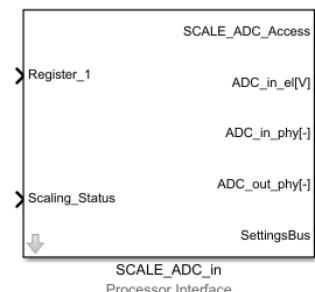


Figure 304: SCALE ADC in block

Block Dialog The dialog only provides a short block description.

Input One Register is used to get the relevant data from the FPGA. These include, the feedback voltage, the output bit length, output binary point, output type (fix or float), Flag for Board/ADC ID and Board/ADC ID feedback.

Output The processor input block has the following outputs:

Name	Unit	Description	Format
SCALE_ADC_Access	Bus	Bus feedback containing the SCALE_ADC_in block's most significant signals	-
ADC_out_phy	[-]	Actual physical output of the ADC which depends on the scaling	-
ADC_in_phy	[-]	Scaled ADC output with offset	-
ADC_out_el	[V]	Actual electrical output voltage of the ADC	-
SettingsBus	Bus	Feedback for the customer which settings are used in the FPGA block	-

Interface Examples

Processor blocks

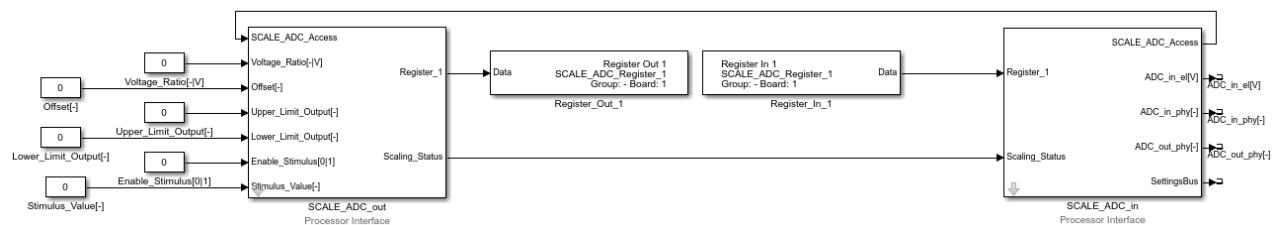


Figure 305: Processor interface

FPGA blocks

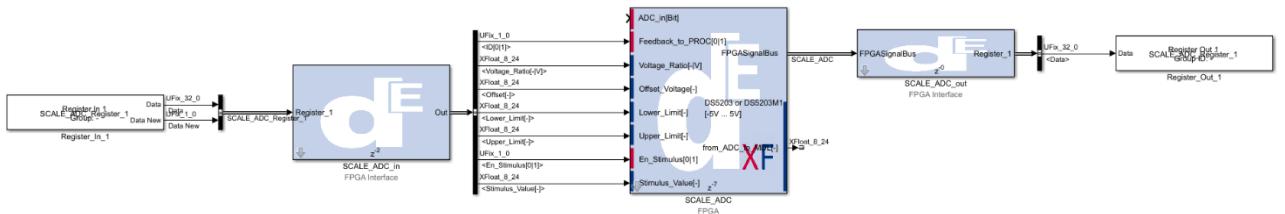


Figure 306: FPGA interface

IO-Access: Proc_2_DAC

Description / Overview	Provides the functionality to access the DACs only via processor interface. This block set is always used for those DACs which are not connected to the FPGA model, so that they can be used like standard DACs (like those from the DS2211 for example).
-------------------------------	---

The blockset contains the following elements:

- Processor Interface: Proc_2_DAC_out (Processor Interface)
- FPGA Interface: Proc_2_DAC_in (FPGA Interface)
- FPGA: Proc_2_DAC (FPGA Main Component)

Processor Output

Block

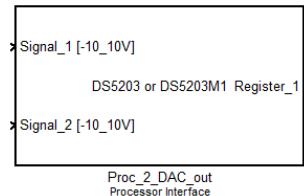


Figure 307: Proc_2_DAC_out block

Block Dialog

The processor output blockset contains the following dialog:

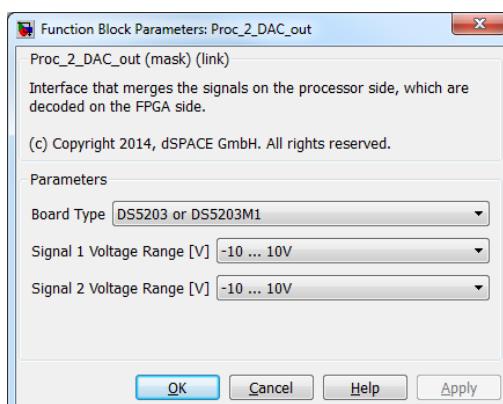


Figure 308: Proc_2_DAC_out dialog

Name	Unit	Description	Range
Board Type	Popup	Selection of the actual board type: DS1552 DS1553 DS2655M1 DS5203 or DS5203M1 DS6651	-
Signal 1 Voltage Range	Popup	Selection of the actual voltage range of signal 1	-
Signal 2 Voltage Range	Popup	Selection of the actual voltage range of signal 2	-

The same settings must be done in the FPGA Interface block GUI.

Input

The Proc_2_DAC_out block has the following inputs:

Name	Unit	Description	Range
Signal_1	[V]	Output voltage of signal 1	Board specific
Signal_2	[V]	Output voltage of signal 2	Board specific

Output

The processor out block provides one output register whose sectioning is shown below:

Register 1

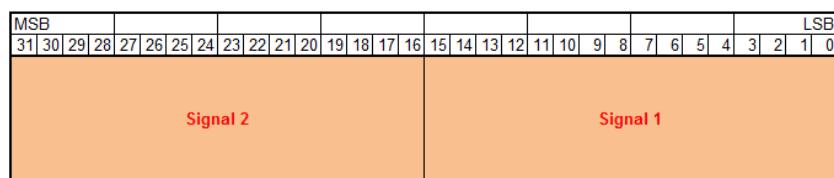


Figure 309: Proc_2_DAC_out Register 1

Name	Used Bits	Description	Range
Signal_1	15 ... 0	Output voltage of signal 1	Board specific
Signal_2	31 ... 16	Output voltage of signal 2	Board specific

FPGA Main Component

Block

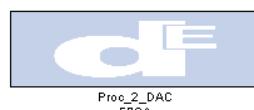


Figure 310: Proc_2_DAC Main block

Block Dialog

The FPGA main blockset contains the following dialog.

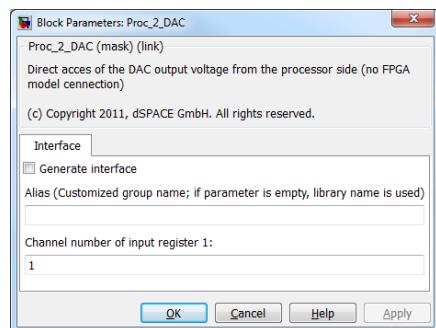


Figure 311: Proc_2_DAC dialog

The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter. After generating the interface, you can delete the FPGA main block in the model.

Input

The FPGA main component has no inputs.

Output

The FPGA main component has no outputs.

Interface Examples

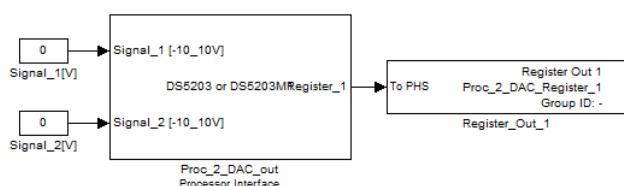
Processor blocks

Figure 312: Processor interface

FPGA blocks

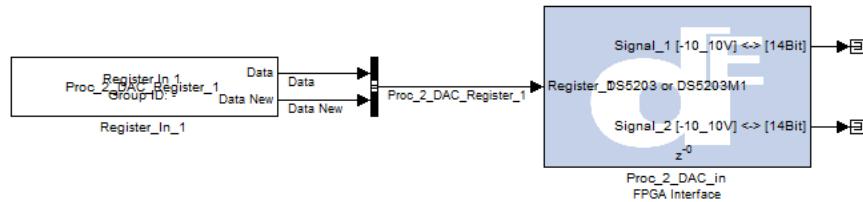
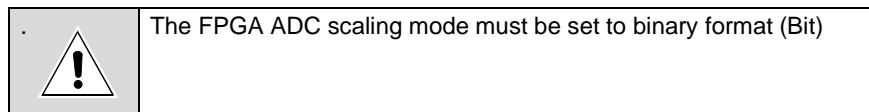


Figure 313: FPGA interface

IO-Access: ADC_2_Proc

Objective	Provides the functionality to access the ADCs via processor interface. This block set is always used for those ADCs which are not connected to the FPGA model, so that they can be used like standard ADCs (like those from the DS2211 for example).
------------------	--



The blockset contains the following elements:

- FPGA: ADC_2_Proc (FPGA Main Component)
- FPGA Interface: ADC_2_Proc_out (FPGA Interface)
- Processor Interface: ADC_2_Proc_in (Processor Interface)

FPGA Main Component

Block



Figure 314: ADC_2_Proc Main block

Block Dialog

The FPGA main blockset contains the following dialog.

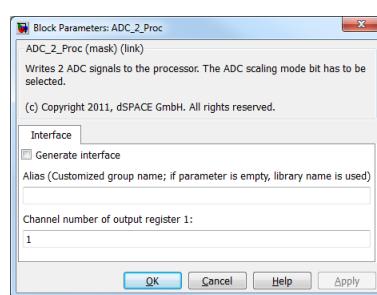


Figure 315: ADC_2_Proc dialog

The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter. After generating the interface, you can delete the FPGA main block in the model.

Input

The FPGA main component has no inputs.

Output The FPGA main component has no outputs.

Processor Input

Block Adapts the FPGA signals for the processor side.

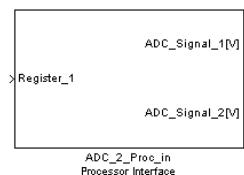


Figure 316: ADC_2_Proc_in block

Block Dialog The processor input blockset contains the following dialog:

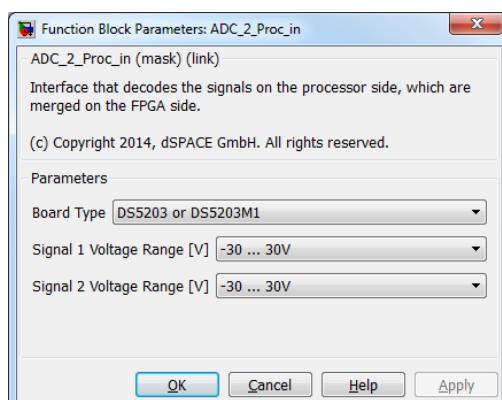


Figure 317: ADC_2_Proc_out dialog

Name	Unit	Description	Range
Board Type	Popup	Selection of the actual board type: DS1552 DS1553 DS2655M1 DS5203 or DS5203M1 DS6651	-
Signal 1 Voltage Range	Popup	Selection of the actual voltage range of signal 1	-
Signal 2 Voltage Range	Popup	Selection of the actual voltage range of signal 2	-

The same settings must be done in the FPGA Interface block GUI.

Input

The processor input block has the following inputs:

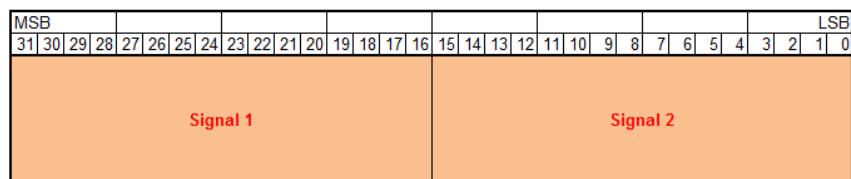
Register 1

Figure 318: ADC_2_Proc_in Register 1

Name	Used Bits	Description	Range
Signal_1	15 ... 0	Output voltage of signal 1	Board specific
Signal_2	31 ... 16	Output voltage of signal 2	Board specific

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Signal_1	[V]	Input voltage of signal 1	Board specific
Signal_2	[V]	Input voltage of signal 2	Board specific

Interface Examples

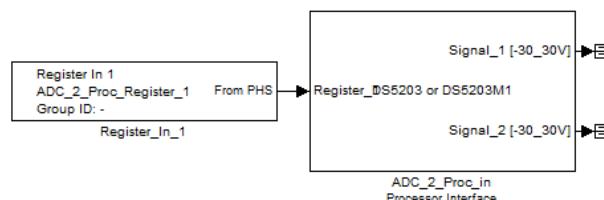
Processor blocks

Figure 319: Processor interface

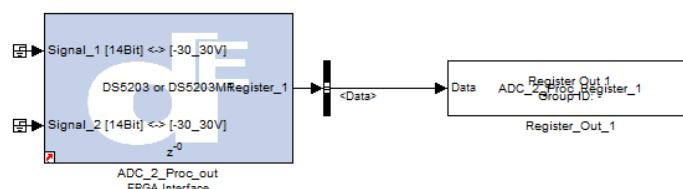
FPGA blocks

Figure 320: FPGA interface

IO-Access: Proc_2_PWM

Description / Overview Provides the functionality to generate a simple PWM-signal with parameterization via processor interface. This block set is always used for those digital channels which are not connected to the FPGA model, so that they can be used like standard PWM out channels (like those from the DS2211 for example).

The blockset contains the following elements:

- Processor Interface: Proc_2_PWM_out (Processor Interface)
- FPGA Interface: Proc_2_PWM_in (FPGA Interface)
- FPGA: Proc_2_PWM (FPGA Main Component)

Processor Output

Block

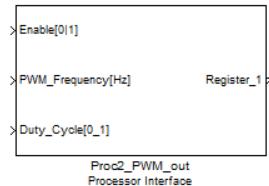


Figure 321: Proc_2_PWM_out block

Block Dialog

The processor output block contains the following dialog.

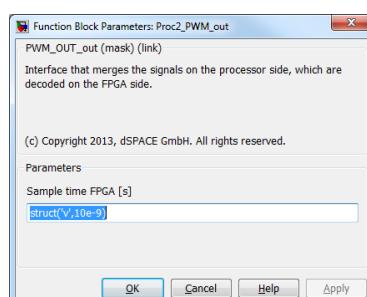


Figure 322: Proc_2_PWM_out dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Sample time FPGA	[s]	Sample time of the FPGA	-

Input

The Proc_2_PWM_out block has the following inputs:

Name	Unit	Description	Range
Enable	[-]	Enable of the output signal	0 1
PWM_Frequency	[Hz]	Frequency of the PWM output	$1 / [(0 \dots 2^{16}-1) * T_{FPGA}]$
Duty_Cycle	[-]	Duty cycle of the PWM output	0 ... 1

Output

The processor out block provides one output register whose sectioning is shown below:

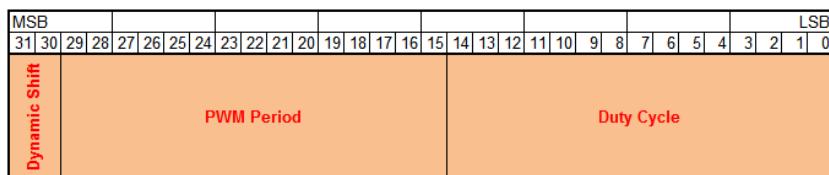
Register 1

Figure 323: Proc_2_PWM_out Register 1

Name	Used Bits	Description	Range
Duty_Cycle	14 ... 0	Duty Cycle of PWM output	0 ... 1
PWM_Period / Enable	29 ... 15	PWM Period (if PWM Period is set to 0 → output is disabled)	$1 / [(0 \dots 2^{15}-1) * T_{FPGA} * Dynamic_Shift]$
Dynamic Shift	31 ... 30	Enable Output / Dynamic Shift	0 ... 3

FPGA Main Component**Block**

Figure 324: Proc_2_PWM_out Main block

Block Dialog

The FPGA main blockset contains the following dialog.

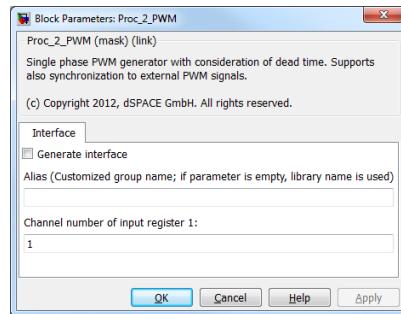


Figure 325: Proc_2_PWM_out dialog

The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter. After generating the interface, you can delete the FPGA main block in the model.

Input The FPGA main component has no inputs.

Output The FPGA main component has no outputs.

Interface Examples

Processor blocks

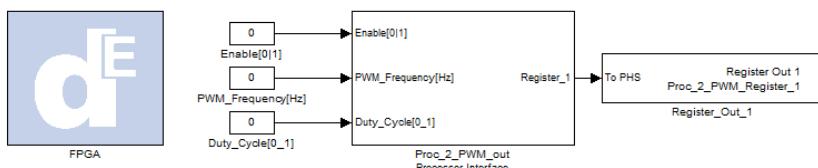


Figure 326: Processor interface

FPGA blocks

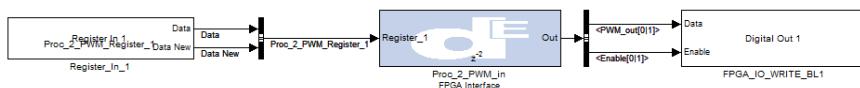


Figure 327: FPGA interface

IO-Access: PWM_2_Proc

Objective	Provides the functionality to a simple PWM signal via processor interface. This block set is always used for those digital inputs which are not connected to the FPGA model, so that they can be used like standard PWM inputs (like those from the DS2211 for example).
------------------	--

The blockset contains the following elements:

- FPGA: PWM_2_Proc (FPGA Main Component)
- FPGA Interface: PWM_2_Proc_out (FPGA Interface)
- Processor Interface: PWM_2_Proc_in (Processor Interface)

FPGA Main Component

Block



Figure 328: PWM_2_Proc Main block

Block Dialog

The FPGA main blockset contains the following dialog.

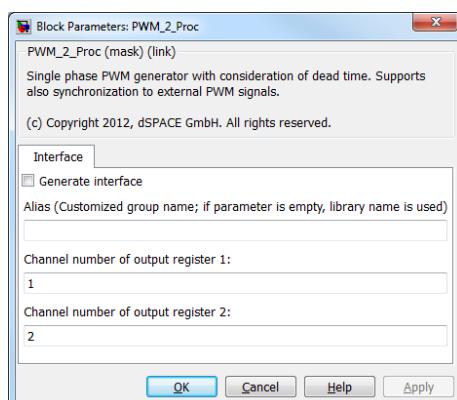


Figure 329: PWM_2_Proc dialog

The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter. After generating the interface, you can delete the FPGA main block in the model.

Input

The FPGA main component has no inputs.

Output The FPGA main component has no outputs.

Processor Input

Block Adapts the FPGA signals for the processor side.

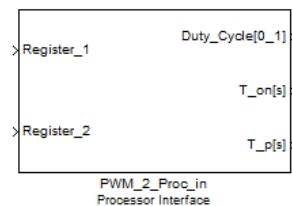


Figure 330: PWM_2_Proc_in block

Block Dialog The processor output blockset contains the following dialog.

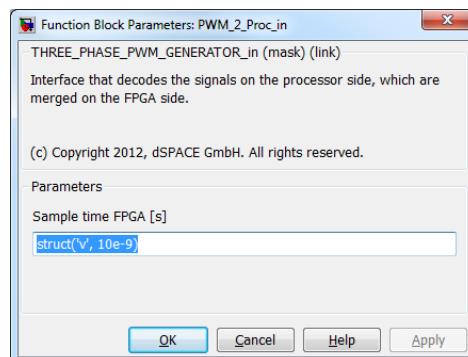


Figure 331: PWM_2_Proc_in dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Sample time FPGA	[s]	Sample time of the FPGA	-

Input The processor input block has the following inputs:

Register 1

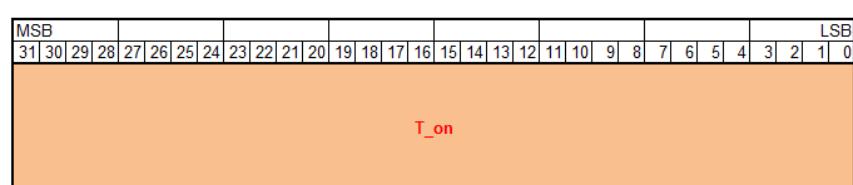


Figure 332: PWM_2_Proc_in Register 1

Name	Used Bits	Description	Range
T_on	31 ... 0	On time of the PWM signal	0 ... 2^31-1

Register 2

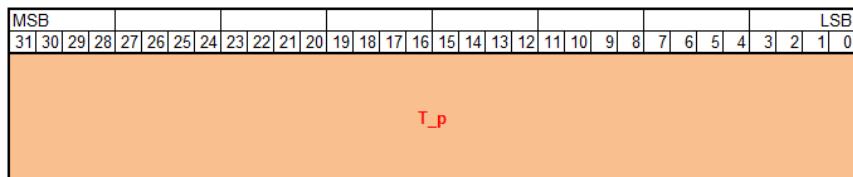


Figure 333: PWM_2_Proc_in Register 2

Name	Used Bits	Description	Range
T_p	31 ... 0	Period time of the PWM signal	0 ... 2^31-1

Output

The processor input block has the following outputs:

Name	Unit	Description	Range
Duty_Cycle	[-]	Duty cycle of the PWM signal	0 ... 1
T_on	[s]	On time of the PWM signal	(0 ... 2^31-1) * T_FPGA
T_p	[s]	Period time of the PWM signal	(0 ... 2^31-1) * T_FPGA

Interface Examples

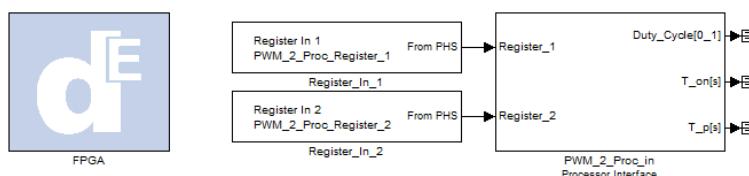
Processor blocks

Figure 334: Processor interface

FPGA blocks

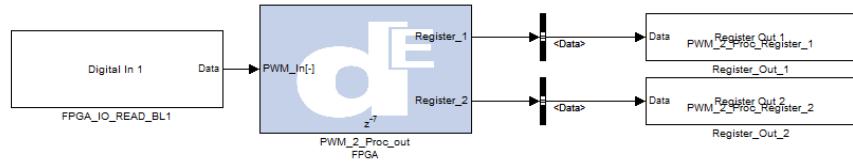


Figure 335: FPGA interface

IO-Access: MULT PURPOSE DIG OUT

Description / Overview	Provides the functionality to generate a digital output, a PWM output or an angle based output which can be configured during model build.
-------------------------------	--

The blockset contains the following elements:

- Processor Interface: MULIT_PURPOSE_DIG_OUT_out (Processor Interface)
- FPGA Interface: MULIT_PURPOSE_DIG_OUT_in (FPGA Interface)
- FPGA: MULIT_PURPOSE_DIG_OUT (FPGA Main Component)

Processor Output

Block	The overview of the processor output block depends on the actual settings of the block GUI. In the following figure the angle based output with external input ports is configured.
--------------	---

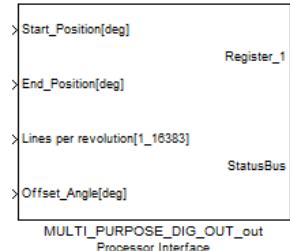


Figure 336: MULT PURPOSE DIG OUT_out block

Block Dialog	The processor output block depends on the actual setting in the GUI. The overall parameters of the GUI are described in the table below.
---------------------	--

Name	Unit	Description	Range
Processor fixed step time	[s]	The sample time at which the processor blocks are to be simulated.	-
Sample time FPGA	[s]	Sample time of the FPGA	-

The following dialog is used if the output type “Digital Output” or “PWM Output” is used.

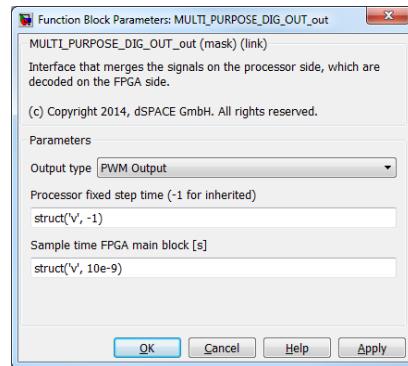


Figure 337: MULIT_PURPOSE_DIG_OUT_out dialog (Output type: “Digital Output” or “PWM Output”)

The dialog blockset has no specific parameters in this output type. If the output type “Angle based output” is used, the following GUI is available.

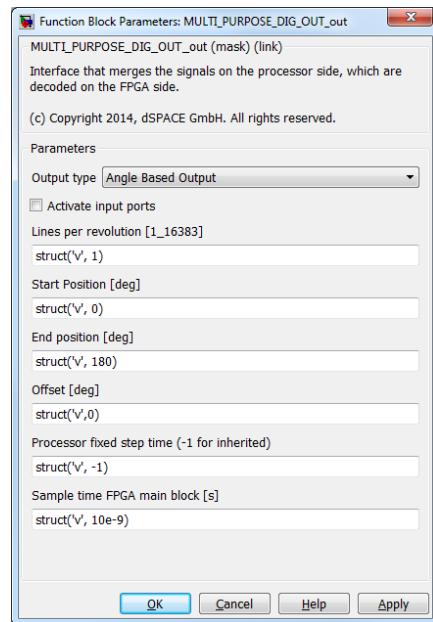


Figure 338: MULIT_PURPOSE_DIG_OUT_out dialog (Output type: “Angle Based Output”)

The following specific parameters can be adjusted in the GUI.

Name	Unit	Description	Range
Activate input ports	[-]	Activate input ports for the parameterization. If the checkbox is enabled, the following GUI parameter are disabled and connected to input ports.	-
Lines per revolution	[-]	Lines per revolution of the APU based output	1 ... 16383
Start Position	[deg]	Start position (rising edge; 0 → 1) of the digital output	0 ... 360
End position	[deg]	End position (falling edge; 1 → 0) of the digital output	0 ... 360
Angle Offset	[deg]	Angle offset between mechanical APU angle and APU of the simulated output behaviour	0 ... 360

Input

The Input depends on the actual defined output type which can be defined in the GUI. Each specific output type has his own input signals.

Output type: Digital Output

Name	Unit	Description	Range
Digital Out	[-]	Enable of the output signal	0 1

Output type: Digital Output

Name	Unit	Description	Range
Duty Cycle	[0_1]	Duty cycle of the PWM output	0_1
PWM Frequency	[Hz]	Frequency of the PWM	0 ... 5 MHz

Output type: Angle Based Output

The input is depending on the GUI settings of the checkbox "Activate input ports". If the checkbox is selected, the parameter of the GUI (described before) is connected to an input port.

Output

The processor out block has got one output register which provide the functionality of a simple multiplexer which is depending on the actual GUI settings. Because of the complex visualization, no register overview is shown here.

FPGA Main Component

Block

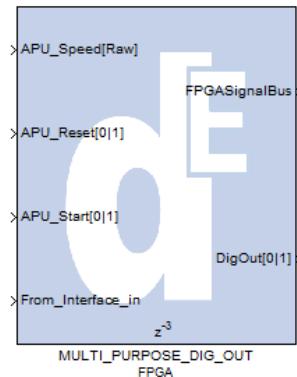


Figure 339: MULIT_PURPOSE_DIG_OUT Main block

Block Dialog

The FPGA main blockset contains the following dialog.

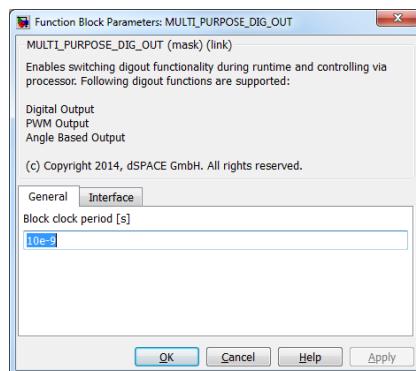


Figure 340: MULIT_PURPOSE_DIG_OUT_out dialog



The parameters of the input and output registers can be defined for automatic interface generation (When checkbox is enabled) on the page Interface. The dialog also contains an additional button to start interface generation. For more details about automatic interface generation, refer to Automatic Interface Generation chapter.

Input

main block has the following inputs:

Name	Unit	Description	Format
APU_Speed	[Raw]	Actual speed	Fix_30_0
APU_Reset	[-]	Reset on high state	Bool
APU_Start	[-]	Start on high state	UFix_1_0
From_Interface_in	BUS	Configuration bus which collected all specific parameters from interface	-

Output

The APU main block has the following outputs with a maximal latency of two:

Name	Unit	Description	Format
FPGASingalBus	Bus	Bus containing the most significant signals of this block	-
DigOut	[-]	Digital output	Bool

Interface Examples

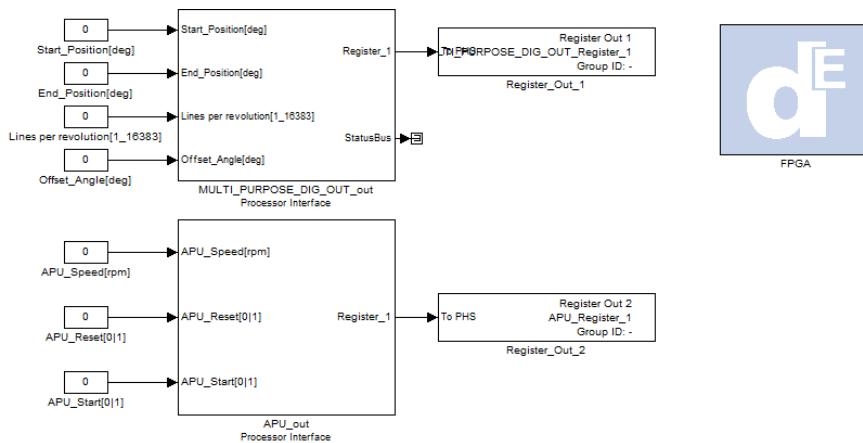
Processor blocks

Figure 341: Processor interface

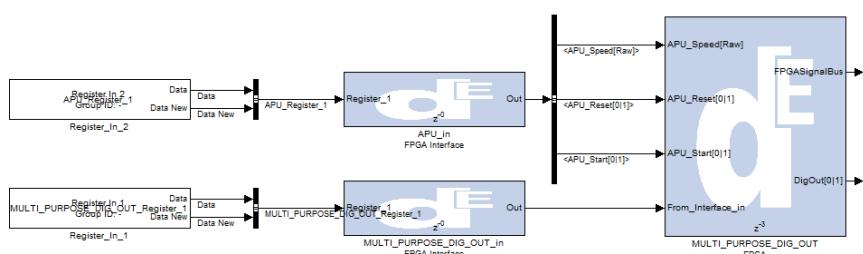
FPGA blocks

Figure 342: FPGA interface

Watchdog

Objective This sub-library contains a watchdog for resetting FPGA signals into a safe state if the communication with the processor times out. The watchdog is divided into 2 components:

- a. The central component for generating the alive pulse signal on the processor side and setting the timeout flag on the FPGA side and
- b. The execution components, which are inserted for each signal which is required to enter a safe state in case of a watchdog timeout.

Content The section contains the following components:

- **WATCHDOG_CENTRAL**
- **WATCHDOG_EXECUTION**

WATCHDOG_CENTRAL

Objective The central component of the watchdog is inserted exactly once. The processor part has to be placed in a task which is regularly called.

Content The blockset contains the following elements:

- | | |
|-------------------------------|---|
| ▪ Processor Interface: | WATCHDOG_CENTRAL_out
(Processor Interface) |
| FPGA Interface: | WATCHDOG_CENTRAL_in
(FPGA Interface) |
| FPGA: | WATCHDOG_CENTRAL
(FPGA Main Component) |

Processor Output

Block Merges the processor signals and writes them to the FPGA.



Figure 343: WATCHDOG_CENTRAL_out block

Block Dialog

The processor output block provides the following dialog:

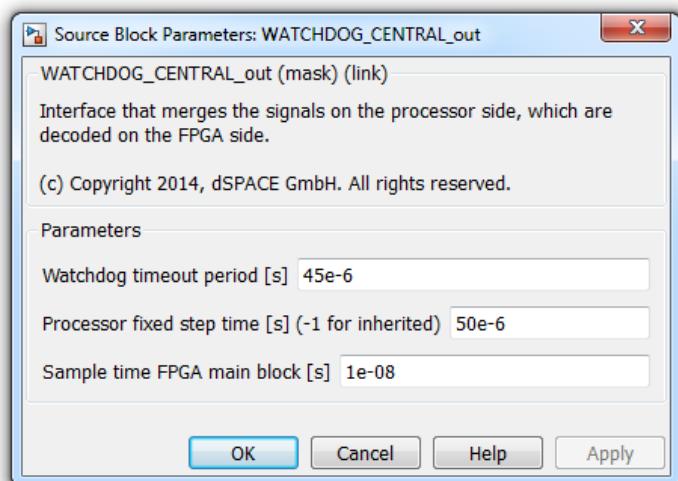


Figure 344: WATCHDOG_CENTRAL_out dialog

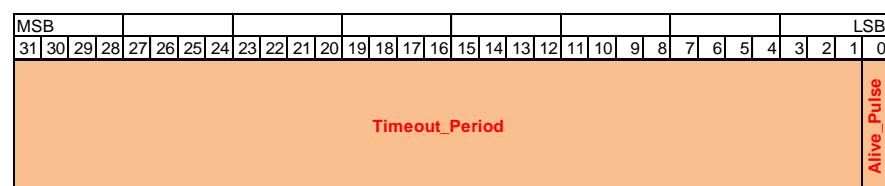
The blockset dialog has the following parameters:

Name	Unit	Description	Range/Resolution
Watchdog timeout period	s	The time after which signals are to be set into a safe state if the processor model does not respond	Range: 0...17.18s Resolution: 8ns (*)
Processor fixed step time	s	The processor sample time for offline simulation	-
Sample time FPGA main block	s	The sample time the FPGA main component is clocked with. Usually this is the FPGA clock rate, however in case of downsampling of the FPGA component this parameter has to be adapted.	-

- (*) Assuming FPGA sample time of 8 ns. For other sample times, the ranges and resolutions scale accordingly.

Output

The Processor Out block outputs 1 register content. The sectioning is shown below:

Register 1

FPGA Main Component

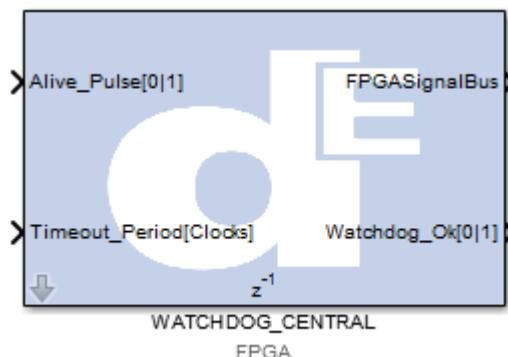
Block

Figure 345: WATCHDOG_CENTRAL FPGA Main block

Block Dialog

You can define the parameters of the input and output registers for automatic interface generation. The dialog also contains a button to start interface generation. For more details about automatic interface generation, refer to the XSG Utils documentation chapter Automatic Interface Generation.

Input

The main block has the following inputs:

Name	Unit	Description	Format
Alive_Pulse	-	A pulse from the processor model, signalizing the model is still executed.	Bool
Timeout_Period	Clocks	The maximum period allowed for no response of the model before setting the "Watchdog_Ok" output to 0.	UFix_31_0

Output

The main block has the following outputs:

Name	Unit	Description	Format
FPGASignalBus	-	Contains all output signals generated by the FPGA main component.	Bus
Watchdog_Ok	-	The output signal indicating that the processor model is still responding.	Bool

FPGA Resources

The FPGA (XC7K325T) resources occupied by the component are negligible.

WATCHDOG_EXECUTION

Description / Overview

The WATCHDOG_EXECUTION block has to be inserted for all signals which are to set into a safe state on watchdog timeout.

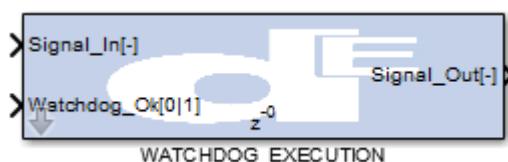
Block

Figure 346: WATCHDOG_EXECUTION block

Block Dialog

The block provides the following dialog:

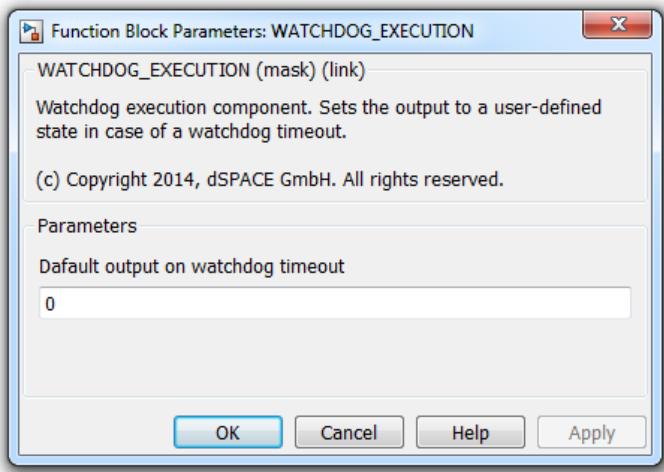


Figure 347: WATCHDOG_EXECUTION dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Default output on watchdog timeout	-	The default safe signal state to be assigned on a watchdog timeout	-inf...inf

Input

The WATCHDOG_EXECUTION block has the following inputs:

Name	Unit	Description	Format
Signal_In	-	The signal to which the watchdog check has to be assigned.	-
Watchdog_Ok	-	Flag indicating if the processor model is still responding. This signal is generated by the WATCHDOG_CENTRAL component.	Bool

Output

The WATCHDOG_EXECUTION block has the following outputs:

Name	Unit	Description	Format
Signal_Out	-	Outputs either the input signal or – in case of a watchdog timeout – the safe default output state.	-

FPGA Resources

The FPGA (XC7K325T) resources are occupied by the component are negligible.

Small APPS

Content

See the following sections for information about the blocks of the subsystem SMALL APPS. These blocks located provide an easy solution for the most used functions, such as edge detection, clock recovery, timing counter, switch off delay, etc. These blocks do not provide a complete interface blockset, but are designed to simplify the implementation of user-defined functionality in user-defined FPGA code.

EDGE DETECTION

Description / Overview

The EDGE_DETECTION block detects a falling or rising edge or both (change of input) and generates an output trigger impulse. Additional to that, the block detects if a signal has changed in relation to a specified previous sample (the delay can be specified, that is the very important point). This can for example be used to compare time-multiplexed signals. The block supports Fixed and Floating point (Single and Double precision) datay types.

Block



Figure 348: EDGE_DETECTION block (Floating Point Variant) for Comparision Sample setting = 1

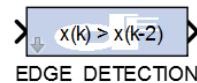


Figure 349: EDGE_DETECTION block for Comparision Sample setting > 1

Block Dialog

The FPGA block contains the following dialog.

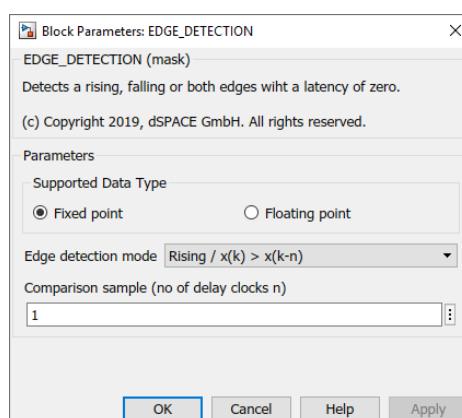


Figure 350: EDGE_DETECTION dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Edge detection mode	[-]	Lets you select the edge detection mode	<ul style="list-style-type: none"> - Falling / $x(k) < x(k-n)$ - Rising / $x(k) > x(k-n)$ - Both / $x(k) \neq x(k-n)$

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input of the edge detection	Floating Point: User-defined – No Bool supported! Fixpoint: Bool + UFix + Fix

Output

The EDGE_DETECTION block has the following outputs:

Name	Unit	Description	Format
Out	[0 1]	Output of the edge detection block	UFix_1_0

Workflow / Example

The following example shows the transfer behavior of the FPGA block.

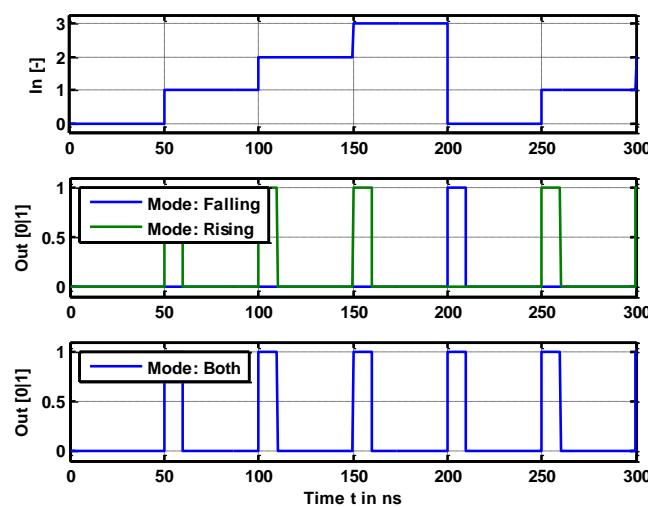


Figure 351: Workflow of the EDGE_DETECTION block

RELAY

Description / Overview

The RELAY block detects when the input signal crosses the on value in positive direction (rising edge), the output state is set to high state. When the input signal crosses the off value in negative direction (falling edge), the output state is set to low state. The Switch on value must be greater than switch off value. If not, the values are automatically swapped.

Block

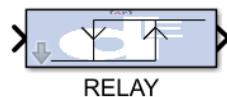


Figure 352: RELAY block

Block Dialog

The FPGA block contains the following dialog.

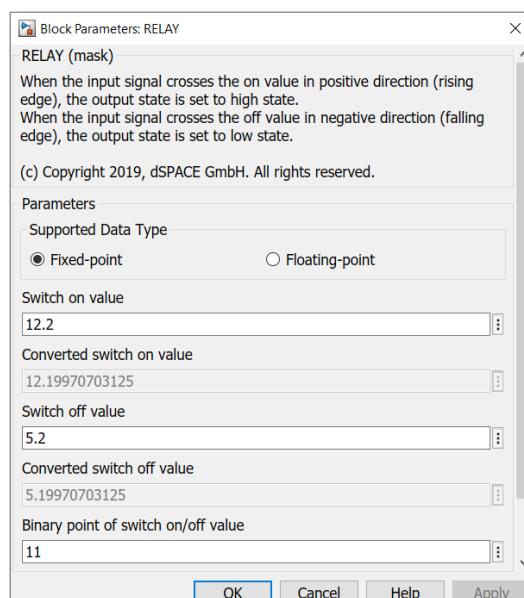


Figure 353: RELAY dialog for fixed point

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Switch on value	[-]	Switch on value	-
Converted switch on value	[-]	Converted switch on value in fixed-point format	-
Switch off value	[-]	Switch off value	-
Converted switch off value	[-]	Converted switch off value in fixed-point format	-
Binary point of switch on/off value	[-]	Binary point of switch on/off value	-

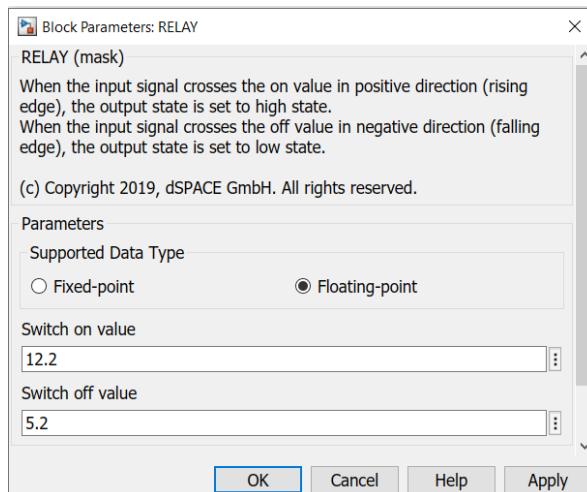


Figure 354: RELAY dialog for floating point

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Switch on value	[-]	Switch on value	-
Switch off value	[-]	Switch off value	-

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	User-defined

Output

The RELAY block has the following outputs:

Name	Unit	Description	Format
Out	[0 1]	Output	Ufix_1_0

ABS

Description / Overview

The ABS block outputs the absolute value of the input.

Block

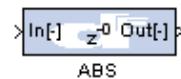


Figure 355: ABS block

Block Dialog

The FPGA block contains the following dialog.

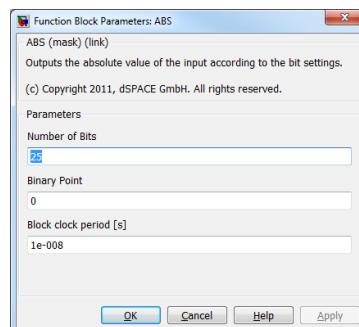


Figure 356: ABS dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Number of bits	[-]	Number of output bits	User-defined
Binary Point	[-]	Binary point of the output	User-defined
FPGA clock period period	[s]	Clock period of the FPGA	-

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	Signed values

Output

The ABS block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Absolute value of the input	UFix_x_x (bit length and binary point are user-defined via dialog)

Workflow / Example

The following example shows the transfer behavior of the FPGA block.

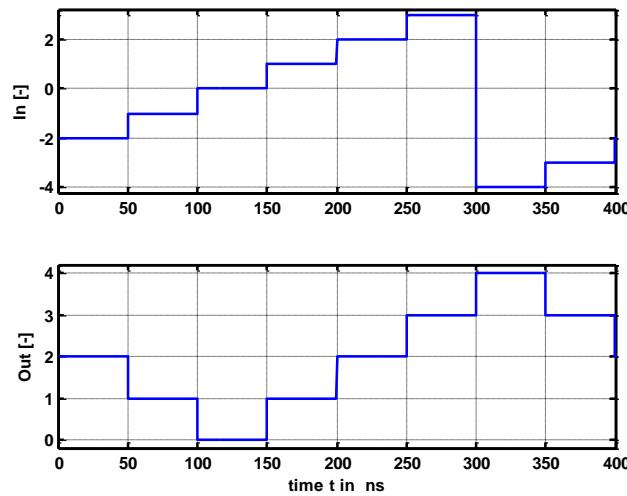


Figure 357: Workflow of the ABS block

COMP2CONST

Description / Overview

The Comp2Const block compares the input value for different modes to a user defined constant value. This block supports both fixed and floating point data types. For fixed point data type, the number of bits and the binary point of the constant value can be user defined. Single and double precision is supported for the floating point data type.

Block

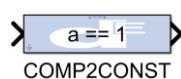


Figure 358: COMP2CONST block for fixed point

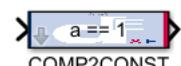


Figure 359: COMP2CONST block for floating point

Block Dialog

The FPGA block contains the following dialog.

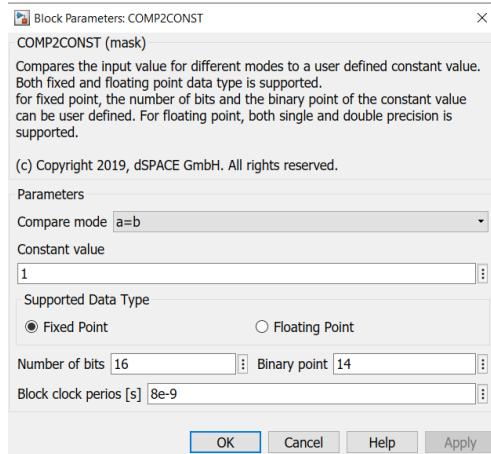


Figure 360: COMP2CONST dialog for fixed point

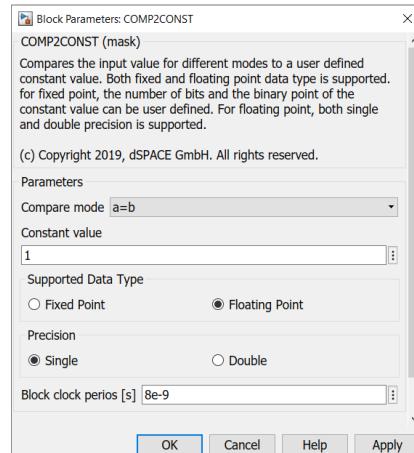


Figure 361: COMP2CONST dialog for floating point

The dialog blockset has the following parameters for fixed point:

Name	Unit	Description	Range
Compare mode	[]	Define the actual compare mode; available modes: a = b a! = b a < b a > b a <= b a >= b	-
Const value	[-]	Constant value	-
Number of bits	[-]	Number of bits of the constant value	-
Binary point	[-]	Binary point of the constant value	-

The dialog blockset has the following parameters for floating point:

Name	Unit	Description	Range
Compare mode	[-]	Define the actual compare mode; available modes: a = b a! = b a < b a > b a <= b a >= b	-
Const value	[-]	Constant value	-

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	User-defined

Output

The COMP2CONST block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	UFix_1_0

MOD

Description / Overview

The Mod block computes modulo based on the input value and the user defined divisor. This block supports both fixed and floating point data types. For fixed point data type, the binary point of the divisor can be user defined. For floating point, single precision is supported.

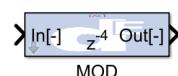
Block

Figure 362: MOD block for fixed point

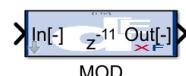


Figure 363: MOD block for floating point

Block Dialog

The FPGA block contains the following dialog.

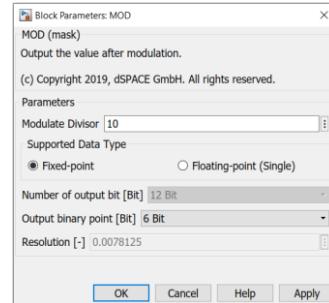


Figure 364: MOD dialog for fixed point

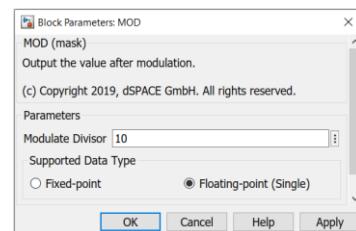


Figure 365: MOD dialog for floating point

The dialog blockset has the following parameters for fixed point:

Name	Unit	Description	Range
Modulate Divisor	[-]	Divisor value	-
Number of bits	[-]	Number of bits of the divisor value	-
Binary point	[-]	Binary point of the divisor value	-

The dialog blockset has the following parameters for floating point:

Name	Unit	Description	Range
Modulate Divisor	[-]	Divisor value	-

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	User-defined

Output

The MOD block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	User-defined

SINE / COSINE

Description / Overview	The SINE and COSINE blocks output the sine and cosine depending on an input address (0 – 2^14-1) which represents 0 – 360 degrees with a maximal latency of two.
-------------------------------	--

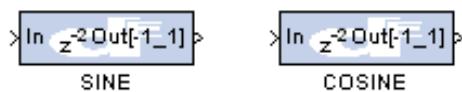
Block


Figure 366: SINE / COSINE blocks

Block Dialog

The FPGA main blockset contains the following dialog.

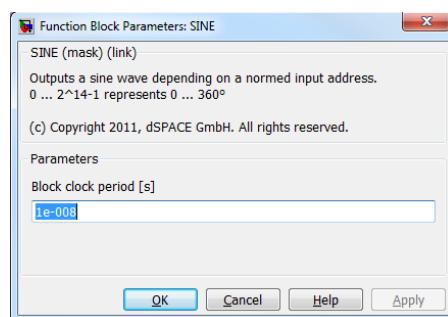


Figure 367: SINE / COSINE dialog

The dialog only provides a short block description with a parameter to configure the clock period of the FPGA.

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Normalized input address 0 ... 2^14-1 represents 0 ... 360°	UFix_14_0; 0 ... 2^14-1 represents 0 ... 360°

Output

The SINE / COSINE blocks have the following outputs:

Name	Unit	Description	Format
Out	[-]	Sine / cosine function depending on the input address	Fix_14_13

Workflow / Example

The following example shows the transfer behavior of the FPGA block.

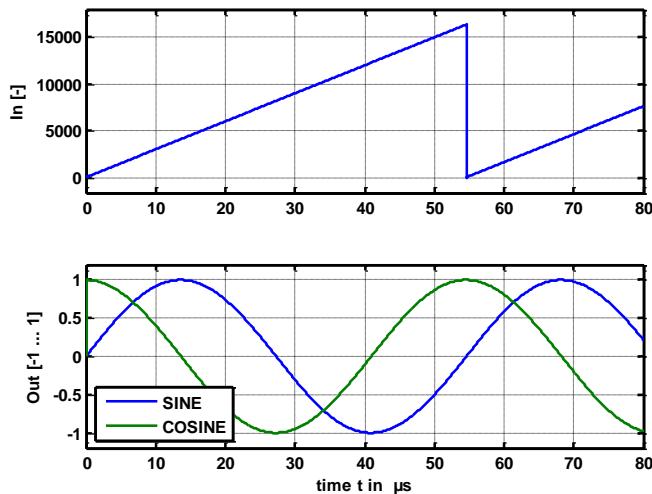


Figure 368: Workflow of the SINE / COSINE blocks

SWITCH OFF DELAY

Description / Overview Delays a Boolean input signal with the number of parameterized delay steps with an additional latency of zero.

Block



Figure 369: SWITCH_OFF_DELAY block

Block Dialog

The FPGA block contains the following dialog.

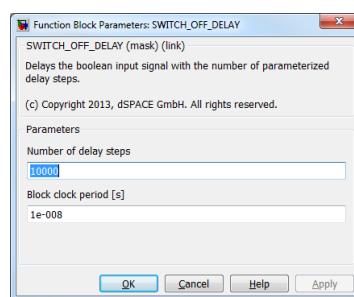


Figure 370: SWITCH_OFF_DELAY dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Number of Delay Steps	[-]	Number of delay steps.	0 ... 2^10-1
FPGA clock period	[s]	Clock period of the FPGA	-

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	Bool

Output

The SWITCH_OFF_DELAY block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Delayed output	Bool

Workflow / Example

The following example shows the transfer behavior of the FPGA block.

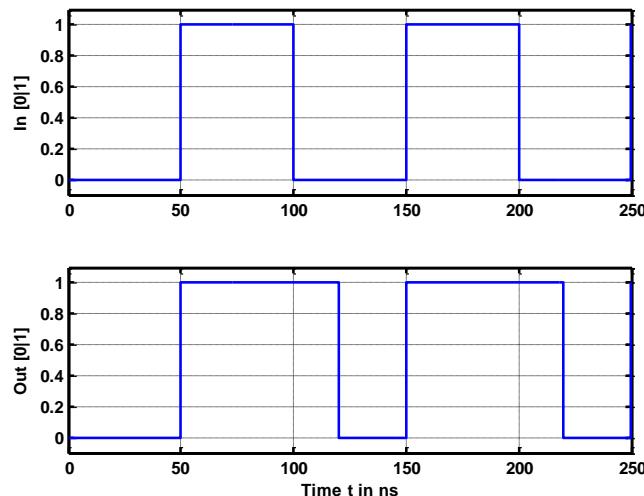


Figure 371: Workflow of the ABS block

MULTIFUNCTION BLOCK

Description / Overview

The MULTIFUNCTION_BLOCK combines the functionalities of a Delay, a Switch Left and a Switch Right function.



Be careful of the bit length and the binary point of this block's input and output. The output has the same bit length and binary point as the input.

Block



Figure 372: MULTIFUNCTION_BLOCK block

Block Dialog

The FPGA block contains the following dialog.

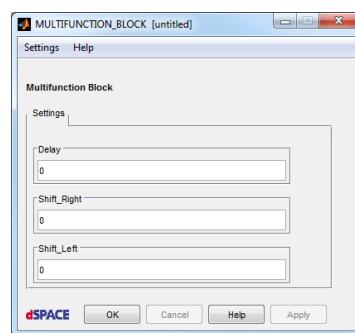


Figure 373: MULTIFUNCTION_BLOCK dialog

The MULTIFUNCTION_BLOCK blockset has the following parameters:

Name	Unit	Description	Range
Delay	[-]	Number of delay steps	User-defined
Shift Right	[-]	Number of shift bits to the right	User-defined
Shift Left	[-]	Number of shift bits to the left	User-defined

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	Unsigned and signed values

Output

The MULTIFUNCTION_BLOCK block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	Unsigned and signed values

Workflow / Example

The following example shows the transfer behavior of the FPGA block with a delay of three and a right shift of one bit.

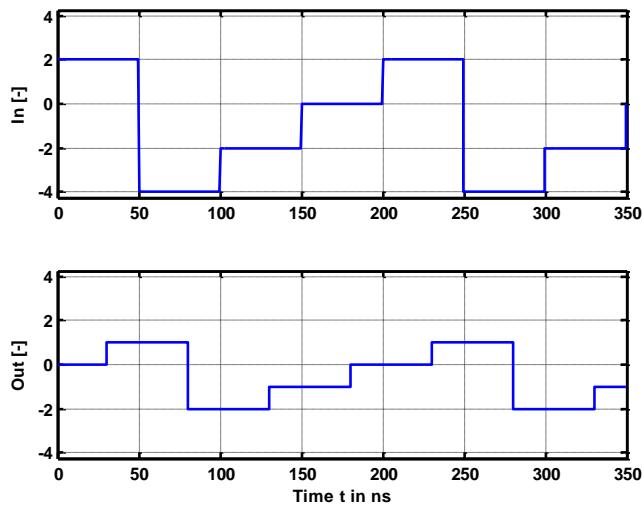


Figure 374: Workflow of the MULTIFUNCTION_BLOCK block

DOWNSAMPLE

Description / Overview

The DOWNSAMPLE block realizes a specific downsampling of the input signals. If a trigger edge is detected in the downsampling period an additional trigger will be generated at the output. For further detailed please refer to the workflow chapter. This block can be used to relax the timing if the timing constrain is very closer to the FPGA clock rate is used.

Block

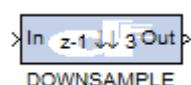


Figure 375: DOWNSAMPLE block

Block Dialog

The FPGA block contains the following dialog.

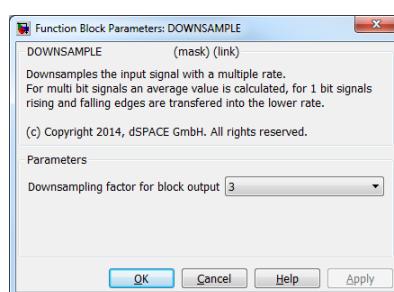


Figure 376: DOWNSAMPLE dialog

The DOWNSAMPLE blockset has the following parameters:

Name	Unit	Description	Range
Downsample factor	[-]	Downsample factor of the block output	User-defined

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	User-defined

Output

The DOWNSAMPLE block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	User-defined

Example

The following example shows the transfer behavior of the FPGA block for a “digital” signal (Bool or UFix1_0).

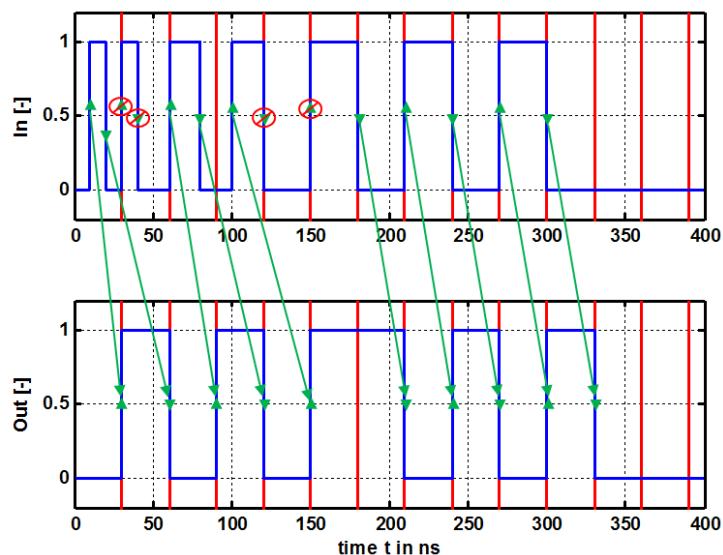


Figure 377: Workflow of the DOWNSAMPLE block for “digital” signals

The DOWNSAMPLE block analyzes the actual signal in the faster input task. If a rising or falling edge is detected, the block stores this behavior and performs an equidistant output on the faster output. With this simple logic trigger signals can be transferred to the slower output task.

If the block transfers an “analog” signal (unsigned or signed values with a greater numbers of bits as one) is shown in the figure below.

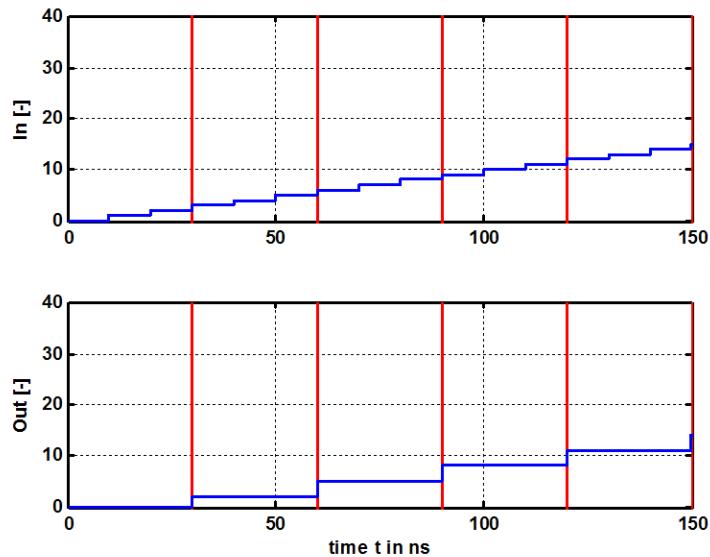


Figure 378: Workflow of the DOWNSAMPLE block for “analog” signals

For “analog” signals only the downsampling functionality is implemented.

Usage in model / Example

To get a better overview of the usage of the DOWNSAMPLE block in a model the following illustration is used.

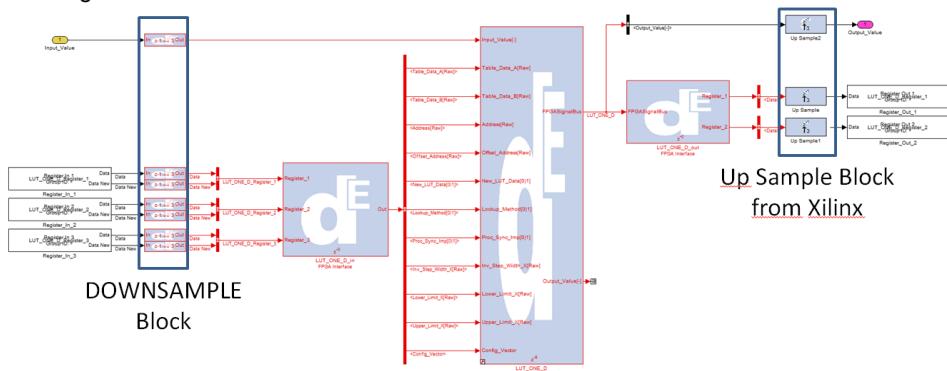


Figure 379: Adoptions for downsampling a model part

All input signals of the FPGA component are routed through the DOWNSAMPLE block as shown in the figure. The FPGA model is calculated in a lower sampled task (marked with the foreground color red) with the downsampling factor of three. The FPGA main component must have the right block period as shown in the figure below.



Figure 380: Adoptions in the block GUI of the FPGA components

To satisfy the original sampling rate the output signals must be up sampled to the original task with the usage of the “Up Sample” block of the Xilinx library.

Separate Timing and Average Functionalities

Description / Overview

The following blocks (TIMING_COUNTER_PROCESSOR_SYNC; TIMING_COUNTER and SIGNAL_AVERAGE_CALC) let you realize separate average functionalities with different timing behavior. The blocks can be connected in two different ways:

- A.) Calculate average values synchronous to a toggling bit (often used for processor-synchronous average calculation)
- B.) Calculate average values synchronous to a parameterized length of a counter

The connection types are explained in the next chapters.

A.) Block structure

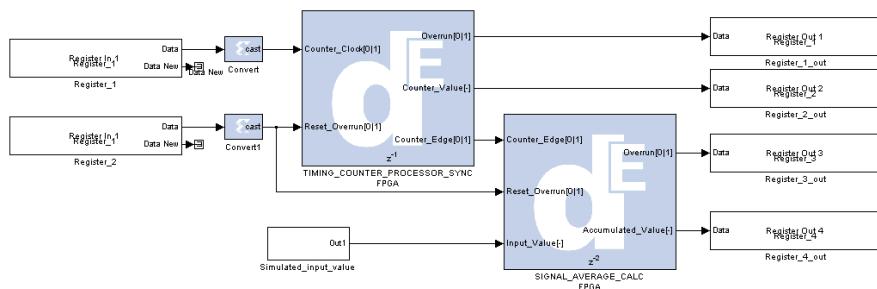


Figure 381: Block structure of average calculation synchronous to a toggling bit

A.) Block Dialog

The dialog only provides a short block description with a parameter to configure the clock period of the FPGA.

A.) Input

The TIMING_COUNTER_PROCESSOR_SYNC block has the following inputs:

Name	Unit	Description	Format
Counter_Clock	[-]	Toggle bit which toggles each processor task	UFix_1_0
Reset_OVERRUN	[-]	Resets the accumulator of the internal counter	Bool

The SIGNAL_AVERAGE_CALC block has the following inputs:

Name	Unit	Description	Format
Counter_Edge	[-]	Counter flag for the output of the average counter value	UFix_1_0
Reset_OVERRUN	[-]	Resets the accumulator of the internal counter	Bool
Input_Value	[-]	Input value to calculate the average	Fix_32_0

A.) Output

The TIMING_COUNTER_PROCESSOR_SYNC block has the following outputs:

Name	Unit	Description	Format
Overrun	[-]	Overrun flag of the counter	Unsigned and signed values
Counter_Value	[-]	Accumulated values of the counted FPGA sample steps	U_Fix_32_0
Counter_Edge	[-]	Counter flag for the output of the average counter value	UFix_1_0

The SIGNAL_AVERAGE_CALC block has the following outputs:

Name	Unit	Description	Format
Overrun	[-]	Overrun flag of the counter	Unsigned and signed values
Accumulated_Value	[-]	Accumulated values between the toggling bit of the input Counter_Edge	Fix_32_0

A.) Workflow / Example

The following example shows the transfer behavior of the FPGA block with a simulated toggle bit of 200e-6 s.

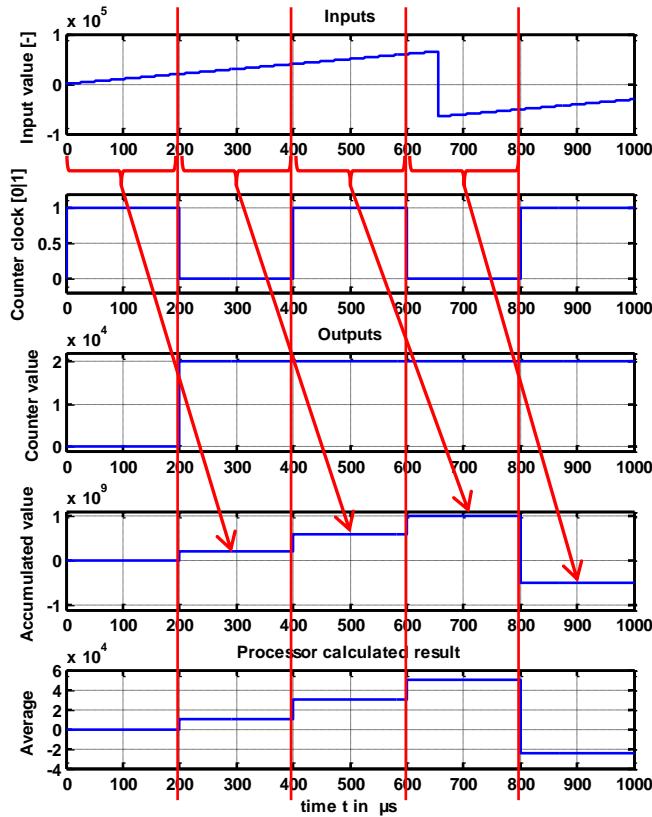


Figure 382: Workflow of the average calculation synchronous to a toggling bit

The actual accumulated value (SIGNAL_AVERAGE_CALC block) and the counter value (TIMING_COUNTER_PROCESSOR_SYNC) are output synchronously to the counter clock. The average is found by dividing the accumulated value by the counter value on a processor board.

B.) Block structure

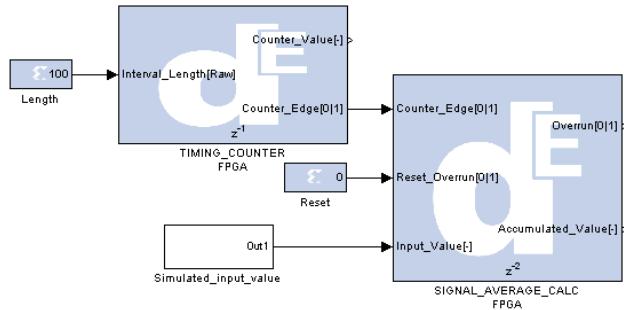


Figure 383: Block structure of average calculation synchronous to a counter length

B.) Block Dialog

The dialog only provides a short block description with a parameter to configure the clock period of the FPGA.

B.) Input

The TIMING_COUNTER block has the following inputs:

Name	Unit	Description	Format
Interval_Length	[-]	Interval length of the average accumulation	UFix_32_0

The SIGNAL_AVERAGE_CALC block has the following inputs:

Name	Unit	Description	Format
Counter_Edge	[-]	Counter flag for the output of the average counter value	UFix_1_0
Reset_OVERRUN	[-]	Resets the accumulator of the internal counter	Bool
Input_Value	[-]	Input value to calculate the average	Fix_32_0

B.) Output

The TIMING_COUNTER_PROCESSOR_SYNC block has the following outputs:

Name	Unit	Description	Format
Counter_Value	[-]	Accumulated values of the counted FPGA sample steps	U_Fix_32_0
Counter_Edge	[-]	Counter flag for the output of the average counter value	UFix_1_0

The SIGNAL_AVERAGE_CALC block has the following outputs:

Name	Unit	Description	Format
Overrun	[-]	Overrun flag of the counter	Unsigned and signed values
Accumulated_Value	[-]	Accumulated values between the toggling bit of the input Counter_Edge	Fix_32_0

B.) Workflow / Example

The following example shows the transfer behavior of the FPGA blocks with a simulated interval length of 25000 FPGA samples.

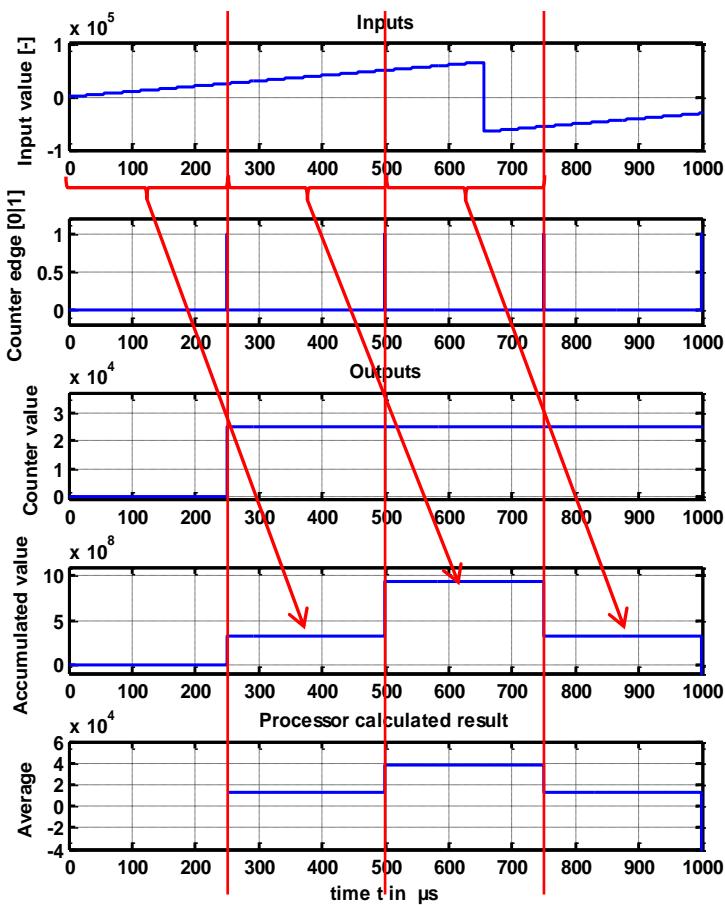


Figure 384: Workflow of the average calculation synchronous to a toggling bit

The actual accumulated value (SIGNAL_AVERAGE_CALC block) and the counter value (TIMING_COUNTER) are output synchronously to the counter edge. The average is found by dividing the accumulated value by the counter value on a processor board.

Discrete First-Order Lag Element (PT1)

Description / Overview	The DISCRET_PT1 block outputs a filtered input signal with the parameterized filter time.
-------------------------------	---

Block

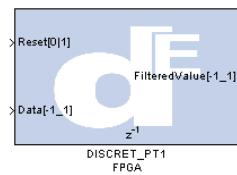


Figure 385: DISCRET_PT1 block

Block Dialog

The FPGA block contains the following dialog.

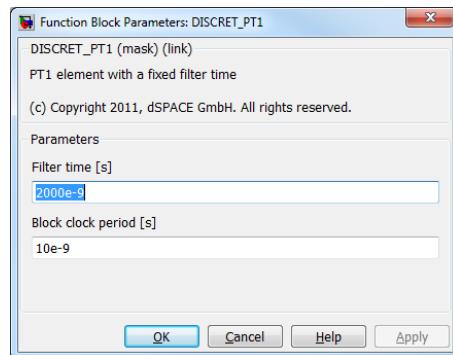


Figure 386: DISCRET_PT1 dialog

The DISCRET_PT1 blockset has the following parameters:

Name	Unit	Description	Range
Filter time	[s]	Filter time of the discrete first-order lag	10E-9 ... 2.62143E-3

Input

The main block has the following inputs:

Name	Unit	Description	Format
Reset	[]	Reset of the first-order lag	Bool
Data	[-]	Data input	Fix_25_24

Output

The DISCRET_PT1 block has the following outputs:

Name	Unit	Description	Format
FilteredValue	[-]	Output	Fix_25_24

Workflow / Example

The following example shows the transfer behavior of the FPGA block.

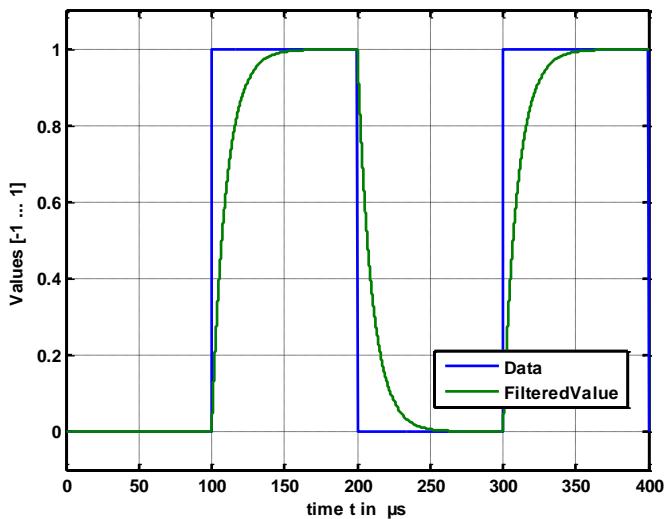


Figure 387: Workflow of the DISCRET_PT1 block

BACKLASH

Description / Overview The BACKLASH block outputs a high signal if the input overshoots the on level and a low signal if the input undershoots the off level.

Block

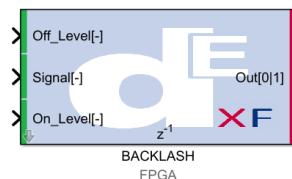


Figure 388: BACKLASH block

Block Dialog The dialog only provides a short block description.

Input The main block has the following inputs:

Name	Unit	Description	Format
Off_Level	[-]	Switch off level	User-defined
Signal	[-]	Signal Input	User-defined
On_Level	[-]	Switch on level	User-defined

Output The BACKLASH block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	UFix_1_0

Workflow / Example

The following example shows the transfer behavior of the FPGA block with a switch on level of 0.25 and an switch off level of -0.25.

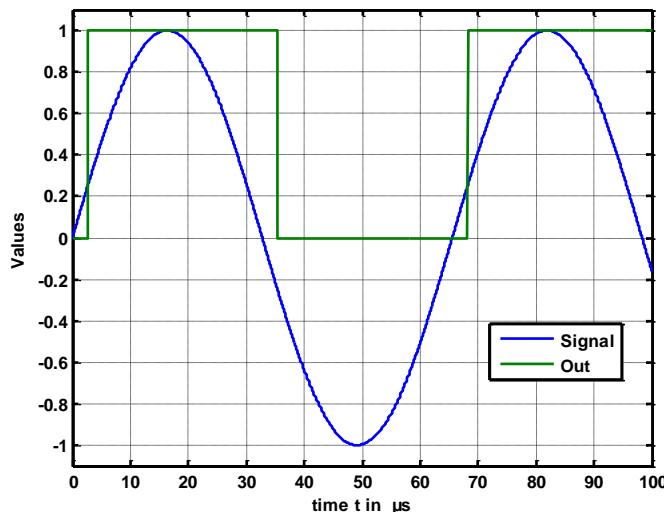


Figure 389: Workflow of the BACKLASH block

VALID_EDGES

Description / Overview

The VALID_EDGES block outputs a valid edge if the edge is set to level zero or one for a defined length of FPGA sample steps.

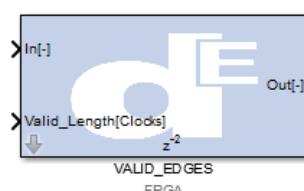
Block

Figure 390: VALID_EDGES block

Block Dialog

The dialog only provides a short block description with a parameter to configure the clock period of the FPGA.

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	U_Fix_1_0
Valid_Length	[-]	Detection of the valid length (in FPGA sample steps) of edge which is interpreted as a valid edge	U_Fix_x_0 (x <= 18)

Output

The VALID_EDGES block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Valid edges	Bool

Workflow / Example

The following example shows the transfer behavior of the FPGA block with a valid length of 10 FPGA samples.

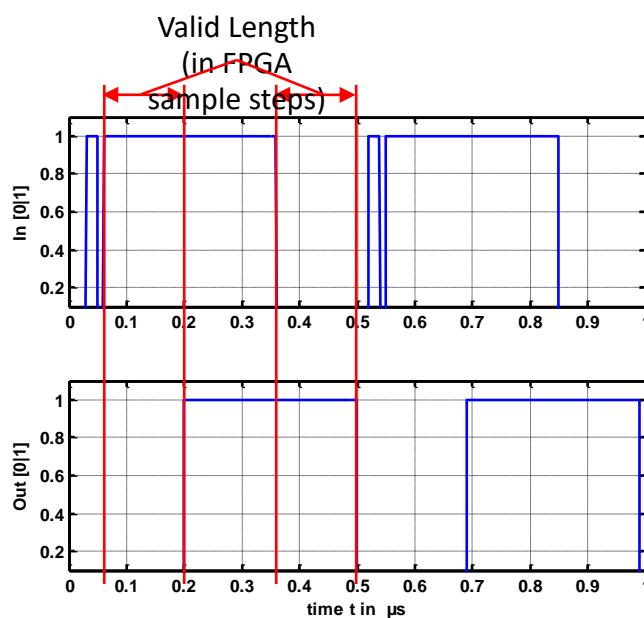


Figure 391: Workflow of the VALID_EDGES block

DYNAMIC_SATURATION

Description / Overview

The DYNAMIC_SATURATION block saturates the output to the defined maximum and minimum output levels. This block supports both fixed and floating point input data type.

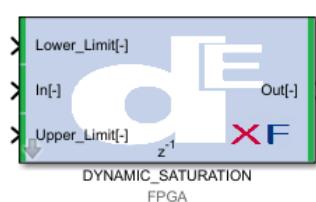
Block

Figure 392: DYNAMIC_SATURATION block

Block Dialog

The FPGA block contains the following dialog.

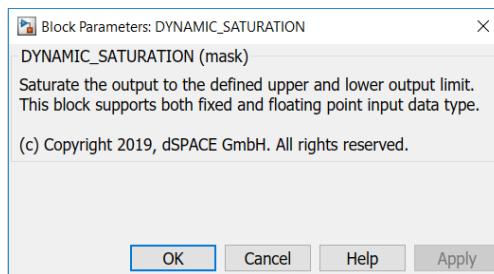


Figure 393: DYNAMIC_SATURATION dialog

Input

The main block has the following inputs:

Name	Unit	Description	Format
Lower_Limit	[-]	Lower limit of the output	User-defined
In	[-]	Input	User-defined
Upper_Limit	[-]	Upper limit of the output	User-defined

Output

The DYNAMIC_SATURATION block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Saturated output	User-defined

Workflow / Example

The following example shows the transfer behavior of the FPGA block with an upper limit of 0.5 and a lower limit of -0.75.

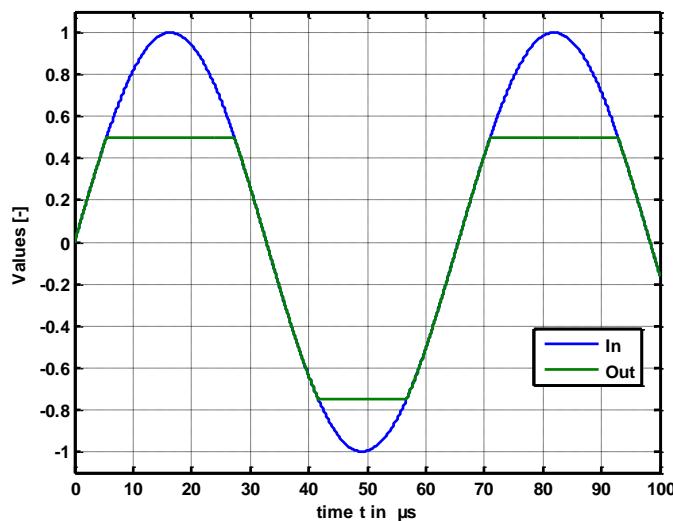


Figure 394: Workflow of the DYNAMIC_SATURATE block

DYNAMIC_DELAY

Description / Overview

The DYNAMIC_DELAY block delays a signal with a user-defined delay parameterized via an input. This block supports both fixed and floating point (single and double precision) input data types.

Block

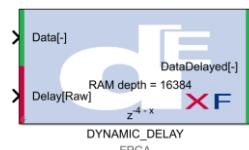


Figure 395: DYNAMIC_DELAY block

Block Dialog

The FPGA block contains the following dialog.

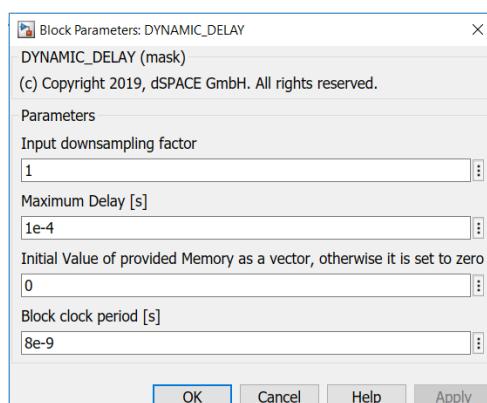


Figure 396: DYNAMIC_DELAY dialog

The DYNAMIC_DELAY blockset has the following parameters:

Name	Unit	Description	Range
Input downsampling factor	[\cdot]	Downsampling factor of the data input	User-defined
Maximum Delay	[s]	Maximum range of the necessary delay	User-defined
Initial value	[\cdot]	Initial value of the delay	User-defined
FPGA clock period	[s]	Clock period of the FPGA	User-defined

Input

The main block has the following inputs:

Name	Unit	Description	Format
Data	[\cdot]	Input of the delay	User-defined
Delay	[\cdot]	Parameterized delay of the data signal	User-defined

Output

The DYNAMIC_DELAY block has the following outputs:

Name	Unit	Description	Format
DataDelayed	[-]	Delayed data output	User-defined

Workflow / Example

The following example shows the transfer behavior of the FPGA block with an input downsampling of 2, a maximum delay of 1e-4s and an initial value of 0.

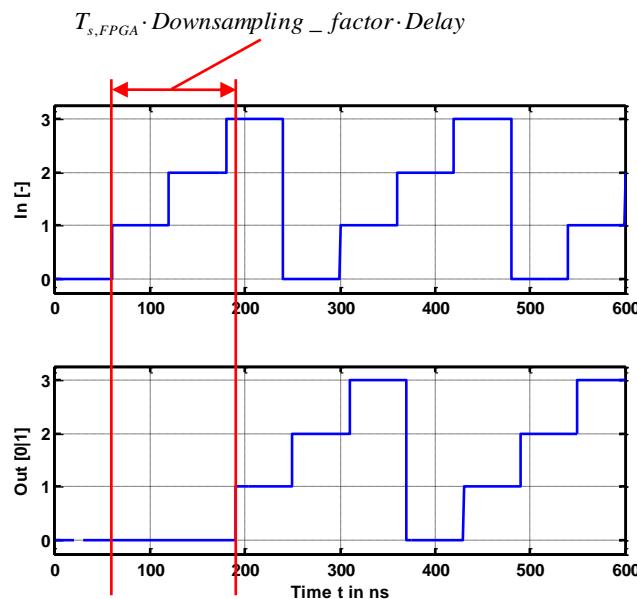


Figure 397: Workflow of the DYNAMIC_DELAY block

DEFAULT_LATCH_TRIGGER_INSERTION

Description / Overview

The DEFAULT_LATCH_TRIGGER_INSERTION block generates a default interrupt impulse if no trigger is measured in the input.

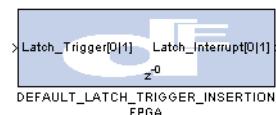
Block

Figure 398: DEFAULT_LATCH_TRIGGER_INSERTION block

Block Dialog

The FPGA block contains the following dialog.

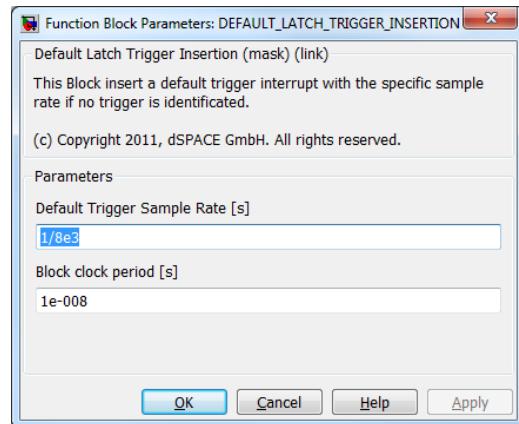


Figure 399: DEFAULT_LATCH_TRIGGER_INSERTION dialog

The DEFAULT_LATCH_TRIGGER_INSERTION blockset has the following parameters:

Name	Unit	Description	Range
Default trigger sample time	[s]	Default trigger sample time if no trigger is detected in the input	User-defined
FPGA clock period	[-]	Clock period of the FPGA	-

Input

The main block has the following inputs:

Name	Unit	Description	Format
Latch_Trigger	[-]	input	Bool

Output

The DEFAULT_LATCH_TRIGGER_INSERTION block has the following outputs:

Name	Unit	Description	Format
Latch_Interrupt	[-]	Interrupt	Bool

Workflow / Example

The following example shows the transfer behavior of the FPGA block.

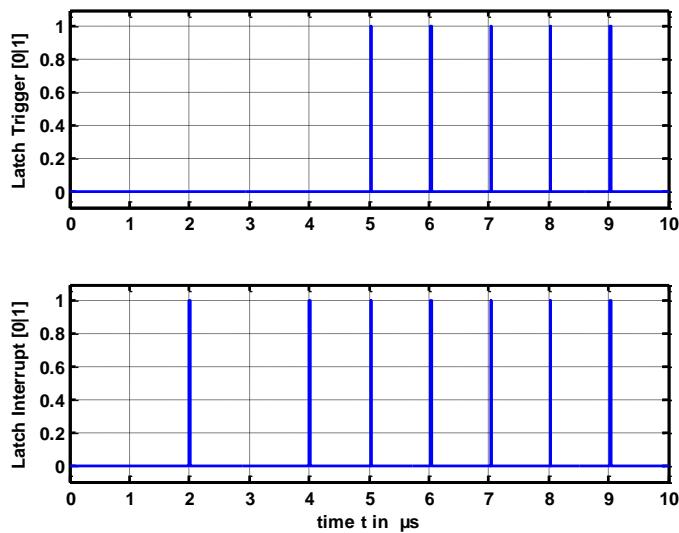


Figure 400: Workflow of the `DEFAULT_LATCH_TRIGGER_INSERTION` block

Park Modulator

Description / Overview

The `PARK_MODULATOR` block calculates the sines and cosines of the angle input to be used as a signal source for the Park-transformation blocks.

Block

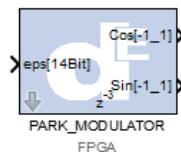


Figure 401: `PARK_MODULATOR` block

Block Dialog

The FPGA block contains the following dialog.

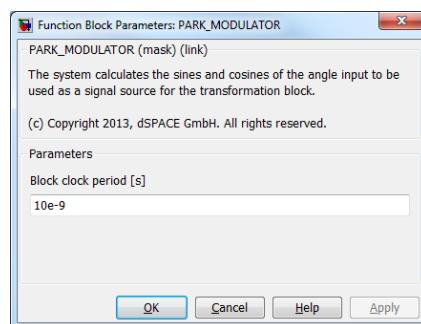


Figure 402: `PARK_MODULATOR` dialog

The `PARK_MODULATOR` blockset has the following parameters:

Name	Unit	Description	Range
FPGA block clock period	[s]	Block clock period	-

Input

The main block has the following inputs:

Name	Unit	Description	Format
eps	[s]	Actual angle	UFix_14_0

Output

The PARK_MODULATOR block has the following outputs:

Name	Unit	Description	Format
Cos	[-1_1]	Cosine track of the angle	Fix_14_13
Sin	[-1_1]	Sinus track of the angle	Fix_14_13

Park Clarke Transformation

Description / Overview

To simulate electric motor applications it's necessary to convert variables between different coordinate systems. Therefore the Clarke and / or Park transformation blocks are included.

Clarke-transformation:

The equation of the system can be written as Clarke-transformation:

Orthogonal components for three-phase system:

$$\begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix}$$

This form is invariant with respect to reference values: $x_\alpha = x_a$ when $x_0 = 0$. Moreover,

$$x_0 = \frac{1}{3} \sum_{i=a,b,c} x_i$$

is the average value of components x_a , x_b and x_c . The corresponding inverse Clarke-transformation is

$$\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix}$$

A different form of Clarke-transformation is invariant with respect to power:

$$\begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix} = \sqrt{\frac{3}{2}} \cdot \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ \frac{\sqrt{2}}{3} & \frac{\sqrt{2}}{3} & \frac{\sqrt{2}}{3} \end{bmatrix} \cdot \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix}$$

The corresponding inverse Clarke-transformation is

$$\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} x_\alpha \\ x_\beta \\ x_0 \end{bmatrix}$$

Park-transformation:

Transformation of orthogonal components to a reference system which is turned by a certain angle. The amount (geometric mean value) is unchanged:

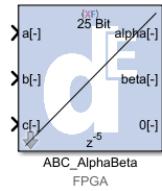
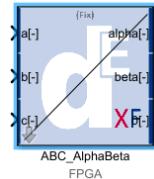
$$\begin{bmatrix} x_a \\ x_b \end{bmatrix} = \begin{bmatrix} \cos(\varepsilon) & \sin(\varepsilon) \\ -\sin(\varepsilon) & \cos(\varepsilon) \end{bmatrix} \cdot \begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix}$$

The corresponding inverse Park-transformation is

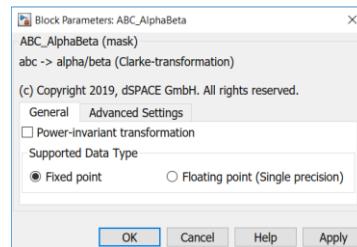
$$\begin{bmatrix} x_\alpha \\ x_\beta \end{bmatrix} = \begin{bmatrix} \cos(\varepsilon) & -\sin(\varepsilon) \\ \sin(\varepsilon) & \cos(\varepsilon) \end{bmatrix} \cdot \begin{bmatrix} x_a \\ x_b \end{bmatrix}$$

To realize a free usage of the different transformation functions four separate blocks are implemented:

- ABC_AlphaBeta
- AlphaBeta_DQ
- DQ_AlphaBeta
- AlphaBeta_ABC

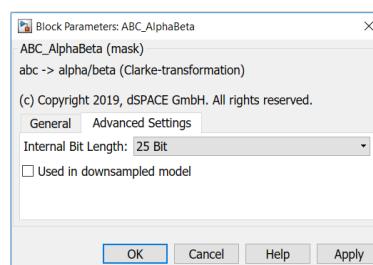
ABC_AlphaBeta**Block****Figure 403: ABC_AlphaBeta block for fixed point****Figure 404: ABC_AlphaBeta block for floating point****ABC_AlphaBeta**

The FPGA block contains the following both dialogs.

Block Dialog**Figure 405: ABC_AlphaBeta dialog - General**

The general dialog has the following parameters:

Name	Unit	Description	Range
Power-invariant transformation	[-]	Checkbox to enable a power invariant transformation	-
Supported Data Type	[-]	Selection of the input data type	-

**Figure 406: ABC_AlphaBeta dialog – Advanced Settings for fixed point**

The general dialog has the following parameters:

Name	Unit	Description	Range
Internal bit length	[-]	Specifies the internal bit length of the transformation	18 – 32 Bit
Used in downsampled model	[-]	Checkbox to enable or disable internal delays	On Off

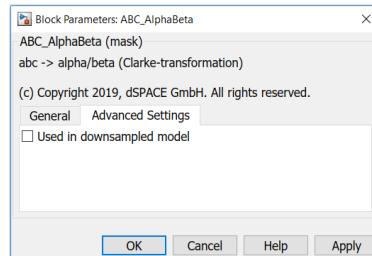


Figure 407: ABC_AlphaBeta dialog – Advanced Settings for floating point

Name	Unit	Description	Range
Used in downsampled model	[-]	Checkbox to enable or disable internal delays	On Off

ABC_AlphaBeta

The block has the following inputs:

Input

Name	Unit	Description	Format
a	[-]	data in phase a	User-defined
b	[-]	data in phase b	User-defined
c	[-]	data in phase c	User-defined

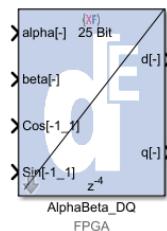
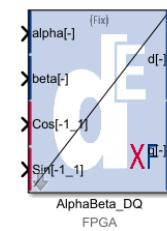
The data format of the inputs is user defined. The internal calculation is normed with the defined bit length, so that the binary point of the input data doesn't matter. The only restriction is that all input formats must be equal.

ABC_AlphaBeta

The block has the following outputs:

Output

Name	Unit	Description	Format
alpha	[-]	data in phase alpha	same as input
beta	[-]	data in phase beta	same as input
0	[-]	Zero component	same as input

AlphaBeta_DQ**Block****Figure 408: AlphaBeta_DQ block for fixed point****Figure 409: AlphaBeta_DQ block for floating point****AlphaBeta_DQ****Block Dialog**

The FPGA block contains a same dialog for “Advanced Settings” as the ABC_AlphaBeta block.

AlphaBeta_DQ

The block has the following inputs:

Input

Name	Unit	Description	Format
alpha	[-]	alpha component	User-defined
beta	[-]	beta component	User-defined
Cos	[-1_1]	Cosine component of the angle (output of the park modulator block)	Fix_14_13
Sin	[-1_1]	Sine component of the angle (output of the park modulator block)	Fix_14_13

The data format of the inputs is user defined. For fixed point input data type, the internal calculation is normed with the defined bit length, so that the binary point of the input data doesn't matter. The only restriction is that the alpha and beta input format must be equal.

AlphaBeta_DQ

The block has the following outputs:

Output

Name	Unit	Description	Format
d	[-]	data in direct axis	same as input
q	[-]	data in quadrate axis	same as input

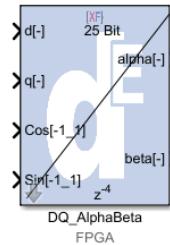
DQ_AlphaBeta**Block**

Figure 410: DQ_AlphaBeta block for fixed point

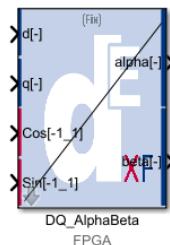


Figure 411: DQ_AlphaBeta block for floating point

DQ_AlphaBeta**Block Dialog**

The FPGA block contains a same dialog for “Advanced Settings” as the ABC_AlphaBeta block.

DQ_AlphaBeta

The block has the following inputs:

Input

Name	Unit	Description	Format
d	[-]	Component in direct axis	User-defined
q	[-]	Component in quadrate axis	User-defined
Cos	[-1_1]	Cosine component of the angle (output of the park modulator block)	Fix_14_13
Sin	[-1_1]	Sine component of the angle (output of the park modulator block)	Fix_14_13

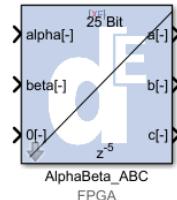
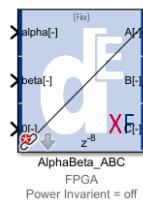
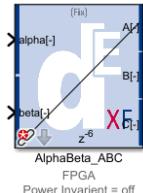
The data format of the inputs is user defined. For fixed point input data type, the internal calculation is normed with the defined bit length, so that the binary point of the input data doesn't matter. The only restriction is that the direct and quadrate axis format must be equal.

DQ_AlphaBeta

The block has the following outputs:

Output

Name	Unit	Description	Format
alpha	[-]	alpha component	same as input
beta	[-]	beta component	same as input

AlphaBeta_ABC**Block****Figure 412: AlphaBeta_ABC block for fixed point with zero component****Figure 413: AlphaBeta_ABC block for floating point with zero component****Figure 414: AlphaBeta_ABC block for floating point without zero component****AlphaBeta_ABC****Block Dialog**

The FPGA block contains a same dialog for “General” and “Advanced Settings” as the ABC_AlphaBeta block.dialog.

AlphaBeta_ABC

The block has the following inputs:

Input

Name	Unit	Description	Format
alpha	[-]	alpha component	User-defined
beta	[-]	beta component	User-defined

The data format of the inputs is user defined. For fixed point input data type, the internal calculation is normed with the defined bit length, so that the binary point of the input data doesn't matter. The only restriction is that all input formats must be equal. The zero component input is optional.

AlphaBeta _ABC

The block has the following outputs:

Output

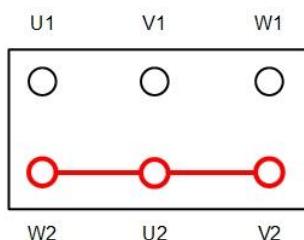
Name	Unit	Description	Format
a	[-]	data in phase a	same as input
b	[-]	data in phase b	same as input
c	[-]	data in phase c	same as input

Star / Delta conversion

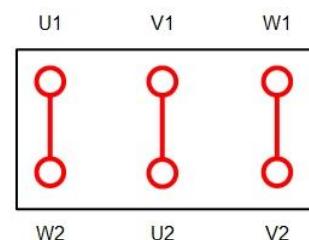
Description / Overview

The system represents a current (or voltage) conversion from three-phase supply variables to three-phase machine variables (in both directions) depending on how the machine is configured.

The following illustration shows the two main configurations:



Three-phase machine, star connected



Three-phase machine, delta connected

Figure 415: Machine configuration

For the star connected machine, the terminals U₂, V₂, W₂ are interconnected in the terminal box on the machine.

For the delta connected machine, the terminal pairs (U₁, W₂), (V₁, U₂), (W₁, V₂) are interconnected (clockwise) and the terminal pairs (U₁, V₂), (V₁, W₂), (W₁, U₂) are interconnected (counterclockwise) in the terminal box on the machine.

Current and voltage conversion of star connected machine

For the case where the input and the output variables are star, the machine phases have a common neutral point as shown in Figure below.

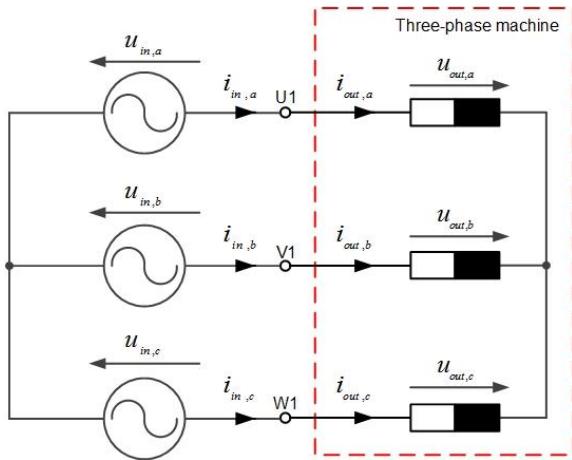


Figure 416: Machine configuration star

With respect to those variables the following expressions are valid

$$\begin{bmatrix} u_{out,a} \\ u_{out,b} \\ u_{out,c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{in,a} \\ u_{in,b} \\ u_{in,c} \end{bmatrix}$$

Note that the zero sequence voltage is not considered here in the transformation.

The inverse are given by

$$\begin{bmatrix} u_{in,a} \\ u_{in,b} \\ u_{in,c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{out,a} \\ u_{out,b} \\ u_{out,c} \end{bmatrix}$$

$$\begin{bmatrix} i_{in,a} \\ i_{in,b} \\ i_{in,c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i_{out,a} \\ i_{out,b} \\ i_{out,c} \end{bmatrix}$$

Current and voltage conversion of delta connected machine

For the case where the input variables are star and the output variables are delta (clockwise), the machine phases have not a common neutral point as shown in Figure below.

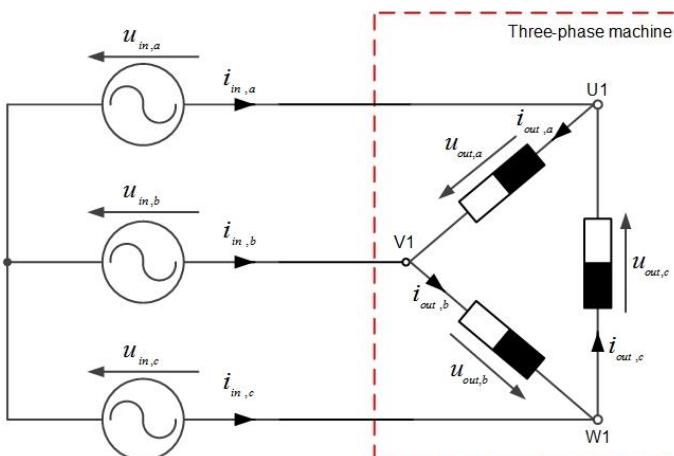


Figure 417: Machine configuration delta

With respect to those variables the following expressions are valid

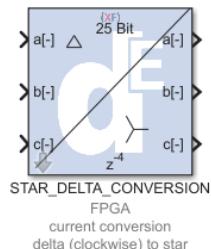
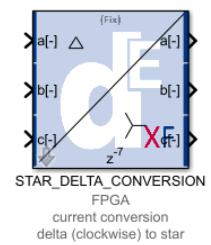
$$\begin{bmatrix} u_{out,a} \\ u_{out,b} \\ u_{out,c} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{in,a} \\ u_{in,b} \\ u_{in,c} \end{bmatrix} \quad \begin{bmatrix} i_{out,a} \\ i_{out,b} \\ i_{out,c} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & -\frac{1}{3} & 0 \\ 0 & \frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & 0 & \frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} i_{in,a} \\ i_{in,b} \\ i_{in,c} \end{bmatrix}$$

Note that the zero sequence voltage is not considered here in the transformation.

The inverse are given by

$$\begin{bmatrix} u_{in,a} \\ u_{in,b} \\ u_{in,c} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & 0 & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} & 0 \\ 0 & -\frac{1}{3} & \frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} u_{out,a} \\ u_{out,b} \\ u_{out,c} \end{bmatrix} \quad \begin{bmatrix} i_{in,a} \\ i_{in,b} \\ i_{in,c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} i_{out,a} \\ i_{out,b} \\ i_{out,c} \end{bmatrix}$$

Block

**Figure 418: STAR_DELTA_CONVERSION block for fixed point****Figure 419: STAR_DELTA_CONVERSION block for floating point**

Block Dialog

The FPGA block contains the following both dialogs.

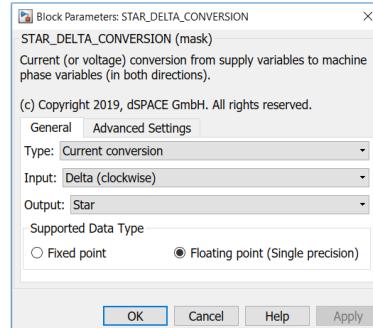


Figure 420: STAR_DELTA_CONVERSION dialog - General

The general dialog has the following parameters:

Name	Unit	Description
Type	[-]	Type of conversion: <ul style="list-style-type: none"> • Current conversion • Voltage conversion
Input	[-]	Input signals need to be transformed: <ul style="list-style-type: none"> • Star • Delta (clockwise) • Delta (counterclockwise)
Output	[-]	Into which quantities the output signals are transformed: <ul style="list-style-type: none"> • Star • Delta (clockwise) • Delta (counterclockwise)
Supported Data Type	[-]	Type of input signals: <ul style="list-style-type: none"> • Fixed point • Floating point (single precision)

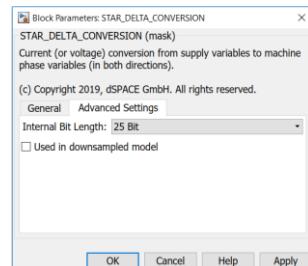


Figure 421: STAR_DELTA_CONVERSION dialog for fixed point – Advanced Settings

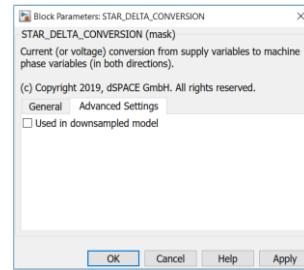


Figure 422: STAR_DELTA_CONVERSION dialog for floating point – Advanced Settings

The advanced settings dialog has the following parameters:

Name	Unit	Description	Range
Internal bit length for fixed point input data type	[-]	Specifies the internal bit length of the transformation	18 – 32 Bit
Used in downsampled model	[-]	Checkbox to enable or disable internal delays	On Off

PARALLEL_2_SERIAL

Description / Overview This block takes several input signals of equal data type (bus or vector) and puts out a single element, specified by the index input. In conjunction with the SERIAL_2_PARALLEL block, it can be used for resource sharing by time multiplexing.

Block

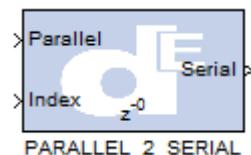


Figure 423: PARALLEL_2_SERIAL block

Block Dialog

The block provides the following dialog:

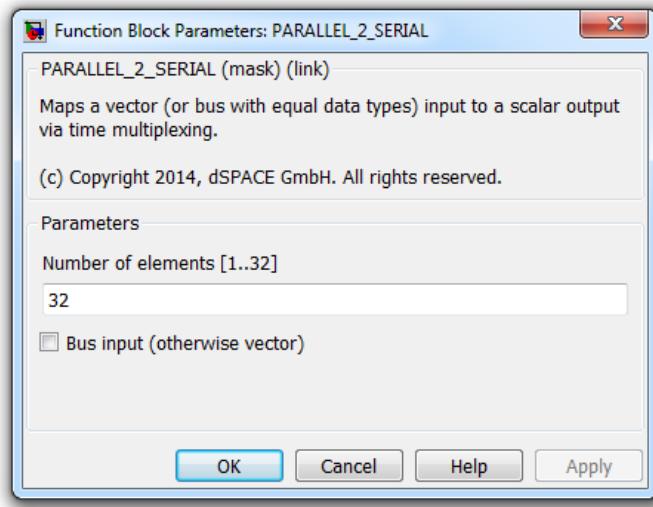


Figure 424: PARALLEL_2_SERIAL dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Number of elements	-	The size of the input vector or bus	1...32
Bus input	-	Specifies if the input signal is a bus or vector	on / off

Input

The PARALLEL_2_SERIAL block has the following inputs:

Name	Unit	Description	Format
Parallel	-	Bus or vector with input signal of equal data type.	-
Index	-	The index of the element to be provided to the serial output (0...31).	UFix_5_0

Output

The PARALLEL_2_SERIAL block has the following outputs:

Name	Unit	Description	Format
Serial	-	The selected element of the input vector/bus.	Bool

SERIAL_2_PARALLEL

Description / Overview

This block takes a serial input signal and its index and converts it to a parallel output (vector or bus). In conjunction with the PARALLEL_2_SERIAL block, it can be used for resource sharing by time multiplexing.

Block

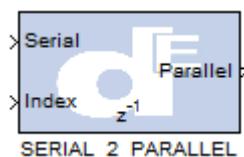


Figure 425: SERIAL_2_PARALLEL block

Block Dialog

The block provides the following dialog:

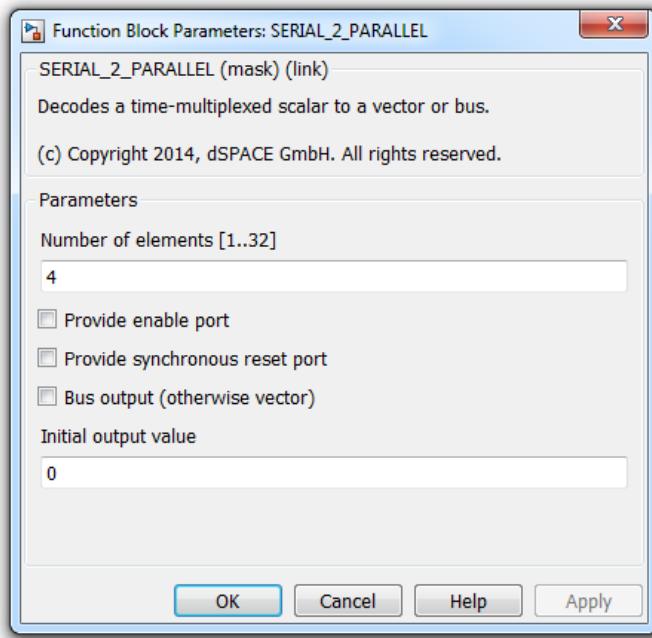


Figure 426: SERIAL_2_PARALLEL dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Number of elements	-	The size of the output vector or bus	1...32
Provide enable port	-	If this option is set, the block will provide an additional enable port. If enable is 0, the output will not be updated in this case.	on / off
Provide synchronous reset port	-	If this option is set, the block will provide an additional reset port. If reset is 1, all outputs will be reset to the initial value.	on / off
Bus output	-	Specifies if the output signal shall be a bus or vector.	on / off
Initial output value	-	The initial output of the bus or vector elements before being updated with serial data.	-inf...inf

Input

The SERIAL_2_PARALLEL block has the following inputs:

Name	Unit	Description	Format
Serial	-	The serial (time-multiplexed) input data.	-
Index	-	The index of the element to be provided to the serial output (0...31).	UFix_5_0
Enable		In case enable port is activated, samples are only latched when this input is high.	Bool
Reset		In case reset port is activated, this input resets certain time division multiplexed samples.	Bool

Output

The SERIAL_2_PARALLEL block has the following outputs:

Name	Unit	Description	Format
Parallel	-	The generated parallel signal vector or bus.	-

SERIAL_SAMPLE_HOLD

Description / Overview

This block holds the state of time division multiplexed inputs until the enable input is set. It is also possible to reset serial input signals.

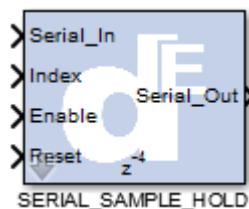
Block

Figure 427: SERIAL_SAMPLE_HOLD block

Block Dialog

The block provides the following dialog:

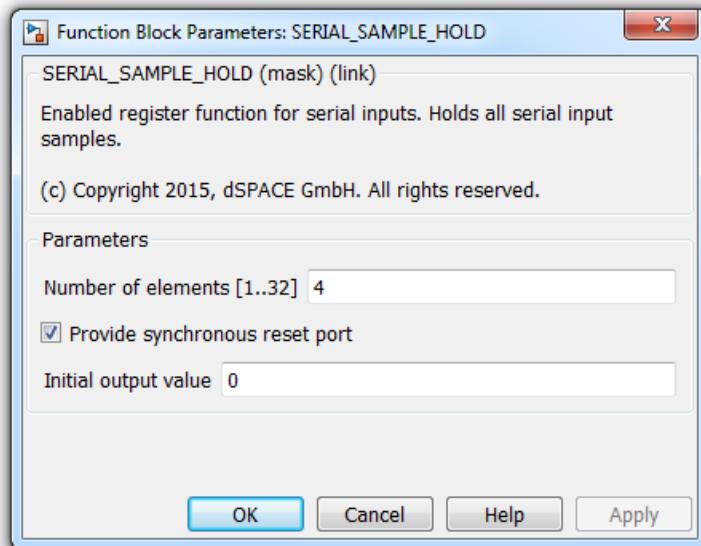


Figure 428: SERIAL_SAMPLE_HOLD dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Number of elements	-	The number of time division multiplexed input signals.	1...32
Provide synchronous reset port	-	If this option is set, the block will provide an additional reset port. If reset is 1, the current serial input signal will be reset to the initial value.	on / off
Initial output value	-	The initial output of the bus or vector elements before being updated with serial data.	-inf...inf

Input

The SERIAL_SAMPLE_HOLD block has the following inputs:

Name	Unit	Description	Format
Serial_In	-	The serial (time division multiplexed) input data.	-
Index	-	The index of the element to be provided to the serial output (0...31).	UFix_5_0
Enable		Serial samples are latched when this input is high.	Bool
Reset		In case reset port is activated, this input resets certain time division multiplexed samples.	Bool

Output

The SERIAL_SAMPLE_HOLD block has the following outputs:

Name	Unit	Description	Format
Serial_Out	-	The latched time division multiplexed signals.	-

NATURAL_NUMBER_DIVIDER

Description / Overview

This block performs a natural number division with integer quotient (Q) and division rest (R) output. The specialty about this divider is the fact that it is extremely cheap concerning resources (area) occupied. While standard dividers from the Xilinx blockset require several multipliers with an area complexity $O(n^2)$, as well as additional adders and other combinatorial logic ($O(n)$), this divider only required one single adder and some other cheap combinatorial logic, so the area complexity is $O(n)$. In this case, n is the bit width of the input signals. number inputs (1, 2, 3 ...) and only provides natural number outputs (quotient and division rest). However if the input and output signals are shifted accordingly (which costs nothing in hardware), also fixed-point signals can be divided using this block. Division of negative numbers is possible as well, by feeding the absolute values of input signals to the inputs N and D, and switching the sign of the outputs Q and R if exactly one of the input signals is <0. Delay and update rate of the block are dependent on the range of the quotient output Q (number of clocks = number of Q bits + 1).

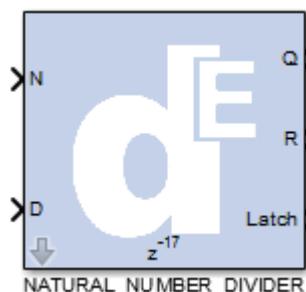
Block

Figure 429: NATURAL_NUMBER_DIVIDER block

Block Dialog

The block provides the following dialog:

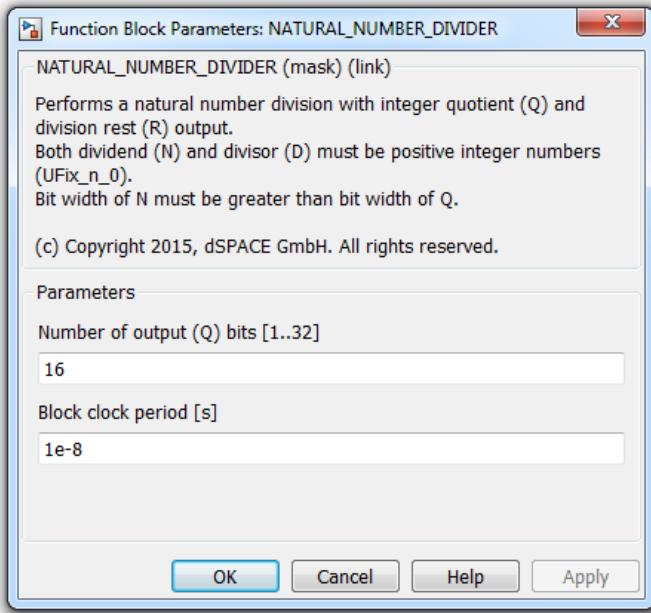


Figure 430: NATURAL_NUMBER_DIVIDER dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Number of output bits	-	Specifies the range of the quotient output Q. The delay and update rate are calculated as: number of clocks = number of Q bits + 1.	1...32
Block clock period	s	The clock period this block is sampled with. Usually, this period is equal to the overall FPGA clock rate, however if downsampling is applied, this parameter has to be adapted.	0...inf

Input

The NATURAL_NUMBER_DIVIDER block has the following inputs:

Name	Unit	Description	Format
N	-	The dividend. Must be a positive integer number with a bit width greater than the specified bit width of Q.	UFix_x_0
D	-	The divisor. Must be a positive integer number. The bit width can differ from the bit width of N.	UFix_x_0

Output

The NATURAL_NUMBER_DIVIDER block has the following outputs:

Name	Unit	Description	Format
Q	-	The integer quotient output.	UFix_x_0
R	-	The division rest.	UFix_x_0
Latch	-	Impulse which is raised for one clock cycle, indicating two events: 1. New input data was latched in the last clock cycle. 2. New output data will be provided in the next clock cycle.	Bool

XSG_2_SL_IF

Description / Overview

This block is used to connect any FPGA signal to a Simulink scope for offline simulation by converting it to a Simulink signal (double). The input signal can be a scalar, vector or bus. The scope adapter also offers the possibility to define a certain range of bits for each input signal to display.

Block

Figure 431: XSG_2_SL_IF block

Block Dialog

The block provides the following dialog:

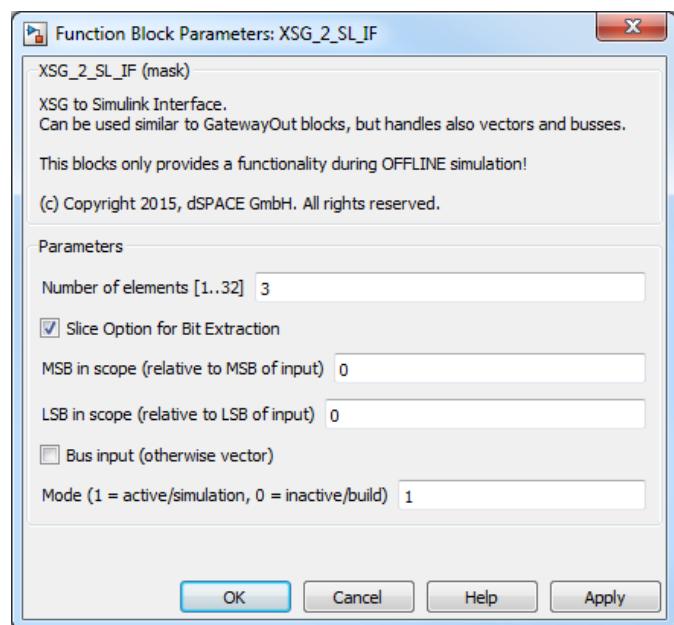


Figure 432: XSG_2_SL_IF dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Number of elements	-	The number of input elements of a vector or bus.	1...32
Slice Option for Bit Extraction	-	Activate an internal slice Operation block and the MSB and LSB can be set in the GUI	
MSB in scope	-	The MSB of the data to be displayed in the scope, relative to the MSB of the input signal. Set to 0 to display all bits. If a value other than 0 is set here, the raw data bits (UFix_n_0) of the selection are displayed in the scope!	-inf...0
LSB in scope	-	The LSB of the data to be displayed in the scope, relative to the LSB of the input signal. Set to 0 to display all bits. If a value other than 0 is set here, the raw data bits (UFix_n_0) of the selection are displayed in the scope!	0...inf
Bus input	-	Set this option if the input is a bus.	on / off
Mode	-	Flag for activating or deactivating the scope adapter. For FPGA build, all scope adapters have to be deactivated, as RTI FPGA will create error messages otherwise.	0 1

Input

The XSG_2_SL_IF block has the following inputs:

Name	Unit	Description	Format
In	-	The FPGA input signal.	-

Output

The XSG_2_SL_IF block has the following outputs:

Name	Unit	Description	Format
Out	-	The Simulink output signal.	-

VECTOR_2_BITS

Description / Overview This block is used to convert a Simulink vector of n Boolean (bool or UFix_1_0) elements to a single data word with n bits (UFix_n_0). The LSB corresponds to the lowest in the Simulink vector.

Block

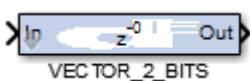


Figure 433: VECTOR_2_BITS block

Block Dialog

The block provides the following dialog:

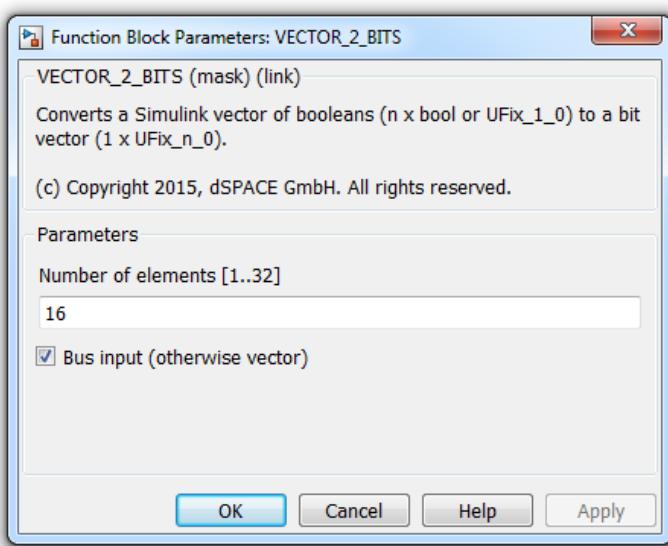


Figure 434: VECTOR_2_BITS dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Number of elements	-	The number of input vector elements.	1...32
Bus input	-	Specifies if the input signal is a bus or vector.	on / off

Input

The BITS_2_VECTOR block has the following inputs:

Name	Unit	Description	Format
In	-	The input vector (n x bool or UFix_1_0).	-

Output

The BITS_2_VECTOR block has the following outputs:

Name	Unit	Description	Format
Out	-	The output word (1 x UFix_n_0).	-

BITS_2_VECTOR

Description / Overview This block is used to convert a single data word with n bits (UFix_n_0) elements to a Simulink vector of n Boolean (Bool or UFix_1_0). The LSB corresponds to the lowest in the Simulink vector.

Block

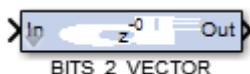


Figure 435: BITS_2_VECTOR block

Block Dialog

The block provides the following dialog:

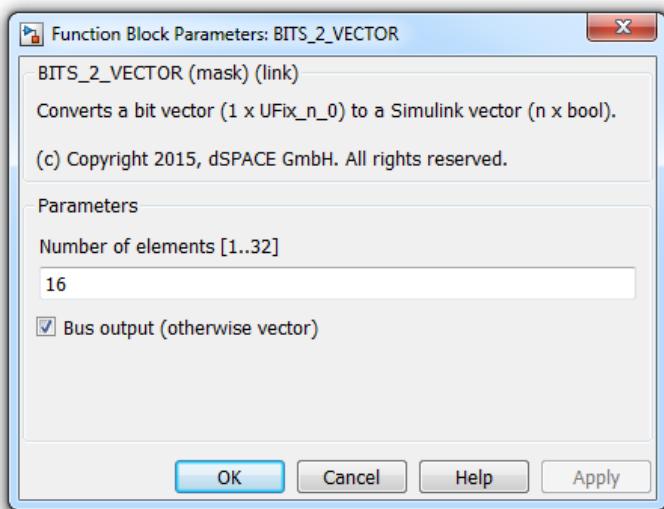


Figure 436: BITS_2_VECTOR dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Number of elements	-	The number of input vector elements.	1...32
Bus output	-	Specifies if the output signal shall be a bus or vector.	on / off

Input

The BITS_2_VECTOR block has the following inputs:

Name	Unit	Description	Format
In	-	The input word ($1 \times \text{UFix_n_0}$).	-

Output

The BITS_2_VECTOR block has the following outputs:

Name	Unit	Description	Format
Out	-	The output vector ($n \times \text{Bool}$).	-

RECIPROC

Description / Overview

This block provides the reciprocal of the input. This block supports two data types; fixed point with input (UFix_26_26) and floating point with single precision.

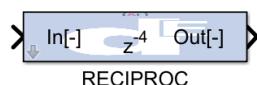
Block

Figure 437: RECIPROC block for fixed point



Figure 438: RECIPROC block for floating point

Block Dialog

The block provides the following dialog:

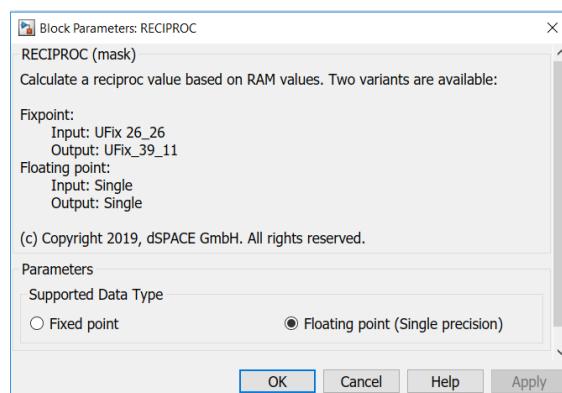


Figure 439: RECIPROC dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Supported Data Type	-	Type of input and output data type	For fixed : UFix_26_26 For Floating-point: XFloat_8_24

Input

The RECIPROC block has the following input:

Name	Unit	Description	Format
In	-	Input (UFix_26_26 or XFloat_8_24)	-

Output

The RECIPROC block has the following output:

Name	Unit	Description	Format
Out	-	Output (UFix_39_11 or XFloat_8_24)	-

BUS_FCN

Description / Overview

Sometimes it is useful or necessary to delay all signals of a simulink bus in a customized model on the FPGA. In this case the BUS_FCN block can be used. Only scalar and muxed signals are supported.

Block

Figure 440: BUS_FCN block (direct feedthrough activated)

Block Dialog

The block provides the following dialog:

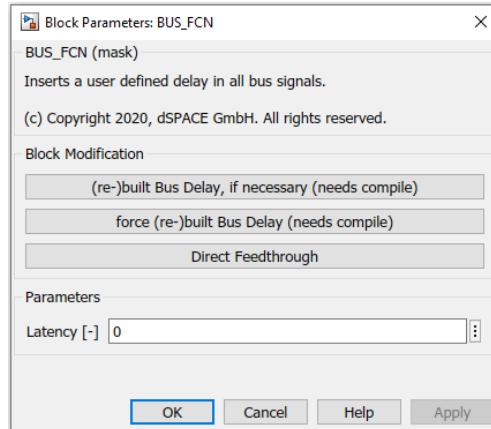


Figure 441: BUS_FCN dialog

The block dialog has the following parameters:

Name	Unit	Description	Range
Button: (re-) build Bus Delay, if necessary	-	Build the internal bus structure with delay blocks in each signal; (re-) build of the internal structure will proceed if necessary; if a (re-) build is necessary the model must be able to compile (successful CTRL-D)	-
Button: Force (re-) build Bus Delay	-	Forces a (re-) build of the internal structure; model must be able to compile (successful CTRL-D)	-
Direct Feedthrough	-	Internal bus delay block will be deleted and the in- and output are directly connected	
Latency	-	Latency of the overall delay	[0 .. x]

Input

The BUS_FCN block has the following input:

Name	Unit	Description	Format
In	-	Simulink input bus	-

Output

The BUS_FCN block has the following output:

Name	Unit	Description	Format
Out	-	Delayed output bus	-

Processor based parts

Overview

Overview

The XSG Utils library also contains several processor based utils. The following processor parts are available:

- Small Apps

The following section describes the different functionalities in detail.

Small APPS

Content

See the following sections for information about the blocks of the subsystem SMALL APPS. These blocks located provide an easy solution for the most used functions, such as slice, signal conversion, shift,...

Coding Fixpoint PROC2FPGA: Signal_2_Raw

Description / Overview

The Signal_2_Raw block converts a fixpoint value to a signal without binary point and in the unsigned format with respect to the two's compliment. The output unit is "Raw".

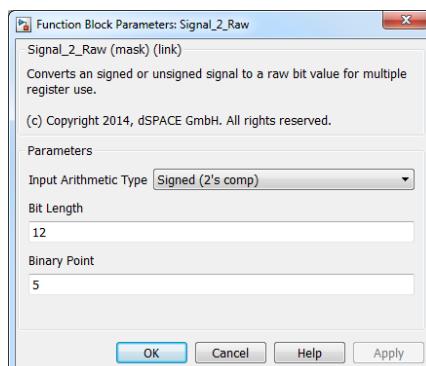
Block



[Figure 442: Signal_2_Raw block](#)

Block Dialog

The block contains the following dialog.



[Figure 443: Signal_2_Raw dialog](#)

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Input Arithmetic Type	[-]	Lets you select format of the input signal	- Signed (2'scomp) - Unsigned
Bit Length	[-]	Definition of the input bit length	1 – 32
Binary Point	[-]	Definition of the binary point	0 - 31

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	User-defined

Output

The Signal_2_Raw block has the following outputs:

Name	Unit	Description	Format
Out	[Raw]	Output	UFix

Workflow / Example

The following example shows the transfer behavior of the block at the following settings:

- Arithmetic Type: Signed (2'scomp)
- Bit Length: 12
- Binary Point: 5
 - ➔ Minimum value: -64
 - ➔ Maximum value: 63.96875

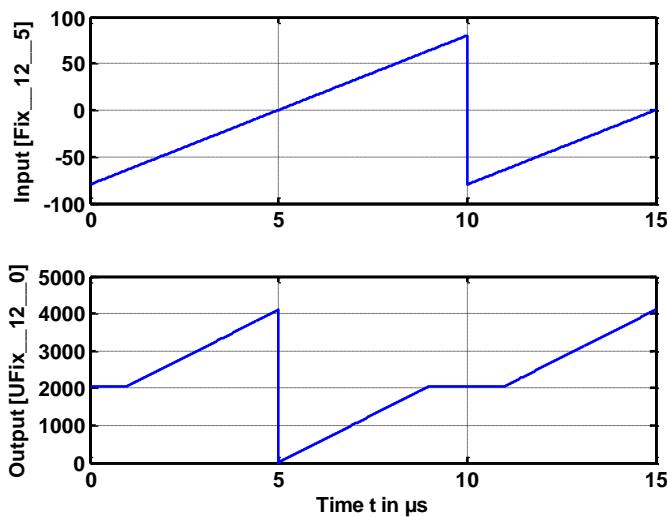


Figure 444: Workflow of the block

Coding Fixpoint PROC2FPGA: Shift

Description / Overview The Shift block shifts the input signal with the specific value. This can be used to shift bits in the right position of a register.

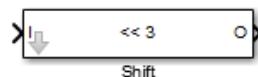
Block

Figure 445: Shift block

Block Dialog

The block contains the following dialog.

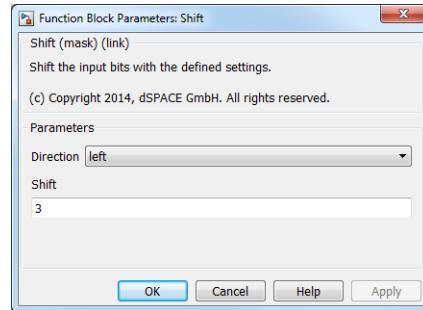


Figure 446: Shift dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Direction	[-]	Lets you select direction for shifting	- left - right
Shift	[-]	Definition of the number of bits for shifting	1 – 32

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	User-defined

Output

The Shift block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	User-defined

Coding Fixpoint FPGA2PROC: Slice

Description / Overview The Slice block slices specific bits of the input value. The output of the value is offset free.

Block

Figure 447: Slice block

Block Dialog

The block contains the following dialog.

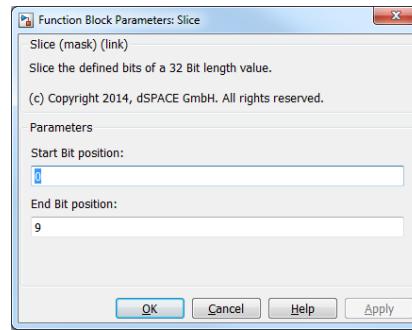


Figure 448: Slice dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Start Bit Position	[-]	Lets you specify the start bit	0 – 32
End Bit Position	[-]	Lets you specify the end bit	0 – 32

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[-]	Input	UFix_32_0

Output

The Slice block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	User-defined

Coding Fixpoint FPGA2PROC: Raw_2_Signal

Description / Overview

The Raw_2_Signal block converts a signal with the unit Raw to a user defined format. It represents the corresponding block to the "Singal_2_Raw" block.

Block

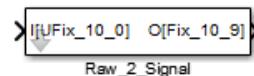


Figure 449: Raw_2_Signal block

Block Dialog

The block contains the following dialog.

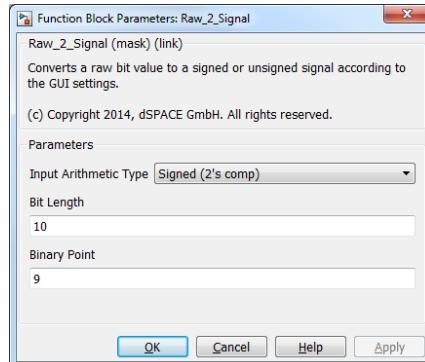


Figure 450: Raw_2_Signal dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Input Arithmetic Type	[-]	Format of the input signal	- Signed (2'scomp) - Unsigned
Bit Length	[-]	Definition of the input bit length	1 – 32
Binary Point	[-]	Definition of the binary point	0 - 31

Input

The main block has the following inputs:

Name	Unit	Description	Format
In	[Raw]	Input	UFix

Output

The Raw_2_Signal block has the following outputs:

Name	Unit	Description	Format
Out	[-]	Output	Fix

Coding Floating - point PROC2FPGA: Float_2_IEEE754Codec

Description / Overview

The Float_2_IEEE754Codec block converts a floating - point value from Simulink (e.g. Double) to an IEEE754 coded signal without binary point and in the unsigned format. The output unit is "Raw". On the FPGA only hardware neutral (bounds no additional FPGA resources) reinterpret is necessary.

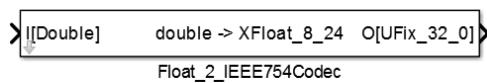
Block

Figure 451: Float_2_IEEE754Codec block

Block Dialog

The block contains the following dialog.

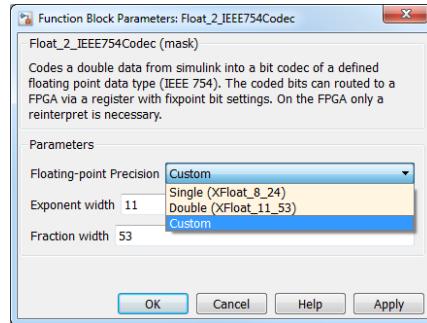


Figure 452: Float_2_IEEE754Codec dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Floating-point Precision		Lets you select the output codec format of the input signal; if the custom Precision is set, the following GUI parameters are available	- Single - Double - Custom
Exponent width	[-]	Definition of the exponent width	1 – 11
Fraction width	[-]	Definition of the fraction width	1 - 53

Input

The main block has the following inputs:

Name	Unit	Description	Format
I	[-]	Input	User-defined

Output

The Float_2_IEEE754Codec block has the following outputs:

Name	Unit	Description	Format
O	[Raw]	Output	UFix

Coding Floating - point FPGA2PROC: IEEE754Codec_2_Float

Description / Overview

The IEEE754Codec_2_Float block converts a IEEE754 coded signal without binary point and in the unsigned format to a floating - point value from Simulink (e.g. Double).

Block

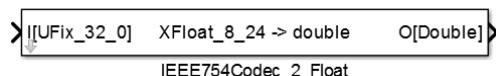


Figure 453: IEEE754Codec_2_Float block

Block Dialog

The block contains the following dialog.

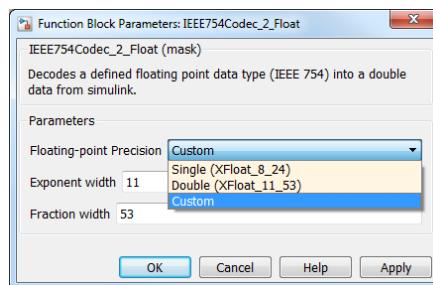


Figure 454: IEEE754Codec_2_Float dialog

The dialog blockset has the following parameters:

Name	Unit	Description	Range
Floating-point Precision		Lets you select the input codec format; if the custom Precision is set, the following GUI parameters are available	- Single - Double - Custom
Exponent width	[-]	Definition of the exponent width	1 – 11
Fraction width	[-]	Definition of the fraction width	1 - 53

Input

The main block has the following inputs:

Name	Unit	Description	Format
I	[Raw]	Input	UFix

Output

The IEEE754Codec_2_Float block has the following outputs:

Name	Unit	Description	Format
O	[-]	Output	User-defined

Demo

Overview

Overview

The XSG Utils library provides one demo model.

It consists of a Matlab Simulink model and a ControlDesk project backup.



The XSG Utils Demos require the XSG Utils Interface library and the XSG Utils library.

The demos are precompiled for the following hardware configurations:

Processor Model	FPGA Model
DS1006	
DS1007	
DS2502	DS2655 (7K160) + DS2655M1 Module

In case that a different hardware configuration should be used, e.g. DS5203 (7K325) with mounted multi IO module (DS5203M1), it is necessary to rebuild the FPGA model.

Demo Installation

Initial Installation

The Demo Model setup is separated from the XSG Utils installation. To install the demo it is necessary to open the XSG Utils library and double click the demo block. The following dialog will appear:

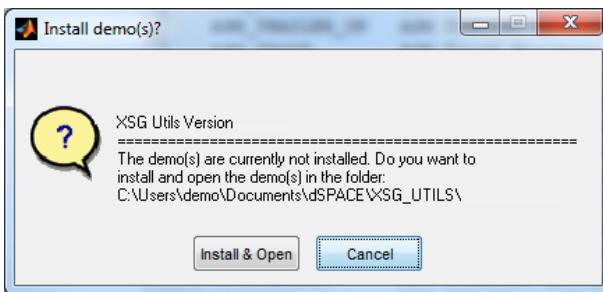


Figure 455: Demo installation dialog (first installation)

The Demo models and experiments will be installed to: My Documents\dSPACE\xsg_utils\20.1. Hence, the demo folder installation path, shown in the demo installation dialog will adjust, according to the user's login name.

Demo Folder Structure

The demos are copied, using the following folder structure:

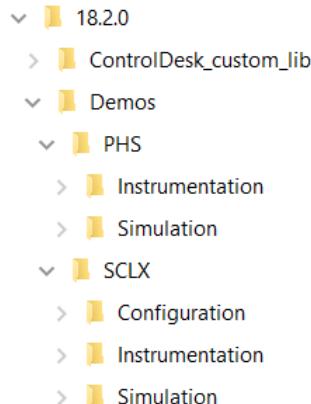


Figure 456: Demo folder structure after demo model installation

The Simulation folder contains the Matlab Simulink model and for PHS, it contains build results for DS1006 and 1007.

The Instrumentation folder contains the ControlDesk Project Backup.

If Scalexio platform is used an additional folder Configuration contains the Configuration Desk Project Backup.

Reinstallation

After the initial demo installation is done, the install demo dialog will change. The dialog now provides the options to open the demo models, or to reinstall the demo, as shown in the figure below.

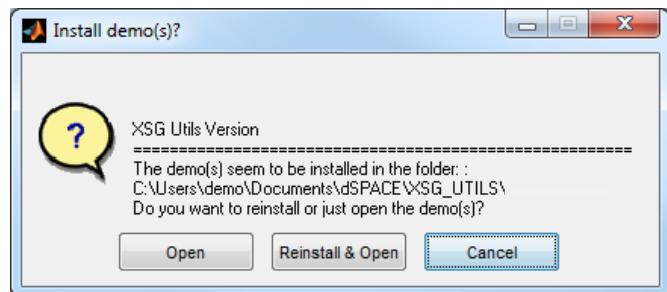


Figure 457: Demo installation dialog (after initial installation)

	Any changes on the demo model and experiment will be lost, when reinstalling the demo!
--	--

Utils Demo

Overview

After the demos are installed the actual used platform (PHS-Bus based system or IOCNET based system (SCALEXIO)) is used and the specific model will be loaded. If both platforms are available a selection box will ask which model will be used.

The Utils demo shows an exemplary integration of an angular processing unit, a wave table encoder and a three phase sine wave generator simulation utilizing the XSG Utils library.

For this demo implementation, three main components from the XSG Utils library are used. These are the APU, the WAVE_TABLE_GENERATOR and the SINE_GENERATOR main components.

Applying a certain speed information to the APU unit, simulates a rotating shaft, on which an analog rotary encoder (Wave table encoder), with a user defined wave form, is mounted.

The Sine Generator is used to generate three symmetric sine waves which for example can be used for simulation of the current from a phase electromotor.

The output of the encoder and the sine generator are routed via a dynamic scaling and stimulus block (SCALE_DAC) to the DAC channels. The values before and after the scaling are monitored within the FPGA clock rate on a scope (MULTI_SCOPE), as well as the six ADC channels.

Processor Model

To get a better overview of the general model structure the following schematic illustration and the Simulink model root is shown in the next figures.

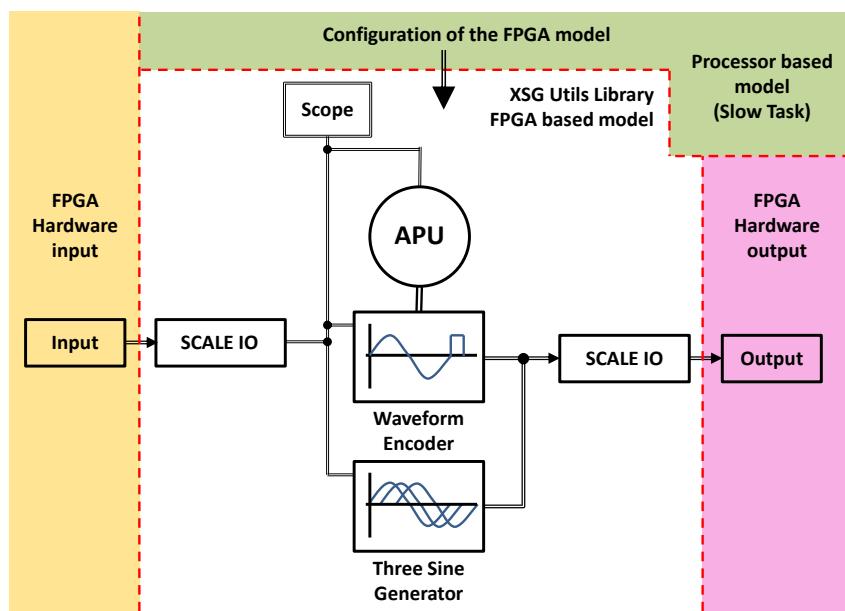


Figure 458: Schematic overview of the model structure

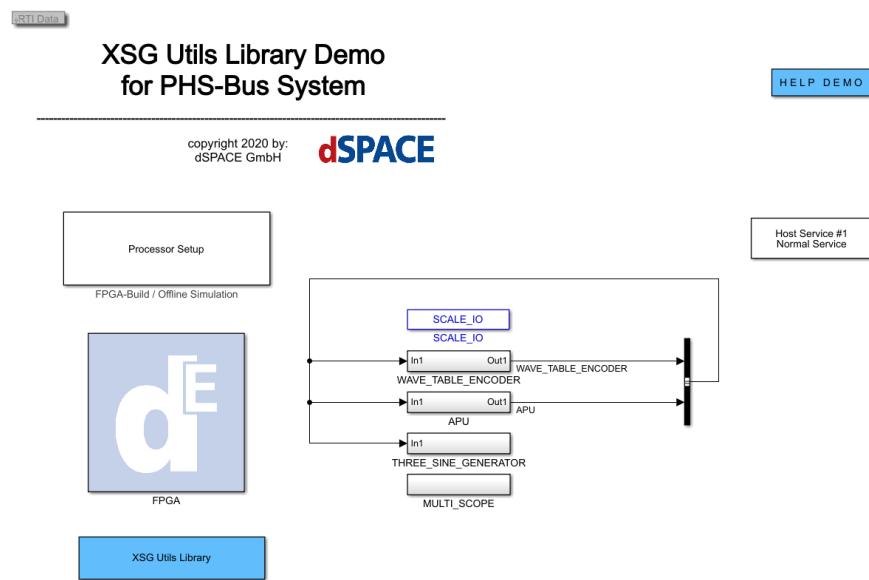


Figure 459: Utils demo Simulink model root

The processor setup block is located in the top left corner of the model. Here you can find the Link to the actual FPGA configuration which will be downloaded to the target FPGA

The FPGA model is directly placed below the processor setup block.

Furthermore, the processor interfaces for the FPGA model are located on the model root and are divided into subsystem for the corresponding component. It includes the APU_Out, APU_In, WAVE_TABLE_ENCODER_out, WAVE_TABLE_ENCODER_in, THREE_SINE_GENERATOR_out, MULTI_SCOPE_out, MULTI_SCOPE_in and interfaces for scaling.

In addition, the Library_Version_Info block is used to ensure that processor and FPGA model are built with the same XSG Utils library version.

FPGA Model

The FPGA model is located on the Utils demo model root, below the processor setup block. The FPGA model is shown in the following figure:

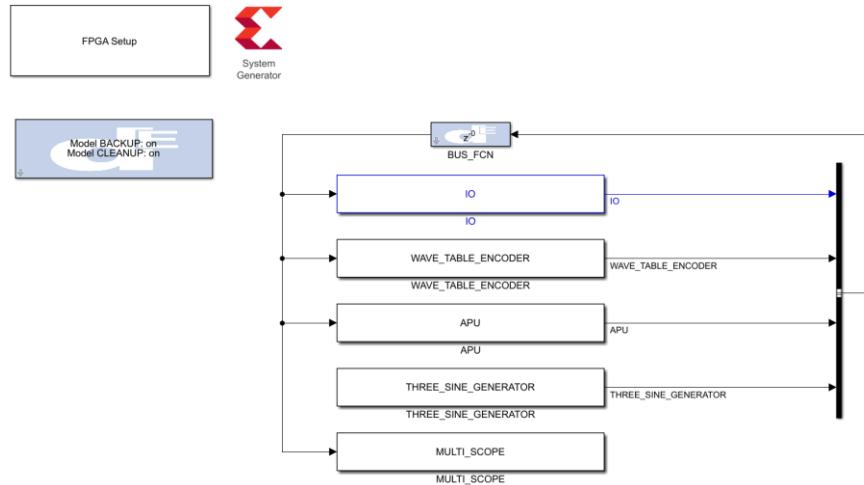


Figure 460: Utils demo FPGA model

The FPGA setup block is located in the top left corner of the FPGA model. Here the specification of desired hardware platform (framework) has to be made. All hardware in- and outputs are located in the subsystem IO. Each subsystem represents a fully implemented function like the APU, a Wave-table-encoder, ... All necessary model inputs from the PHS-Bus are located in these subsystems, too. Model variables (e.g. the speed of the APU) which are necessary from one subsystem (e.g. APU) to another subsystem (e.g. Wave_Table_Encoder) and the hardware IO signals are routed over the overall Bus.

DS5203 IO Mapping

The following table summarizes the model signals, which are mapped to DS5203 IO channels:

DS5203 IO Channel	Model Signal / Function
ADC 1 to 6	No special function, can be freely supplied by the user
DAC 1	Wave table encoder output
DAC 2	Sine phase A output
DAC 3	Sine phase B output
DAC 4	Sine phase C output
DAC 5	Stimulus signal
DAC 6	Stimulus signal

DS2655 IO Mapping

The following table summarizes the model signals, which are mapped to DS2655 IO channels:

DS5203 IO Channel	Model Signal / Function
Analog In – Ch: 11 [Mod: 1] to Analog In – Ch: 15 [Mod: 1]	No special function, can be freely supplied by the user
Analog Out – Ch: 16 [Mod: 1]	Wave table encoder output
Analog Out – Ch: 17 [Mod: 1]	Sine phase A output
Analog Out – Ch: 18 [Mod: 1]	Sine phase B output
Analog Out – Ch: 19 [Mod: 1]	Sine phase C output
Analog Out – Ch: 20 [Mod: 1]	Stimulus signal

ControlDesk Environment

To open the prepared ControlDesk backup, first start ControlDesk. Then, select File - Open – Project + Experiment from Backup.

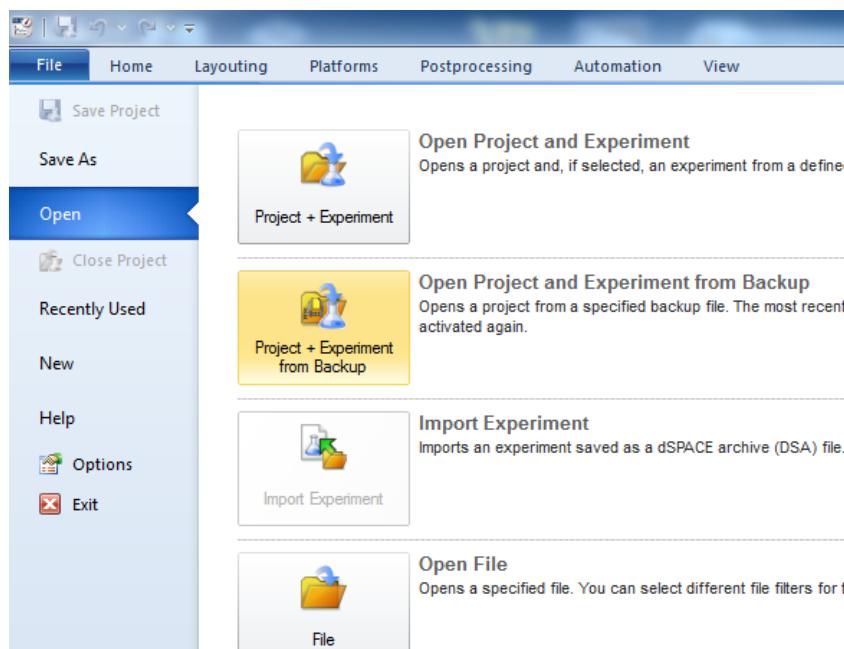


Figure 461: Open ControlDesk backup

Navigate to the demo installation directory at: My Documents\ dSPACE\ XSG_Utils\20.1\ Utils\ <system>(PHS/SCLX)\ Instrumentation\ and select the UTILS_<system>(PHS/SCLX)_Demo.ZIP file.

After the backup is loaded, the following layout is shown:

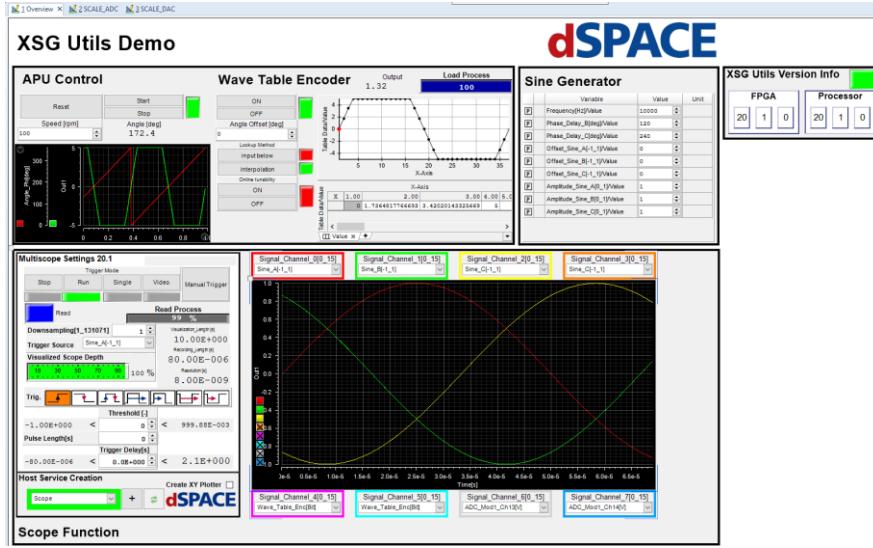


Figure 462: ControlDesk Utils Demo layout

Now, register the processor board and start the measurement. The real-time application will be downloaded.

After the real-time application is downloaded and the real-time process is executed, the XSG Utils Lib Version Info will show the recognized versions numbers for the FPGA and Processor model.

The Overview layout is organized into function groups. On the upper left hand side you can find the control of the angular processing unit (APU), and the actual shape of the wave table (online tunable). The upper right group provides the parameterization of the 3 phase sine generator; the generated outputs can be visualized on the Multiscope, below. The bottom of the layout shows the version information. To change the scaling of the analog in- and outputs the following layouts can be used.

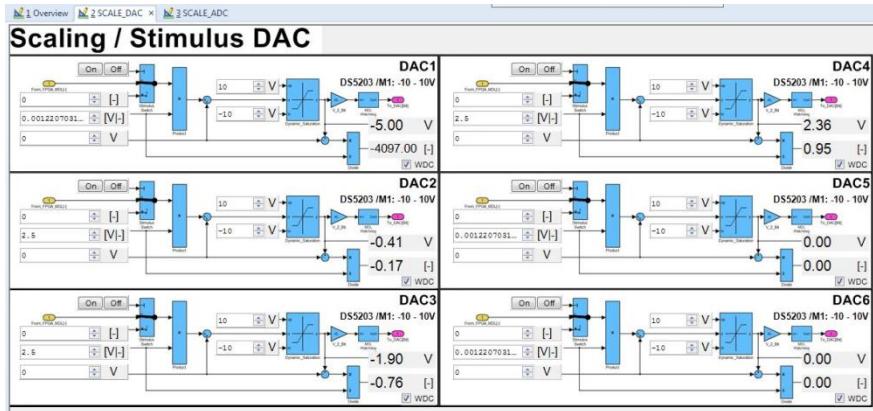


Figure 463: ControlDesk Utils Demo;

Layout: SCALE_DAC

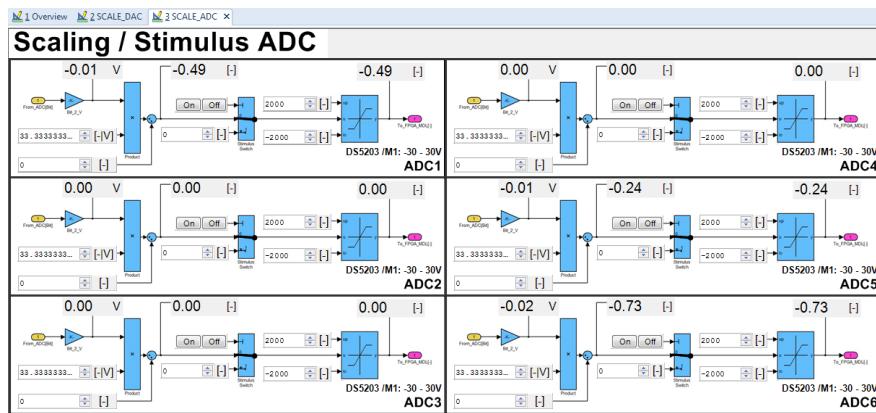


Figure 464: ControlDesk Utils Demo;

Layout: SCALE_ADC

ConfigurationDesk Environment

In case of using an IOCNET system (SCALEXIO) an additional ConfigurationDesk environment is also available. To open the prepared ConfigurationDesk backup, first start ConfigurationDesk. Then, select File - Open – Project + Experiment from Backup.

Navigate to the demo installation directory at: My Documents\ dSPACE\ XSG_Utils\20.1\ Utils\SCLX\ Configuration\ and select the UTILS_SCLX_Demo.ZIP file.

Useful functions for ControlDesk

Overview

Overview

To fasten up the layouting and usability of the most frequently used XSG Utils functions in ControlDesk, the solution also contains special designed **Custom Instruments** (CI) for the following model parts:

- SCALE_DAC
- SCALE_ADC
- MULTISCOPE
- SCOPE

In the following chapter the import of the custom instruments and the usage will be described. To simplify the afterwards variable connection of these Custom Instruments in a ControlDesk experiment a **CI Auto Signal Connector** function is also included.

Custom Instruments

General Description

ControlDesk distinguishes between instrument categories and libraries. Libraries contain all standard instruments. These instruments can be added to diverse categories. For example, you can use the Custom Instrument category to save frequently used instruments with their settings. Categories are collections of instruments which you can change according to your needs.

To get a quick start up the XSG Utils Solution support such Custom Instruments especially designed for the most frequently blocks like the SCALE_ADC, MULTISCOPE, ... as shown in the figure bellow.

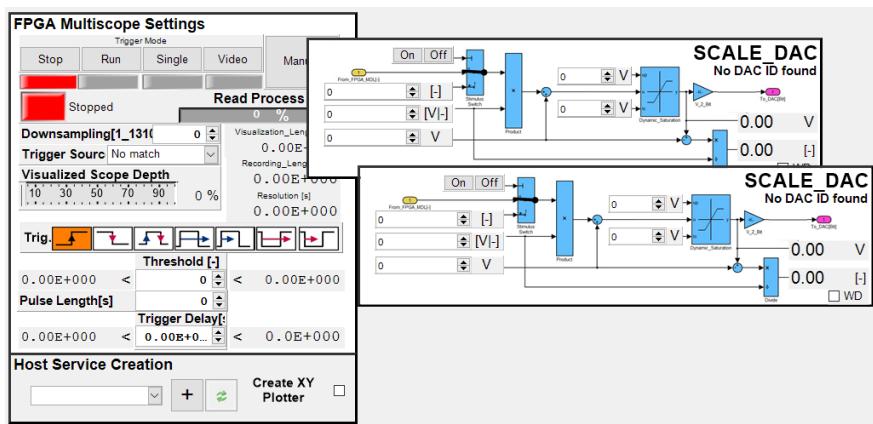


Figure 465: Custom Instruments of the XSG UTILS Solution

Import of the Custom Instruments in ControlDesk

To use the XSG Utils Custom Instruments in your own layouts the library must be imported in ControlDesk. Therefore, start ControlDesk and open one time the XSG Utils library in MATLAB. After that the folder structure will be installed as described in Figure 456.

Right click in the “Instrument Selector” in a free space in ControlDesk and select the command “Import Library” as shown in the figure bellow:

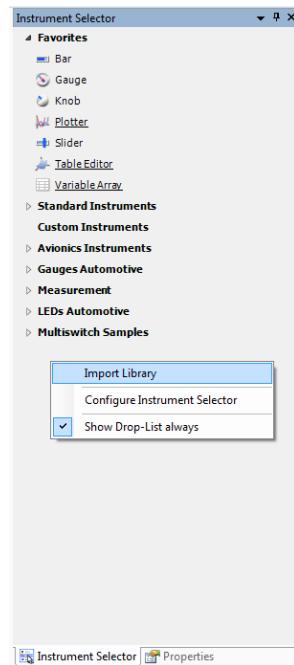


Figure 466: Import Custom Instruments Library in ControlDesk

Select the .ilx library file of the folder My Documents\ dSPACE\ XSG_UTILS\20.1\ ControlDesk _custom_lib\ XSG_Utils_Custom Instruments.ilx. Afterwards a new chapter of is integrated and the following custom instruments are available as shown in the figure bellow:

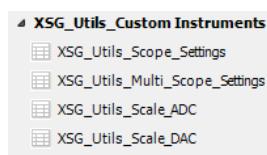
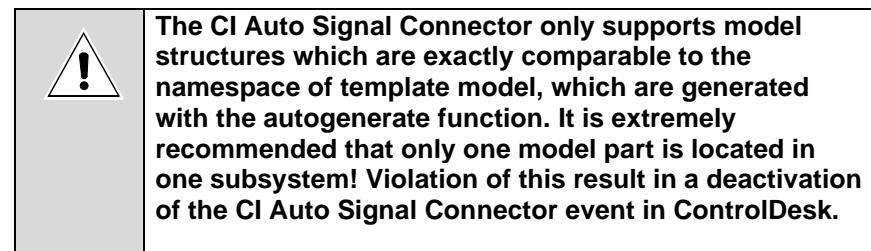


Figure 467: XSG Utils Custom Instruments

The custom instruments are now available for usage in the layouts and the variable mapping of each subcomponent can be done.

CI Auto Signal Connector

Description	To simplify and speed up the routing of the different variables to the subcomponent blocks of each custom instrument an internal function “CI Auto Signal Connector” is supported by the XSG Utils Solution. In the following chapter the installation and the workflow is described in detail.
--------------------	---



Usage of the CI Auto Signal Connector	To get a better overview of the usage of the CI Auto Signal Connector a template implementation of the SCALE_ADC model part (Figure 305 / Figure 306) is used. After the FPGA / Processor build and download to the hardware, the following model root is available in ControlDesk as described in the figure below. Additionally, the custom instrument for the SCALE_DAC functionality is also shown.
--	---

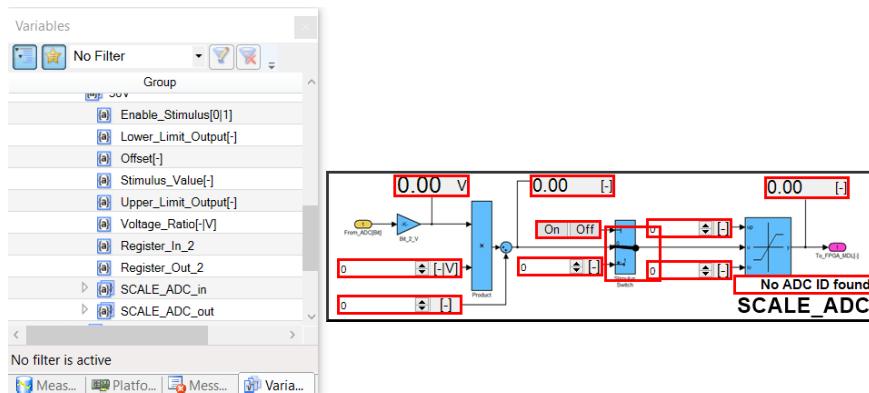


Figure 468: Example model root and the custom instrument for a SCALE_ADC.

To activate the auto-connection-function, drag and drop a variable of the subsystem SCALE_ADC_in or SCALE_ADC_out to the frame field of the custom instrument, which is marked in red in the figure below.

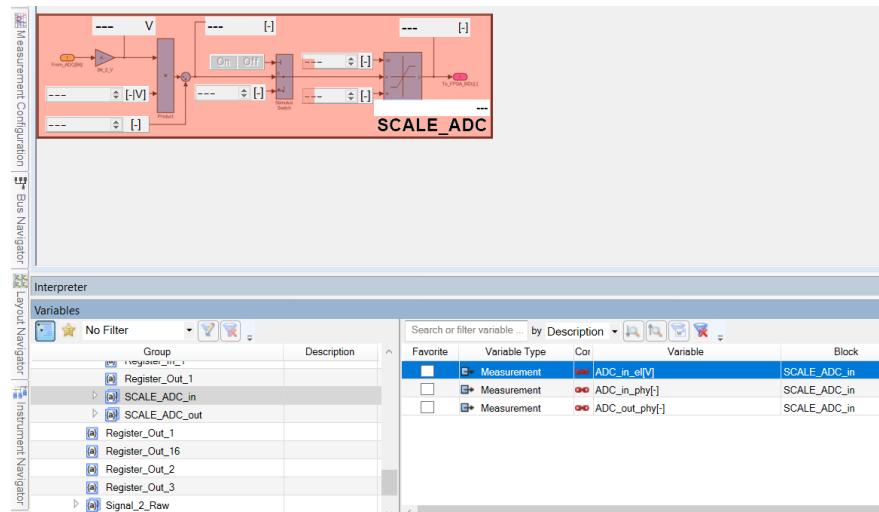


Figure 469: Drag and drop action of a variable to the custom instrument frame.

After the variable is dropped to the frame of the custom instrument, the auto-connection-function will be activated and all other variables will be connected to the standard defined model variables with a feedback in the Interpreter ribbon as shown in the figure bellow.



Figure 470: Feedback of the action of the auto-connection-function

Usage of Scope custom instrument

The steps involving the variable mapping and plotter generation for custom instruments XSG_Utils_Scope_Settings and XSG_Utils_Multi_Scope_Settings are identical and therefore, are shown only for XSG_Utils_Multi_Scope_Settings.

Drag and drop any Parameter from **MULTI_SCOPE_in** or **MULTI_SCOPE_out** processor interface onto the frame of the XSG_Utils_Multi_Scope_Settings frame as shown in Figure 471.

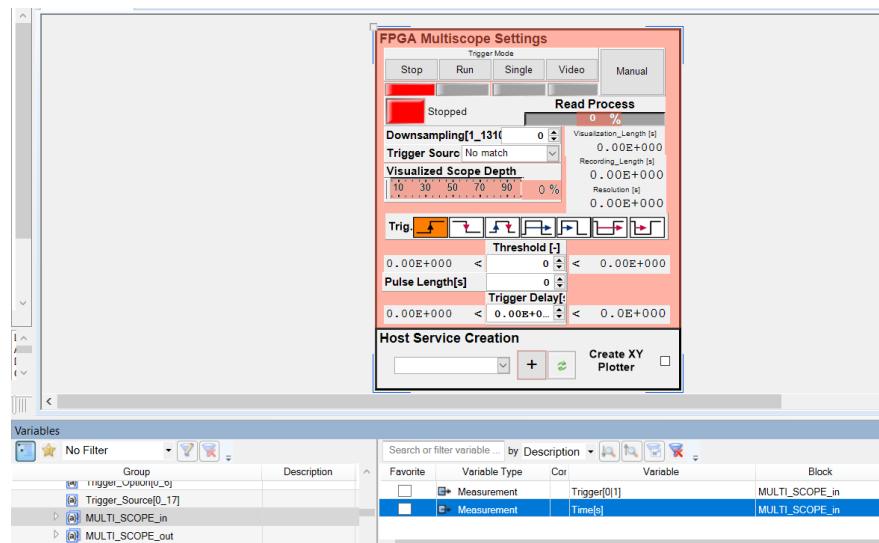


Figure 471: Variable Mapping using drag and drop

After the variable mapping is completed, a dialog box pops up, prompting to select the CSV file containing the trigger or channel list, for the selection box elements. It is automatically saved (when the block FPGA Build Action is integrated inside the FPGA model before FPGA Build. If the block is not used you can create them via the GUI of the FPGA (Multiscope block) in the XSG_Sol_Folder under ControlDest_config_data.

Afterwards automatically, a new measurement raster and a new XY-Plotter, inclusive all data connections and selection boxes, is created.

If the automatic addition of raster and generation of XY Plotter is not desired, select 'Cancel' as shown in Figure 472. In this case, only the variables will be mapped to the custom instrument. Each step contains a 'Cancel' button, and the automatic generation can be terminated at any of the following steps, by selecting the 'Cancel' option or the X button at the top right corner of the dialog box.

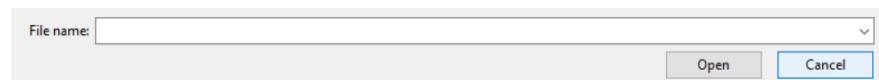


Figure 472: Automatic Plotter generation terminates when Cancel is selected

Once the trigger/channel list is added to the instrument, the next step is adding a raster. If the model contains more than one task (synchronous or asynchronous), the dialog box pops up as shown in Figure 473, prompting to select a base raster (duration or sample count).

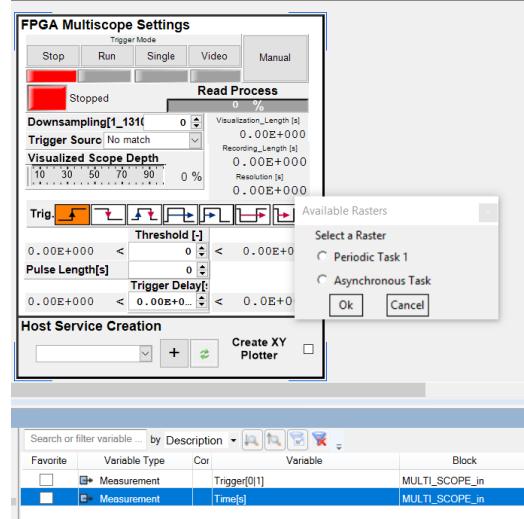


Figure 473: Following Dialog pops up when model contains more than one task

When a base raster is selected, the desired raster name must be entered in the dialog box. If the entered raster name already exists, the dialog prompts for a different name.

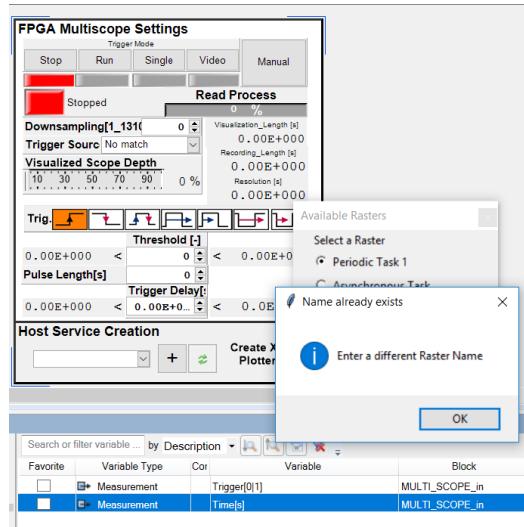


Figure 474: Following Dialog pops up when raster name already exists

If the base raster field is left empty, the default raster is considered as base and is used to create the new raster. After the new raster is added, the XY Plotter is generated with signals connected to X and Y axis and corresponding selection boxes with a channel list (contains the signal label connected to the FPGA main block) which is exemplarily shown in Figure 475.

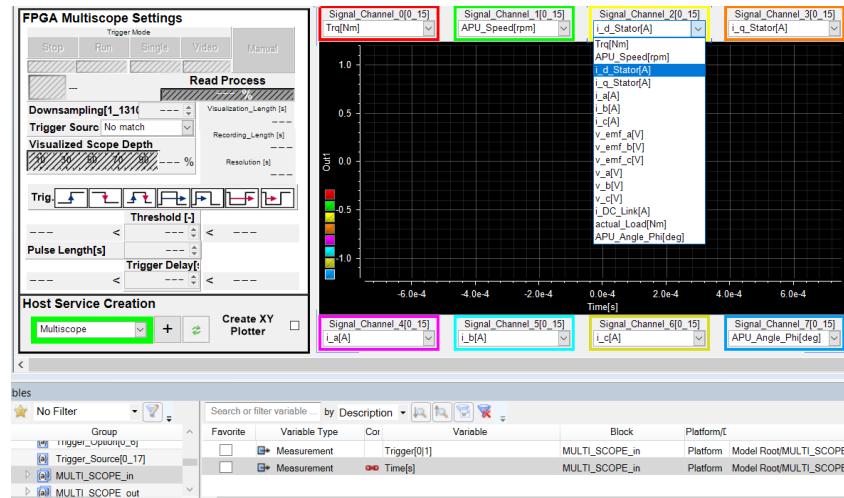


Figure 475: Custom Instrument with generated XY Plotter

Once the online calibration is started the visualized scope depth – slider instrument (marked in red) can be used to change the recording- and with this also the visualization- duration.

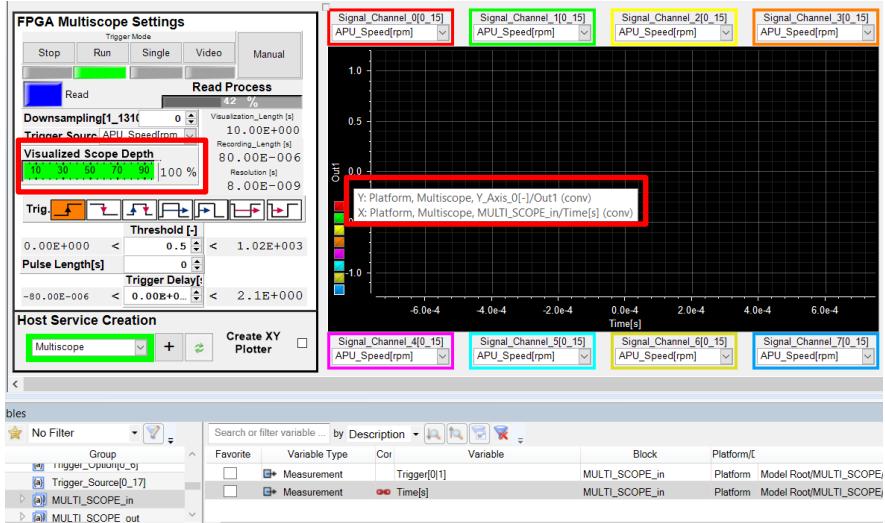


Figure 476: Variation of duration trigger of the corresponding raster

Here, when the slider is moved, the duration is correspondingly changed in the measurement raster, the signals are connected to (here: Multiscope).

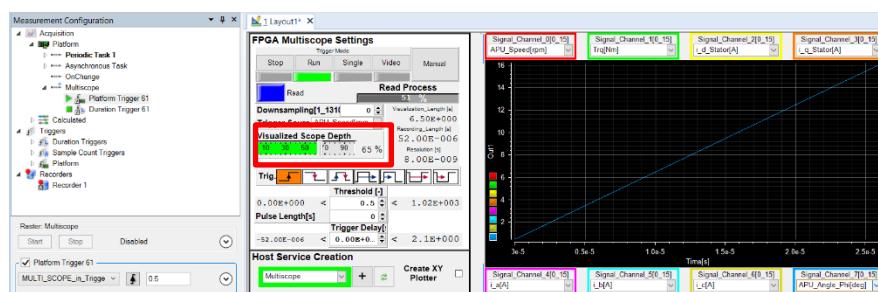
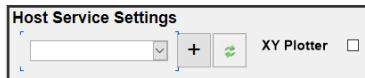


Figure 477: Variation of visualization length in corresponding raster**Host Service Settings**

The Frame ‘Host Service Settings’ contains additional icons as shown in Figure 478, for adding additional raster and XY plotters. **All these icons can be used only after the variables are mapped to the instrument.**

**Figure 478: Host service settings frame with additional icons**

The ‘**Select Raster**’ selection box displays the list of measurement rasters.



The **add** icon can be used to add new raster. On clicking this icon, the dialog box similar to the one shown in Figure 473 appears. As explained earlier, after the base task is selected, the raster name must be entered. This new raster is created with the trigger condition and by default, the duration raster has the duration of 10 s whereas the sample count raster has 10000 samples.



The **refresh** icon can be used to update the raster list. Once updated, the 'Select Raster' selection box displays all the rasters available in the Measurement Configuration pane.



The **XY Plotter** check box can be used to generate additional plotters and the signals in this plotter will be mapped to the selected raster name in the 'Select Raster' selection box.

Additional dialog box

If the signals connected to the instrument already have a raster, the existing raster dialog box appears as shown in Figure 479 .

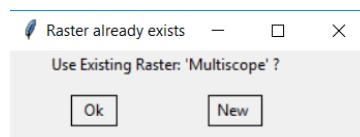


Figure 479: Existing Raster dialog box

The option Ok can be selected, if the existing raster, in this case, Multiscope is to be used. The option 'New' adds a new raster with a desired raster name.

An Example is illustrated below:

This Multiscope signals are already connected to the raster 'Multiscope', as shown in red.

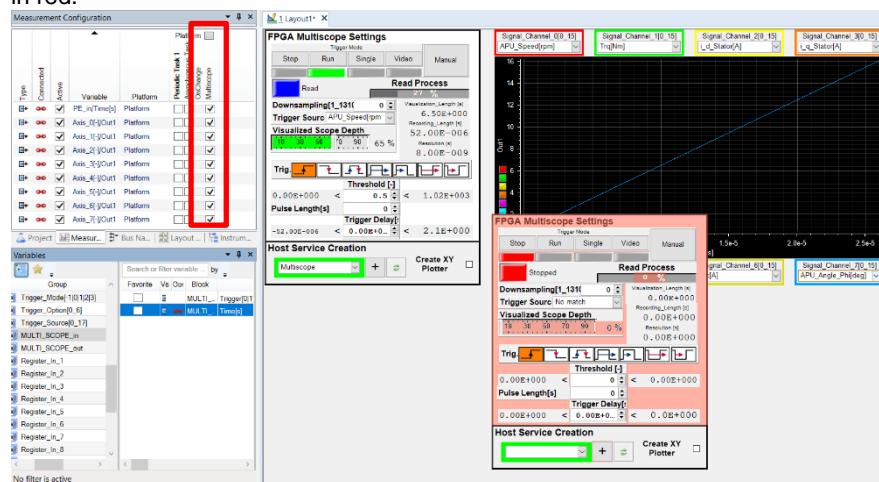


Figure 480: Example of existing Raster dialog box

Now, when the same signal (for instance MULTI_SCOPE_in -> Time[s]) is connected to a new instrument as shown above, the following dialog box appears.

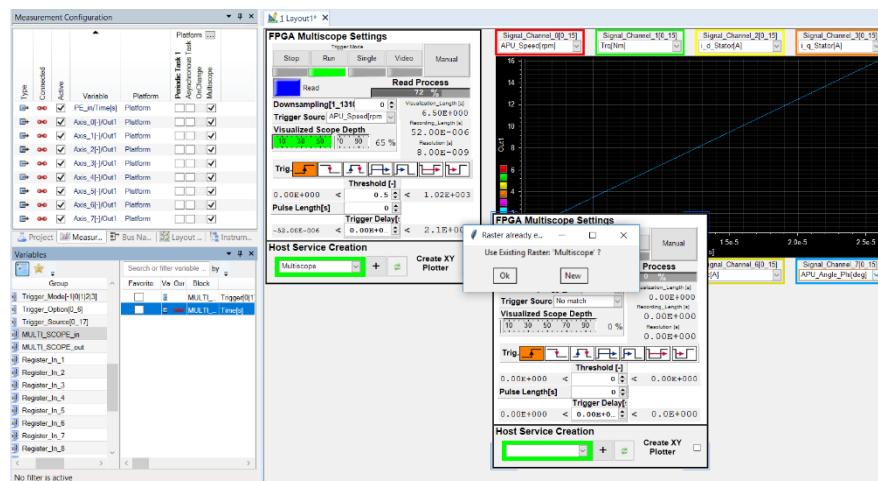


Figure 481: Example of existing Raster dialog box

Parameter Change

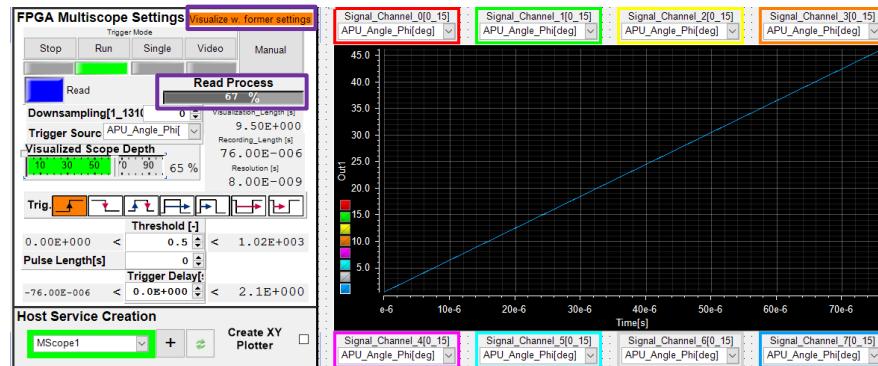


Figure 482: Parameter change during the read process

If a parameter is changed (for example, Downsampling), when the **Read process** is between 1 - 100%, the signals are plotted with the former settings as shown by the orange LED on top of the custom instrument (FPGA Scope Settings) in Figure 482: Parameter change during the read process Figure 482. This means that the changes become effective, when the read process is complete (100%) and can be visualized in the adjacent plotter. The parameter changes also become instantaneously effective when the trigger mode is either Stop or Armed.

Manual signal mapping of custom instruments

Description The signal mapping of the custom instruments can also be done manually. Therefore, please find the signal mapping below.

SCALE_DAC

Signal Mapping In the next picture an overview of the possible Instruments of the SCALE_DAC instrument in Control Desk is shown.

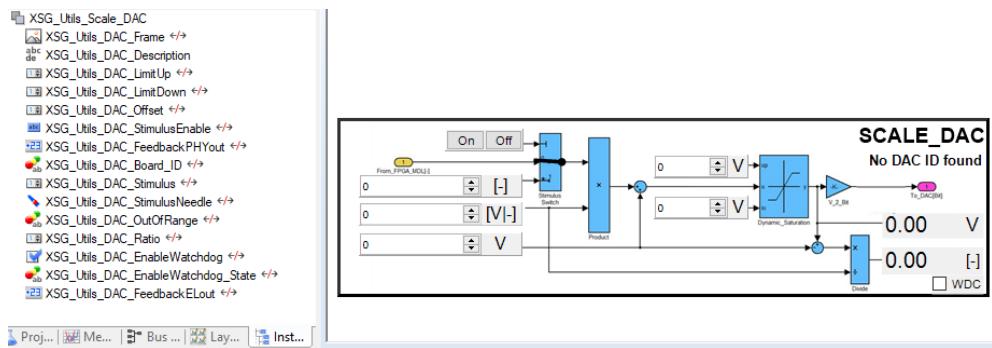


Figure 483: Instruments of the SCALE_DAC instrument

Instrument Name: XSG_Utils_DAC_	Simulink variable:
LimitUp	Block Input: Upper_Limit_Output_Voltage[V]
LimitDown	Block Input: Lower_Limit_Output_Voltage[V]
Offset	Block Input: Offset_Voltage[V]
StimulusEnable	Block Input: Enable_Stimulus[0 1]
FeedbackPHYout	Block Output: DAC_out_phy[-]
Board_ID	Block Output: Board_DAC_ID[-]
Stimulus	Block Input: Stimulus_Value[-]
StimulusNeedle	Block Input: Enable_Stimulus[0 1]
Ratio	Block Input: Voltage_Ratio[V/-]
EnableWatchdog	Block Variable: Const_en_watchdog
EnableWatchdog_State	Block Variable: Const_en_watchdog
FeedbackELout	Block Output: DAC_out_el[V]

SCALE_ADC

Signal Mapping In the next picture an overview of the possible Instruments of the SCALE_DAC instrument in Control Desk is shown.

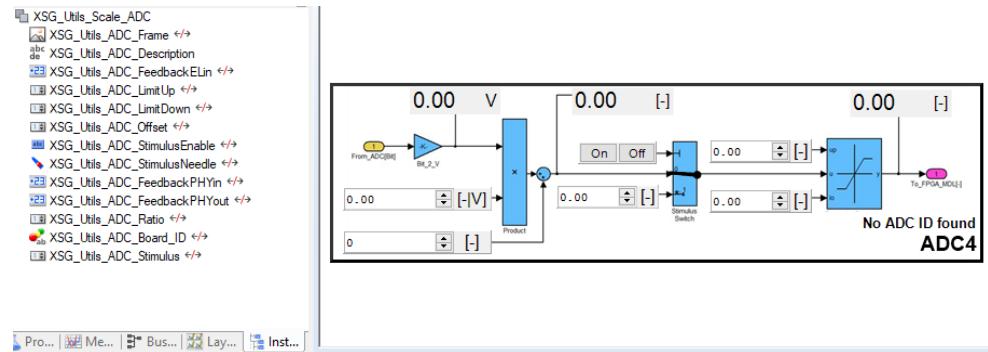


Figure 484: Instruments of the SCALE_ADC instrument

Instrument Name:	Simulink variable:
FeedbackELin	Block Output: ADC_in_el[V]
LimitUp	Block Input: Upper_Limit_Output[-]
LimitDown	Block Input: Lower_Limit_Output[-]
Offset	Block Input: Offset[-]
StimulusEnable	Block Input: Enable_Stimulus[0 1]
StimulusNeedle	Block Input: Enable_Stimulus[0 1]
FeedbackPHYin	Block Output: ADC_in_phy[-]
FeedbackPHYout	Block Output: ADC_out_phy[-]
Ratio	Block Input: Voltage_Ratio[-/V]
Board_ID	Block Output: Board_DAC_ID[-]
Stimulus	Block Input: Stimulus_Value[-]

SCOPE / MULTISCOPE

Signal Mapping

In the next picture an overview of the possible Instruments of the SCOPE / MULTISCOPE instrument in Control Desk is shown.

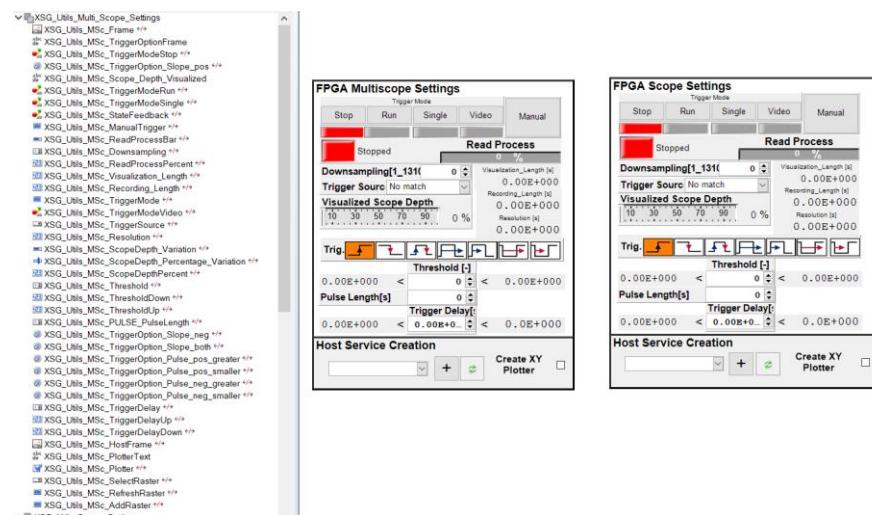


Figure 485: Instruments of the SCOPE / MULTISCOPE instrument

Instrument Name: XSG_Utils_Sc_ XSG_Utils_MSc	Simulink variable:
TriggerModeStop	Block Output: Trigger_State[0:stop_1:run_2:single_3:video]
TriggerModeRun	Block Output: Trigger_State[0:stop_1:run_2:single_3:video]
TriggerModeSingle	Block Output: Trigger_State[0:stop_1:run_2:single_3:video]
StateFeedback	Block Output: Trigger_State[0:stop_1:run_2:single_3:video]
Threshold	Block Input: Threshold[-]
ManualTrigger	Block Input: Manual_Trigger[-]
ReadProcessBar	Block Output: Read_Process[%]
Downsampling	Block Input: Downsampling[1_131071]
ReadProcessPercent	Block Output: Read_Process[%]
Visualization_Length	Block Output: Visualization_Length[s]
Recording_Length	Block Output: Recording_Length [s]
TriggerMode	Block Input: Trigger_Mode[-1 0 1 2 3]
ThresholdDown	Block Output: Trigger_Lower_Limit[-]
TriggerModeVideo	Block Output: Trigger_State[0:stop_1:run_2:single_3:video]
ThresholdUp	Block Output: Trigger_Upper_Limit[-]
TriggerSource	Block Input: Triger_Source
ScopeDepth_Percentage_Variation	Block Input: Const_Scope_Depth[%]
ScopeDepth_Variation	Block Input: Const_Scope_Depth[%]
ScopeDepthPercent	Block Input: Const_Scope_Depth[%]
TriggerDelay	Block Input: Trigger_Delay[s]
TriggerDelayDown	Block Output: TriggerDelay_Lower_Limit[-]
TriggerDelayUp	Block Output: TriggerDelay_Upper_Limit[-]
Resolution	Block Output: Resolution[s]
PULSE_PulseLength	Block Input: Pulse_Length[s]
TriggerOption_Slope_pos	Block Input: Trigger_Option[0_6]
TriggerOption_Slope_neg	Block Input: Trigger_Option[0_6]
TriggerOption_Slope_both	Block Input: Trigger_Option[0_6]
TriggerOption_Pulse_pos_greater	Block Input: Trigger_Option[0_6]
TriggerOption_Pulse_pos_smaller	Block Input: Trigger_Option[0_6]
TriggerOption_Pulse_neg_greater	Block Input: Trigger_Option[0_6]
TriggerOption_Pulse_neg_smaller	Block Input: Trigger_Option[0_6]

Templates

Overview

Overview

The XSG Utils library provides two template models to get a quick startup of programming a FPGA. Template models are available for the following FPGA Boards:

- DS5203 (7K325) with onboard I/O
- DS5203 (7K325) with Multi-I/O Module (DS5203M1)
- DS2655 FPGA Base Module + DS2655M1 I/O Module



The Template models require the XSG Utils library.

Utils Template Models

Location

The Utils template models are located in the XSG Utils library as shown in the figure below

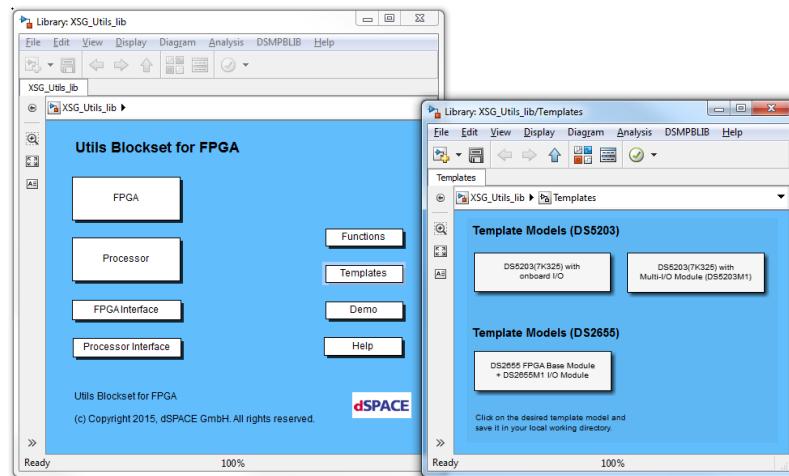


Figure 486: Utils Template Models

The desired model will be loading by double clicking on the desired block.

	<p>It is recommended to save the template model in your local working directory. Saved modifications in the original template models could not be recovered.</p>
--	--

DS5203 (PHS-Bus based)

The template model contains the following structure:

Model Structure (processor side)

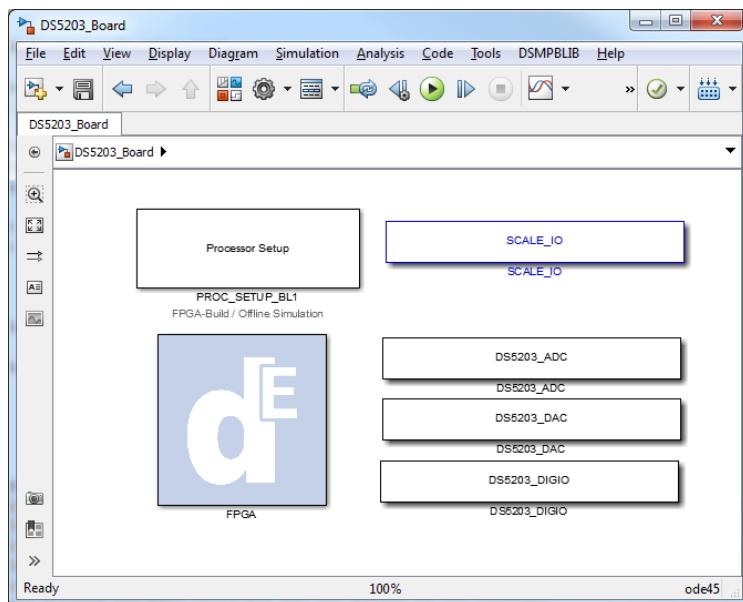


Figure 487: Model structure of the DS5203 (7K325) with onboard I/O (processor-side)



The following chapter describes only the template model for the DS5203(7K325) with onboard I/O. The behavior of the DS5203(7K325) with Multi-I/O Module (DS5203M1) template model is the same but with a different number of I/O and registers.

In the right upper corner an interface block for scaling the in- and outputs and for getting the version info of the blockset are insert. Additional to that, stimulus interface blocks for the analog inputs (DS5203_ADC), analog outputs (DS5203_DAC) and the digital I/O are located in the right corner, too. To insert your own plant model, replace the subsystems DS5203_ADC, DS5203_DAC and DS5203_DIGIO.



Current model describes a template stimulus model.

To insert your own plant model, replace the " DS5203_ADC " , the "DS5203_DAC" and the " DS5203_DIGIO " subsystem.

DS5203 (PHS-Bus based)

The FPGA model contains the model structure is described in the figure bellow.

Model Structure (FPGA side)

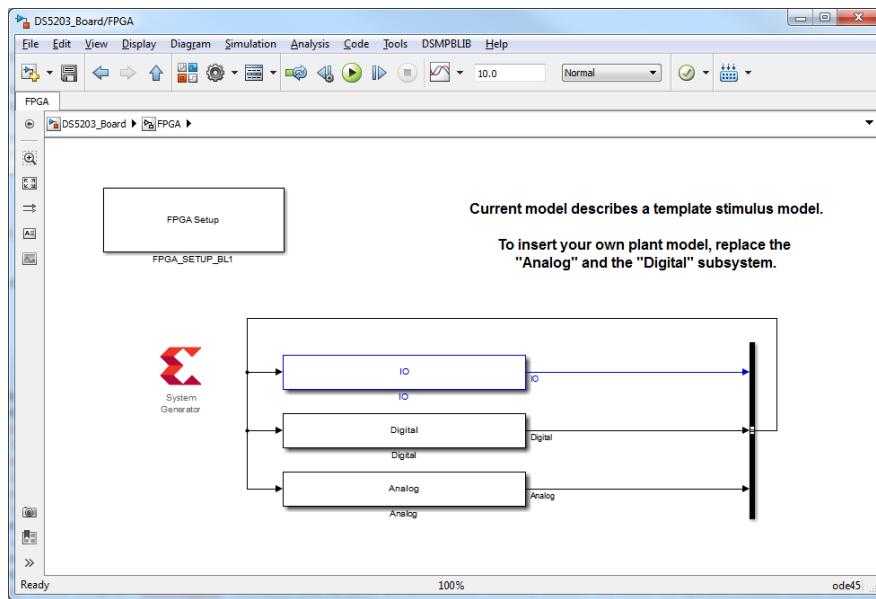


Figure 488: Model structure of the DS5203 (7K325) with onboard I/O (FPGA side)

The FPGA model part is divided into the IO, Digital and Analog subsystems. In the IO subsystem the overall IO of the FPGA board is inserting as shown in the figure bellow.

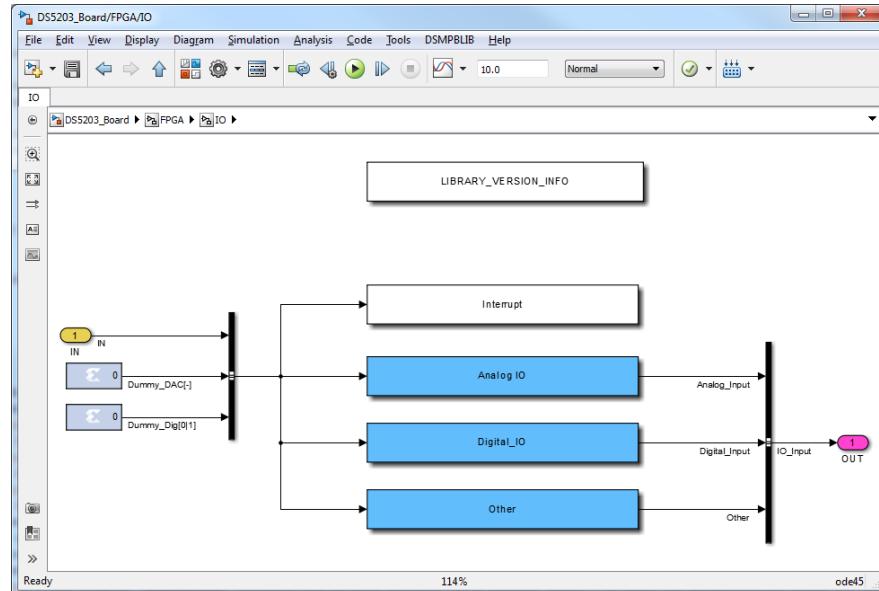


Figure 489: IO subsystem of the FPGA side

Each signal is separated in an overall group like “Analog IO”, “Digital IO” or “Other”. All interrupts which are generated on the FPGA are stored in the subsystem “Interrupt”. To get a better overview of the internal structure of the subsystems the “Analog IO” subsystem is used and shown in the figure below.

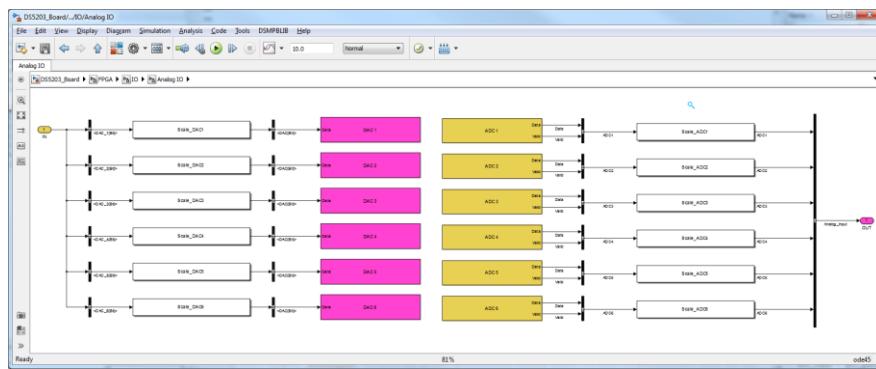


Figure 490: Analog IO subsystem of the FPGA side

For routing external signals to each output bus, selector blocks are inserting. If the port is not necessary a dummy variable for the analog and digital output is available.



Digital ports can be configured as inputs or outputs. In this template model 8 channels are configured as input and 8 channels as output!

Additional to the IO subsystem, stimulus subsystem for the digital and analog in- and outputs are available.



Current model describes a template stimulus model.

To insert your own plant model, replace the "Analog" and the "Digital" subsystem.

To use this simple model for a stimulus test of the DS5203 Board (7K325) a precompiled version for both FPGA boards is available and is called:

- DS5203 (7K325) with onboard I/O
Simulink model: DS5203_Board.slx
FPGA precompiled file: DS5203_Board.ini
- DS5203 (7K325) with Multi-I/O Module (DS5203M1)
Simulink model: DS5203_M1_Board.slx
FPGA precompiled file: DS5203_M1_Board.ini

DS2655 (SCALEXIO based)

The template model contains two separate models and represents a template model with one DS2655M1 I/O Module. The processor side (model name: DS2655M1_PROC_part) contains the following structure:

Model Structure (processor side)

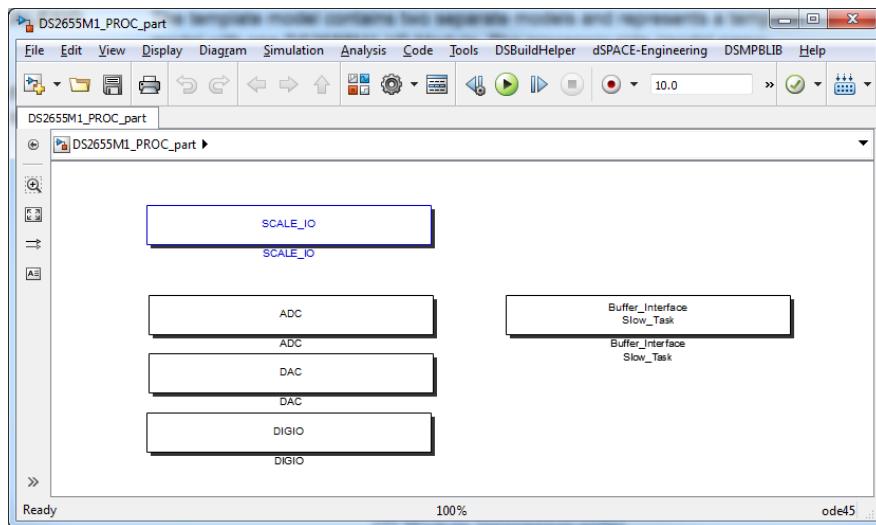


Figure 491: Model structure of the DS2655 FPGA Base Module + DS2655M1 I/O Module (processor-side)

In the left upper corner the version information of the actual FPGA model and the scaling of the in- and output can be found. The analog I/O can be manipulated by the SCALE_ADC and SCALE_DAC blocks of the XSG Utils library during runtime. For the digital I/O a stimulus block is used. Additional to that, stimulus interface blocks for the analog inputs (ADC), analog outputs (DAC) and the digital I/O are located in the right corner, too. To insert your own plant model, replace the subsystems ADC, DAC and DIGIO.

	<p>Current model describes a template stimulus model.</p> <p>To insert your own plant model, replace the " ADC " , the " DAC " and the " DIGIO " subsystem.</p>
---	---

DS2655/DS6601/ DS6602 (SCALEXIO based)

The FPGA model (separate model with model name: DS2655M1_FPGA_part) contains the model structure as described in the figure bellow.

Model Structure (FPGA side)

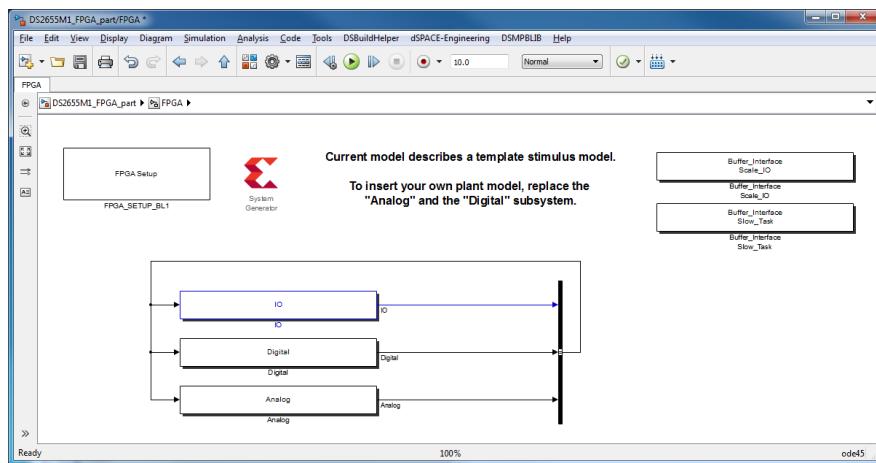


Figure 492: Model structure of the DS2655 FPGA Base Module + DS2655M1 I/O Module (FPGA side)

The FPGA model part is divided into the IO and stimulus subsystem for the digital and analog I/O subsystems. In the IO subsystem the overall IO of the FPGA board is inserted as shown in the figure below. Scaling and stimulus functionalities are realized with the SCALE_ADC and SCALE_DAC blocks of the XSG Utils library.

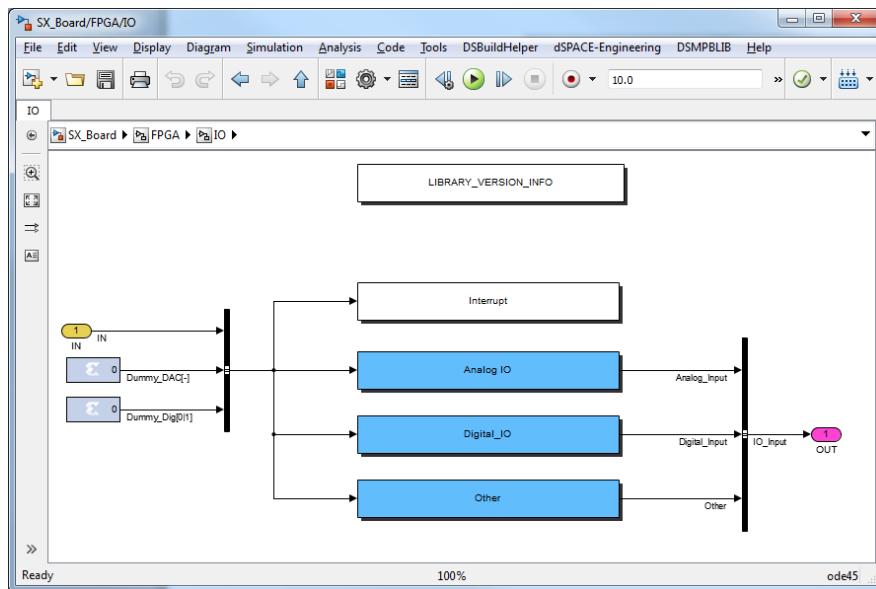


Figure 493: IO subsystem of the FPGA side

Each signal is separated in an overall group like “Analog IO”, “Digital IO” or “Other”. All interrupts which are generated on the FPGA are stored in the subsystem “Interrupt”. To get a better overview of the internal structure of the subsystems the “Analog IO” subsystem is used and shown in the figure below.

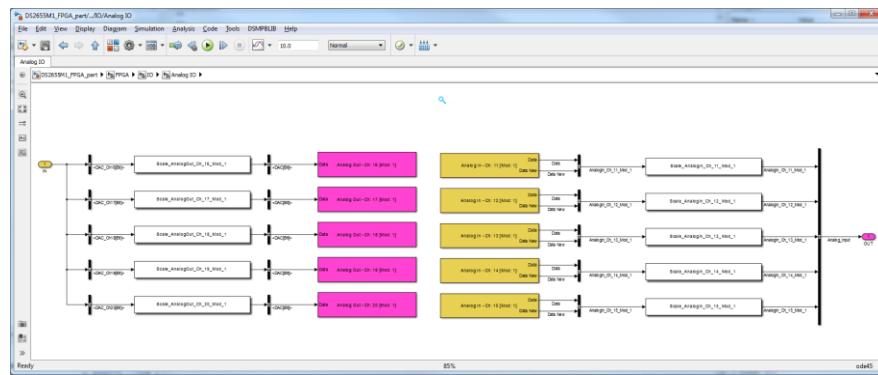


Figure 494: Analog IO subsystem of the FPGA side

For routing external signals to each output, bus selector blocks are inserting. If the port is not necessary, a dummy variable for the analog and digital output is available.

To use this simple model for a stimulus test of the DS2655 Module with one DS2655M1 Module a precompiled version is available and is called:

- DS2655 FPGA Base Module + DS2655M1 I/O Module

Simulink model processor side: DS2655M1_PROC_part.slx

Simulink model FPGA side: DS2655M1_FPGA_part.slx

FPGA precompiled file: DS2655M1_Board.ini

Functions

Overview

Overview

The XSG Utils library provides several functions to simplify the work with an FPGA model. These functions are not linked to a specific processor or FPGA model part and can be only offline used.

The different functions of the both library parts are shown in the figure below.

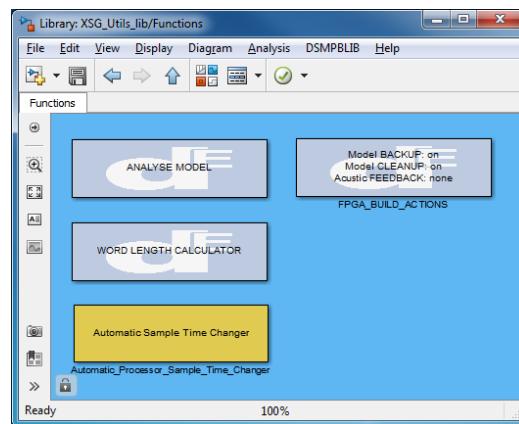


Figure 495: Functions of the XSG Utils library

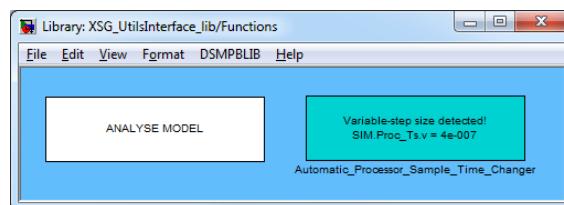


Figure 496: Functions of the XSG Utils Interface library

Additional to these, helpful matlab functions are also included which can be called from user code or the command window.

Analyze Model

Objective

For having an easy and fast overview about the used Library elements an automatic LOG file generation and analyse function tool is included. In the next sections each functionality is described in detail. The different analyze and file generation functions are selectable as shown in the figure below.

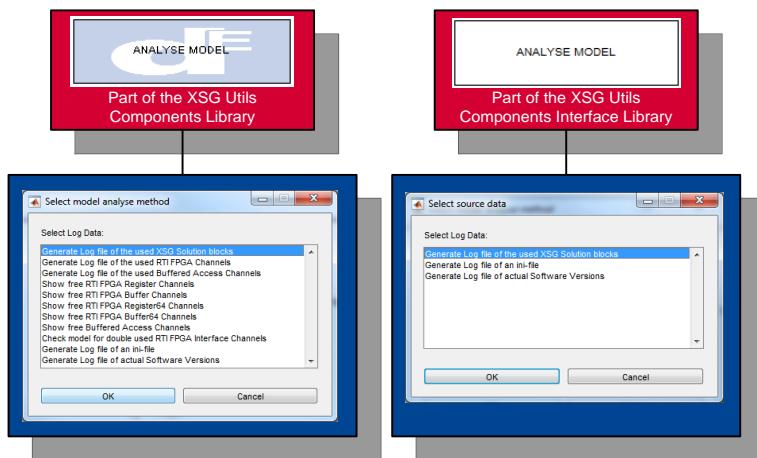


Figure 497: Selection boxes of the ANALYSE_MODEL and the LOG_FILE_GENERATION block

If a log file is generated the result is coded in an html format which can be read with a standard Internet Browser.

Example of a LOG File Generation of the used XSG Solution library blocks

The following example describes the procedure to perform an automatically generation of a LOG file which carries the used XSG Utils library blocks in the actual Simulink (FPGA) model. Therefore, open the model from which you want to generate a LOG file and insert the ANALYSE_MODEL (part of the XSG Utils library) or ANALYSE_MODEL (part of the XSG Utils Interface library) block which is located in the

- XSG_Utils_lib:
XSG_Utils.lib/Functions/ ANALYSE_MODEL
- XSG_UtilsInterface_lib:
XSG_UtilsInterface.lib/Functions/ ANALYSE_MODEL

Start the LOG file generation by double clicking on the block and selecting "**Generate Log file of the used XSG Solution blocks**". Afterwards a selection of the log data can be selected, as shown in the figure below.

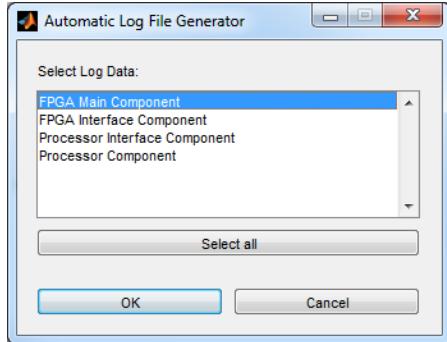


Figure 498: Selection of the searched blocks for the logged data

The different subtitle stands for:

- **FPGA Main Component:**
All FPGA Main Component blocks searched and settings are stored in the LOG file.
- **FPGA Interface Component:**
All FPGA Interface Component blocks searched and settings are stored in the LOG file.
- **Processor Interface Component:**
All Processor Interface Component blocks searched and settings are stored in the LOG file.
- **Processor Component:**
All Processor Component blocks are searched and settings are stored in the LOG file.

The result after clicking the ok button is a log file near the actual model. The save directory is specified with (% parallel to model root)

XSG_SOL_Folder\Manual__XSG_Utils.lib\Log_files\<date>_XSG_BLKS.lib.html.

Example LOG File Generation of the used RTI FPGA channels or Buffer Access Channels

The following example describes the procedure to perform an automatically generation of a LOG file which carries the used RTI FPGA Interface channels of the actual Simulink model.



Note that only the Register and Buffer on the FPGA side are analyzed and consulted for the report!

Therefore, open the model from which you want to know the free ports and insert the ANALYSE_MODEL block which is part of the XSG Utils library.

- **XSG_Utils.lib:**
XSG_Utils.lib/Functions/ ANALYSE_MODEL

- XSG_UtilsInterface_lib:
XSG_UtilsInterface_lib/Functions/ ANALYSE_MODEL

Start the analyze function by double clicking on the block and selecting “Generate Log file of the used RTI FPGA Channels”. The result after clicking the ok button is a log file near the actual model. The save directory is specified with (% parallel to model root)

XSG_SOL_Folder\Manual__XSG_Utils_lib\Log_files\<date>_RTI_PROC_CHANNELS.html.

Example of the analyze functionality to show free RTI FPGA Register and Buffer channels

The following example describes the procedure to perform an analyze function to show the free PHS Register or Buffer channels of the actual Simulink model.



Note that only the Register and Buffer on the FPGA side are analyzed and consulting for the report!

Therefore open the model from which you want to know the free interface channels and insert the ANALYSE_MODEL block which is part of the XSG Utils library.

- XSG_Utils_lib:
XSG_Utils_lib/Functions/ ANALYSE_MODEL

Start the analyze function by double clicking on the block and selecting “Show free RTI FPGA PHS Register Channels” to show you the free register channels or “Show free RTI FPGA PHS Buffer Channels” to show you the free buffer channels of the PHS interface. The result after clicking the ok button is a popup with the free PHS interface channels of the in- and output of the register or buffer as shown in the figure below.

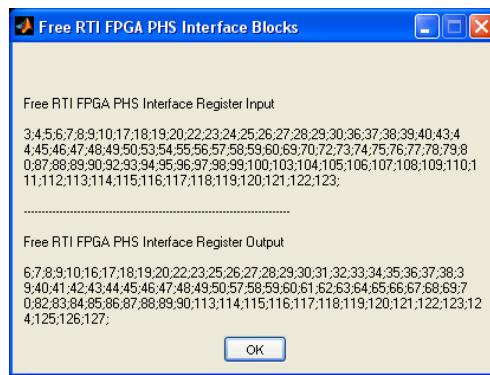


Figure 499: Example of the popup message of the free interface channels

Example of a check for double used RTI FPGA channels

To minimize the error rate by a Xilinx build process it is assumed to check if double used RTI FPGA Interface channels exist in the actual model. This can occur if template models are build with the automatic generate function of the XSG Utils library and the result is copy in the actual FPGA model but the used channels are also used for other functionalities.



Note that only the Register and Buffer on the FPGA side are analyzed and consulted for the report!

To perform this check open the actual model and insert the ANALYSE_MODEL block which is part of the XSG Utils library.

- XSG_Utils_lib:
XSG_Utils_lib/Functions/ ANALYSE_MODEL

Start the analyze function by double clicking on the block and selecting “Check model for double used RTI FPGA Channels”. If the algorithm detects double used RTI FPGA interface channels a report is generated and is stored in the current work directory of Matlab with the name (%MDL_Name)_Check_Double_Used_RTI_FPGA_CHANNELS.html. If the algorithm detects no double used PHS interface channels, a popup message is generated.

Example LOG File Generation of the used RTI FPGA Interfaces of a generated ini file

The following example describes the procedure to perform an automatically generation of a LOG file which carries the used RTI FPGA channels and version and board information of a generated ini file. Therefore open the ANALYSE_MODEL (part of the XSG Utils library) or ANALYSE_MODEL (part of the XSG Utils Interface library) block in a current Simulink model which is located in the

- XSG_Utils_lib:
XSG_Utils_lib/Functions/ ANALYSE_MODEL
- XSG_UtilsInterface_lib:
XSG_UtilsInterface_lib/Functions/ ANALYSE_MODEL

Start the LOG file generation by double clicking on the block and selecting “Generate Log file of an ini-file”. Afterwards a user selection of the investigated ini file is necessary as the figure shown below.

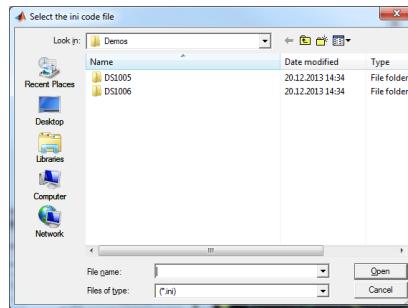


Figure 500: Selection of the investigated ini file

The result after clicking the ok button is a log file near the actual model. The save directory is specified with (% parallel to model root)
 XSG_SOL_Folder\Manual__XSG_Utils_lib\Log_files\<date>_RTI_FPGA
 _INT_INI.html.

Example LOG File Generation of the actual Software Version

The following example describes the procedure to perform an automatically generation of a LOG file which carries the actual Software versions of Matlab toolboxes.

Therefore, open the model from which you want to know the free ports and insert the ANALYSE_MODEL block which is part of the XSG Utils library.

- XSG_Utils_lib:
XSG_Utils_lib/Functions/ ANALYSE_MODEL
- XSG_UtilsInterface_lib:
XSG_UtilsInterface_lib/Functions/ ANALYSE_MODEL

Start the analyze function by double clicking on the block and selecting "Generate Log file of actual Software Versions". The result after clicking the ok button is a log file near the actual model. The save directory is specified with (% parallel to model root)
 XSG_SOL_Folder\Manual__XSG_Utils_lib\Log_files\<date>_Version_Info.html.

Example generate r2b and b2r files from the used Buffer2Register Resiter2Buffer blocks

The following example describes the procedure to generate the connection description files (r2b and b2r) from the used Buffer2Register and Register2Buffer blocks.



Note that only the Buffer2Register and Register2Buffer blocks on the FPGA side are analyzed!

Therefore, open the model from which you want to know the free interface channels and insert the ANALYSE_MODEL block which is part of the XSG Utils library.

- XSG_Utils_lib:
XSG_Utils_lib/Functions/ ANALYSE_MODEL

Start the generation by double clicking on the block and selecting “Generate r2b and b2r files”. The result after clicking the ok button is the connection files near the actual model. The save directory is specified with (% parallel to model root)
XSG_SOL_Folder\Manual__XSG_Utils_lib\BufferAccess

Word Length Calculator

Objective

The XSG Utils library is implemented in Fix point computation which requires a conversion of the floating point to a defined fixpoint datatype. To calculate an optimal datatype for a user defined value a Word Length Calculator is implemented.

The data type is calculated based on a maximal value using different computations methods. The workflow and the computation methods are described in the next section.

Workflow

The figure below shows an overview of the word length calculator GUI.

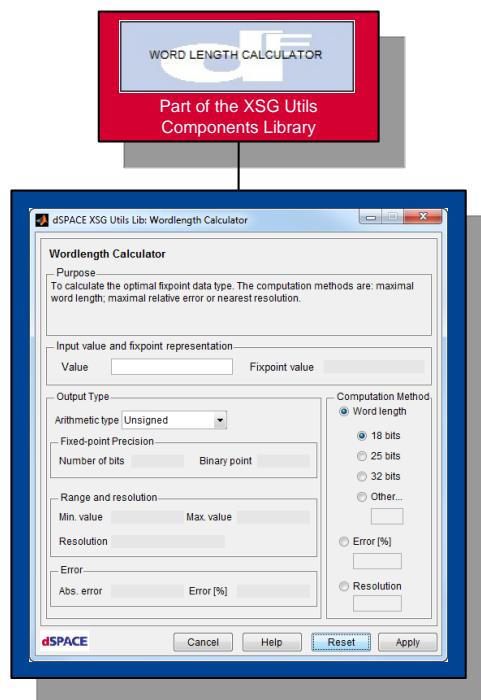


Figure 501: Overview of the Word Length Calculator GUI

The floating point value can be entered in the value in the value field, which can be positive or negative, scalar or a variable, which is defined in the workspace. It can also be a matlab known constant like "pi". Not accepted are values like "Inf", "Nan" or complex values are not supported.

The Input value is considered as the maximal value that should be represented.

Also an arithmetic type can be set to define if the value is signed or unsigned.

Therefore the popup "Arithmetic type" can be used.

Given a maximum value, three computation methods are supported which are selected per radio button on the right side of the GUI. The three methods are:

- **Word length:** computation by given maximal available bit. Three preset values can be select per radio buttons: 18 (standard), 25 and 32 bits. For other bitwidth the other radio button can be selected and the number of

bit has to be inserted in the field below which must be a positive integer value. Note that the option "Other..." can be selected only when the computation method is set to "Wordlength". If the given word length is not enough to represent the input value in fix point, the maximal field will be highlighted in red. If the input value is negative and the arithmetic type is set to "unsigned" the minimal value will be set to zero and will be highlighted in red, too.

- **Error [%]:** this method is oriented by the relative error between the estimated fix point representation and the input value. The maximal used word length with this method is set to 130 bit.
- **Resolution:** this computation is based on a maximal resolution. The input value can be a positive value less or equal to one. The fix point data type is calculated that at least the given resolution can be represented. For example, by a given resolution '0.1' the best nearest can be '0.0625' and need therefore 4 bits for the decimal part.

The computation automatically starts when all the necessary input values are available

The functionality of the "Word length calculator" can also comfortable called per command line. The detailed workflow of the function called "xsg_utils_wordlength_calculator" is shown in a separate subchapter in the section "Useful command line called functions".

Automation FPGA Build Actions

Objective

To perform automated actions after FPGA Build and useful settings of the rtifpga blockset the following block can be integrated in the FPGA model as shown in figure below.

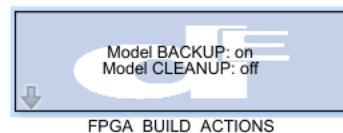


Figure 502: FPGA_BUILD_ACTIONS block

The different actions of this block will be explained in detail in the chapters below.

GUI settings

The following GUI settings are available.

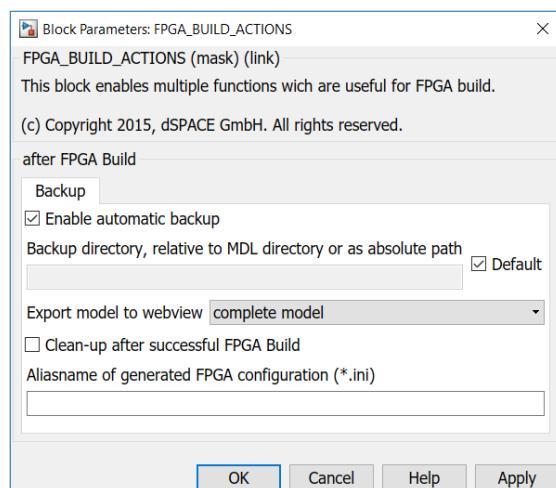


Figure 503: GUI of the FPGE_BUILD_ACTION block

rtifpga

In the group “rtifpga” special functions of the rtifpga blockset can be modified. In this case the automatic channel number adaption of the rtifpga blockset can be en- or disabled. The actual state of this settings is also available in matlab workspace (variable: “G_RTIFPGA_COPYBLOCK_CALLBACK_ENABLE”).

Enable automatic backup

Actions after the FPG build are in the group “after FPGA Build” and are listed below. If the automatic backup function is enabled, an additional checkbox and edit field will be visible.

Backup folder path:

In this field the backup folder of the generated data can be specified. If the checkbox “Default” will be selected, the generated backup folder will be set to the default path with respect to the actual model.

Popmenu “Export model to webview”:

An additional export of the FPGA model to a web view can be enabled by selection of the specific popup. The generate web view zip file is equal to the Matlab export function to web.

Checkbox “Clean-up after successful FPGA Build”:

If a cleanup after successfully build is preferred, please enables the corresponding function in the GUI. If the function is enabled the temporary files and the FPGA Build folder will be deleted.

Edit field “Aliasname of generated FPGA configuration

If an alias name is defined in this field, the generated ini file will be renamed.

Result after FPGA Build:

The algorithm will generate the following folder structure after successfully FPGA build (the example shows a default generated):

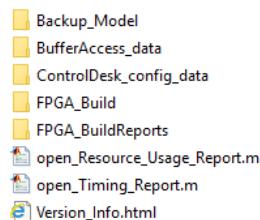


Figure 504: Generated folder structure

The structure will cover the following substructures:

XSG_Sol_Folder \<date>_<modelname>

- *Backup_Model*
Copy of the Simulink FPGA model: <modelname>.slx. If the keyword “FPGA” is found in the modelname (e.g.: Test_FPGA_v1.slx) the algorithm search for a corresponding processor model with the keyword “PROC” (in this case: Test_PROC_v1.slx). If the web view export is activated the web view zip file is also located in this folder. Additional to that the actual matlab workspace will be stored in the file ws.mat.
- *BufferAccess_data*
Optional folder if the FPGA model uses Buffer2Register / Register2Buffer blocks for processor FPGA communication. The specific communication description files (r2b / b2r) are located her.
- *ControlDesk_config_data*
Optional folder if blocks are used in the FPGA which supports configuration files for ControlDesk.
- *FPGA_Build*
Copy of the generated *.ini file which is renamed to the Alias

➤ *FPGA_BuildReports*

Copy of the generated file (*.rpt) which give feedback of the hardware resource usage of the FPGA build. The file can be opened via Notepad.

Copy of the timing result. The actual timing can be displayed by using the following workflow:

- Open the backup model which is stored in the subfolder Backup_Model
- Execute the following command in the command window of Matlab xlAnalyzeTiming(bdroot(gcbh),[pwd,'\\FPGA_Timing'])

➤ *Main folder*

In the main folder a Version_Info.html file is also stored which listed all activated Matlab toolboxes during FPGA build.

- *Open_Resource_Usage_Report.m*

Open this M-file and run it to open the Resource Utilization report.

- *Open_Timing_Report.m*

Open this M-file and run it to open the Timing report.

Necessary changes for Remote FPGA Build

The following steps are necessary to enable the FPGA backup function on the remote Build Server (additionally to the rtifpga specific settings):

- Copy the file

`<Solution\InstallationRoot>\Matlab\Python\xsg_utils_rtifpgahook.py`

To the destination of the unzipped FPGABuildServer.zip

`<ServerFolder>\FPGABuildServer\FPGABuildServer\Python`

Reserve Communication Channels for Automatic Interface Generation

Objective

In some cases it is useful to reserve a defined slot of communication channels (Register, Buffer, Buffer Access, ...) which are interpreted via the automatic interface generation. To activate this feature drag and drop the following block, which is shown in the figure below, in the source model.



Figure 505: Reserved_Comm_Channels block

In the GUI, which is shown in the figure below, the reserved slot can be configured.

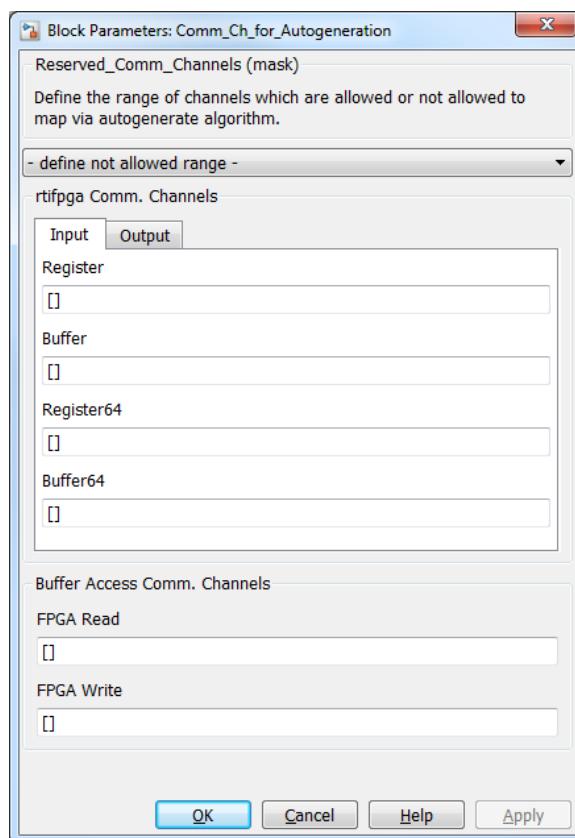


Figure 506: Reserved_Comm_Channels GUI

If this block is located in the model, the automatic interface generation will check the actual GUI entries.

To get a better overview of the channel space each case is described in detail.

- Interface generation without reserved channels:



Figure 507: Comm_Channels without reserved channels

If the reserved block is not used, the automatic interface generation uses the next free channels of the actual FPGA board.

- Interface generation with defined **not allowed** channels:

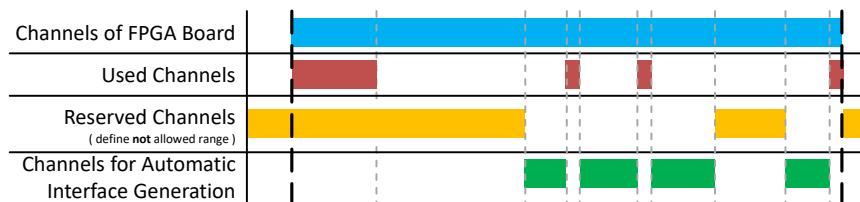


Figure 508: Comm_Channels with not allowed channels

In the shown example the reserved block is add to the model with defined not allowed channels. The automatic interface generation uses the free space of channels between the used and the reserved channels which are available on the actual FPGA board.

- Interface generation with defined **allowed** channels:

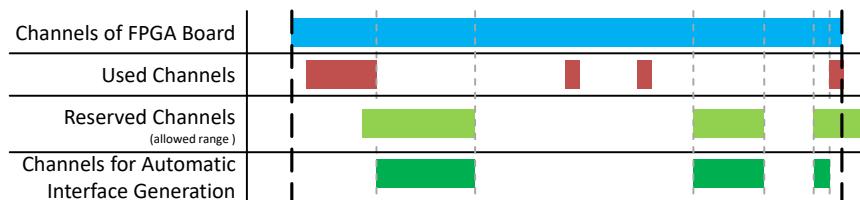


Figure 509: Comm_Channels with allowed channels

In the shown example the reserved block is add to the model with defined allowed channels. The automatic interface generation uses the space of channels which are available on the actual FPGA board, which are defined in the block GUI and which are not actual used in the model.

Useful command line called functions

Content

Additional to the functions which can be called by double-clicking on the block (like ANALYSE_MODEL function), helpful matlab tools are also included which can be called from user code or the command window. In the following section these functions are described in detail.

xsg_utils_fcn

Description / Overview

Often a list of the used FPGA blocks of the rti_fpga library or the actual frame work of the FPGA model is necessary for a self-made user automatization. Therefore the xsg_utils_fcn can be used. The possible callbacks are:

- [existingBlks] = xsg_utils_fcn('FIND_RTI_FPGA_BLKS', hmodel)
- [FPGA_Framework] =


```
xsg_utils_fcn('GET_RTI_FPGA_FRAMEWORK',hModel)
```
- [Ports] = xsg_utils_fcn('MAX_RTI_FPGA_CHANNELS',FPGA_Framework)
- [List] = xsg_utils_fcn('USED_RTI_FPGA_CHANNELS',hModel)
- [List] = xsg_utils_fcn('FREE_RTI_FPGA_CHANNELS',hModel)
- [List] = xsg_utils_fcn('CHECK_RTI_FPGA_DOUBLE_USED_CHANNELS',hModel)
- [List] = xsg_utils_fcn(' GET_RTI_FPGA_INI_DATA ',inipath)

In the subsections bellow the different callback methods of the function are described in detail.

xsg_utils_fcn('FIND_RTI_FPGA_BLKS',...)

This function search all FPGA specific blocks, like DAC, ADC, Register, Buffer and return a variable structure with block handles. The method can be called with:

[existingBlks] = xsg_utils_fcn('FIND_RTI_FPGA_BLKS', hModel)

The transfer parameter (hmodel) can be the actual model path or the actual subsystem. The function will search only in the defined parameter the specific FPGA blocks. The parameter hModel could be:

- gcs (get current system; get pathname of current system)
- gcb (get current block; get pathname of current block)
- gcbh (get current block handle; get handle of current block)
- bdroot (return name of top-level Simulink system)

For detailed information of these internal matlab functions please refer to the matlab documentation / help.

xsg_utils_fcn('GET_RTI_FPGA_FRAMEWORK', ...)	This function returns the actual FPGA framework of the FPGA model which is defined in the FPGA setup block. The method can be called with: [FPGA_FRAMEWORK] = xsg_utils_fcn('GET_RTI_FPGA_FRAMEWORK', hModel)
	The transfer parameter (hmodel) can be the actual model path or the actual subsystem.
xsg_utils_fcn('MAX_RTI_FPGA_CHANNELS', ...)	Returns the maximum number of available FPGA channels (only Register and Buffer) which depends on the actual FPGA framework of the model. The returned parameter is an array of the available ports. The method can be called with: [Ports] = xsg_utils_fcn('MAX_RTI_FPGA_CHANNELS', hModel)
	The transfer parameter (hmodel) can be the actual model path or the actual subsystem.
xsg_utils_fcn('USED_RTI_FPGA_CHANNELS', ...)	Return an array of used FPGA channels (only Register and Buffer). The method can be called with: [List] = xsg_utils_fcn('USED_RTI_FPGA_CHANNELS', hModel)
	The transfer parameter (hmodel) can be the actual model path or the actual subsystem.
xsg_utils_fcn('FREE_RTI_FPGA_CHANNELS', ...)	Return an array of free FPGA channels (only Register and Buffer), which also depends on the actual FPGA framework of the model. The method can be called with: [List] = xsg_utils_fcn('FREE_RTI_FPGA_CHANNELS', hModel)
	The transfer parameter (hmodel) can be the actual model path or the actual subsystem.
xsg_utils_fcn('CHECK_DOUBLE_USED_RTI_FPGA_CHANNELS', ...)	Return an array of double used FPGA channels (only Register and Buffer). The method can be called with: [List] = xsg_utils_fcn('CHECK_DOUBLE_USED_RTI_FPGA_CHANNELS', hModel)
	The transfer parameter (hmodel) can be the actual model path or the actual subsystem.
xsg_utils_fcn('GET_RTI_FPGA_INI_DATA', ...)	Return the data of a FPGA description file (ini-file). The method can be called with: [List] = xsg_utils_fcn('GET_RTI_FPGA_INI_DATA', inipath)

The transfer parameter (inipath) defines the path of the ini-file.

xsg_utils_wordlength_calculator

Description / Overview	Often a conversion of a defined floating point datatype to a fixpoint datatype is necessary for a self-made user automatization. Therefore the xsg_utils_wordlength_calculator can be used. The possible callbacks are:
▪ [dataType] =	xsg_utils_wordlength_calculator(value,methodString,parameter,sigflag)
▪ [dataType] =	xsg_utils_wordlength_calculator(value,'WORD_LENGTH',maxWordLength,sigflag)
▪ [dataType] =	xsg_utils_wordlength_calculator(value,'RESOLUTION',maxWordLength,sigflag)
▪ [dataType] =	xsg_utils_wordlength_calculator(value,'ERROR',maxWordLength,sigflag)

In the subsections below the different callback methods of the function are described in detail.

The function returns for all this different calls a structure (dataType) with information on the estimated data type which is defined by:

- dataType.**value**: the value which can be represented with the estimated data type
- dataType.**nbits**: the total number of bit for the representation
- dataType.**bpoint**: binary point position
- dataType.**minval**: the minimal value that can be represented
- dataType.**maxval**: the maximal value that can be represented
- dataType.**res**: the resolution (LSB) of the estimated data type
- dataType.**abs_error**: the absolute error between the representable value and the input value
- dataType.**rel_error**: the relative error between the representable value and the input value
- dataType.**str**: the Xilinx representation of the computed data type

xsg_utils_wordlength_calculator(value,'WORD_LENGTH',...) This function estimates the data type of an input value based on the maximal bit length given as parameter. The method can be called with:

[dataType] = xsg_utils_wordlength_calculator(value,'WORD_LENGTH', maxWordLength,sigflag)

The parameter value represents the maximal value which can be represented and can be a numerical datatype, nor NaN nor Inf.

The constant string 'WORD_LENGTH' represents the computation method.

The parameter maxWordLength represent the maximal word length available for the computation.

The parameter signflag determinates if the computed value can be positive. His value must be 1 or true if the computed value can be signed or 0 or false in the other case.

After the bits for the sign flag, an integer part is calculated the other bits (if they are) are assigned to the decimal part. If more bits for the decimal part are available, more resolution getting better. The output of the function is a structure similar as described in the section before.

xsg_utils_wordlength_calculator(value,'RESOLUTION',...)

This function estimates the data type of an input value based on maximal resolution given as a parameter. The method can be called with:

```
[dataType] = xsg_utils_wordlength_calculator(value,'RESOLUTION',
maxResolution,singflag)
```

The parameter value represents the maximal value which can be represented and can be a numerical datatype, nor NaN nor Inf.

The constant string 'RESOLUTION' represents the computation method.

The parameter maxResolution represent the maximal resolution the estimated data type should have. It must a positive number less or equal than one.

The parameter signflag determinates if the computed value can be positive. His value must be 1 or true if the computed value can be signed or 0 or false in the other case.

The computed data type is calculated so that his resolution is less or equal the given maximal resolution (maxResolution). The output of the function is a structure similar as described in the section before.

xsg_utils_wordlength_calculator(value,'ERROR',...)

This function estimates the data type of an input value based on maximal relative error given as parameter. The method can be called with:

```
[dataType] = xsg_utils_wordlength_calculator(value,'ERROR',
maxRelError,singflag)
```

The parameter value represents the maximal value which can be represented and can be a numerical datatype, nor NaN nor Inf.

The constant string 'ERROR' represents the computation method.

The parameter maxRelError represent the maximal relative error in percent (%) between the input value (value) and his fix point representation. This parameter must a positive number not equal to zero.

The parameter signflag determinates if the computed value can be positive. His value must be 1 or true if the computed value can be signed or 0 or false in the other case.

The computed data type is calculated, so that the error between the input value and his fix point representation is less or equal than the given maximal relative error (maxRelError). The output of the function is a structure similar as described in the section before.

xsg_utils_action_buffer_access

Description / Overview

If model parts are copied from one to another model or corresponding blocks of the buffer access are deleted, wrong or defect blocks must be repaired by hand. To speed up this repair the xsg_utils_action_buffer_access can be used. The possible callbacks are:

- `xsg_utils_action_buffer_access('DELETE_INVALID', hModel)`
- `xsg_utils_action_buffer_access('RESTORE_INVALID', hModel)`

In the subsections bellow the different callback methods of the function are described in detail.

`xsg_utils_action_buffer_access('DELETE_INVALID', hModel)`

This function search and delete all invalid buffer access blocks. The feedback will be generated in the command window of matlab. The method can be called with:

`xsg_utils_action_buffer_access('DELETE_INVALID', hModel)`

The parameter hModel represents the pointer to the actual system or subsystem of the model which will be analyzed.

`xsg_utils_action_buffer_access('RESTORE_INVALID', hModel)`

This function search and restore all invalid buffer access blocks which have no corresponding block. The feedback will be generated in the command window of matlab. The method can be called with:

`xsg_utils_action_buffer_access('RESTORE_INVALID', hModel)`

The parameter hModel represents the pointer to the actual system or subsystem of the model which will be analyzed.

MATLAB Tools (shortcuts)

Content

In order to make the model preparation (Offline and Online), updating buffer interfaces and generating log files easier, some shortcuts have been added to the MATLAB/Simulink tools tab. In the following section the usage of these shortcuts is described in detail.

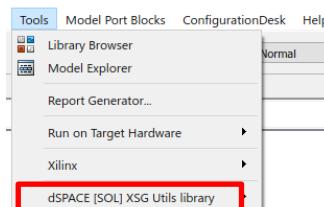


Figure 510: XSG Utils shortcuts under Tools pane

Prepare model for: Offline simulation This function is described at length here: Menu bar option for Offline simulation in MATLAB/Simulink

Prepare model for: PROC simulation This function is described at length here: Menu bar option for Offline simulation in MATLAB/Simulink

Analyze model for XSG purpose This function is described at length here: Analyze Model

Open word length calculator This function is described at length here: Word Length Calculator

Update Buffer Interface

Description

The ‘Update Buffer Interface’ shortcut can be used to generate/update the buffer communication between Processor and FPGA. An example is shown below:

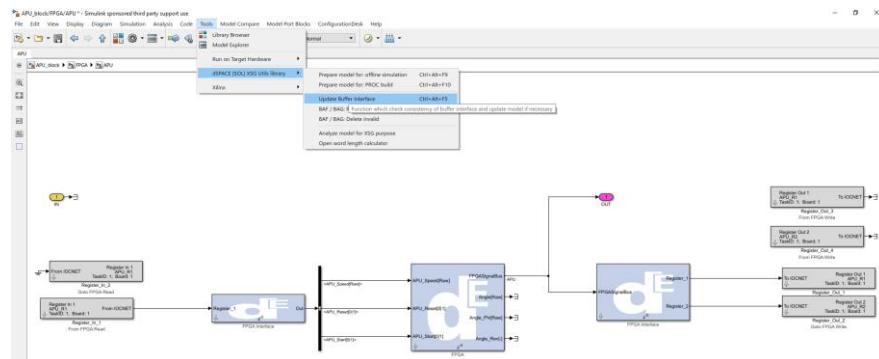


Figure 511: Update Buffer interface

In the above example, after Automatic Interface Generation, the update buffer interface from the Tools menu is selected. Following GUI appears, by default both FPGA and Processor side is selected, click ‘OK’.

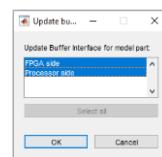


Figure 512: Update buffer interface GUI

After clicking ‘OK’, the interface is generated, and the progress is displayed in the MATLAB Command Window as shown below.

```
Command Window
=====
----- Update Buffer Access Interface -----
-----
Check / Update FPGA part of model ...
Rebuild/Build: BBR Board:1 Task_ID:1 ...
-> DONE - 24.9 sec.
Rebuild/Build: RBB Board:1 Task_ID:1 ...
-> DONE - 22.9 sec.

-----
Check / Update Processor part of model ...
Read settings for Processor update from FPGA side ...
-> DONE
Rebuild/Build: B2R_out Board:1 Task_ID:1 ...
-> DONE - 2 sec.
Rebuild/Build: R2B_in Board:1 Task_ID:1 ...
-> DONE - 1.7 sec.

-----
-> Total time amount - 54.4 sec.
```

Figure 513: Buffer interface progress in MATLAB Command Window

The end of the buffer interface generation is identified, when the ‘Total time amount’ is displayed in the command window. The Buffer_in and Buffer_out blocks are generated as on the FPGA and on the Processor, side as shown below (in Yellow and Magenta subsystems).

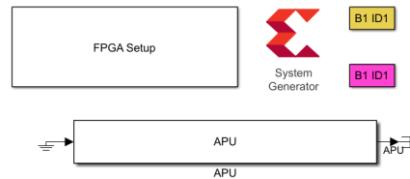


Figure 514: FPGA after the Buffer interface is generated

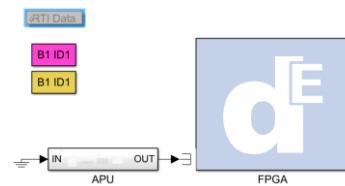


Figure 515: Processor after the Buffer interface is generated

BAF / BAG: Restore invalid

Description

The Buffer access block consists of ‘From’ and ‘Goto’ set. If a model contains only ‘From’ or it contains only ‘Goto’, then it is called an ‘Invalid’ block, and they can be restored by selecting ‘BAF/BAG: Restore invalid’ from the Tools menu. An example is shown below.

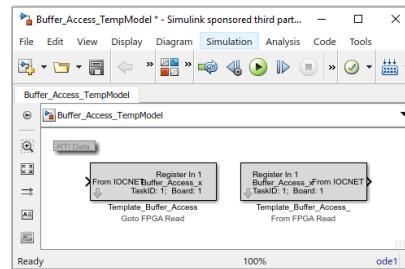


Figure 516: Valid Buffer Access blocks

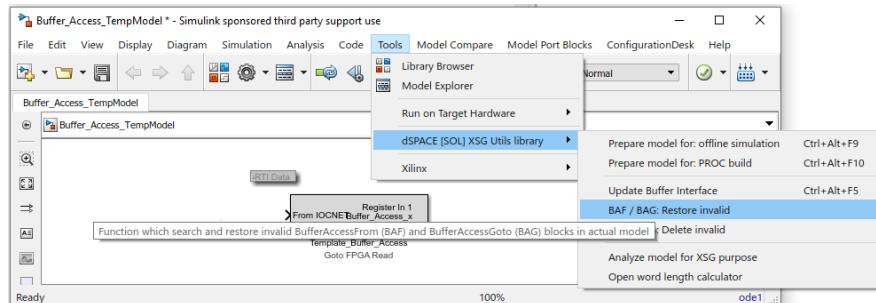


Figure 517: Restore corresponding ‘From’ block with ‘Restore invalid’

BAF / BAG: Delete invalid

Description

The Buffer access block consists of ‘From’ and ‘Goto’ set. If a model contains only ‘From’ or it contains only ‘Goto’, then it is called an ‘Invalid’ block, and they can be deleted by selecting ‘BAF/BAG: Delete invalid’ from the Tools menu. An example is shown below.

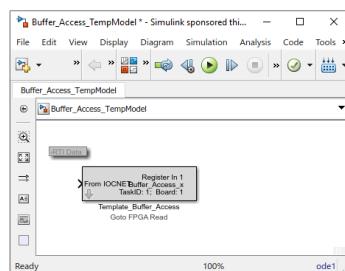


Figure 518: Invalid Buffer Access blocks

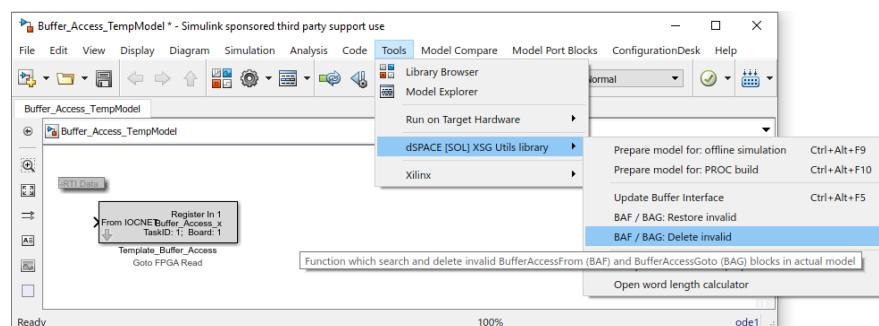


Figure 519: Delete invalid ‘Goto’ block with ‘BAF/BAG: Delete invalid’