

**XSG Injector Emulation Interface Solution**

# **Software User Guide**

**20.1.0 – Jun 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:	<a href="mailto:info@dspace.de">info@dspace.de</a>
Web:	<a href="http://www.dspace.com">http://www.dspace.com</a>

## How to Contact dSPACE Support

To contact dSPACE if you have problems and questions, fill out the support request form provided on the website at <http://www.dspace.com/go/supportrequest>.

The request form helps the support team handle your difficulties quickly and efficiently.

In urgent cases contact dSPACE via phone: +49 5251 1638-941 (General Technical 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.com/go/support> for software updates and patches.

## Important Notice

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

© 2020 by:  
dSPACE GmbH  
Rathenaustraße 26  
33102 Paderborn  
Germany

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

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

# Contents




<b>About This Document .....</b>	<b>4</b>
<i>Conventions Used in dSPACE User Documentation .....</i>	<i>4</i>
<b>Introduction.....</b>	<b>5</b>
Solenoid Injectors .....	6
Injector Emulation Features.....	8
<b>Target Hardware .....</b>	<b>9</b>
<b>Simulation Model.....</b>	<b>11</b>
FPGA Model .....	12
Simulink Interface Model.....	15
Feedback Voltage Look Up Table .....	20
Register Mappings .....	21
<b>Quick Start Guide .....</b>	<b>23</b>
How to Integrate the Injector Emulation Interface in an Existing Project.....	24
How to Create a Feedback Voltage LUT .....	31
<b>XSG Injector Emulation IF Library .....</b>	<b>34</b>
Injector .....	35
Processor Output.....	35
Processor Input.....	37
Feedback Voltage .....	39
Processor Output.....	39
Processor Input.....	43
Load Control .....	45
Processor Output.....	45
Processor Input.....	47
<b>ControlDesk .....</b>	<b>48</b>
Injector Control Layout.....	49
Feedback Voltage Layout .....	52
Multiscope Layout .....	55

# About This Document

## Conventions Used in dSPACE User Documentation

### Symbols

dSPACE user documentation uses the following symbols:

Admonition	Description
 <b>DANGER</b>	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
 <b>WARNING</b>	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
 <b>CAUTION</b>	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
<b>NOTICE</b>	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	Indicates tips that can make your work easier.

### Naming conventions

dSPACE user documentation uses the following naming conventions:

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

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

# Introduction

---

<b>Objective</b>	This document deals with the dSPACE XSG Injector Emulation (IE) Interface Solution. The objective of this document is, to explain the main functionality of the dSPACE Injector Emulation, how to integrate this solution in other Simulink models and how to control the injector emulation in ControlDesk.
------------------	--

---

---

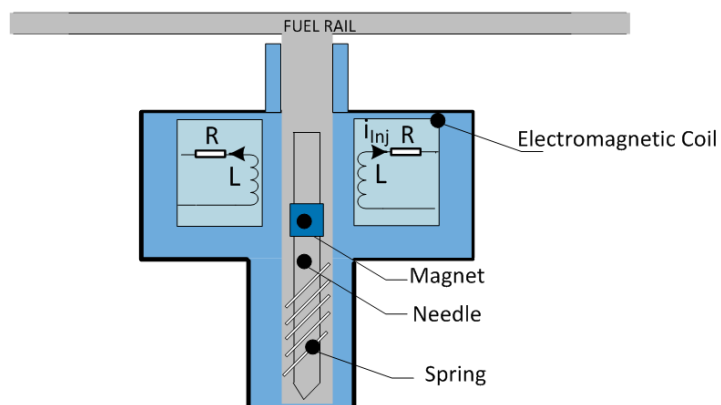
<b>Where to go from here</b>	Information in this section
------------------------------	-----------------------------

<b><i>Solenoid Injectors</i></b>	6
<b><i>Injector Emulation Features</i></b>	8

## Solenoid Injectors

### Basics

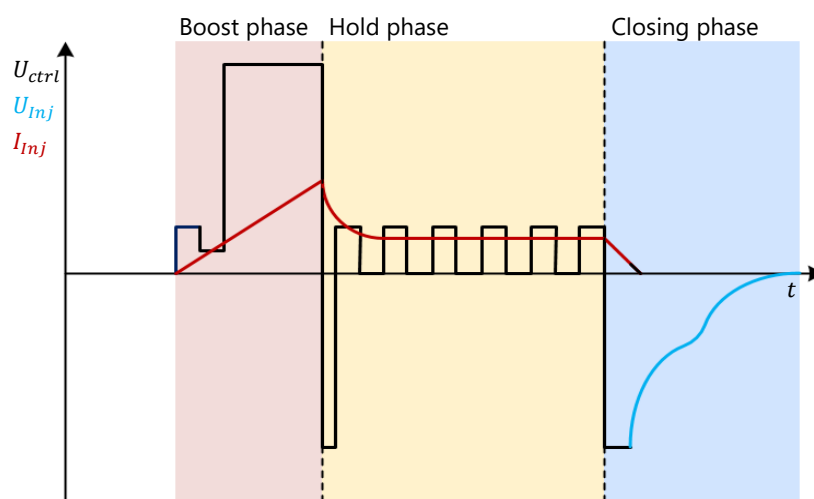
The following figure shows the structure of a solenoid injector. The injector valve is controlled by the engine ECU, by applying a voltage pulse pattern to the coil



circuit. The needle moves upwards and the valve opens. When the coil closes, it induces a negative voltage into the coil circuit – the so called feedback voltage. Due to an increasing demand of ecofriendly vehicles, an accurate measurement of the injection quantity is necessary. By evaluating the closing behavior of the injector valve, the ECU is able to control the fuel injection very precisely.

### Injector Signals

A typical injector control pattern of the ECU is depicted in the figure below.



The black signal in the upper plot shows the ECU control signal. The ECU starts to apply a high voltage pulse to the injector coil (boost phase), to open the injector valve. To hold the valve in an open state, the ECU applies a pulse pattern to the

coil (hold phase). The red line shows the resulting current in the boost and hold phase. The current increases rapidly in the boost phase. In the hold phase the current is controlled to a certain current level. The current decreases to Zero in the closing phase.

When the ECU stops controlling the valve, the needle induces a negative voltage (feedback voltage) into the coil circuit (closing phase). This voltage is shown by the blue signal in the upper plot of figure above. The signal shape of the induced voltage depends on the injection time and the rail pressure.

The dSPACE XSG Injector Emulation Solution is able to emulate the injector current (red) and the feedback voltage (blue).

# Injector Emulation Features

---

## Overview

The implementation of the dSPACE Injector Emulation system includes the following features:

- Emulation of the injector current flow (in the boost and hold phase).
  - Parameterizable injector coil circuit (resistance, inductance incl. saturation effects and hysteresis)
  - Injection time measurement.
- Emulation of the feedback voltage in the closing phase.
  - Interchangeable 3D look up table (LUT) for the closing behavior of the valve.
- Failure Insertion
  - Defective 3D-LUT for closing behavior
  - Ahead of time valve closing
  - Delayed valve closing
  - Ramp closing (instead of LUT)

All parameters can be changed online.



# Target Hardware

## Overview

Due to very short injection times, the injector emulation places great demands on the hardware setup. Therefore dSPACE has developed a special injector emulation board called EV1139 which is controlled via an FPGA with fast IO channels.

## Required Hardware

For an injector emulation SCLAXIO system with four injection channels, the following hardware is needed besides an SCALEXIO computational node (CN):

- 1x DS2655 FPGA Base Board
- 2x DS2655M1 Multi-IO Module (1 module for 2 injector emulation channels)
- 4x EV1139 Injector Emulation Board

### Tip

For a detailed description of the hardware, please refer to the specific hardware documentation of the boards.

## FPGA I/O

The communication between EV1139 and FPGA is realized by digital and analog FPGA I/O Channels. The table below shows the used I/O channels.

DS2655 M1 Module	I/O Channel	I/O Direction	Function
1	DIG_IO1	Out	Enable Injector Ch. 1
1	DIG_IO2	In	Status Injector Ch. 1
1	DIG_IO3	Out	Switch high impedance state of Injector Ch. 1
1	DIG_IO5	Out	Enable Injector Ch. 2
1	DIG_IO6	In	Status Injector Ch. 2
1	DIG_IO7	Out	Switch high impedance state of Injector Ch. 2
1	ADC1	In	Voltage measurement Injector Ch. 1
1	ADC2	In	Current measurement Injector Ch. 1
1	ADC3	In	Voltage measurement Injector Ch. 2
1	ADC4	In	Current measurement Injector Ch. 2
1	DAC1	Out	Control value (+) Injector Ch. 1
1	DAC2	Out	Control value (-) Injector Ch. 1
1	DAC3	Out	Control value (+) Injector Ch. 2
1	DAC4	Out	Control value (-) Injector Ch. 2

2	DIG_IO1	Out	Enable Injector Ch. 3
2	DIG_IO2	In	Status Injector Ch. 3
2	DIG_IO3	Out	Switch high impedance state of Injector Ch. 3
2	DIG_IO5	Out	Enable Injector Ch. 4
2	DIG_IO6	In	Status Injector Ch. 4
2	DIG_IO7	Out	Switch high impedance state of Injector Ch. 4
2	ADC1	In	Voltage measurement Injector Ch. 3
2	ADC2	In	Current measurement Injector Ch. 3
2	ADC3	In	Voltage measurement Injector Ch. 4
2	ADC4	In	Current measurement Injector Ch. 4
2	DAC1	Out	Control value (+) Injector Ch. 3
2	DAC2	Out	Control value (-) Injector Ch. 3
2	DAC3	Out	Control value (+) Injector Ch. 4
2	DAC4	Out	Control value (-) Injector Ch. 4

# Simulation Model

---

## Overview

The simulation model implements a simplified physical injector model of the coil circuit, as well as the control for the injector emulation hardware. It is separated into the two following parts:

**Precompiled FPGA model:** Implements the physical model of the injectors and controls the emulation hardware. This model is precompiled and cannot be changed.

**Simulink interface model:** User interface for the FPGA model, to transmit parameters and control commands online. This model has to be included in your model, if you want to use the injector emulation solution.

The FPGA runs with 8 ns clock time. The FPGA I/O module measures the injection voltage signal from the ECU and emulates the corresponding injector current in the boost and hold phase. When the ECU stops controlling the injector, the system has to emulate the induced feedback voltage of the closing needle instead of the injector current (closing phase). Therefore a state machine with two different emulation states - current mode and voltage mode - is implemented on the FPGA.

The interface model communicates with the FPGA model via buffer access. All parameter for the injector emulation can be configured online in the processor model and will be passed to the FPGA.

---

## Where to go from here

Information in this section

<b><i>FPGA Model</i></b>	12
<b><i>Simulink Interface Model</i></b>	15
<b><i>Feedback Voltage Look Up Table</i></b>	20

## FPGA Model

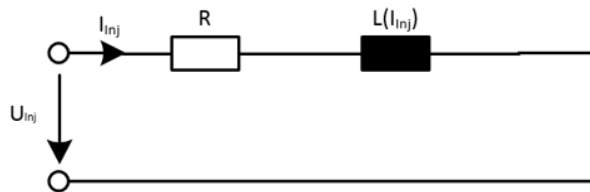
### Overview

The FPGA model contains a simplified physical model of the coil circuit to emulate the injector current and an LUT based implementation for the emulation of the feedback voltage. A simple state machine is implemented which switches automatically between current emulation and voltage emulation mode.

The FPGA code is precompiled and cannot be changed by the user. How to add the FPGA code to your project is described in section *How to Integrate the Injector Emulation Interface in an Existing Project* on Page 24.

### Current Mode

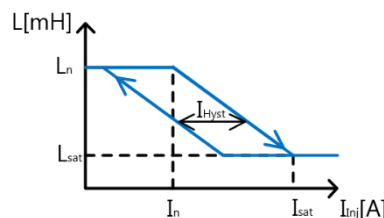
In the current mode the control voltage of the ECU  $U_{inj}$  is measured and the corresponding current  $I_{inj}$  is computed by means of the following equivalent circuit diagram for the injector coil.



The corresponding linear differential equation is

$$U_{inj}(t) = RI_{inj}(t) + L(I_{inj}(t)) \frac{dI_{inj}(t)}{dt}.$$

It is possible to parameterize the resistance  $R$  and the inductance  $L$  of the injector coil. Besides, a saturation effect of the inductance of the coil can be considered by decreasing the inductance  $L$  for higher injector currents. This leads to a steeper slope of the current curve for higher currents in the boost phase. The following figure shows the inductance  $L$  over the injector current  $I_{inj}$ .



Additionally a hysteresis effect is considered in the current model. The width of the hysteresis  $I_{Hyst}$  can be parameterized online.

The following relations apply to the implementation:

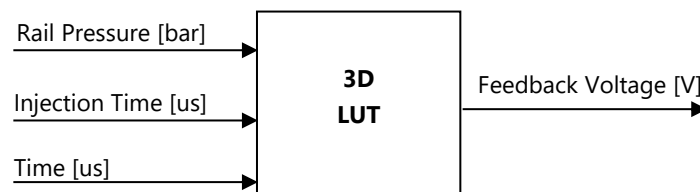
$$I_{Hyst} \leq I_n \leq I_{sat}$$

and

$$L_n \geq L_{sat}$$

### Voltage Mode

In the voltage mode the Injector Emulation system emulates the closing behavior of the injector valve. Therefore a feedback voltage of the injector coil is emulated after the ECU stops controlling the injector. The voltage curve depends on the rail pressure and the injection time and is implemented as a 3D look up table (LUT).



The user is able to change the LUT data online. The LUT has to be in a special format which is described in section *Feedback Voltage Look Up Table* on page 20.

There are implemented two LUTs which are switchable online (e.g. one with correct feedback voltage and one with a fault behavior) to simulate injector faults. Besides, there are implemented some more injector fault simulations. You can shift the LUT to the left or right, to simulate ahead or delayed closing of the valve or you can replace the LUT value with a simple ramp function.

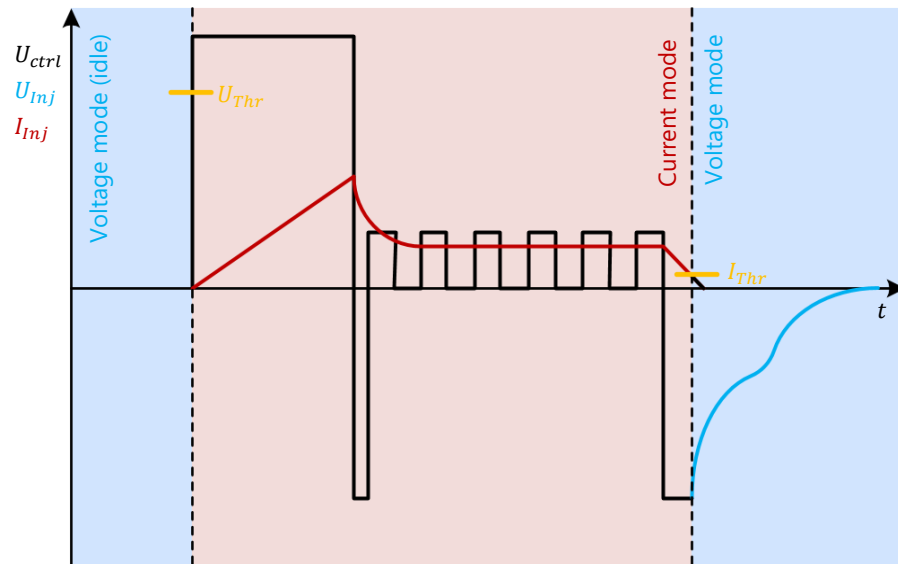
To save resources on the FPGA both LUTs are used for all injectors. There is no possibility to assign an individual LUT for each injector, whereas the injector faults can be switched individually for each injector.

#### Note

The feedback voltage will only be generated if the injector current exceeds a certain threshold in the previous boost phase. The threshold is parameterizable during runtime (default: 4 A) for each injector.

### Switching between Current mode and Voltage mode

As mentioned above, a state machine with two states is implemented, to switch between the current emulation mode and voltage emulation mode. The following figure illustrates the states related to the voltage and current signals.

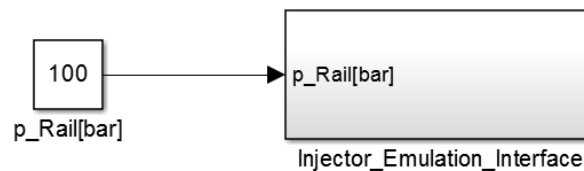


The idle state respectively default mode is the voltage mode. A state transition to the current mode is performed when the measured injector control voltage  $U_{ctrl}$  of the ECU exceeds the threshold  $U_{Thr}$  for a specific time (default 5 us). In this case, the ECU starts with the injection phase. The current mode switches to the voltage mode when the emulated current drops below the configurable current threshold  $I_{Thr}$  and the boost voltage is not active:  $U_{ctrl} < U_{Thr}$ . All thresholds are parameterizable during runtime.

## Simulink Interface Model

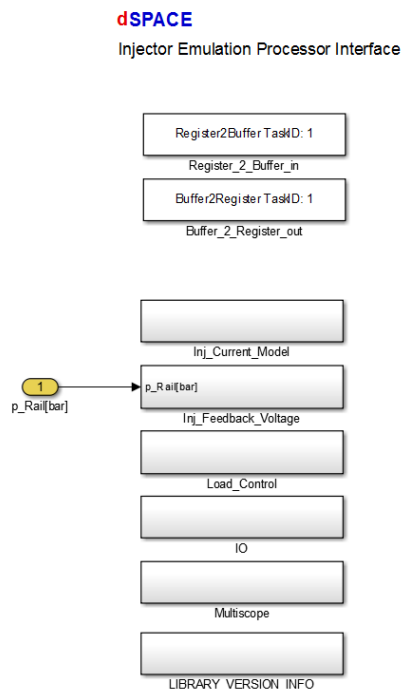
### Overview

The Simulink interface model is required for the data exchange with the precompiled FPGA model. All parameters will be passed to the FPGA model via buffer access. The corresponding Simulink interface model for the precompiled FPGA model is included in the Demo model which comes with the XSG IE Solution. Because of a fixed mapping of the contained registers for data exchange with the FPGA it is absolutely necessary to copy the Simulink interface from the demo model. The Simulink interface model contains the `Injector_Emulation_Interface` subsystem depicted in the figure below.



This subsystem should be copied to your existing Simulink project, as described in section *How to Integrate the Injector Emulation Interface in an Existing Project* on page 24.

The `Injector_Emulation_Interface` subsystem contains the following model:

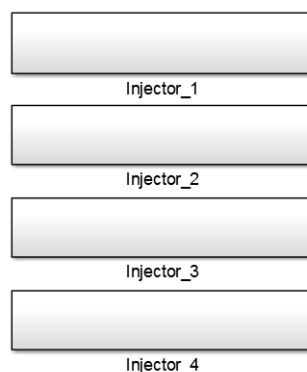


The Register\_2\_Buffer\_in and Buffer\_2\_Register\_out blocks are necessary for the connection to ConfigurationDesk and for routing the incoming and outgoing data to the corresponding subsystems.

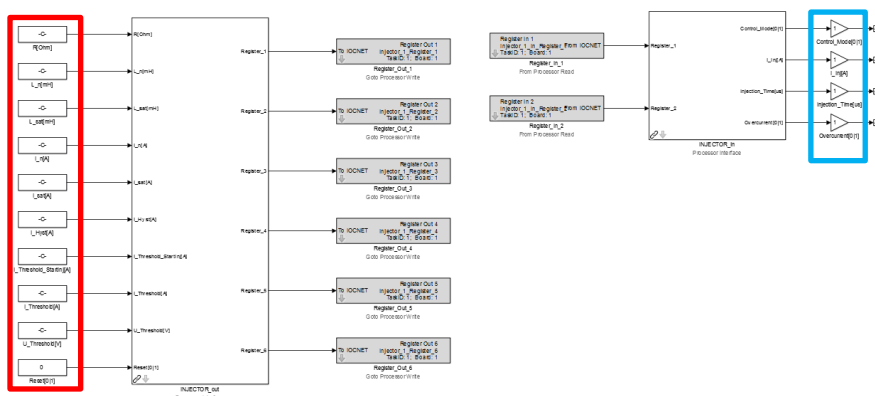
The most important subsystems for the control and parameterization of the injector emulation are the Inj\_Current\_Model, the Inj\_Feedback\_Voltage and the Load\_Control subsystem. These subsystems are explained in details below. All other subsystem are needed for the functionality of the system but they should not be changed.

### Inj\_Current\_Model Subsystem

The Inj\_Current\_Model subsystem contains all parameters for the injectors which are necessary for the current emulation. The content of this subsystem is shown in the next figure.



There is one subsystem for each injector, so the injectors can be parameterized independently. The next figure shows the content of one of these subsystems. The parameters inside the four identical subsystem can be found in constant blocks on the left side (red frame). The feedback values from the FPGA can be accessed on the right side (blue frame) as depicted in the figure below.

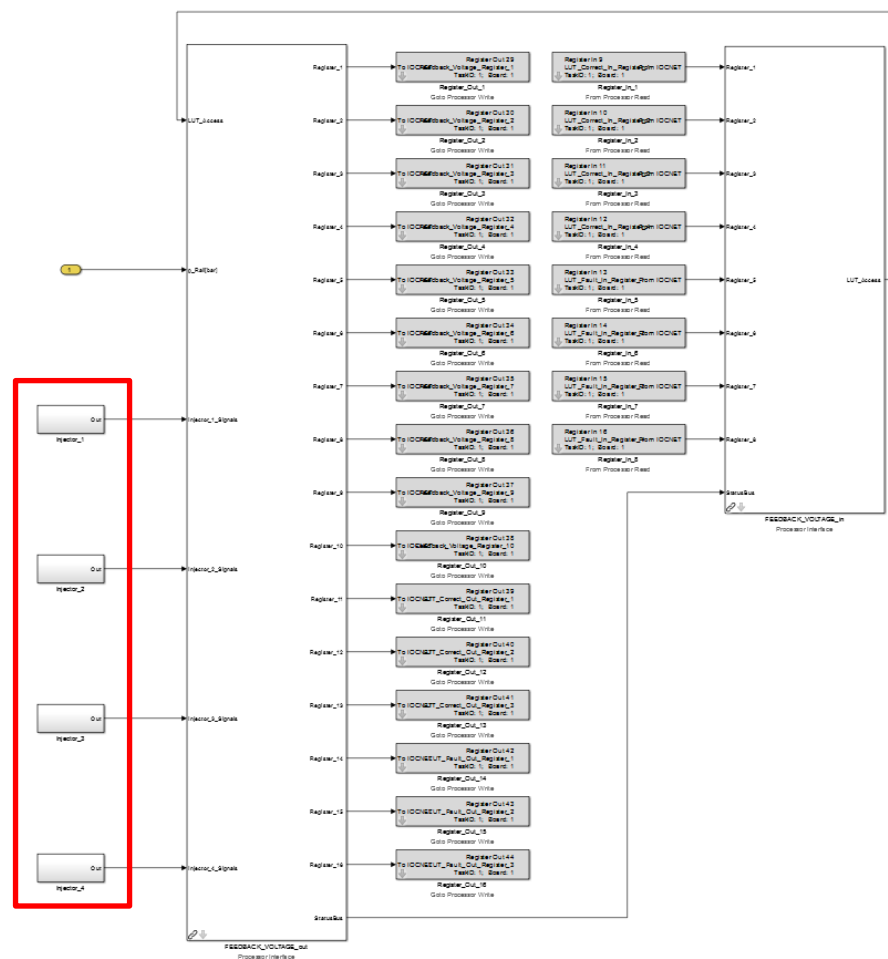


The mapping to and from the data exchange registers is done by the INJECTOR\_out and INJECTOR\_in blocks. These blocks are part of the XSG IE Interface Library. All inputs, outputs and mask parameters are explained in chapter *Injector* on page 34.



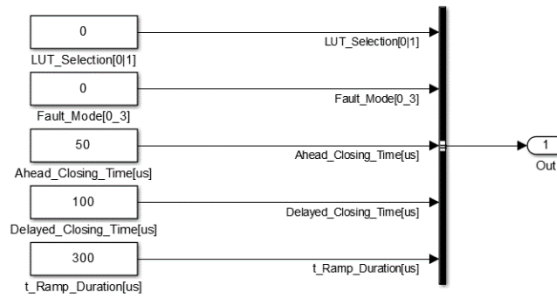
### Inj\_Feedback\_Voltage Subsystem

In the `Inj_Feedback_Voltage` subsystem you can set the feedback voltage LUT data as well as failure simulation settings for each cylinder. The following figure shows the content of the subsystem.



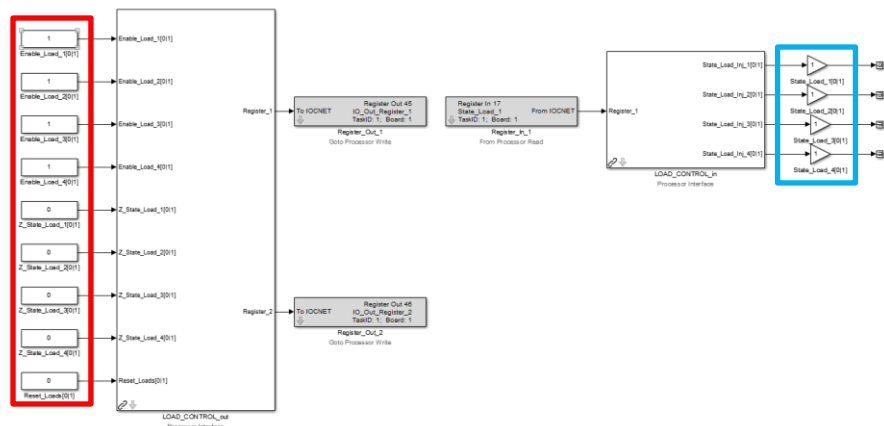
The mapping to and from the data exchange registers is done by the `FEEDBACK_VOLTAGE_out` and `FEEDBACK_VOLTAGE_in` blocks. In the mask dialog of the `FEEDBACK_VOLTAGE_out` subsystem you can enter the LUT data vectors/matrices. This block is part of the XSG IE Interface Library and is described in chapter *Feedback Voltage* on page 39.

The subsystems on the left (see red frame) contain constant blocks, where the feedback voltage can be manipulated for each cylinder (see figure below).



### Load\_Control Subsystem

The Load\_Control subsystem summarizes control parameters for the EV1139 injector emulation hardware. The content is shown in the next figure.



On the left side (red frame) you can enable/disable each load board or set it into a high impedance state (z-State). On the right side (blue frame) the states of the loads are available.

The mapping to and from the data exchange registers is done by the LOAD\_CONTROL\_out and LOAD\_CONTROL\_in blocks. These blocks are part of the XSG IE Interface Library. All inputs, outputs and mask parameters are explained in chapter *Load Control* on page 45.

### IO Subsystem

The IO subsystem contains an interface for the scaling of the FPGA I/O and some input filters. It is preconfigured for the connection to the load board.

**NOTICE****Fixed I/O scaling**

Do not change any settings inside this subsystem. Otherwise the communication between load board and FPGA will not work properly.

---

**Multiscope Subsystem**

The *Multiscope* subsystem contains the interface to the FPGA multi scope instrument. The multi scope is a measurement instrument for FPGA internal signals. It is part of the XSG Utils Library. All inputs, outputs and mask parameters are explained in the XSG Utils documentation. How to use the multi scope in ControlDesk is explained in section *Multiscope Layout* on page 55.

---

**Library\_Version\_Info Subsystem**

The *Library\_Version\_Info* block compares the FPGA code version with the processor interface. It is part of the XSG Utils Library. For further information, please refer to the XSG Utils documentation.

## Feedback Voltage Look Up Table

### Overview

This section defines the special fixed format for the implemented 3D feedback voltage look up table (LUT).

### Format Definition

The LUT has to be in the following format:

- LUT data:  $Y \times X \times Z = 1000 \times 21 \times 3$  values with index vectors:

LUT Dim.	Name	Vector Dim.	Unit	Default Vector
X	Injection Time	1 x 21	μs	[300, 485, ..., 3815, 4000]
Y	Time	1 x 1000	μs	[0, 3, 6, ..., 2994, 2997]
Z	Rail pressure	1 x 3	bar	[100, 250, 400]

- X, Y and Z index vectors have fixed length and must to be equidistant.
- The default vectors define minimum and maximum index vector values:

LUT Dim.	Range of minimum vector element	Range of maximum vector element
X	$X_{min} = 0 \dots 512 \mu s$	$X_{max} = X_{min} \dots 4096 \mu s$
Y	$Y_{min} = 0 \mu s$ [Fixed]	$Y_{max} = 2997 \mu s$
Z	$Z_{min} = 0 \dots 128 \text{ bar}$	$Z_{max} = Z_{min} \dots 512 \text{ bar}$

- Y (Time) index vector has to start with 0.
- LUT data has to end with 0 value in Y dimension.
- LUT data has to be normalized in range [0 ... 1]. As normalization factor, please use negative freewheeling voltage of the ECU (usually between -60V and -70V).

How to create a LUT in this format is described in section *How to Create a Feedback Voltage LUT* on page 31.

The LUT data and index vectors have to be entered into the block dialog of the `FEEDBACK_VOLTAGE_out` block (see *Feedback Voltage*, page 39), before building the model code. The LUT data will be transmitted to the FPGA model on model start-up. The transmission may take some time. Between the LUT data points a linear interpolation is preformed online.

#### Note

The default LUT data and index vectors contains only dummy data, to define minimum and maximum values of the LUT.

# Register Mappings

## Overview

For the data exchange between the Processor Interface and the FPGA model the XSG Utils Buffer Interface is used. Due to the precompiled FPGA model the mapping of the registers for the data exchange is fixed.

## Processor Output Registers

The following table shows the output registers which are used in the processor model.

Injector IF	Part of Solution	Used Registers	Register Out No.:
Injector 1	XSG IE	6	1-6
Injector 2	XSG IE	6	8-13
Injector 3	XSG IE	6	15-20
Injector 4	XSG IE	6	22-27
Feedback Voltage	XSG IE	16	29-44
Load Control	XSG IE	2	45-46
Library Version Info	XSG Utils	1	7
Scale DAC 1	XSG Utils	1	47
Scale DAC 2	XSG Utils	1	48
Scale DAC 3	XSG Utils	1	49
Scale DAC 4	XSG Utils	1	50
Scale ADC U_Inj 1	XSG Utils	1	51
Scale ADC U_Inj 2	XSG Utils	1	52
Scale ADC U_Inj 3	XSG Utils	1	53
Scale ADC U_Inj 4	XSG Utils	1	54
Scale ADC I_Inj 1	XSG Utils	1	55
Scale ADC I_Inj 2	XSG Utils	1	56
Scale ADC I_Inj 3	XSG Utils	1	57
Scale ADC I_Inj 4	XSG Utils	1	58
Discrete PT1 U_Inj 1	XSG Utils	1	59
Discrete PT1 U_Inj 2	XSG Utils	1	60
Discrete PT1 U_Inj 3	XSG Utils	1	61
Discrete PT1 U_Inj 4	XSG Utils	1	62
Discrete PT1 I_Inj 1	XSG Utils	1	63
Discrete PT1 I_Inj 2	XSG Utils	1	64
Discrete PT1 I_Inj 3	XSG Utils	1	65
Discrete PT1 I_Inj 4	XSG Utils	1	66
Multiscope 1	XSG Utils	2	67-68
Multiscope 2	XSG Utils	2	69-70

## Processor Input Registers

The following table shows the input registers which are used in the processor model.

Injector IF	Part of Solution	Used Registers	Register Out No.:
Injector 1	XSG IE	2	1-2
Injector 2	XSG IE	2	3-4
Injector 3	XSG IE	2	5-6
Injector 4	XSG IE	2	7-8
Feedback Voltage	XSG IE	8	9-16
Load Control	XSG IE	1	17
Library Version Info	XSG Utils	1	18
Scale DAC 1	XSG Utils	1	21
Scale DAC 2	XSG Utils	1	22
Scale DAC 3	XSG Utils	1	23
Scale DAC 4	XSG Utils	1	24
Scale ADC U_Inj 1	XSG Utils	1	25
Scale ADC U_Inj 2	XSG Utils	1	26
Scale ADC U_Inj 3	XSG Utils	1	27
Scale ADC U_Inj 4	XSG Utils	1	28
Scale ADC I_Inj 1	XSG Utils	1	29
Scale ADC I_Inj 2	XSG Utils	1	30
Scale ADC I_Inj 3	XSG Utils	1	31
Scale ADC I_Inj 4	XSG Utils	1	32
Discrete PT1 U_Inj 1	XSG Utils	1	19
Discrete PT1 U_Inj 2	XSG Utils	1	20
Discrete PT1 U_Inj 3	XSG Utils	1	51
Discrete PT1 U_Inj 4	XSG Utils	1	52
Discrete PT1 I_Inj 1	XSG Utils	1	53
Discrete PT1 I_Inj 2	XSG Utils	1	54
Discrete PT1 I_Inj 3	XSG Utils	1	55
Discrete PT1 I_Inj 4	XSG Utils	1	56
Multiscope 1	XSG Utils	2	33-41
Multiscope 2	XSG Utils	2	42-50

# Quick Start Guide

---

<b>Objective</b>	The quick start guide helps you to integrate the injector emulation software into your existing project. It is recommended to read this chapter before using the injector emulation.
------------------	--

---

---

<b>Where to go from here</b>	Information in this section
------------------------------	-----------------------------

---

<b>How to Integrate the Injector Emulation Interface in an Existing Project</b>	24
<b>How to Create a Feedback Voltage LUT</b>	31

# How to Integrate the Injector Emulation Interface in an Existing Project

## Overview

This section shows you, how to integrate the injector emulation interface in an existing SCALEXIO project.

### NOTICE

#### Demo application is required

Due to a fixed mapping of the contained registers for data exchange with the FPGA, it is absolutely necessary to start with the Simulink Demo model which comes with the XSG IE Interface Solution.

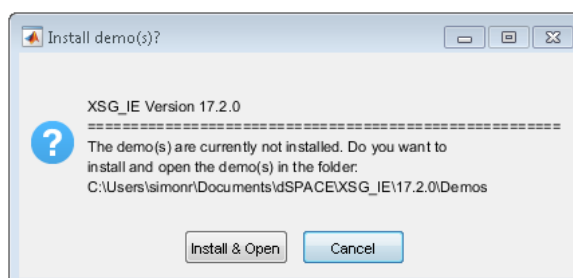
## Precondition

You have an existing SCALEXIO project with a ConfigurationDesk project and a Simulink model, where you want to integrate the injection emulation interface. The following steps assume that you have installed the XSG Injection Emulation Solution properly.

## Method Simulink

### To integrate the injector emulation interface in your existing model ...

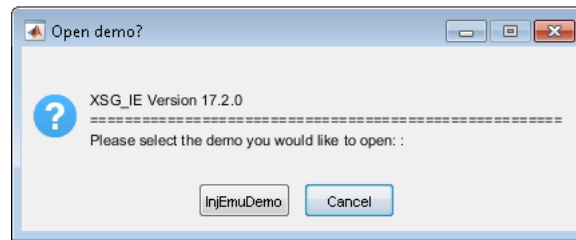
- 1 Open MATLAB.
- 2 Open your existing Simulink model, where the injector emulation interface shall be integrated.
- 3 Enter `XSG_IEInterface_lib` in the MATLAB Command Window to open the XSG Injector Emulation Interface Library.
- 4 Double-click the Demo button.
- 5 The following dialog appears:



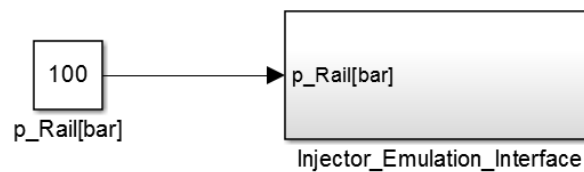
Click **Install & Open** to copy the demo project in your local dSPACE folder.

- 6 In the next dialog click **InjEmuDemo** to open the demo project.



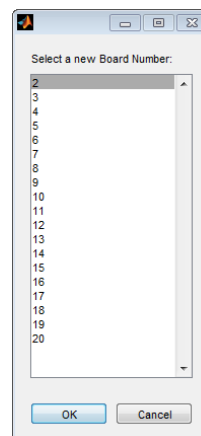


- 7 The MATLAB path switches to the demo folder and the demo model opens.



To get more information about the structure of the demo model, please refer to *Simulink Interface Model* on page 15.

- 8 Copy the *Injector\_Emulation\_Interface* subsystem and paste it into your existing Simulink model.
- 9 If you already using another FPGA interface in your model, the following Dialog appears. Please choose a new board number for the copied interface.



- 10 Copy the following files from the demo directory to your Simulink project directory:

```
%InjEmuDemo_Path%/IniFiles/XSG_IE_ini.m
```

```
%InjEmuDemo_Path%/IniFiles/FeedbackVoltageLUT3D_dSPACE_Dummy.mat
```

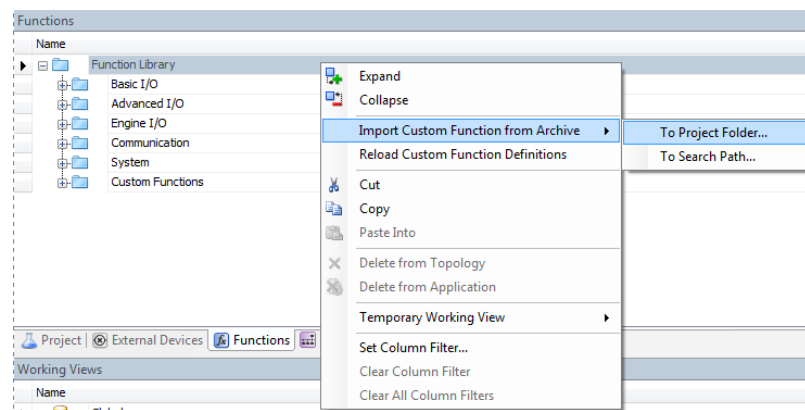
The *XSG\_IE\_ini.m* script contains initial values for the parameters of the injector emulation. This script has to be called before compiling the model, to write the necessary parameters to the MATLAB workspace. The *FeedbackVoltageLUT3D\_dSPACE\_Dummy.mat* file contains the LUT data for the feedback voltage.

- 11 Make sure that the `XSG_IE_ini.m` script will be called automatically when opening your Simulink model. This is necessary to initialize the `XSG_IE` structure in the MATLAB workspace which is needed by the injector emulation interface. You can use Simulink callbacks or you can add it to your project specific `go.m` script.
- 12 Connect the input port `p_Rail[bar]` of the Injector Emulation Interface to the calculated rail pressure in your model.

#### Method ConfigurationDesk

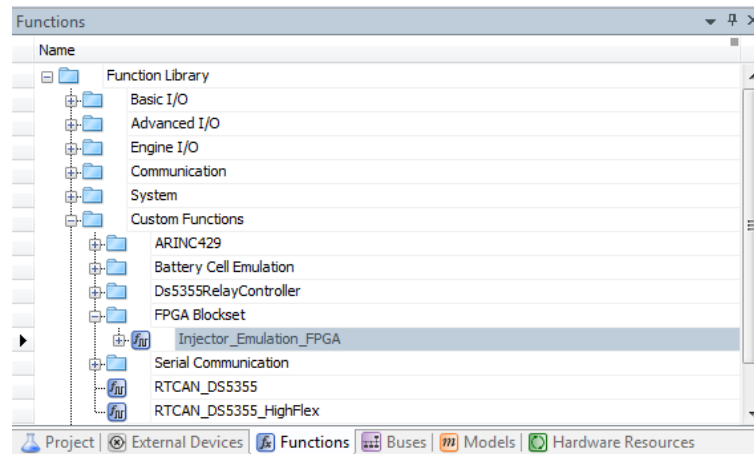
#### To add the precompiled FPGA code to your existing ConfigurationDesk project ...

- 1 Open ConfigurationDesk and load your ConfigurationDesk project which belongs to your existing Simulink model.
- 2 Right-click the Function Library and click Import Custom Function from Archive → To Project Folder ...

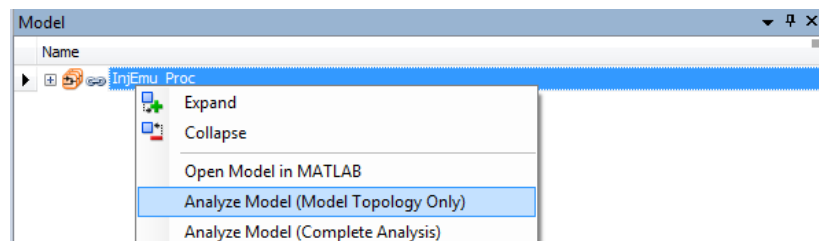


- 3 Change the file type to FPGA model INI files (\*.ini) and select the precompiled FPGA code `XSG_IE_FPGA_V01.ini` located in the installation directory of the XSG IE IF Solution. (Default installation path: `C:\dSPACE\XSG_IE_Solution 17.2.0\FPGA_Configuration`). Click Open.

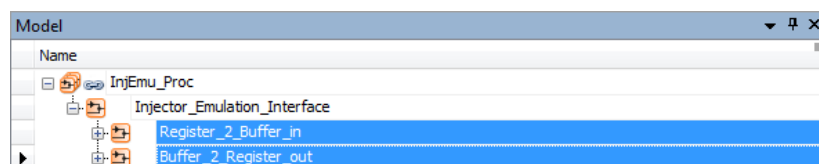
- 4 A new FPGA Custom IO Function appears in the function library.



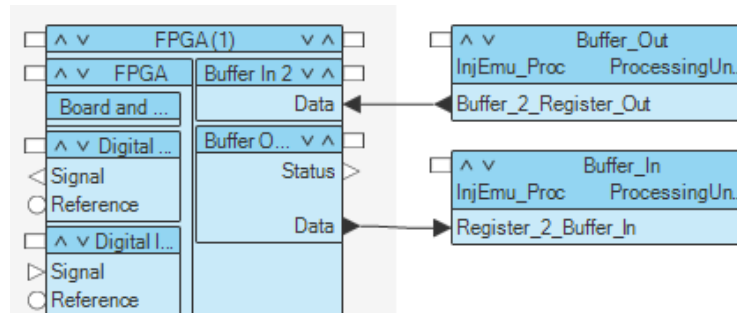
- 5 Add the Injector\_Emulation\_FPGA function to your application via drag and drop.
- 6 Assign a hardware channel (DS2655 + 2x M1 modules) in the properties of the FPGA function.
- 7 In the Model Window, right-click your existing model and select Analyze Model (Model Topology Only)



- 8 Locate the Injector\_Emulation\_Interface subsystem in the tree view of the Model window and drag and drop the Register\_2\_Buffer\_in and the Buffer\_2\_Register\_out model ports into your application.



- 9 Connect the model ports to the FPGA IO Function as shown below.

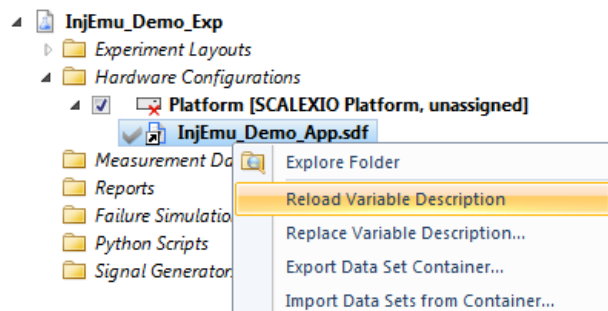


- 10 Build the model.

#### Method ControlDesk

#### To add Injector Emulation specific layouts to your existing ControlDesk project ...

- 1 Start ControlDesk.
- 2 Open your existing ControlDesk project.
- 3 In the Project explorer right-click on your variable description file (sdf-File) and select Reload Variable Description.

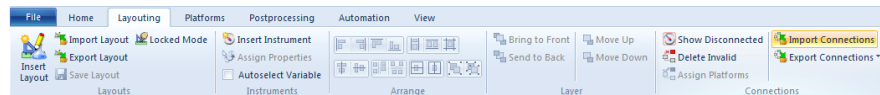


- 4 Right-click on the Experiment Layout folder and click Import Layout. Select all three Layouts (\*.lax) located in the installation directory (Default installation path: C:\dSPACE\XSG\_IE\_Solution 17.2.0\Layouts) and import them.
- 5 Open the connection file (Variable\_Connections.conx) located in the same directory as in step 4 with a text editor of your choice to adapt the variable paths according to your Simulink model.
- 6 Perform the following search and replace action:

Search string: //Model Root/Injector\_Emulation\_Interface/  
 Replace sting: //Model  
 Root/%PATH\_TO\_INTERFACE%/Injector\_Emulation\_Interface/

%PATH\_TO\_INTERFACE% is a placeholder for the location of the Injector\_Emulation\_Interface subsystem in your model, which you have copied in the section above.

- 7 Save the changes and close the file.
- 8 In the Layouting navigation pane, click **Import Connections** and select your modified connection file.

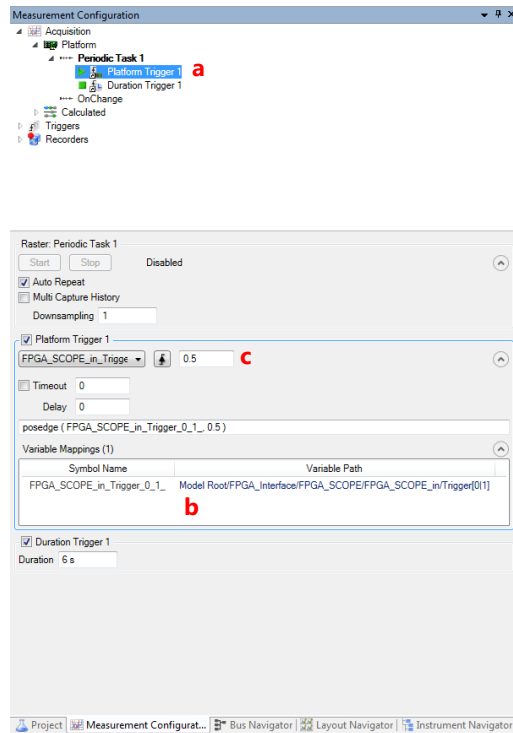


- 9 All instruments should now be connected to the corresponding variables.

#### Note

If the instruments are not connected yet, check the platform name of your platform in ControlDesk. If the name is not **Platform** reopen the connection file (as in step 5) and search and replace all occurrences of **Platform** with the name of your platform. Save the file and try to import the modified connection file again as described in step 8.

- 10 To ensure a proper triggering of the FPGA Multiscope, connect the signal `FPGA_Interface/FPGA_SCOPE/FPGA_SCOPE_in/Trigger[0|1]` to the Platform Trigger of your application.
  - a. Select the Platform trigger of the 1ms Task (default name is: Periodic Task 1)
  - b. Drag and drop the trigger signal from the Variable browser to the Variable Mappings area in the Measurement Configurations tab.
  - c. Select the signal in the drop-down menu and set trigger to rising edge and the threshold to 0.5.



## Result

You integrated the injector emulation interface in your existing Simulink, ConfigurationDesk and ControlDesk projects. Now you are able to control and parameterize the injector hardware.

# How to Create a Feedback Voltage LUT

## Overview

This chapter explains, how to create a feedback voltage LUT which can be used for the injector emulation solution.

## Precondition

You read the chapter *FPGA Model* where the LUT format is defined and you have recorded measurement data of the feedback voltage of a real injector for different rail pressures and injection times.

In the following it is assumed, that your measurement data has the following format and is available in the MATLAB Workspace:

Matlab Variable Name	Dimension	Unit	Content / Data
InjectionTimes	1 x K	µs	All injection times of the measurement in increasing order.
RailPressures	1 x L	bar	All rail pressures of the measurement in increasing order.
Time	1 x M	µs	Time vector
MeasData	M x K x L	V	Matching measurement data for the upper index vectors. All voltage values have to be negative.

### Tip

If you only have one measurement, you can use scalars for `InjectionTimes` and `RailPressures` as well.

## Method

### To create a feedback voltage LUT for the Injector Emulation interface ...

- 1 Open MATLAB
- 2 Load your measurement data (as defined above) into the MATLAB workspace.
- 3 Call the `xsg_ie_createLUT` script with the following command:

```
xsg_ie_createLUT(Time, InjectionTimes, RailPressures,
MeasData, MinInjectionTime_out, MaxInjectionTime_out,
MinRailPressure_out, MinRailPressure_out)
```

- a. `MinInjectionTime_out` defines the minimum injection time value contained in the generated LUT.

- b. `MaxInjectionTime_out` defines the maximum injection time value contained in the generated LUT.
- c. `MinRailPressure_out` defines the minimum rail pressure value contained in the generated LUT.
- d. `MaxRailPressure_out` defines the maximum rail pressure value contained in the generated LUT.

- 4 If your input data has the defined format above the following text will appear in the command window:

```
Input data OK. Processing data ...
Normalized measurement data with factor: -65.7
Measurement data was interpolated successfully.
Write LUT3D_x_values, LUT3D_y_values, LUT3D_z_values,
LUT3D_data and LUT3D_norm_factor to MATLAB workspace.
Saved Feedback_Voltage_LUT.mat
Saved Feedback_Voltage_LUT.csv
```

- 5 Note down the normalization factor printed in the command window. (see previous step, blue Number).
- 6 The `Feedback_Voltage_LUT.mat` file and a `Feedback_Voltage_LUT.csv` file are saved in your current working directory. The mat-File contains the LUT data and the index vectors. The csv file can be used to update the LUT data in ControlDesk during online calibration.

#### Note

Please make sure to enter the generated LUT data vector and the normalization factor in the mask dialog of the `FEEDBACK_VOLTAGE_out` block (see page 39). If you use the initialization scripts which comes with the injector emulation interface demo, please go on with the following steps.

- 7 Copy the generated `Feedback_Voltage_LUT.mat` to your `Inifiles` directory located in your Simulink project directory.
- 8 To update the LUT in the initialization script, please open the `XSG_IE_ini.m` file which comes with the injector emulation interface demo and change the filename in line 90 to:

```
load('Feedback_Voltage_LUT.mat');
```

- 9 Enter the new normalization factor (you noted down in step 5) in the `init.` script in line 120:

```
XSG_IE.FeedbackVoltage.normFactor.v =
%your_new_factor%
```

- 10 Run the `XSG_IE_ini.m` script or rerun the `go.m` script.



---

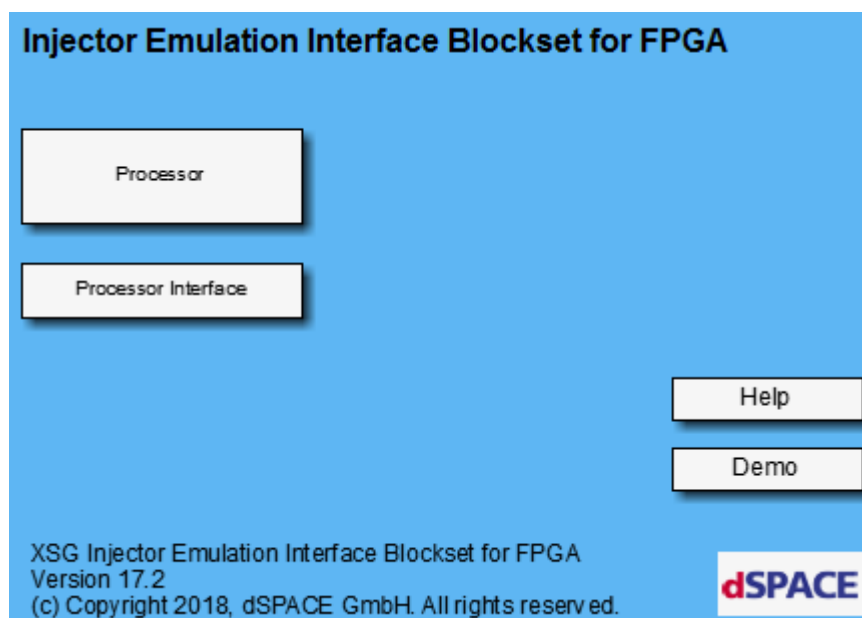
**Result**

You generated a LUT for the Injector Emulation interface.

# XSG Injector Emulation IF Library

## Objective

This chapter describes all available blocks of the XSG Injector Emulation (IE) IF library. You can open the library by typing `xsg_IEInterface_lib` to the command window in MATLAB. The library is shown in the figure below.



The processor interface subsystem contains all XSG interface blocks which are needed for the injector emulation. These blocks are explained in the following sections.

## Where to go from here

Information in this section

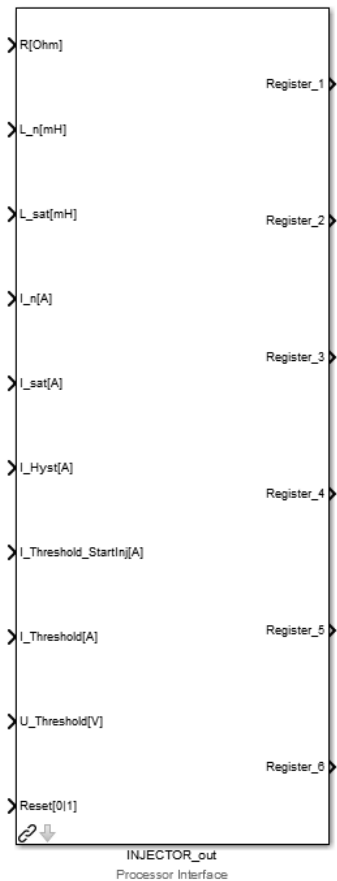
<b><i>Injector</i></b>	35
<b><i>Feedback Voltage</i></b>	39
<b><i>Load Control</i></b>	45

# Injector

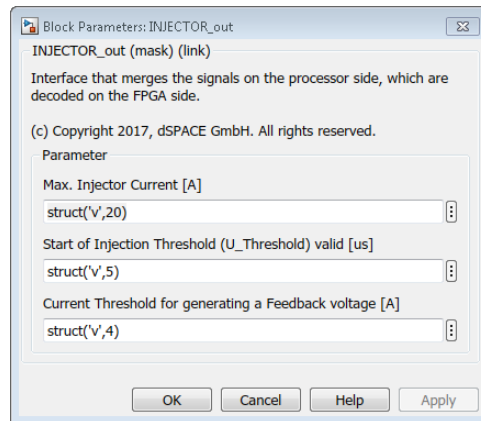
**Objective** The Injector interface blockset maps the injector specific parameters to the corresponding data registers for data exchange with the FPGA model. One Injector interface is needed for each implemented injector model on the FPGA side.

## Processor Output

**Block**



**Block Dialog** The processor output block contains the following dialog:



The dialog block has the following parameters:

Parameter	Unit	Description	Range
Max. Injector Current	[A]	Maximum allowed injector peak current. If the current exceeds this limit, the injector emulation board (EV1139) will be switched off.	0 ... 20 A
Start of Injection Threshold (U_Threshold) valid	[us]	To detect a valid "Start of injection" event, the measured injection signal of the connected ECU has to exceed the U_Threshold for this amount of microseconds. This is necessary to avoid false edge detection caused by cross talking effects.	0 .... 8 us
Current Threshold for generating a Feedback voltage	[A]	For generating a feedback voltage in the voltage phase, this threshold has to be exceeded once in the previous boost phase.	0 ... 15 A

## Input

The INJECTOR\_out block has the following inputs:

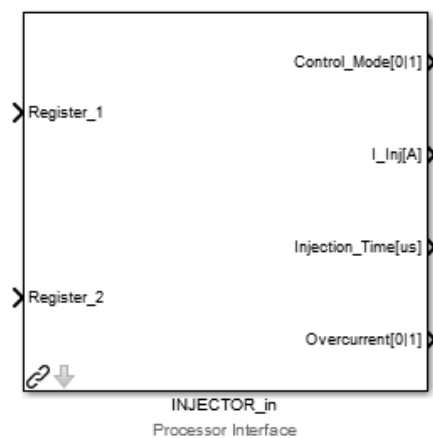
Input	Unit	Description	Range
R	[Ohm]	Resistance of the injector coil	0 ... 15 Ohm
L_n	[mH]	Nominal inductance of the injector coil.	0 ... 10 mH
L_sat	[mH]	Saturation inductance of the injector coil.	0 ... 10 mH
I_n	[A]	Nominal current of the injector coil.	0 ... I_sat A
I_sat	[A]	Saturation current of the injector coil.	I_n ... 20 A
I_Hyst	[A]	Width of Hysteresis	0 ... I_n A
I_Threshold_StartInj	[A]	Current threshold for start of injection time measurement	0 ... 20 A

Input	Unit	Description	Range
I_Threshold	[A]	Current threshold for switching to voltage emulation mode (on falling edge of current signal).	0 ... 20 A
U_Threshold	[V]	Voltage threshold for switching to current emulation mode (on rising edge of voltage signal)	0 ... 100 V
Reset	[0 1]	Reset injector (current) model	0 1

**Output**

The INJECTOR\_out block has 6 register outputs. Each of these outputs has to be connected to a 32 bit register of the RTI FPGA Library. Please refer to the chapter *Register Mappings* on page 21 for details of the mapping.

## Processor Input

**Block****Block Dialog**

The dialog contains only a short block description

**Input**

The INJECTOR\_in block has 2 register inputs. Each of these inputs has to be connected to a 32 bit register of the RTI FPGA Library. Please refer to the chapter *Register Mappings* on page 21 for details of the mapping.

**Output**

The INJECTOR\_in block has the following outputs:

Output	Unit	Description	Range
Control_Mode	[0 1]	Shows the state of the hardware control mode: [0] = Current emulation mode [1] = Voltage emulation mode	0 1
I_Inj	[A]	Actual injector current	0 ... 20 A
Injection_Time	[us]	Injection time of last injection cycle	0 ... 32768 us
Overcurrent	[0 1]	Indicates a detected overcurrent. The hardware is switches off in case of an overcurrent detection.	0 1

**Note**

Please note that these values will only be updated within the processor step size (1 ms). Especially for the control mode and the actual injector current this update time is not sufficient for an appropriate representation of the signal curve. Please use the FPGA Multi Scope to track these values.

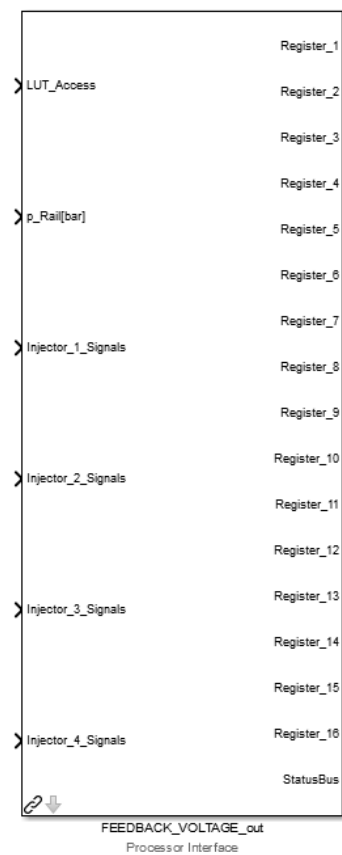
## Feedback Voltage

### Objective

The Feedback Voltage interface block set handles the data exchange of the LUT data with the FPGA model. Besides it maps injector specific failures simulation parameters to the data exchange registers. This interface blockset is used for all four implemented injectors.

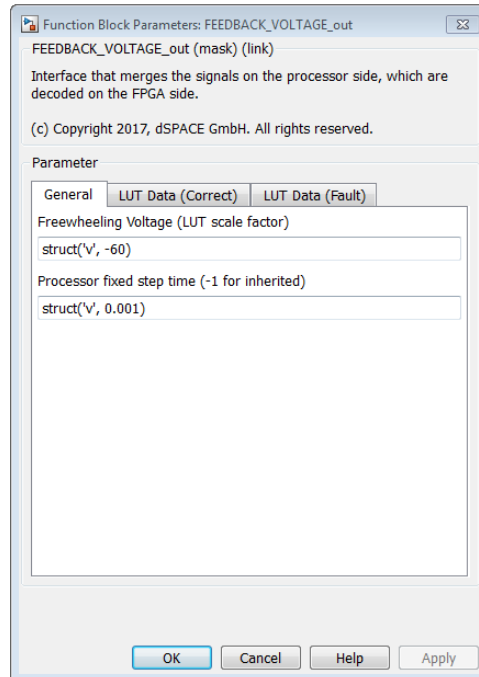
## Processor Output

### Block



### Block Dialog

The FEEDBACK\_VOLTAGE\_out block contains a dialog with three tabs. The General tab is shown below:

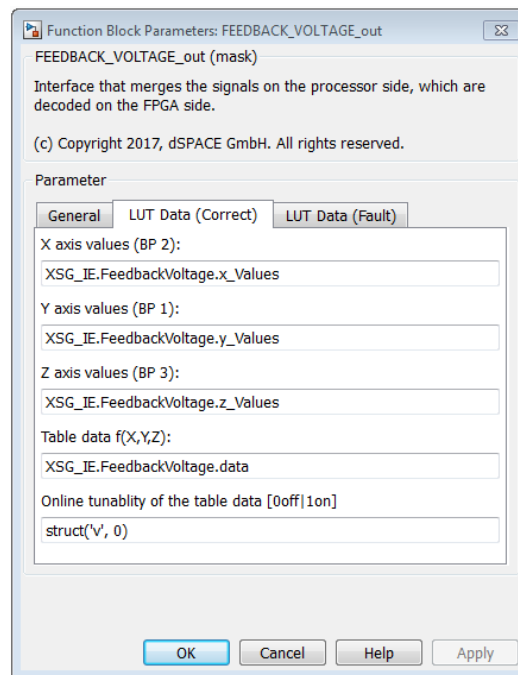


The parameter are explained in the following table:

Parameter	Unit	Description	Range
Freewheeling Voltage (LUT scale factor)	[V]	This value is the feedback voltage LUT scale factor. The normalized LUT output values (0 ... 1) are multiplied with this factor. This factor is ECU and injector specific and represents the negative freewheeling voltage of the injector circuit after the ECU stops controlling the injector current.	-70 ... 0 V
Processor fixed step size	[s]	Step size of the processor model	= Step size of model.



In the second and third tab (LUT Data Correct and LUT Data Fault) of the block dialog you can input your LUT data for the feedback voltage. It is possible to insert two different LUTs (e.g. one with correct behavior, one with fault behavior). You can switch between these LUTs online. The following figure shows one of the two similar tabs:



The parameters are explained in the following table:

Parameter	Unit	Description	Range
X axis values	1x21 [us]	Injection time values for x axis of feedback voltage LUT. The values have to be equidistant.	0 ... 4096 us
Y axis values	1x1000 [us]	Time values for y axis of feedback voltage LUT. The values have to be equidistant.	0 ... 2997 us
Z axis values	1x3 [bar]	Rail pressure values for z axis of feedback voltage LUT. The values have to be equidistant.	0 ... 512 bar
Table data	1000x21x3 [0...1]	Data values of feedback voltage LUT. The value have to be normalized to 0 ... 1.	0 ... 1
Online tunability of the table data	[0 1]	Enable/Disable online tunability of LUT data.	0 1

**Input**

The FEEDBACK\_VOLTAGE\_out block has the following inputs:

Input	Unit	Description	Range
LUT_Access	[-]	Internal Bus for LUT data exchange. This port has to be connected to the LUT_Access output port of the corresponding FEEDBACK_VOLTAGE_in block.	
p_Rail	[bar]	Actual rail pressure for feedback voltage LUT.	0 ... 512 bar
Injector_<X>_Signals	[-]	<X> = [1,2,3,4]; Bus signal for each injector including the following signals:	
- LUT_Selection	[0 1]	Selection of used feedback voltage LUT: [0] = Correct LUT [1] = Fault LUT	0 1
- Fault_Mode	[0_3]	Selection of the fault mode: [0] = No fault. Normal feedback voltage output. [1] = Ahead valve closing. Feedback voltage output starts earlier. [2] = Delayed valve closing. Feedback voltage output starts later. [3] = Ramp closing. Feedback voltage LUT is replaced by a ramp.	0 ... 3
- Ahead_Closing_Time	[us]	Ahead closing time for fault mode 1. The feedback voltage LUT is shifted to the left (early).	0 ... 2000 us
- Delayed_Closing_Time	[us]	Delayed closing time for fault mode 2. The feedback voltage LUT is shifted to the right (late).	0 ... 2000 us
- T_Ramp_Duration	[us]	Ramp duration for fault mode 3. The LUT values are replaced by a simple ramp function with this duration.	0 ... 500 us

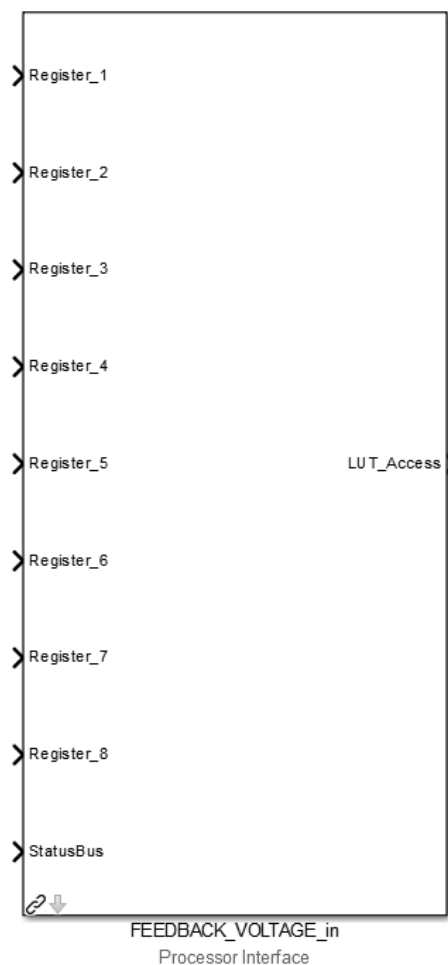
**Output**

The FEEDBACK\_VOLTAGE\_out block has 16 register outputs. Each of these outputs has to be connected to a 32 bit register of the RTI FPGA Library. Please refer to the chapter *Register Mappings* on page 21 for details of the mapping.

The StatusBus contains information for the LUT data exchange with the FPGA. It has to be connected to the corresponding FEEDBACK\_VOLTAGE\_in block.

## Processor Input

### Block



### Block Dialog

The dialog contains only a short block description.

### Input

The FEEDBACK\_VOLTAGE\_in block has 8 register inputs. Each of these inputs has to be connected to a 32 bit register of the RTI FPGA Library.

The StatusBus input port has to be connected to the StatusBus output port of corresponding FEEDBACK\_VOLTAGE\_out block. Please refer to the chapter *Register Mappings* on page 21 for details of the mapping.

**Output**

The FEEDBACK\_VOLTAGE\_in block has the following outputs:

Output	Unit	Description	Range
LUT_Access	[-]	Internal Bus for LUT data exchange. This port has to be connected to the LUT_Access input port of the corresponding FEEDBACK_VOLTAGE_out block.	

# Load Control

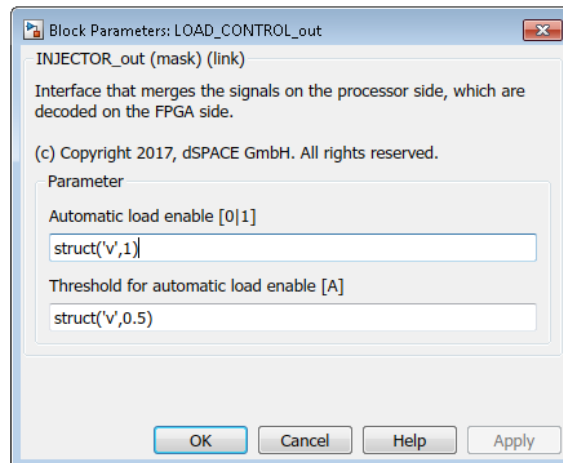
**Objective** The load control interface maps the user control signals to the corresponding data exchange registers of the FPGA. It is possible to enable, reset or activate a high impedance state for the injector emulation channels.

# Processor Output

**Block**



**Block Dialog** The processor output block contains the following dialog:



The parameter are explained in the following table:

Parameter	Unit	Description	Range
Automatic load enable	[0 1]	Automatic load enable mode: [0] = <i>Automatic load enable</i> disabled. The hardware will always be active. This might lead to cross talking problems. [1] = Enabled <i>Automatic load enable</i> . This setting is the preferred one to avoid cross talking effects. The emulation hardware will be only enabled (automatically) if an injection pulse was detected.	0 1
Threshold for automatic load enable	[A]	This threshold has to be reached by the simulated/calculated current to enable the hardware emulation board.	0 ... 4 A

#### Note

The parameters above are only included for stability reasons. Usually there is no need to change these parameters.

#### Input

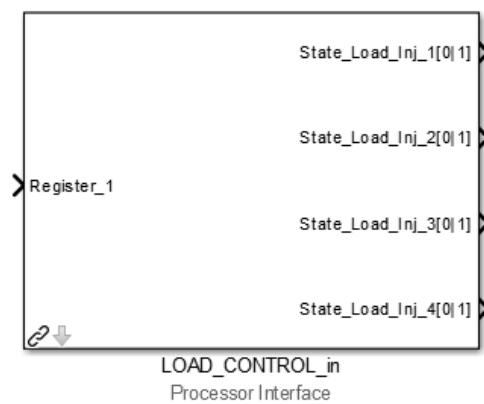
The LOAD\_CONTROL\_out block has the following inputs, with <X> = [1, 2, 3, 4].

Input	Unit	Description	Range
Enable_Load_<X>	[0 1]	Enable load emulation board.	0 1
Z_State_Load_<X>	[0 1]	Switch load board to high impedance (z-) state	0 1
Reset_Loads	[0 1]	Reset faults of all load emulation boards	0 1

**Output**

The LOAD\_CONTROL\_out block has 2 register outputs. Each of these outputs has to be connected to a 32 bit register of the RTI FPGA Library. . Please refer to the chapter *Register Mappings* on page 21 for details of the mapping.

## Processor Input

**Block****Block Dialog**

The dialog contains only a short block description.

**Input**

The LOAD\_CONTROL\_in block has 1 register input. Each of these inputs has to be connected to a 32 bit register of the RTI FPGA Library. Please refer to the chapter *Register Mappings* on page 21 for details of the mapping.

**Output**

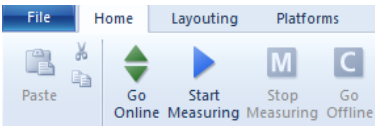
The LOAD\_CONTROL\_in block has the following outputs, with <X> = [1, 2, 3, 4].

Output	Unit	Description	Range
State_Load_Inj_<X>	[0 1]	Returns the state of the load hardware. [0] = Ok [1] = Fault	0 1

# ControlDesk

**Overview** This chapter contains a detailed description of the ControlDesk layouts. The ControlDesk section is especially intended for operators of the Injector Emulation system.

- How to start online calibration**      **To start working with the injector emulation system ...**
- 1 Start ControlDesk NG and load the Injector Emulation project.
  - 2 If not done yet, register the SCALEXIO system by clicking on the Register Platform button on the Platforms tab.
    - Select the SCALEXIO Platform on the left.
    - Enter the IP address of the Platform on the right and click Register.
  - 3 Go to the Home tab and start online calibration by clicking Go Online.



The application will be downloaded to the platform.

- 4 Start the measuring by clicking on the Start Measuring button, if you want to use the Multiscope layout.

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

<b><i>Injector Control Layout</i></b>	49
<b><i>Feedback Voltage Layout</i></b>	52
<b><i>Multiscope Layout</i></b>	55

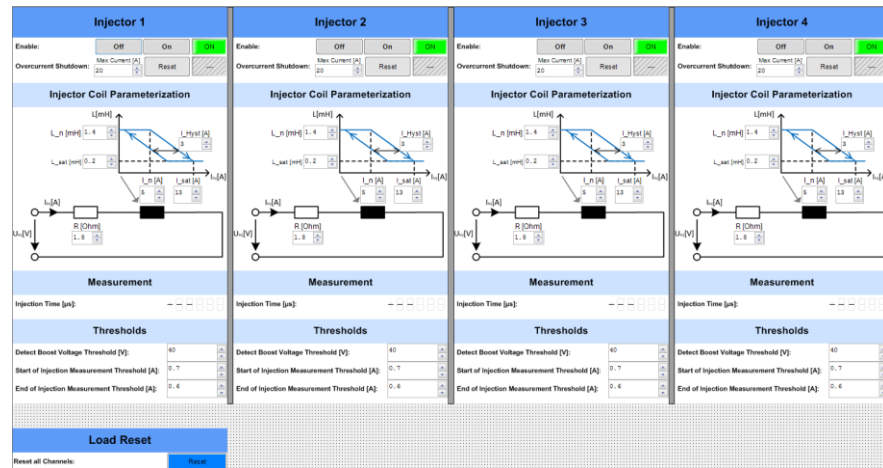


# Injector Control Layout

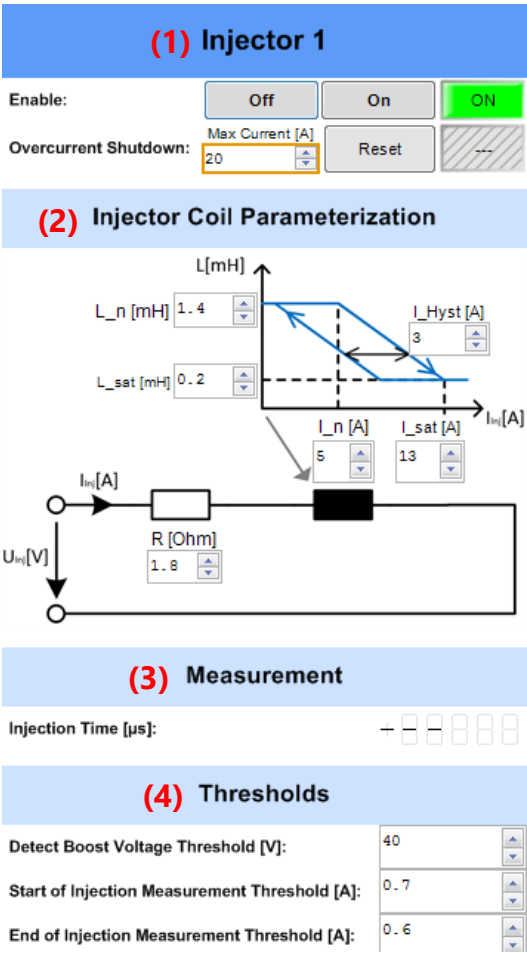
Name `Injector_Control.lay`

## Description

This layout represents the starting point for the operator. With the Injector Control layout the operator is able to enable the injector emulation as well as to parameterize the different injectors. The following figure shows the current version of the layout for four injectors.



At the bottom of the Layout, the user can reset the load hardware to clear all errors. The main part of the layout contains four equal columns for each injector. In this section the first column for the first injector will be discussed in details, which is shown in the next figure.



**General Controls (1)** The upper part of the layout contains some main control switches for the corresponding injector channel. The table below summarizes the functionality of these instruments.

Instrument/Parameter	Description
Enable	Enable <b>[On]</b> or disable <b>[Off]</b> the injector load emulation for the corresponding injector channel.
Overcurrent Shutdown	The model includes an overcurrent safety shutdown. If the emulated injector current exceeds the Limit specified in <b>Max Current [A]</b> , the injector emulation will be disabled. Press the <b>[Reset]</b> button to enable the injector emulation after an overcurrent shutdown.

**Injector Coil Parameterization (2)** In the Injector Coil Parameterization area the resistance and the inductance of the coil can be parameterized. To emulate saturation effects of the inductance, it is possible to set a nominal inductance  $L_n$  and and saturation inductance  $L_{sat}$ . If the emulated injector current is between 0 and  $I_n$ , the current model uses the

constant inductance  $L_n$ . When the current is between  $I_n$  and  $I_{sat}$ , the inductance of the coil decreases lineally from  $L_n$  to  $L_{sat}$ , as it is depicted in figure of the layout. An additional Hysteresis  $I_{Hyst}$  can be configured in the model. For more information of the implemented model, please refer to chapter *FPGA Model* on page 12. The following table summarizes the adjustable coil parameters.

Instrument/Parameter	Description
R [Ohm]	Resistance of the coil
$L_n$ [ $\mu$ H]	Nominal inductance of the coil.
$I_n$ [A]	Nominal current of the coil.
$L_{sat}$ [ $\mu$ H]	Saturation inductance of the coil.
$I_{sat}$ [A]	Saturation current of the coil.
$I_{Hyst}$ [A]	Hysteresis of the inductance.

#### Measurement (3)

The injection time is measured by the FPGA model and passed to the processor model. The start and the end of the measurement can be parameterized in the Threshold area, explained in the next section.

Instrument/Parameter	Description
Injection Time [ $\mu$ s]	Displays the measured injection time

#### Thresholds (4)

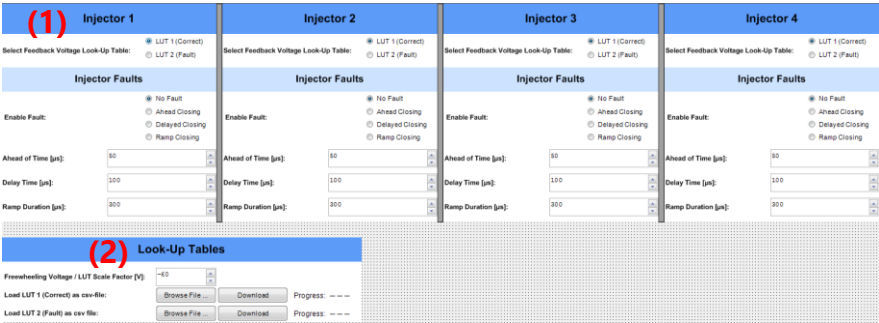
In this section of the layout the start threshold and the end threshold of the injection time measurement can be configured. The end threshold will also be used for the control mode state machine (see *FPGA Model*). If the model recognizes that the current drops below the end threshold, the injector emulation will switch from current to voltage emulation mode, to emulate the feedback voltage of the coil.

Instrument/Parameter	Description
Detect Boost Voltage Threshold [V]	If this voltage threshold is exceeded, the simulation model will switch from voltage to current emulation mode.
Start Injection Measurement Threshold [A]	If the emulated current reaches this threshold, the injection time measurement will be started.
End Injection Measurement Threshold [A]	If the emulated current drops below this threshold, the injection time measurement will be stopped and the simulation model will switch to the voltage mode.

# Feedback Voltage Layout

Name Feedback\_Voltage.lay

Description With this layout, the user is able to switch between two different feedback voltage LUTs as well as insert different failures of the feedback voltage for each emulated injector.



As for the previous control layout, this layout is also split up in four columns (1), one for each injector. In the following sections the first column for the first injector will be explained in details. After that, the Look-Up Table settings (2) are explained.

Injector Faults (1) In this area, you can switch between two different LUTs (Correct and Fault LUT). Besides you can insert some faults on the LUT.

Injector 1

Select Feedback Voltage Look-Up Table:

LUT 1 (Correct)

LUT 2 (Fault)

Injector Faults

No Fault

Ahead Closing

Delayed Closing

Ramp Closing

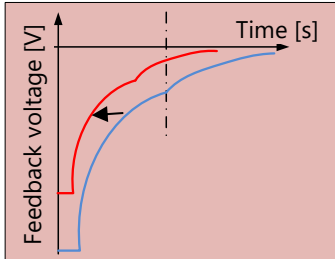
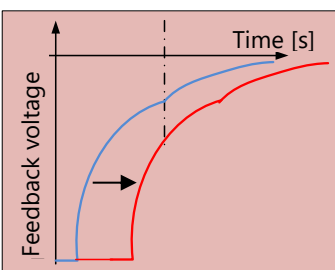
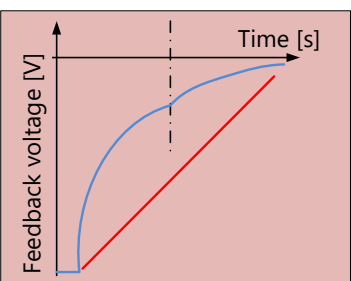
Enable Fault:

Ahead of Time [µs]:50

Delay Time [µs]:100

Ramp Duration [µs]:300

The following table summarizes the parameters.

Instrument/Parameter	Description
Select Feedback Voltage Look-Up Table	Switch between the two different LUTs.
Enable Fault	<p>Enable feedback voltage faults:</p> <p><b>[No Fault]:</b> Fault simulation disabled.</p> <p><b>[Ahead Closing]:</b> Enable ahead closing fault of the injector valve.</p>  <p><b>[Delayed Closing]:</b> Enable delayed closing fault of the injector valve.</p>  <p><b>[Ramp Closing]:</b> Enable ramp closing of the injector valve. The LUT will be replaced by a ramp signal.</p> 
Ahead of Time [us]	Set the ahead of time closing time.
Delay Time [μs]	Set the delay closing time.
Ramp Duration [μs]	Set the duration of the closing ramp.

**Look-Up Tabled (2)**

The user can upload two different look-up tables, to be able to switch between these LUTs online. The whole LUT data can also be exchanged online. To update the LUT data the generated csv-File with the LUT data is needed. How to generate this csv-File is explained in chapter *How to Create a Feedback Voltage LUT* on page 31.

Look-Up Tables

Freewheeling Voltage / LUT Scale Factor [V]:

Load LUT 1 (Correct) as csv-file:

Load LUT 2 (Fault) as csv file:

Browse File ...
Download

Browse File ...
Download

Progress: ---

Progress: ---

The following table summarizes the parameters of the figure above:

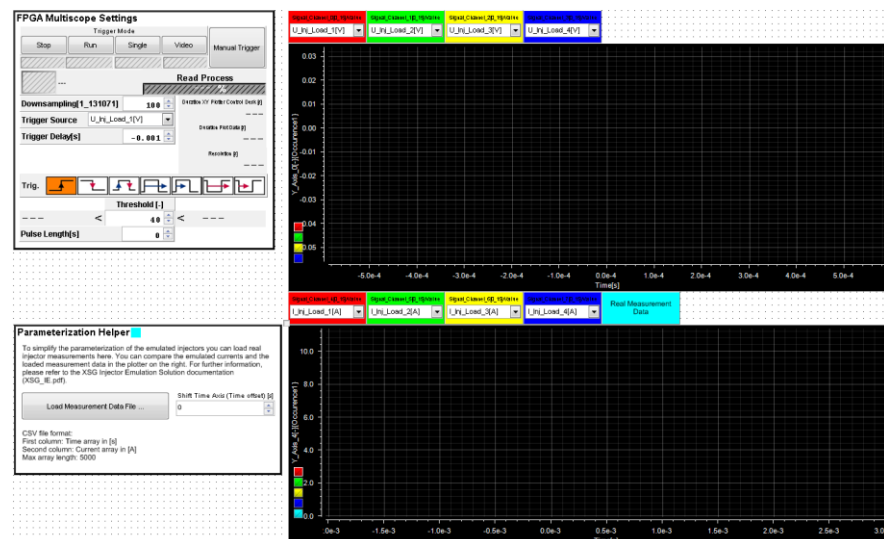
Instrument/Parameter	Description
Freewheeling Voltage / LUT Scale Factor [V]	All LUTs are normalized (Values [0 ... 1]) to the ECUs feedback voltage. The normalization factor has to be entered here. This factor has to be negative and represents the freewheeling voltage of the injector circuit.
Load LUT 1 (Correct) as csv file	You can change the data of LUT 1 (Correct) here. <b>[Browse File...]:</b> A dialog opens. You have to select the csv-File which was generated by the <code>xsg_ie_createLUT</code> script. <b>[Download]:</b> Click download, to transfer the new LUT data to the FPGA model.
Load LUT 2 (Fault) as csv file	You can change the data of LUT 2 (Fault) here. The options are explained in the row above.

# Multiscope Layout

**Name** Multiscope.lay

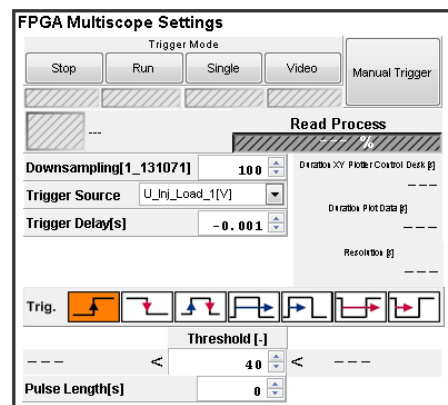
## Description

The multiscope layout grants access to selected internal FPGA signals. It basically works like an oscilloscope. With the colored drop-down menus above the two time plotters, the user can chose the signal, which shall be plotted. In addition a helper tool in implemented to simplify the parametrization of the emulated injectors. The handling of the plotters is explained in the next section.



## FPGA Multiscope Settings

To use this layout, the user has to start the measurement in ControlDesk first. The multiscope implementation on the FPGA measures the selected signals and passes the recorded data to the processor model. The FPGA Multi Scope has similar control options as an oscilloscope.



The following table explains the controls of the multiscopes settings.

Instrument/Parameter	Description
Trigger Mode	Select between different trigger modes: <b>[Stop]</b> : Trigger is off. No recording of new Data. <b>[Run]</b> : Trigger runs continuously (Auto mode). <b>[Single]</b> : Trigger is armed for a single shot. <b>[Video]</b> : Continuous measurement, without trigger. <b>[Manual Trigger]</b> : Start a single measurement manually.
Downsampling [1_131071]	The FPGA runs with a base sample rate of 8 ns. A down sampling rate for the measurement can be specified here. The resulting resolution and measurement duration will be displayed in the instruments on the right hand side.
Trigger Source	Select the source signal for the trigger.
Trig.	Select an edge type for triggering (rising, falling, both)
Trigger Delay [s]	Define a trigger delay. This value may also be negative.
Threshold	Define a trigger level for triggering the measurement.
Pulse Length [s]	Defines the pulse lengths, if pulse trigger mode is selected.

As mentioned in the section above, the user can use the colored drop-down menus to select different FPGA signals.



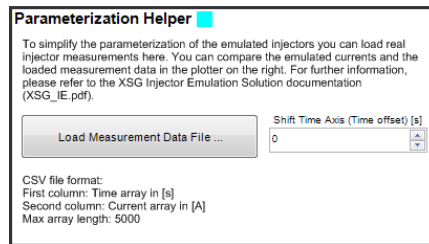
Currently the following signals can be measured, where <X> is a placeholder for the injector number 1-4.

Instrument/Parameter	Description
U_Inj_Load_<X>[V]	Voltage measurement of the specific injector channel.
I_Inj_Load_<X>[A]	Current measurement of the specific injector channel.
Feedbackvoltage_<X>[V]	Desired emulated feedback voltage of the specific injector.
I_Inj_<X>[A]	Desired emulated current of the specific injector.

#### Parameterization Helper

To simplify the parameterization of the emulated injector you can load a current measurement of a real injector to the real-time application. This signal can be displayed in the plotter together with the emulated injector current to compare both signals. The figure below shows the controls of the parameterization helper.





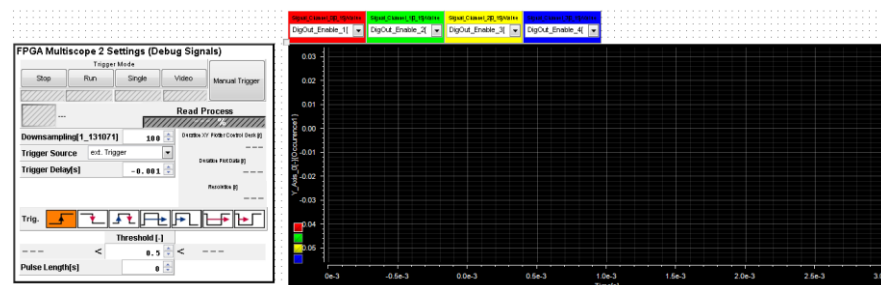
The following table explains the usage of the controls.

Instrument/Parameter	Description
Load Measurement Data File...	Loads a real measurement to the real-time application. On click, a file selection dialog opens. You have to select a csv file with the measurement data. The csv file has to be in the following format: <ul style="list-style-type: none"> <li>- Two columns with a maximum of 5000 values (rows).</li> <li>- First column: Time array in [s]. (Suggested value range: 0 ms to 3 ms)</li> <li>- Second column: Corresponding current in [A] (Value range usually between 0 A and 13 A)</li> </ul>
Shift Time Axis (Time offset) [s]	Shifts the real measurement to the left or right, to match the starting point of the real measurement with the emulated injector current.

The loaded measurement will be triggered automatically together with the multiscopes trigger. To enable the signal plotting you have to double click the cyan colored box at the y-axis of the plotter.

### FPGA Multiscopes (Debug Signals)

For debugging purposes you can find a second Multiscopes in the bottom of the layout.



Currently the following signals can be measured, where <X> is a placeholder for the injector number 1-4.

Instrument/Parameter	Description
DigOut_Enable_<X>[0/1]	Digital Output for enabling the EV1139 board.

Instrument/Parameter	Description
DigIn_Status_<X>[0 1]	Digital Output for EV1139 status feedback.
Control_Mode_<X>[0 1]	Control Mode of FPGA model (0=Current Emulation, 1= Voltage Emulation)
Fbd_Voltage_Req_<X>[0 1]	This signal indicates if the current in the boost phase exceeded a certain threshold. If this threshold (default: 4 A) is exceeded this signal has the value 1 and the feedback voltage will be generated after the current phase. The signal is reset when detecting the next boost pulse.