

# ETAS INCA-SIP V7.5



User Guide

## Copyright

The data in this document may not be altered or amended without special notification from ETAS GmbH. ETAS GmbH undertakes no further obligation in relation to this document. The software described in it can only be used if the customer is in possession of a general license agreement or single license. Using and copying is only allowed in concurrence with the specifications stipulated in the contract.

Under no circumstances may any part of this document be copied, reproduced, transmitted, stored in a retrieval system or translated into another language without the express written permission of ETAS GmbH.

© Copyright 2024 ETAS GmbH, Stuttgart

The names and designations used in this document are trademarks or brands belonging to the respective owners.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://mathworks.com/trademarks) for a list of additional trademarks.

INCA-SIP V7.5 | User Guide R01 EN | 03.2024

# Contents

1	Introduction .....	6
1.1	Intended Use .....	6
1.2	Target Group .....	6
1.3	Classification of Safety Messages .....	6
1.4	Safety Information .....	7
1.5	Data Protection .....	8
1.6	Data and Information Security .....	8
2	About INCA-SIP .....	9
2.1	Connecting the Simulink® Model to INCA .....	9
2.2	Measuring and Calibrating the Model .....	10
2.3	Technical Overview .....	10
2.3.1	Basic Operation .....	10
2.3.2	INCA-SIP MATLAB® Components .....	11
2.3.3	Connection MATLAB® Side .....	11
2.3.4	Connection INCA Side .....	11
2.3.5	Handling Measurements .....	12
2.3.6	Simulink® - INCA Synchronization .....	12
2.3.7	Handling Calibrations .....	13
3	Installation .....	14
3.1	System Requirements .....	14
3.1.1	Hardware Requirements .....	14
3.1.2	Software Requirements .....	14
3.2	Installing .....	14
3.2.1	Installing INCA-SIP V7.5 .....	14
3.2.2	Registering a C++ Compiler .....	14
3.2.3	Start Menu .....	15
3.2.4	Installation Path of INCA-SIP .....	15
3.2.5	MATLAB® Script Files .....	15
3.3	Licensing .....	15
4	Working with INCA-SIP .....	17
4.1	MATLAB® Search Path Modification .....	17
4.2	Switching between INCA Versions .....	17
4.3	Menu Items .....	17
4.3.1	Connect to INCA .....	17

4.3.2	ECUs .....	19
4.3.3	Configuration .....	23
4.3.4	MDF Configuration .....	24
4.4	Macros .....	27
4.5	Naming Conventions .....	28
4.5.1	Calibrations and Measurements .....	28
4.5.2	Groups .....	28
4.6	INCA-SIP Emulation Mode .....	28
4.6.1	Real-Time Emulation Mode .....	28
4.6.2	Fast Emulation Mode .....	29
4.7	INCA-SIP and Block Reduction .....	29
4.8	Simulation Modes .....	29
4.9	Referenced Models Support .....	30
4.10	Refusing Calibrations .....	30
4.11	Busy ECU .....	30
4.12	Starting and Stopping the Experiment .....	30
4.13	Simulink.Signal and Asap2.Signal Support .....	31
4.14	Simulink.Parameter and Asap2.Parameter Support .....	31
4.15	Error, Warning and Information Messages .....	31
4.16	DLL Mode .....	31
4.16.1	Creating the DLL .....	32
4.16.2	Creating a DLL Search Script .....	33
4.16.3	Creating Object Using DllAsap2ObjectFactory .....	35
4.17	Writing Dataset to MDF File .....	40
4.18	Using INCA Remotely .....	42
4.19	Custom Hooks .....	44
4.19.1	setVariableFilterFunctionHandle .....	44
4.19.2	setVariableCustomizationFunctionHandle .....	46
4.19.3	setVariableRenamingFunctionHandle .....	48
4.19.4	setGroupRenamingFunctionHandle .....	49
4.19.5	MScripts.VariableWrapper .....	51
4.19.6	MScripts.AccessInfo .....	56
4.20	Supported Simulink® Block Types .....	57
4.21	Restrictions and Other Behaviors .....	59
4.21.1	RAM Calibration .....	59
4.21.2	Parallel Measurements with Other Hardware .....	59

4.21.3	Pausing Measurements .....	59
5	Tutorial .....	60
6	Contact Information .....	62
	Index .....	63

# 1 Introduction

## 1.1 Intended Use

INCA and INCA add-ons are developed and approved for automotive applications and procedures as described in the user documentation for INCA and INCA add-ons.

INCA-SIP (INCA Simulink® Integration Package) is an INCA add-on that provides measurement and calibration access to MATLAB®/Simulink® modelled ECU functions via INCA. The INCA-SIP add-on is an additional Simulink® Toolbox.

INCA and the INCA add-ons are intended to be used in industrial labs and in test vehicles.

ETAS GmbH cannot be made liable for damage that is caused by incorrect use and not adhering to the safety information.

## 1.2 Target Group

This software product and this user guide address qualified personnel working in the fields of automotive ECU development and calibration, as well as system administrators and users with administrator privileges who install, maintain, or uninstall software. Specialized knowledge in the areas of measurement and ECU technology is required. This includes calibration procedures, software such as INCA and MDA and software algorithms of systems you would like to calibrate.

## 1.3 Classification of Safety Messages

Safety messages warn of dangers that can lead to personal injury or damage to property:



### **DANGER**

---

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



### **WARNING**

---

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



### **CAUTION**

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

### **NOTICE**

**NOTICE** indicates a situation that, if not avoided, could result in damage to property.

## 1.4 Safety Information

Observe the following safety information when working with INCA and INCA add-ons:



### **WARNING**

#### **Risk of unexpected vehicle behavior**

Calibration activities influence the behavior of the ECU and the systems that are connected to the ECU.

This can lead to unexpected vehicle behavior, such as engine shutdown as well as breaking, accelerating, or swerving of the vehicle.

Only perform calibration activities if you are trained in using the product and can assess the possible reactions of the connected systems.



### **WARNING**

#### **Risk of unexpected vehicle behavior**

Sending messages via bus systems, such as CAN, LIN, FlexRay, or Ethernet, influences the behavior of the systems connected to it.

This can lead to unexpected vehicle behavior, such as engine shutdown as well as breaking, accelerating, or swerving of the vehicle.

Only perform the sending of messages via a bus system if you have sufficient knowledge in using the respective bus system and can assess the possible reactions of the connected systems.

Adhere to the instructions in the ETAS Safety Advice and the safety information given in the online help and user guides. Open the ETAS Safety Advice in the INCA help menu ? > **Safety Advice**.

## 1.5 Data Protection

If the product contains functions that process personal data, legal requirements of data protection and data privacy laws shall be complied with by the customer. As the data controller, the customer usually designs subsequent processing. Therefore, he must check, if the protective measures are sufficient.

## 1.6 Data and Information Security

To securely handle data in the context of this product, see the INCA Help section "Data and Information Security".



## 2 About INCA-SIP

INCA-SIP (INCA Simulink<sup>®</sup> Integration Package) is an INCA add-on that provides measurement and calibration access to MATLAB<sup>®</sup>/Simulink<sup>®</sup> modelled ECU functions via INCA. The INCA-SIP add-on is an additional Simulink<sup>®</sup> Toolbox.

If you follow the instructions in chapter "[Installation](#)" on page 14, a new menu item will be available in Simulink<sup>®</sup>. This menu item establishes a connection between INCA and your Simulink<sup>®</sup> model.

As part of the connection procedure, INCA-SIP creates various INCA objects (.a2l, .s19, INCA workspace) based upon the user's model. In addition, INCA-SIP provides a number of search scripts that enable measuring and calibration of models using standard Simulink<sup>®</sup> blocks. Customer-specific blocks can also be measured and calibrated. To support such customer-specific blocks, custom search scripts need to be written.

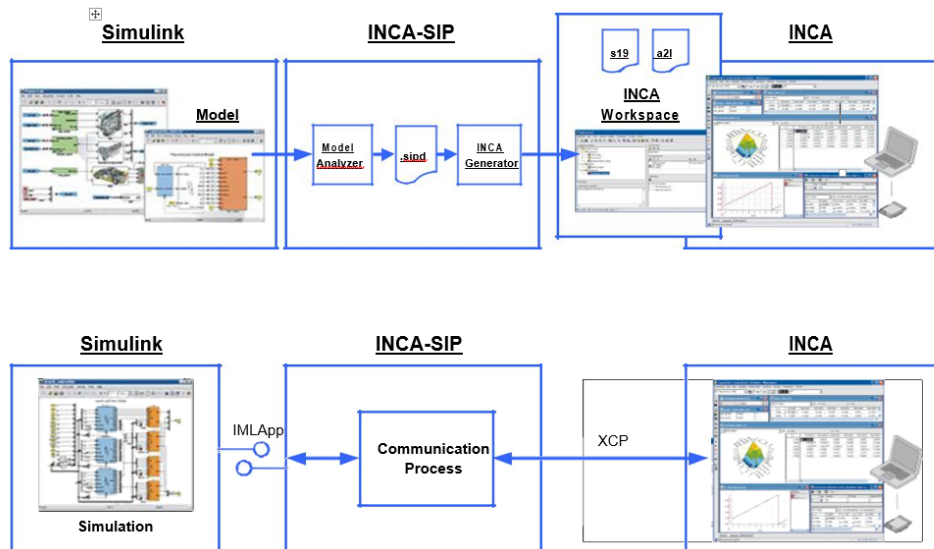
### 2.1 Connecting the Simulink<sup>®</sup> Model to INCA

During the model connection phase, INCA-SIP starts an analysis of the model blocks (see the upper part in the figure chapter "[Measuring and Calibrating the Model](#)" on the next page). You can customize the analysis to deal with customer-specific blocks through custom search scripts.

After a successful connection, all parameters required for measurements and calibration access are determined and all artifacts required by INCA are generated.

## 2.2 Measuring and Calibrating the Model

Within the INCA environment, the INCA-SIP add-on is presented as an XCP device. INCA uses XCP to send and receive data to and from INCA-SIP, while INCA-SIP exchanges data with the model via MATLAB®'s COM API interface (see figure lower part). Measurements from a Simulink® model can be received in INCA when the Simulink® model is running, while calibration is also possible even if the model is not under simulation, i.e. stopped or paused.



Analyzing the model (upper part) and M/C access (lower part) INCA-SIP

## 2.3 Technical Overview

### 2.3.1 Basic Operation

INCA-SIP is a suite of applications, libraries, and MATLAB® scripts that enables measurement and calibration of Simulink® models by INCA. Most of INCA-SIP functionality has been implemented in MATLAB® scripts. This enables extensive customization of the suite, such as supporting customer-specific calibration and measurement blocks. To an MC application INCA-SIP appears as an ASAM MCD-1 XCP V1.1.0 server that represents one or more “virtual ECUs” depending on the underlying Simulink® model. The application such as INCA interacts with these “ECUs” as if they were real physical devices. Underneath, INCA-SIP exchanges measurement and calibration data with a Simulink® model using the Microsoft COM API which communicates between the INCA-SIP Server executable and the Simulink® Model.

## 2.3.2 INCA-SIP MATLAB® Components

To use INCA-SIP, a path to INCA-SIP installation directory must be added to the MATLAB® search path list. The installation contains `s1_customization.m` script, which tells Simulink® to add INCA-SIP menu items to its **Tools** menu. Most of MATLAB® specific functionality is implemented in an **ETAS SIP** Simulink block, which needs to be added to the model being calibrated. Adding this block to a model loads the INCA-SIP model containing the custom Simulink blocks and loads the block.

## 2.3.3 Connection MATLAB® Side

With INCA-SIP, the connection is always initiated on the Simulink® side. The connection process starts when the user selects **Connect to INCA** from the INCA-SIP submenu. During connection, a MATLAB® script performs a `find_system()` call to retrieve all blocks in the model and loops through them to find blocks that are suitable for calibration, such as Constant or Lookup Table blocks, and named signals that could be used as measurements. The results of this analysis are saved in three files:

File Format	Description
*.sipd	SIPD is an XML file specific to INCA-SIP, which contains descriptions of all measurements, calibrations, events, groups, and conversion methods found during the analysis.
*.a2l	A2L contains essentially the same information, but in ASAM MCD2 MC (ASAP2) format that is recognized by most compliant MC applications. Applications such as INCA use this file to create measurement and calibration variables for their variable selection dialogs.
*.s19	S19 is a binary file containing memory offsets of the measurements and calibrations in the Motorola S19 format.

Once the connection has been established, all subsequent interaction, for example, sending calibration values and requesting measurement data during visualization, is initiated by the MC application such as INCA. After connection, the model is started in a paused state until a measurement is started.

## 2.3.4 Connection INCA Side

Once the A2L and S19 files have been generated, a MATLAB® script starts the INCA-SIP COM server. After the server starts, the script starts INCA, creates all the necessary INCA artefacts (the folder, the workspace, and the project) and prepares the INCA Experiment Environment for visualization via a series of calls to INCA COM API. Information in the A2L file is used to create the “A2l project” in INCA and represent each virtual ECU from the model as an “Ethernet System

Simulation device" in the INCA Hardware Configuration Editor. After these steps are complete, the user is ready to select MC variables, configure the experiment layout and start visualization. As a final step, the connection script makes a COM API call on the INCA-SIP server to start its XCP listener thread. At this point the connection process is complete and INCA-SIP XCP server (slave) is ready to accept commands from INCA.

### 2.3.5 Handling Measurements

When configuring the ECU, the rasters chosen in the ECU dialogue window reflect how often INCA will receive updates on the model's measurements. If no custom rasters are selected, a default raster of 100 ms is defined.

These rasters cannot be smaller than Simulink<sup>®</sup>'s in-built solver step and must be a multiple of this solver step as well. Small rasters will lead to inefficient execution time of the model due to the increased communication between INCA and Simulink<sup>®</sup>.

During the model run, each time this raster time is triggered in Simulink<sup>®</sup>, a COM event is sent to the INCA-SIP executable and triggers INCA-SIP to get the values of all variables selected in INCA for that simulation point in time and cache it. If multiple rasters are triggered at the same time for the same variable, it will use a locally cached value.

In the case the variable is from a Simulink<sup>®</sup> ECU, for each value the INCA-SIP executable will communicate via COM to the Simulink<sup>®</sup> model to get its latest value. If it is a DLL ECU, the INCA-SIP executable will read the DLL loaded in memory.

The SIP Server retrieves this measurement data and sends it to INCA in response to XCP calls from INCA Target Server. The data appears in INCA as if it was received from a physical ECU.

### 2.3.6 Simulink<sup>®</sup> - INCA Synchronization

The main purpose of this COM event approach from Simulink<sup>®</sup> is to throttle down measurement data transmission rate to match the desired raster value. A Simulink<sup>®</sup> model updates at its own solver step which can be several orders of magnitude higher than the default raster of 100 ms and at a much faster rate at which INCA can update its user interface, throttling the communication.

The INCA-SIP XCP server would get such a request through the COM event and all variables will have to be updated before it continues the simulation time, to ensure synchronization at a particular point in time.

This can be seen by connecting an oscilloscope to the signal in the Simulink<sup>®</sup> model: the graph will slow down to match the measurement rate determined by the raster.

### 2.3.7 Handling Calibrations

Changes in calibration values are sent to INCA-SIP in the form of XCP Server requests. These changes are applied to corresponding block parameters in the model only when INCA visualization is running. To fulfill the request, INCA-SIP server posts a message to the Simulink<sup>®</sup> window of the running model whose handle has been obtained when the connection was established. Simulink<sup>®</sup> then executes an m-script, which sets the value of the block parameter specified in the message.

Calibration is also possible even if the model is not under simulation, that means stopped or paused. These changes will be reflected to the model once the measurement is resumed in INCA.

## 3 Installation

### 3.1 System Requirements

To work with INCA-SIP the following requirements must be fulfilled.

#### 3.1.1 Hardware Requirements

The hardware requirements for the add-on are the same as for INCA and MATLAB®/Simulink®. Refer to the respective manuals.

#### 3.1.2 Software Requirements

To work with INCA-SIP V7.5 you need the following software components:

- INCA V7.5 including current hotfixes
- MATLAB®/Simulink® (For supported versions, refer to the release notes.)

### 3.2 Installing

In this section you will find a description of the installation procedure and additional information about start menu entries and MATLAB® path modifications.

#### 3.2.1 Installing INCA-SIP V7.5

This section describes how to install INCA-SIP V7.5.

##### To install INCA-SIP

1. Close all open ETAS programs.
2. Depending on your company-specific regulations, the installation files are provided on DVD or on a network drive.

By using the DVD, the installation starts automatically. If this is not the case, execute the `Autostart.exe` file.

*or*

If you install the program from a network drive, execute the `Setup_ServicePack.exe`.

3. In the column "Install" select the checkboxes of INCA V7.5, MDA, INCA-SIP and any further add-ons if needed.

#### 3.2.2 Registering a C++ Compiler

If you use Rapid Accelerator Mode in Simulink® with INCA-SIP, you need a C++ compiler registered as a MEX compiler. Therefore, prior to using INCA-SIP with Rapid Accelerator Mode, you have to register an external compiler, such as MSVC (Visual Studio), as a MEX compiler. To register an external compiler, run a "mex - setup" in the desired MATLAB® version before connecting INCA-SIP to INCA. For the entire list of supported compilers, refer to the MATLAB® documentation.

### 3.2.3 Start Menu

The Windows start menu folder contains the following entries once installation has been completed successfully:

- **E > ETAS INCA V7.5 > INCA V7.5 Manuals and Tutorials**  
Opens a folder containing this manual and other support information.
- **E > ETAS > ETAS License Manager**  
Starts the ETAS License Manager where you can view and manage licenses for your ETAS software products (e.g. adding, borrowing, and returning licenses).

### 3.2.4 Installation Path of INCA-SIP

The default installation path for INCA-SIP is:

64-bit                    <drive>:\Program Files\ETAS\AddOn\_SIP7.5\

### 3.2.5 MATLAB<sup>®</sup> Script Files

If INCA-SIP is installed, the MATLAB<sup>®</sup> script files are stored under the following directories:

64-bit                    <drive>:\Program Files\ETAS\AddOn\_  
SIP7.5\+MScripts

The folder "+MScripts" contains MATLAB<sup>®</sup> script files that are either called by INCA-SIP server or used by Simulink<sup>®</sup> to interact with INCA-SIP server and INCA.

## 3.3 Licensing

A valid license is required to use the software. You can obtain a license in one of the following ways:

- from your tool coordinator
- via the self-service portal on the ETAS website at [www.etas.com/support/licensing](http://www.etas.com/support/licensing)
- via the ETAS License Manager

To activate the license, you must enter the Activation ID that you received from ETAS during the ordering process.

For more information about ETAS license management, see the [ETAS License Management FAQ](#) or the ETAS License Manager help.

#### To open the ETAS License Manager help

The ETAS License Manager is available on your computer after the installation of any ETAS software.

1. From the Windows Start menu, select **E > ETAS > ETAS License Manager**.  
The ETAS License Manager opens.
2. Click in the ETAS License Manager window and press F1.

The ETAS License Manager help opens.



## 4 Working with INCA-SIP

This chapter contains information on preparatory measures and a description of how to work with INCA-SIP.

### 4.1 MATLAB® Search Path Modification

To make INCA-SIP available in a particular MATLAB® version, add the INCA-SIP installation path using the MATLAB® command `addpath` followed by `savepath`. Refer to [3.2.4 "Installation Path of INCA-SIP" on page 15](#) for path information.

Once the path is entered, the INCA-SIP menu item will be visible under the "Tools" menu in the Simulink® window.

### 4.2 Switching between INCA Versions

You can have both INCA-SIP V7.5 and any other INCA-SIP V7.x installed on the same computer although you use only one version at a time.

However, if you switch between INCA versions, it will be necessary to manually change the path to the proper INCA-SIP version and to restart MATLAB®.

For more information, refer to ["MATLAB® Search Path Modification" above](#).

### 4.3 Menu Items

The INCA-SIP menu item contains the following functions:

- ["Connect to INCA" below](#)
- ["ECUs" on page 19](#)
- ["Configuration" on page 23](#)
- ["MDF Configuration" on page 24](#)
- Help

#### 4.3.1 Connect to INCA

If you select this feature, INCA-SIP will start processing the associated model as per the configuration settings. For each enabled ECU definition in INCA-SIP, the calibrations and measurements are identified and processed. Each ECU definition is converted to an A2L and an S19 file. When the conversion process is complete, INCA is automatically opened and a workspace is auto-generated in INCA. For each ECU, a device is added in INCA with the corresponding project description and dataset.

INCA-SIP generates an EPK for each ECU.

The EPK is composed of:

- A hashing function which considers the calibrations and measurements present in the ECU
- The IP address of the network interface the ECU is made available on

- The associated TCP port

This ensures that a device on INCA will be connected to the appropriate Simulink<sup>®</sup> model in INCA-SIP.

If the model is modified and you attempt to connect to INCA again, INCA-SIP detects the changes and automatically replaces the datasets in INCA. If you have an open experiment in INCA, INCA-SIP prompts you to save before closing the experiment for you.

After the connection process is complete, INCA-SIP leaves the model in a paused state in "Normal" and "Accelerated" mode. This will speed up the data collection whenever measuring is started in the experiment window in INCA.

### 4.3.2 ECUs

If you select this feature, the **ECUs** dialog box opens.

You have three options to choose for the ECUs being modeled:

- **Add**
- **Remove**
- **Configure**

INCA-SIP supports two types of ECUs, Simulink<sup>®</sup> and DLL ECUs.

Simulink<sup>®</sup> ECUs represent ECUs that have measurements and calibrations which exist as blocks within the Simulink<sup>®</sup> model.

Simulink<sup>®</sup> ECUs are divided in two types: Simulink<sup>®</sup> ECU and Referenced Model Group.

The difference between these two types is that Simulink<sup>®</sup> ECU allows either mapping to the root model or one-to-one mapping to referenced models. On the other hand, Referenced Model Group only allows one to many mapping to referenced models.

DLL ECUs represent ECUs that are used when an S-Function uses third party DLLs in order to compute the S-Function result. Such DLLs can have global variables which can be considered as measurements and/or calibrations. DLL ECUs provide a mechanism that exposes these variables as calibrations and measurements in INCA.

To add a new ECU in Simulink<sup>®</sup>

1. In the **Tools** menu, select **INCA-SIP > ECUs**.  
The "ECUs" dialog box opens.
2. Click **Add** in the "ECUs" dialog box.  
The "Add ECU" dialog box opens.
3. Enter the required information in the following fields:
  - ECU Name:** Name of the ECU. This field is macro compliant, refer to ["Macros" on page 27](#).
  - ECU Type:** Type of the ECU, Simulink<sup>®</sup> or DLL
4. Click **Add**.  
The new ECU is added to the Simulink<sup>®</sup> model.

When you use DLL ECUs, INCA-SIP creates the following folders in the same folder location as the Simulink<sup>®</sup> model.

- A roof folder named "+ <SimulinkModelName>\_DLL".
- Within the "+ <SimulinkModelName>\_DLL" folder, an additional folder named "+ <ECUName>" is created. Note that the name of the ECU must be a valid Simulink<sup>®</sup> package name.
- Within the "+ <ECUName>", two more folders are created:

+Calibrations	This folder contains search functions which add calibrations.
+Measurements	This folder contains search functions which add measurements.

DLL ECUs are auto detected if the model does not have any saved configurations. If the model already has some configurations saved, you can add the DLL ECU manually from the ECUs window.

#### To configure an ECU in Simulink<sup>®</sup>

1. In the **Tools** menu, select **INCA-SIP > ECUs**.  
The "ECUs" dialog box opens.
2. Select the desired ECU.
3. Click **Configure** in the "ECUs" dialog box.  
The "Configure ECU" dialog box opens.
4. Enter your configurations.  
For more information, refer to ["Individual Tab Description" below](#).
5. Click **Save**.

### Individual Tab Description

#### *General*

This tab contains settings related to how a specific ECU will connect to INCA.

<i>Enabled</i>	If enabled, this ECU will be used. If disabled, INCA-SIP will skip processing this ECU.
<i>Regenerate A2L and S19 file</i>	If disabled, INCA-SIP will not generate the A2L and S19 files, assuming that the content of the existing files is correct. This can be used to speed the processing time when connecting. INCA-SIP can decide to ignore this flag if it deems that the existing files are not using the latest formats, or if any of the files are missing. In this case, a corresponding information message will be reported.
<i>ECU Name (read only)</i>	Name of the ECU. Note that this field is macro compliant. For more information, refer to <a href="#">"Macros" on page 27</a> .
<i>S19 File Name</i>	Path containing the name of the S19 generated file. Note that this field is macro compliant. For more information, refer to <a href="#">"Macros" on page 27</a> .
<i>A2L File Name</i>	Path containing the name of the A2L generated file. Note that this field is macro compliant. For more information, refer to <a href="#">"Macros" on page 27</a> .

<i>SIPD File Name</i>	Path containing the name of the SIPD generated file. The SIPD file contains a full description of the ECUs events, calibrations, measurements, and groups. Note that this field is macro compliant. For more information, refer to <a href="#">"Macros" on page 27</a> .
<i>Subsystem (visible only in Simulink® ECUs)</i>	A sub path within the model which this ECU will represent. Processing for measurement and calibrations will be limited to this path and its children. Note that this field is macro compliant. For more information, refer to <a href="#">"Macros" on page 27</a> .
<i>Referenced Model (visible only in Simulink® ECUs)</i>	Name of the referenced model this ECU represents. If set as N/A, the ECU represents the root model.
<i>IPv4 Address</i>	The IPv4 address of the network interface the XCP Slave will be listing on. If an IP address is configured and such IP is no longer valid, INCA-SIP will revert the configuration to the Loopback IP (127.0.0.1).
<i>XCP TCP Port</i>	The XCP Port on which the ECU XCP Slave service will be listening on. If set to -1, INCA-SIP will allocate a random port for this ECU and will save it in the associated configurations.
<i>All/Selection (visible only in Referenced Model Group ECUs)</i>	<b>All</b> includes all referenced models that exist in the connection phase. The selection allows a subset of the referenced models to be mapped to the ECU, which can be selected using the <b>Model List</b> function.

## Events

Events can be defined to create rasters in the ECU, which are calculated in seconds. These decide how often SIP expects updates from the Simulink® model. A default raster of 0.1 seconds is setup in this window.

If the raster time is 0, the DAQ will send the data with each solver step in Simulink®. Small raster values or using raster time 0 with a very small, fixed solver time in Simulink® are possible but can become inefficient and cause to a slow execution.

A raster that is smaller than the fixed solver time or a raster that is not a multiple of the solver is not possible and will be marked invalid on INCA connection. This is because these rasters would not all be able to be triggered and some will be skipped.

Using variable solver time is not recommended as it cannot be determined if the raster will be hit or not, or how often Simulink® will be updated.

## *Search Functions*

The two tables contain the search scripts available for that ECU. They are separated in two tables: one for calibration and one for measurement. Multiple selection is possible by holding **SHIFT** pressed down. You can deselect a single item by holding **CTRL** pressed down.

Simulink<sup>®</sup> ECUs come with default supported calibration search scripts for Simulink<sup>®</sup> blocks, these are: Basic Gain, Basic Lookup 1D, Basic Lookup 2D, Basic Lookup 3D, Table Lookup nD, Direct Lookups nD, Interpolation Blocks nD, Prelookups, and Generic S-Function Masked Parameters.

Note that nD implies all dimensions from 1D to 4D.

For measurements, Simulink<sup>®</sup> ECUs support searching for named lines that are connecting blocks. Each line is followed to its data source(s). The Runtime Objects of the outports composing the measurement will then be used for data collection.

DLL ECUs search scripts for these ECUs need to be written specifically for the custom DLL(s) being calibrated and measured. For more information, refer to chapter "[Creating a DLL Search Script](#)" on page 33 on how to create these custom scripts.

If the same calibration is in multiple ECUs, calibrating it on one ECU will not update the dataset of the other ECUs in INCA. This will result in a checksum error when reinitializing the hardware from INCA.

### 4.3.3 Configuration

If you select this feature, you can configure INCA project creation settings.

#### To configure INCA project creation settings

1. In the **Tools** menu, select **INCA-SIP> Configuration**.  
The "Session Configuration" dialog box opens.
2. Enter your configurations.  
For more information, refer to "[Session Configuration Description](#)" below and "[Remote INCA Configuration](#)" on the next page.
3. Click **OK**.

#### Session Configuration Description

<i>Compile Model</i>	If disabled, when <b>Connect to INCA</b> is pressed, the Simulink <sup>®</sup> Model is not compiled.  You need compilation to generate measurements, hence, if a Simulink <sup>®</sup> ECU is using the BasicNameSignals Search Script and has the Compile Model unchecked, it will result in no measurements being visible in INCA unless the Regenerate A2L and S19 file is unchecked and files already exist.
<i>Open INCA On Connect</i>	If enabled, INCA opens and the automatic configuration of the workspace is done.  If disabled, the A2L and S19 files are still generated (if needed), the XCP slave is initiated, but INCA will neither be opened nor configured.
<i>INCA Database</i>	The path of the directory INCA should save the INCA database for INCA-SIP. Note that this field is macro compliant. For more information, refer to " <a href="#">Macros</a> " on <a href="#">page 27</a> .
<i>INCA Folder</i>	The name of the INCA folder in the INCA Database. Note that this field is macro compliant. For more information, refer to " <a href="#">Macros</a> " on <a href="#">page 27</a> .
<i>INCA Workspace</i>	The name of the INCA project Workspace. Note that this field is macro compliant. For more information, refer to " <a href="#">Macros</a> " on <a href="#">page 27</a> .

## Remote INCA Configuration

<i>Remote INCA Connection</i>	If enabled, INCA-SIP opens INCA on the remote machine via DCOM.
<i>INCA Machine DNS Name/IP Address</i>	The host name or IP address of the remote INCA machine.
<i>Root Network Path</i>	Common UCN path that the INCA machine and the MATLAB® machine can access.

For more information, see ["Using INCA Remotely" on page 42](#).



### Note

The default Simulink® ECU contains the Emulation Mode Controls in the A2L. For more information, refer to ["INCA-SIP Emulation Mode" on page 28](#).

If **Show Workspace/DD Variables** is enabled, each ECU will have a new group in the **Variable Selection Dialog**. The new group contains all the variables used by the model which are either in the global workspace, in the local workspace, or in the data dictionary.

### 4.3.4 MDF Configuration

If you select this feature, you can configure the MDF stimulus.

To configure INCA the MDF stimulus

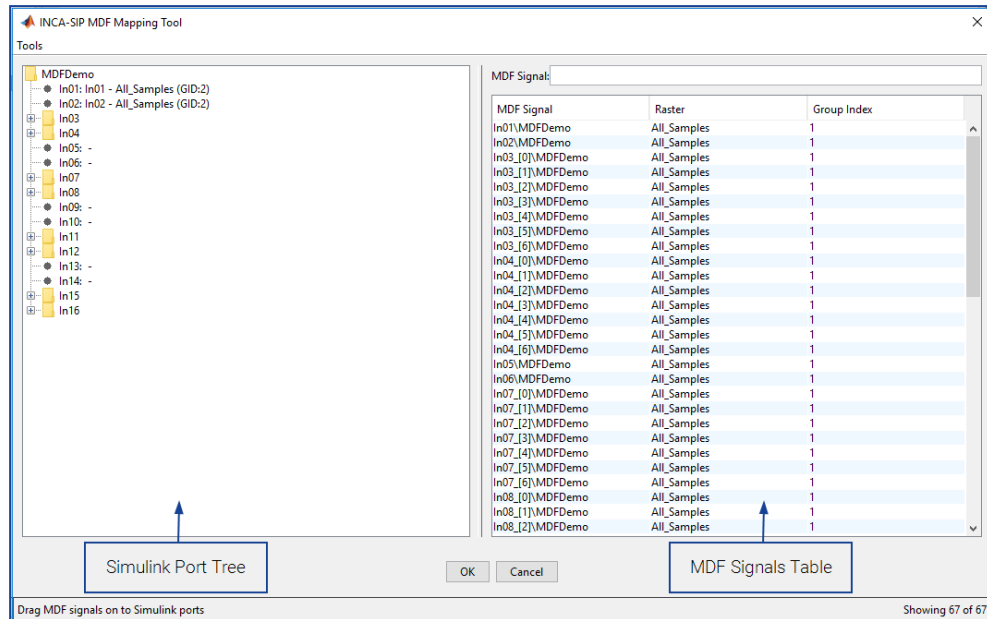
1. In the **Tools** menu, select **INCA-SIP > MDF Configuration**.  
The "MDF Configuration" dialog box opens.
2. Enter your configurations.  
For more information, refer to ["MDF Configuration Description" on the next page](#).
3. Click **OK**.



## MDF Configuration Description

<i>Use MDF Stimulus</i>	If enabled, INCA-SIP will allow loading external stimulus from MDF files to inputs in Simulink <sup>®</sup> . Once connected INCA-SIP will validate that the stimuli are loaded.
<i>Stimulus File</i>	The MDF file from which the stimuli will be loaded and mapped. Note that this field is macro compliant. For more information, refer to <a href="#">"Macros" on page 27</a> .
<i>Mapping File</i>	The configuration file defining the Mapping MDF signals from the Stimulus file to the inputs in the Simulink <sup>®</sup> Model. If the file does not exist, INCA-SIP will generate it automatically when the mapping tool is used. Note that this field is macro compliant. For more information, refer to <a href="#">"Macros" on page 27</a> .
<i>Mapping Tool</i>	Opens the MDF Mapping tool. This tool will allow to map MDF signal to inputs in the Simulink <sup>®</sup> Model. The left hand side tree shows the ports in Simulink <sup>®</sup> to which data can be coupled with. The right hand side table contains all the data from the MDF file. The MDF data can be dragged onto the Mapping Value column to map the data of that column to the required inport. INCA-SIP loads the data for the selected MDF stimuli from time = 0 till time = <simulation time>. If the simulation time is extended, the mapping tool needs to be re-used to load the data to the newly specified simulation time.

## Using the MDF Mapping Dialog



### To map a signal

- In the "MDF Mapping Tool" window, drag the desired MDF signal and drop it on a Simulink<sup>®</sup> port.

The Simulink<sup>®</sup> ports can be identified by the following icon:

or

1. Select a Simulink<sup>®</sup> port.
2. Double-click on the desired MDF signal to map it to the port.

### Clearing MDF Mappings

#### To clear all mappings

- In the "MDF Mapping Tool" window, select **Tools > Clear All Mappings**.

or

- Right-click on the Simulink<sup>®</sup> port tree and select **Clear All Mappings**.

#### To clear individual mappings

1. Select a port or a tree node.
2. Press the <DEL> key.

or

- Right-click on the Simulink<sup>®</sup> tree and select **Clear Selected**.

The selected node and all of its children are cleared.

#### Note

This operation cannot be performed on the root node.

## Updating the Simulink® Ports

To map MDF signals to Simulink® ports, the model must be compiled. INCA-SIP uses the last known configuration (if available) to avoid recompiling the model every time the mapping tool is opened. This last known configuration may be out-dated.

### To update the Simulink® port tree

1. In the "MDF Mapping Tool" window, click on **Tools**.
2. Reload Simulink® ports from the "Mapping" dialog.

## Updating MDF Signal Data

For performance reasons, the MDF data is stored as part of the model workspace. This data may be out of date, so manual updating is necessary. For example, if the MDF file is changed or replaced with the same MDF file name and path.

### To update the MDF signals

1. In the "MDF Mapping Tool" window, click on **Tools**.
2. Reload MDF data from the "Mapping" dialog.



### Note

There are cases where the data is automatically reloaded when connecting to INCA:

- If INCA loads the MDF data, the simulation time is taken into account and only the required amount of data is loaded. If the simulation time is changed to a longer duration, the data is automatically reloaded.
- If the MDF path is changed in the MDF configuration window, the cached MDF data is deleted so that it is automatically reloaded when connecting.
- If a Simulink® port is changed and the assignment becomes invalid, the MDF data for that port is deleted and automatically reloaded (if reassigned) when connecting.

## 4.4 Macros

INCA-SIP supports macro variables in strings to provide you dynamic strings. You can perform the following macro variables:

- `$(SystemName)` : Resolves to the name of the model.
- `$(ECUName)` : Resolves to the name of the ECU (can only be used in ECUs Configuration window).
- `$(INCADataDir)` : Resolves to the INCA data directory path.
- `$(INCASIPDir)` : Resolves to INCA-SIP Installation path.

INCA-SIP also supports environment variables in macros, e.g. `% (APPDATA) %`.

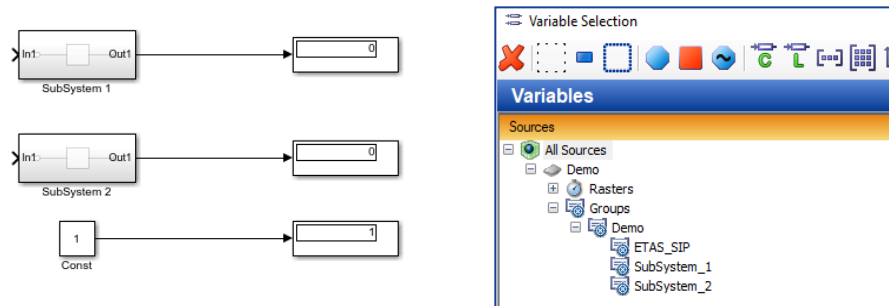
## 4.5 Naming Conventions

### 4.5.1 Calibrations and Measurements

When INCA-SIP processes an ECU, it will generate an A2L file. Hence, any names for measurements and calibrations need to be compliant with the A2L calibrations and measurement names. Non-compliant names are renamed accordingly. In addition, if there are any name conflicts, INCA-SIP adds `_x` to the name, where `x` is an integer in the range 1 to infinity.

### 4.5.2 Groups

You can define measurements and calibrations to exist in `Groups`, shown in INCA's Variable Selection Dialog (VSD). If there are any name conflicts between group names in the same ECU, INCA-SIP will add `_x` to the name, where `x` is an integer in the range 1 to infinity.



The "ETAS\_SIP" Group will contain the controls related to the INCA-SIP Emulation Mode.

## 4.6 INCA-SIP Emulation Mode

INCA-SIP provides two modes of model simulation:

- "Real-Time Emulation Mode"
- "Fast Emulation Mode"

These controls are available in the "Default Simulating ECU" in the A2L file in the "ETAS-SIP" group.

### 4.6.1 Real-Time Emulation Mode

In Real-Time Emulation Mode, the INCA-SIP add-on attempts to approximate real-time simulation. Therefore, if the model is simulating faster than real-time use-case, INCA-SIP will throttle the model simulation to near real-time. On the other hand, INCA-SIP has no effect on model simulation when it is simulating slower than real-time.

This mode is ideal in the case the user would like to calibrate the model during simulation and visualize the effects of the calibration on the simulation.

To activate this mode of operation, within the associated INCA experiment, you need to add the `RealTimeBlock.mode` calibration and select the "Real Time" option. The `RealTimeBlock.realTimeMultiplier` calibration can be used to change the throttling multiplier as described below:

<i>Equal to 1</i>	Represents real time
<i>Less than 1</i>	Represents slower than real time
<i>Greater than 1</i>	Represents faster than real time
<i>Less than or equal to 0</i>	Are equivalent to INF, which represents the Fast Emulation Mode

You can find these calibrations in the "ETAS\_SIP" group in the "Default Simulink ECU".

#### 4.6.2 Fast Emulation Mode

In Fast Emulation Mode, during model simulation, the INCA-SIP add-on transmits data to INCA as fast as possible, while ensuring no data loss.

Therefore, this behavior makes it impractical for data visualization. This mode is ideal in case you would like to make offline calibration changes and then simulate the model for a fixed period of time. This model simulation should be complete in a relatively short period of time, while the measured data can then be analyzed offline through MDA (Measure Data Analyzer).

In order to activate this mode of operation, within the associated INCA experiment, you need to add the `RealTimeBlock.mode` calibration and select the "Fast Emulation" option. You can find these calibrations in the "ETAS\_SIP" group in the "Default Simulink ECU".

#### 4.7 INCA-SIP and Block Reduction

MATLAB<sup>®</sup> supports block reduction by default. The calibrations and measurements can be optimized or eliminated from the simulation if these are deemed to have no effect on the simulation. INCA-SIP ignores such calibrations and measurements. You can enable and disable block reduction from the simulation settings. For more information, refer to the MATLAB<sup>®</sup> documentation.

#### 4.8 Simulation Modes

MATLAB<sup>®</sup> provides various simulation modes. INCA-SIP supports the following modes:

- Normal Mode
- Accelerated Mode
- Rapid Accelerated Mode

### Note

Rapid Accelerated Mode needs a working registered C++ compiler to work. For more information, refer to "[Registering a C++ Compiler](#)" on [page 14](#).

### Note

Due to changes in new MATLAB® versions, Rapid Accelerated mode is only supported on MATLAB® Version 2018b and below.

## 4.9 Referenced Models Support

Referenced models are normal models in Simulink®. These models are referenced by a parent model, allowing the parent model to link to the referenced model's inputs and outputs. When configuring an ECU, it can be configured to point to a specific referenced model by choosing the referenced model's name from the combo box.

Referenced models are shown as a separate ECU in INCA.

## 4.10 Refusing Calibrations

There can be cases where MATLAB® Simulink® refuses calibrations sent by INCA. One such example is when you use a workspace variable as table data and axis data at the same time. In such case, the axis data need to be increasing monotonically while the table data has no such restrictions. If the table data is calibrated and monotonicity is broken, MATLAB® will refuse this calibration. The error code `CALMEM_REQUESTNOTVALID` being returned by XCP is displayed in the INCA Monitor window.

## 4.11 Busy ECU

Some requests may be time consuming and will place the ECU in a busy state. One such example is when calibrating a large array at one go by doing increment/decrement on all elements together. In such case, the ECU is busy with the execution of this request and INCA will show a dialog until the ECU is free again and is able to receive the next command.

## 4.12 Starting and Stopping the Experiment

To start the experiment

1. In the INCA Experiment window, click  or press F11.

The simulation in Simulink® behaves as follows:

Start if the simulation was in a stopped state.

or

Continue if simulation was in a paused state.

#### To stop the experiment

1. In the INCA Experiment window, click stop  or press F9.

The simulation in Simulink® behaves as follows:

Pause in Normal and Accelerated Mode.

or

Stop in Rapid Accelerated Mode.



#### Note

The simulation state is changed in the above cases only if there are some measurements selected in the INCA experiment.

### 4.13 Simulink.Signal and Asap2.Signal Support

In Simulink® ECUs, if a named line is named after a Simulink.Signal or Asap2.Signal, the measurement will take the "Description", "Unit", "Min" and "Max" from the Simulink.Signal or Asap2.Signal variable.

### 4.14 Simulink.Parameter and Asap2.Parameter Support

In Simulink® ECUs, if a supported block has a parameter with the content of a workspace variable of type Simulink.Parameter or Asap2.Parameter, the calibration will take the "Description", "Unit", "Min" and "Max" from the Simulink.Parameter or Asap2.Parameter variable.

### 4.15 Error, Warning and Information Messages

INCA-SIP displays error, warning and information messages during the connection and simulation process.

Messages related to an open Simulink® model are logged in the Simulink®'s Diagnostic Viewer.

#### To open Simulink®'s Diagnostic Viewer

1. In the Simulink® editor, click **View > Diagnostic Viewer**.

MATLAB® also opens the Diagnostic Viewer whenever an error message is logged.

### 4.16 DLL Mode

In MATLAB®, you can write an S-function which internally uses DLLs at some stage of its processing. The INCA-SIP DLL mode provides support to calibrate and measure global variables within a DLL.

### 4.16.1 Creating the DLL


You can find a sample Visual Studio Project under

`<drive>:\ETASData\INCA7.5\Demo\AddOn_INCA-SIP7.5\SIPDllDemo.`

The contents of the "SIPDllDemo" folder are described below.

"legacy_alg"	Contains the sample DLL which we will be calibrating and measuring from INCA. When the solution is built, it will copy the DLL and the map file to the "x64" folder. The map file is used to determine the module and variables offset.
"x64"	Used in a 64-bit MATLAB® version, this folder contains the model and the DLL search scripts. To use this project, the <code>legacy_alg_sfcn.c</code> file needs to be compiled using <code>mex legacy_alg_sfcn.c</code> .

In the source code, there are a number of global variables which can be measured or calibrated.

 **Note**

No synchronization mechanisms are in place between SIP and the DLL. Therefore, race conditions can occur if INCA-SIP is not the only actor calibrating the variables.

INCA-SIP supports the following types for DLL Mode:

INCA-SIP DLL Type	C Type	Size	Remarks
<code>MScripts.DllMode.</code> <code>DllAsap2Object.UINT8</code>	<code>uint8_t</code>	8	
<code>MScripts.DllMode.</code> <code>DllAsap2Object.UINT16</code>	<code>uint16_t</code>	16	
<code>MScripts.DllMode.</code> <code>DllAsap2Object.UINT32</code>	<code>uint32_t</code>	32	
<code>MScripts.DllMode.</code> <code>DllAsap2Object.UINT64</code>	<code>uint64_t</code>	64	Calibrations Only
<code>MScripts.DllMode.</code> <code>DllAsap2Object.INT8</code>	<code>int8_t</code>	8	
<code>MScripts.DllMode.</code> <code>DllAsap2Object.INT16</code>	<code>int16_t</code>	16	



INCA-SIP DLL Type	C Type	Size	Remarks
MScripts.DllMode. DllAsap2Object.INT32	int32_t	32	
MScripts.DllMode. DllAsap2Object.INT64	int64_t	64	
MScripts.DllMode. DllAsap2Object.BOOL	uint8	8	
MScripts.DllMode. DllAsap2Object.FLOAT	float	32	IEEE-754
MScripts.DllMode. DllAsap2Object.DOUBLE	double	64	IEEE-754



### Note

int64 and uint64 will only work with calibrations. If used in measurements, these will be seen in INCA and in the Variable Selection Dialog but cannot be selected.

## 4.16.2 Creating a DLL Search Script

You can find sample scripts under

```
<drive>:\ETASData\INCA7.5\Demo\AddOn_INCA-SIP7.5\SIPDLLDemo\x64\+SipDllDemo_DLL\+DllDemo
```

Each search script is defined by having a function defined by the following function signature:

```
<Function Name>(dataModel, sRootSystem)
```

The "dataModel" is the object to which calibrations and measurements need to be added. "sRootSystem" is a string with the name of the model.

### DataModel Function Calls

Function	addCalibration (calibration)	
Parameters	calibration	Calibration Description Object
Return Value	None	
Description	Adds this Calibration Description Object to the data model to be shown in INCA. All axes associated to the calibration will be added automatically.	
Function	addCalibrationAxis (axis)	

Parameters	<code>axis</code>	Calibration Axis Description Object
Return Value	None	
Description	Adds this Calibration Axis Object to the data model to be shown in INCA. This should only be used if you intend to show a stand-alone calibration axis which was not associated to any Calibration Object.	
Function	<code>addMeasurement (measurement)</code>	
Parameters	<code>measurement</code>	Measurement Description Object
Return Value	None	
Description	Adds this Measurement Description Object to the data model to be shown in INCA.	

Once you have defined a DLL calibration and have set all its properties, you need to add it to the data model. You can achieve this by:

```
dataModel.addCalibration(calibration);
```



### Note

An axis object should only be linked to one calibration. Should you want to link the same module offset to another calibration as an axis, another instance of the axis should be created.

If the calibration had a linked axis to it, you do not need to add the axis since these will be added automatically.

If an axis is not linked to any calibration, you can add it by:

```
dataModel.addCalibrationAxis(axis);
```

Once you have defined a DLL measurement and have set all its properties, you need to add it to the data model. You can achieve this by:

```
dataModel.addMeasurement(measurement);
```

### 4.16.3 Creating Object Using DllAsap2ObjectFactory

#### Creating Calibrations

The `MScripts.DllMode.DllAsap2ObjectFactory` provides the following methods to create calibrations:

Function	<code>createScalarCalibration(name, moduleOffset, cDataType, dllName, groupPath)</code>	
Parameters	<code>name</code>	Name as it will be shown in INCA
	<code>moduleOffset</code>	Module offset address hex string of the variable in the DLL
	<code>cDataType</code>	Type of the data at the module offset
	<code>dllName</code>	The name of the DLL where the module offset is located
	<code>groupPath</code>	Group in which the calibration will be shown in INCA
Return Value	Calibration Description Object	
Description	Creates a scalar calibration which is visualized as a simple calibration in INCA.	

Function	<code>createTableCalibration(name, moduleOffset, cDataType, dllName, groupPath)</code>	
Parameters	<code>name</code>	Name as it will be shown in INCA
	<code>moduleOffset</code>	Module offset address hex string of the variable in the DLL
	<code>cDataType</code>	Type of the data at the module offset
	<code>dllName</code>	The name of the DLL where the module offset is located
	<code>groupPath</code>	Group in which the calibration will be shown in INCA

	<code>arraySize</code>	The size of the array in vector format (e.g. [1 3] or [2 3 4])
Return Value	Calibration Description Object	
Description	Creates a n-dimensional calibration which is visualized as a calibration in INCA.	
Function	<code>createFixedAxis(offset, distance, numberOfAxisPoints, asap2Description)</code>	
Parameters	<code>offset</code>	Start value of the Fixed Axis
	<code>distance</code>	Increment Size
	<code>numberOfAxisPoints</code>	Number of elements in the axis
Return Value	Fixed Axis Description Object	
Description	Creates fixed axis. This can only be linked to a n-dimension calibration. Note that all n-dimension calibrations have assigned to them a fixed axis by default .	
Function	<code>createCalibrationAxis(name, moduleOffset, cDataType, dllName, groupPath, arraySize)</code>	
Parameters	<code>name</code>	Name as it will be shown in INCA
	<code>moduleOffset</code>	Module offset address hex string of the variable in the DLL
	<code>cDataType</code>	Type of the data at the module offset
	<code>dllName</code>	The name of the DLL where the module offset is located
	<code>groupPath</code>	Group in which the calibration will be shown in INCA
	<code>arraySize</code>	The size of the array in vector format (e.g. [1 3] or [2 3 4])
Return Value	Calibration Axis Description Object	
Description	Creates an axis. This can be linked to a n-dimension calibration or added to the data model without being linked to a Table Calibration.	

Function	<code>createCalibrationAxisWithMeasurementKey (name, moduleOffset, cDataType, dllName, groupPath, arraySize, sMeasurmentKey, hDataModel)</code>	
Parameters	<code>name</code>	Name as it will be shown in INCA
	<code>moduleOffset</code>	Module offset address hex string of the variable in the DLL
	<code>cDataType</code>	Type of the data at the module offset
	<code>dllName</code>	The name of the DLL where the module offset is located
	<code>groupPath</code>	Group in which the calibration will be shown in INCA
	<code>arraySize</code>	The size of the array in vector format (e.g. [1 3] or [2 3 4])
	<code>sMeasurmentKey</code>	The Measurement Key of the measurement which relates to this axis
	<code>hDataModel</code>	Handle to the data Model
Return Value	Calibration Axis Description Object	
Description	Creates an axis. This can be linked to a n-dimension calibration or added to the data model without being linked to a Table Calibration.	



### Note

It is the responsibility of the user to ensure that the module offset and the array sizes are correct.

## Linking an Axis to a Calibration

If you would like to link a DLL Axis Calibration to a DLL Table Calibration, you can do the following on Calibration Description Objects:

Function	<code>setAxis (axis, index)</code>	
Parameters	<code>axis</code>	Calibration Axis Description Object
	<code>index</code>	Index of axis e.g. 1=X axis, 2=Y axis, 3=Z axis, 4=W axis

---

Return Value	None
--------------	------

---

Description	Associates an axis to Calibration Description Object.
-------------	---

---

## Creating Measurements

The `MScripts.DllMode.DllAsap2ObjectFactory` provides the following methods to create measurements:

---

Function	<code>createScalarMeasurement(name, moduleOffset, cDataType, dllName, groupPath, arraySize)</code>	
----------	--	--

---

Parameters	name	Name as it will be shown in INCA
------------	------	----------------------------------

---

moduleOffset	Module offset address hex string of the variable in the DLL
--------------	---

---

cDataType	Type of the data at the module offset
-----------	---------------------------------------

---

dllName	The name of the DLL where the module offset is located
---------	--

---

groupPath	Group in which the calibration will be shown in INCA
-----------	--

---

Return Value	Measurement Description Object
--------------	--------------------------------

---

Description	Creates a Scalar measurement which is visualized as a simple measurement in INCA.
-------------	---

---



---

Function	<code>createArrayMeasurement(name, moduleOffset, cDataType, dllName, groupPath, arraySize)</code>	
----------	---	--

---

Parameters	name	Name as it will be shown in INCA
------------	------	----------------------------------

---

moduleOffset	Module offset address hex string of the variable in the DLL
--------------	---

---

cDataType	Type of the data at the module offset
-----------	---------------------------------------

---

dllName	The name of the DLL where the module offset is located
---------	--

---

groupPath	Group in which the calibration will be shown in INCA
-----------	--

---

	<code>arraySize</code>	The size of the array in vector format (e.g. [1 3] or [2 3 4])
Return Value	Measurement Description Object	
Description	Creates a n-dimensions measurement which is visualized as a collection of simple measurements in INCA.	

### Note

The name must follow ASAP2 naming conventions. If it is not adhered to, INCA-SIP renames the measurement/calibration to make it compliant. Similarly, if the same name has already been used in the ECU, then `_x` will be appended to the name, where x is an auto-incrementing integer ranging from 1 to infinity.

### Note

It is the responsibility of the user to ensure that the module offset and the array sizes are correct.

## General Options

All objects can have a description, minimum and maximum values set using the function calls below.

Function	<code>setAsap2Description (sDescriptionText)</code>	
Parameters	<code>sDescriptionText</code>	Description Text
Return Value	None	
Description	Sets description for object.	

Function	<code>setAsap2MinValue (dMin)</code>	
Parameters	<code>dMin</code>	Minimum Value
Return Value	None	
Description	Sets minimum allowed value.	

Function	<code>set2AsapMaxValue (dMax)</code>	
Parameters	<code>dMax</code>	Maximum Value
Return Value	None	
Description	Sets maximum allowed value.	

Function	<code>addGroupPath (sGroupPath)</code>	
Parameters	<code>sGroupPath</code>	Group in which the calibration will be shown in INCA
Return Value	None	
Description	Adds another group path where the object is to be seen in INCA.	

## 4.17 Writing Dataset to MDF File

The MDF write functionality allows you to take any Simulink<sup>®</sup> dataset, for example one generated by the logging of Simulink<sup>®</sup> signals, and output it to an MDF file.

### To write dataset to MDF file via MATLAB<sup>®</sup> command line

- Call the command `MScripts.MDF.Write (parameter1, 'parameter2', parameter3)`.

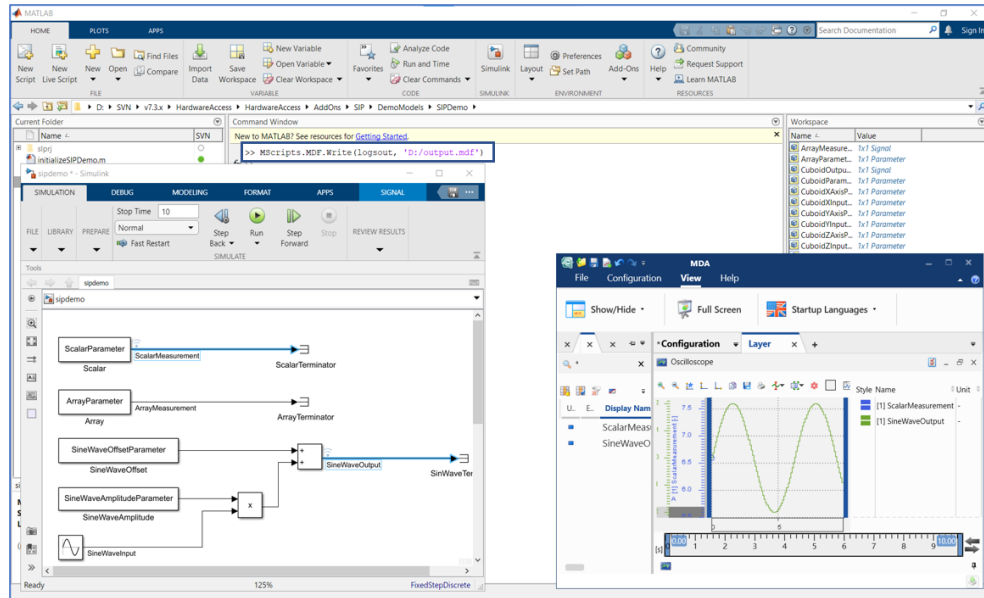
<code>parameter1</code>	MATLAB <sup>®</sup> /Simulink <sup>®</sup> dataset you want to write to an MDF file.
<code>parameter2</code>	Path to which you want to write the MDF file.
<code>parameter3</code>	Parameter instructs the MDF writer to copy the most recent value of every signal in the dataset to the last timestamp. This additional data point is added to help an MDF viewer, such as ETAS MDA, to display a line instead of just one point. Possible values are <code>true</code> or <code>false</code> . This parameter is optional.

#### Example:

```
MScripts.MDF.Write(logsout, 'D:/output.mdf', false)
```

The following screenshot shows the MDF write functionality in MATLAB<sup>®</sup> /Simulink<sup>®</sup> and MDA. Signals were logged using the signal logging feature of Simulink<sup>®</sup>. Then the MDF write command `MScripts.MDF.Write(logsout, 'D:/output.mdf')` was executed and the output was loaded into the MDA.





### Note

You can set Simulink<sup>®</sup> to log signals with a repeated value every tick if you set its **Sample Time** to **-1**.

If you use the lifetime callbacks of the model, you can set up Simulink<sup>®</sup> to automatically call the MDF write functionality whenever a model is paused, stopped, or saved, for example. Refer to the official MATLAB<sup>®</sup> documentation on how to set up model callbacks.

### To write an MDF file via GUI

1. In the **Tools** menu, click **INCA-SIP > MDF Configuration**.

The MDF Configuration dialog box opens.

2. In the MDF Export area, activate **Use Signal Logging**.

or

Click **Change Settings** and in the Simulink<sup>®</sup> Configuration Parameter dialog box on the **Data Import/Export** tab, activate the signal logging setting.

3. In the **Export File** field, define the path and the name of the MDF file.
4. To hold the last signal value until a signal changes its value in the MDF export file, activate **Hold Last Value**.
5. To create an MDF export file automatically when you click the **Stop** button in Simulink<sup>®</sup>, activate **Automatically Export on Stop**.
6. To automatically increment the filename of the MDF export file, activate **Auto Increment Filename**.

In case of a file collision, INCA-SIP appends an additional counter **\_xx** to the end of the file name. In case of a new session or a change of the file name, INCA-SIP resets the counter to **\_01**.

 **Note**

If this option is deactivated, previously created MDF export files will be overwritten.

7. To export the MDF file immediately, click **Export Now**.
8. Click **Apply**.
9. Click **OK**.

## 4.18 Using INCA Remotely

Remote INCA access is required if you want to use one machine to operate a Simulink<sup>®</sup> model while using INCA on another machine to calibrate the Simulink<sup>®</sup> model.

To run INCA remotely, INCA-SIP and INCA must be installed on the INCA machine and the remote machine.

### *Setting up the Remote INCA Connection*

To set up the Remote INCA Connection, perform the following actions:

- ["To adjust the Window Component Services settings " below](#)
- ["To add a user or a group to the INCA DCOM Config" below](#)
- ["To adjust the INCA-SIP settings in MATLAB<sup>®</sup>" on the next page](#)

#### To adjust the Window Component Services settings

1. Open the Windows desktop app **Component Services**.
2. Expand all **Component Services** folders.
3. Right-click on **My Computer** and select **Properties**.
4. In the **Default Properties** tab, make the following settings:
  - Enable **Enable Distributed COM on this computer**.
  - Set the **Default Authentication Level** to **Connect**.
  - Set the **Default Impersonation Level** to **Identify**.
5. Click **Apply**.
6. Click **OK**.

#### To add a user or a group to the INCA DCOM Config

1. Open the Windows desktop app **Component Services**.
2. Expand all **Component Services** folders.
3. In the **DCOM Config** folder, right-click on **INCA7x Server** and select **Properties**.
4. Select the **Security** tab.
5. In the **Launch and Activation Permissions** list, enable **Customize**.
6. Click **Edit**.

7. In the **Group or user names** list, click **Add** and add a new user or a new group.
8. Click **OK**.
9. In the **Permissions for SYSTEM** list, enable all **Allow** check boxes.
10. Click **OK**.
11. Click **Apply**.
12. Click **OK**.

If you add a user or a group, ensure that:

- The added user is the same user who is using MATLAB<sup>®</sup>.
- The added group contains the user who is using MATLAB<sup>®</sup>.

To allow DCOM communication through the firewall, you may need to change the firewall settings. For more information, refer to the Microsoft documentation.

To adjust the INCA-SIP settings in MATLAB<sup>®</sup>

1. In the **Tools** menu, select **INCA-SIP > Configuration**.
2. Enable the following settings:
  - **Open INCA On Connect**
  - **Remote INCA Connection**
3. In the **INCA Machine DNS Name/IP Address** field, enter the host name or the the IP address of the INCA remote machine.
4. In the **Root Network Path** field, enter a common UCN path that the INCA machine and the MATLAB<sup>®</sup> machine can access.

The **Root Network Path** is used to save the INCA database, the A2L file, and the S19 file. The data will be saved in the remote network path if the configured path is relative. In such case, the path is resolved relative to the configured UCN path. If the path is absolute, then it is the responsibility of the user to make sure that the INCA database path is accessible by the remote INCA machine. The A2L and S19 path will be accessible by both machines using the same UCN.



#### Note

Each ECU has its own TCP Port. Note that each ECU port must be allowed as an incoming port on the MATLAB<sup>®</sup> machine by the firewall. In addition, each ECU has the IP Address which INCA will use to connect to the XCP Slave. If the selected address is not on the same subnet as the INCA machine, INCA-SIP attempts to find an interface which is on the same subnet as the INCA machine. If no such interface is found, manual user action will be needed to identify the appropriate IP Address.

## 4.19 Custom Hooks

INCA-SIP provides hooks to allow:

- Custom filtering
- Custom measurement and calibration naming
- Custom group naming
- Customization of variables

Some use cases of these hooks are:

- Filter unwanted calibrations and measurements
- Create custom own naming conversion for calibrations, measurements, and groups
- Set the display identifier etc..

Custom hooks are function pointers to MATLAB<sup>®</sup> functions.

The hooks can be set in the `MScripts.Configuration` by calling:

- ["setVariableFilterFunctionHandle" below](#)
- ["setVariableCustomizationFunctionHandle" on page 46](#)
- ["setVariableRenamingFunctionHandle" on page 48](#)
- ["setGroupRenamingFunctionHandle" on page 49](#)

The hooks can be reset in the `MScripts.Configuration` by calling:

- `clearGroupRenamingFunctionHandle`
- `clearVariableCustomisationFunctionHandle`
- `clearVariableFilterFunctionHandle`
- `clearVariableRenamingFunctionHandle`

The `MScripts.Configuration` class is a singleton and is global to the MATLAB<sup>®</sup> Instance.

### To get the instance of this class

1. Execute `MScripts.Configuration.getInstance()`.

Any changes done to the configuration class are lost when MATLAB<sup>®</sup> is closed. Therefore, custom hooks would need to be set every time MATLAB<sup>®</sup> is opened.

### 4.19.1 setVariableFilterFunctionHandle

Each calibration or measurement added to a data model is passed to the custom hook if it is set. If the custom function accepts this calibration, it is added to the data model otherwise it is skipped and none of the remaining custom hooks are called with regards to this calibration or measurement.

## Hook Function Signature

Function	<code>bKeep = variableCustomFilter(type, variableInfoWrapper)</code>	
Parameters	<code>bkeep</code> returns	
	<code>true</code>	If the calibration or measurement is to be kept.
	<code>false</code>	If the calibration or measurement is to be filtered out.
	<code>type char[]</code>	Can be one of the following constants: <code>MScripts.DataModelBase.SimulinkdWorkspaceCalibration</code> <code>MScripts.DataModelBase.DLLCalibrationAxis</code> <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.DLLMeasurement</code> <code>MScripts.DataModelBase.SimulinkCalibrationAxis</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkMeasurement</code>
	<code>variableInfoWrapper</code>	Instance of <code>MScripts.VariableWrapper</code> . The <code>variableInfoWrapper</code> is used to determine which calibration or measurement is currently being processed. For more information on how to use the <code>variableInfoWrapper</code> , refer to <a href="#">"MScripts.VariableWrapper" on page 51</a> .



### Note

During this hook, only `getters` should be used on the `variableInfoWrapper`. Any changes needed to be effected to this calibration or measurement can be effected in the `setVariableCustomisationFunctionHandle` hook.

For an example of this hook, see the `CustomHooks variableCustomFilter.m` script located in the "SIPDEMO" folder.

Variable selection before using a custom hook

All Sources (Not Filtered, 7/7 Visible)		
	Name	Role
	_1_D_Lookup_Table	
	CalibrationToFilter	
	LineToFilter	
	OriginalCalibrationName	
	OriginalLineName	
	RealTimeBlock.mode	
	RealTimeBlock.realTimeMultiplier	

Variable selection after using a custom hook

All Sources (Not Filtered, 5/5 Visible)		
	Name	Role
	_1_D_Lookup_Table	
	OriginalCalibrationName	
	OriginalLineName	
	RealTimeBlock.mode	
	RealTimeBlock.realTimeMultiplier	

#### 4.19.2 setVariableCustomizationFunctionHandle

Each calibration or measurement that passes the filter function stage can be customized in this custom hook. To better understand what can be customized, refer to "[MScripts.VariableWrapper](#)" on page 51.

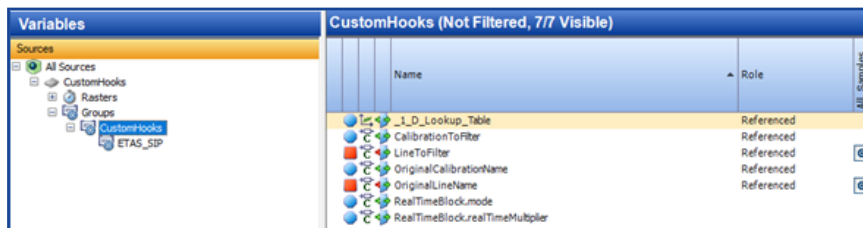
## Hook Function Signature

Function `variableCustomization(type, variableInfoWrapper)`

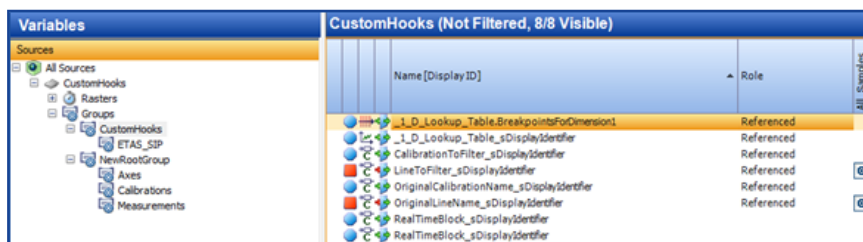
Parameters	<code>type char[]</code>	Can be one of the following constants: <hr/> <code>MScipts.DataModelBase.SimulinkWorkspaceCalibration</code> <hr/> <code>MScipts.DataModelBase.DLLCalibrationAxis</code> <hr/> <code>MScipts.DataModelBase.DLLCalibration</code> <hr/> <code>MScipts.DataModelBase.DLLMeasurement</code> <hr/> <code>MScipts.DataModelBase.SimulinkCalibrationAxis</code> <hr/> <code>MScipts.DataModelBase.SimulinkCalibration</code> <hr/> <code>MScipts.DataModelBase.SimulinkMeasurement</code>
variableInfoWrapper		Instance of <code>MScipts.VariableWrapper</code> . The <code>variableInfoWrapper</code> is used to determine which calibration or measurement is currently being processed. For more information on how to use the <code>variableInfoWrapper</code> , refer to <a href="#">"MScipts.VariableWrapper" on page 51.</a>

For an example of this hook, see the `CustomHooks variableCustomization.m` script located in the "SIPDEMO" folder.

Variable selection before using a custom hook



Variable selection after using a custom hook



### 4.19.3 setVariableRenamingFunctionHandle

Each calibration or measurement that passes the filter function stage can be renamed in this custom hook. Calibrations added due to the `show workspace/data dictionaries variables` are not passed to this hook as these cannot be renamed.

#### *Hook Function Signature*

---

```
Function mappedName = variableNameMapper(incaSIPName, type,
    variableInfoWrapper)
```

---

Parameters	<code>mappedName</code>	New desired mapped name. Cannot be empty.
	<code>incaSIPName</code>	Suggested name from INCA-SIP
	<code>type char[]</code>	Can be one of the following constants: <code>MScripts.DataModelBase.DLLCalibrationAxis</code> <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.DLLMeasurement</code> <code>MScripts.DataModelBase.SimulinkCalibrationAxis</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkMeasurement</code>
	<code>variableInfoWrapper</code>	Instance of <code>MScripts.VariableWrapper</code> . The <code>variableInfoWrapper</code> is used to determine which calibration or measurement is currently being processed. For more information on how to use the <code>variableInfoWrapper</code> , refer to " <a href="#">MScripts.VariableWrapper</a> " on <a href="#">page 51</a> .

---









### Note



- If the mapped name is not ASAP2 compliant, INCA-SIP will correct this to make it compliant.
- If the mapped name is already used, INCA-SIP will append `_x` where x is 1 to inf.
- The `incaSIPName` is the suggested name and it can also be non-unique and/or non ASAP2 compliant at this stage
- During this hook only getters should be used on the `variableInfoWrapper`. Any changes needed to be effected to this calibration or measurement can be effected in the `setVariableCustomisationFunctionHandle` hook.

For an example of this hook, see the `CustomHooks variableNameMapper.m` script located in the "SIPDEMO" folder.

Variable selection before using a custom hook

All Sources (Not Filtered, 7/7 Visible)		
Name	Role	All_Samples
 <code>_1_D_Lookup_Table</code>		
 <code>CalibrationToFilter</code>		
 <code>LineToFilter</code>		
 <code>OriginalCalibrationName</code>		
 <code>OriginalLineName</code>		
 <code>RealTimeBlock.mode</code>		
 <code>RealTimeBlock.realTimeMultiplier</code>		

Variable selection after using a custom hook

CustomHooks (Not Filtered, 7/7 Visible)		
Name [Display ID]	Role	All_Samples
 <code>_1_D_Lookup_Table</code>	Referenced	
 <code>CalibrationToFilter</code>	Referenced	
 <code>CustomCalibrationName</code>	Referenced	
 <code>CustomLineName</code>	Referenced	
 <code>LineToFilter</code>	Referenced	
 <code>RealTimeBlock.mode</code>		
 <code>RealTimeBlock.realTimeMultiplier</code>		

#### 4.19.4 `setGroupRenamingFunctionHandle`

Each group path for all the calibration or measurement that passes the filter function stage can be renamed in this custom hook. Each unique path will only be passed to the hook once.

## Hook Function Signature

Function	<code>mappedName = groupNameMapper(incaSIPName, type, metadata)</code>	
Parameters	<code>mappedName</code>	New desired mapped name. Cannot be empty.
	<code>incaSIPName</code>	Suggested name from INCA-SIP.
	<code>type char[]</code>	Can be one of the following constants: <code>MScripts.DataModelBase.SimulinkGroup</code> <code>MScripts.DataModelBase.DLLGroup</code>
	<code>metadata</code>	Full path of the group, e.g. <code>/model/subsystem/subsubsystem</code> . The metadata is used to determine the group that is currently being processed.



### Note

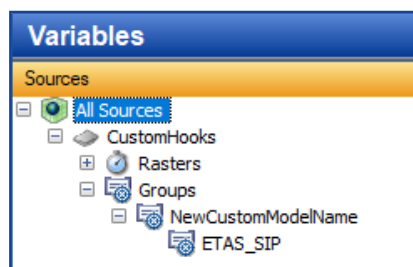
- If the mapped name is not ASAP2 compliant INCA-SIP will correct this to make it compliant.
- If the mapped name is already used INCA-SIP will append `_x` where x is 1 to inf.
- The `incaSIPName` is the suggested name and it can also be non-unique or non-ASAP2 compliant at this stage

For an example of this hook, see the `CustomHooks groupNameMapper.m` script located in the "SIPDEMO" folder.

Variable selection before using a custom hook



Variable selection after using a custom hook



### 4.19.5 MScripts.VariableWrapper

This class is used to wrap calibration and measurements and to allow access for changing specific aspects of its properties.

#### *Properties and functions that can be changed*

Property	sDisplayIdentifier
Description	A2L Display Identifier Entry
Usage with Variable Types	MScripts.DataModelBase.DLLCalibration MScripts.DataModelBase.DLLMeasurement MScripts.DataModelBase.SimulinkCalibration MScripts.DataModelBase.SimulinkMeasurement MScripts.DataModelBase.SimulinkWorkspaceCalibration
Field Type	char []

Property	dMin
Description	A2L variable minimum valued allowed. Used by INCA.
Usage with Variable Types	MScripts.DataModelBase.DLLCalibrationAxis MScripts.DataModelBase.DLLCalibration MScripts.DataModelBase.DLLMeasurement MScripts.DataModelBase.SimulinkCalibrationAxis MScripts.DataModelBase.SimulinkCalibration MScripts.DataModelBase.SimulinkMeasurement MScripts.DataModelBase.SimulinkWorkspaceCalibration
Field Type	number

Property	dMax
Description	A2L variable maximum valued allowed. Used by INCA.
Usage with Variable Types	MScripts.DataModelBase.DLLCalibrationAxis MScripts.DataModelBase.DLLCalibration MScripts.DataModelBase.DLLMeasurement MScripts.DataModelBase.SimulinkCalibrationAxis MScripts.DataModelBase.SimulinkCalibration MScripts.DataModelBase.SimulinkMeasurement MScripts.DataModelBase.SimulinkWorkspaceCalibration
Field Type	number
Property	sDescription
Description	AL2 variable description entry.
Usage with Variable Types	MScripts.DataModelBase.DLLCalibrationAxis MScripts.DataModelBase.DLLCalibration MScripts.DataModelBase.DLLMeasurement MScripts.DataModelBase.SimulinkCalibrationAxis MScripts.DataModelBase.SimulinkCalibration MScripts.DataModelBase.SimulinkMeasurement MScripts.DataModelBase.SimulinkWorkspaceCalibration
Field Type	char[]

Property	<code>numOfDimensions</code>
Description	Number of dimension of the data the variable represents.
Usage with Variable Types	<code>MScripts.DataModelBase.DLLCalibrationAxis</code> <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.DLLMeasurement</code> <code>MScripts.DataModelBase.SimulinkCalibrationAxis</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkMeasurement</code> <code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>
Field Type	number

Property	<code>dataAccessInfo</code>
Description	Used to determine which calibration or measurement this variable wrapper represents.
Usage with Variable Types	<code>MScripts.DataModelBase.DLLCalibrationAxis</code> <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.DLLMeasurement</code> <code>MScripts.DataModelBase.SimulinkCalibrationAxis</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkMeasurement</code> <code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>
Field Type	<code>MScripts.AccessInfo</code>

Function	<code>addGroupPath(&lt;char[] newPath&gt;)</code>
Description	Adds another A2L group to this variable.
Usage with Variable Types	<code>MScripts.DataModelBase.DLLCalibrationAxis</code> <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.DLLMeasurement</code> <code>MScripts.DataModelBase.SimulinkCalibrationAxis</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkMeasurement</code> <code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>
Field Type	<code>MScripts.AccessInfo</code>
Parameter	<p>New A2L group in which this variable is to be shown.</p> <p><code>newPath</code> This path does not need to exist in the Simulink® model. It can be any path.</p>

Function	<code>makeComAxis</code>
Description	Forces axis to be a COM axis in the A2L.
Usage with Variable Types	<code>MScripts.DataModelBase.DLLCalibrationAxis</code> <code>MScripts.DataModelBase.SimulinkCalibrationAxis</code>
Field Type	<code>MScripts.AccessInfo</code>

Property	<code>xAxisAccessInfo</code>
Description	Used to determine which calibration axis this variable element represents.
Usage with Variable Types	<p>Can be used with the following variable types if <code>numOfDimensions &gt;= 1</code>:</p> <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>
Field Type	<code>MScripts.AccessInfo</code>

Property	<code>yAxisAccessInfo</code>
Description	Used to determine which calibration axis this variable element represents.
Usage with Variable Types	Can be used with the following variable types if <code>numOfDimensions &gt;= 2</code> : <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>
Field Type	<code>MScripts.AccessInfo</code>
Property	<code>zAxisAccessInfo</code>
Description	Used to determine which calibration axis this variable element represents.
Usage with Variable Types	Can be used with the following variable types if <code>numOfDimensions &gt;= 3</code> : <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>
Field Type	<code>MScripts.AccessInfo</code>
Property	<code>wAxisAccessInfo</code>
Description	Used to determine which calibration axis this variable element represents.
Usage with Variable Types	Can be used with the following variable types if <code>numOfDimensions = 4</code> : <code>MScripts.DataModelBase.DLLCalibration</code> <code>MScripts.DataModelBase.SimulinkCalibration</code> <code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>
Field Type	<code>MScripts.AccessInfo</code>

**Note**

For all `axisAccessInfo` an exception can also be thrown if the axis in that index is a non-editable axis (Fixed Axis).

- If calls are used on the wrong variable types, exceptions will be thrown.
- `MScripts.AccessInfo` is used to determine which calibration or measurement this variable wrapper represents.

### 4.19.6 MScripts.AccessInfo

This class contains two parameters that are enough to determine which calibration or measurement is being represented. These parameters are `accessParameter1` and `accessParameter2`. For better description, one can make use of more meaningful getters depending on the variable type.

If Variable Type is	<code>MScripts.DataModelBase.SimulinkCalibrationAxis</code>	
	<code>MScripts.DataModelBase.SimulinkCalibration</code>	
Use	<code>getBlockPath</code>	The Block path is the path in the model where the block is located.
	<code>getPropertyName</code>	The property name is the property in the block this calibration is representing.
Note	Helper Function to get the name of the block using the access info: <pre>blockName = MScripts.AccessInfo.getBlockName (&lt;hAccessInfo&gt;);</pre>	
If Variable Type is	<code>MScripts.DataModelBase.SimulinkWorkspaceCalibration</code>	
Use	<code>getBlockPath</code>	The Block path model name.
	<code>getPropertyName</code>	The property name is the name of the workspace variable.



If Variable Type is	<code>MScripts.DataModelBase.SimulinkMeasurement</code>	
Use	<code>getBlockPath</code>	The Block path is the path in the model where the block is located.
	<code>getPortNumber</code>	The port number is the index of the line with respect to the outputs of the block.
Note	Helper Function to get the name of the block using the access info: <code>LineName = MScripts.AccessInfo.getLineName (&lt;hAccessInfo&gt;);</code>	

If Variable Type is	<code>MScripts.DataModelBase.DLLCalibrationAxis</code>	
	<code>MScripts.DataModelBase.DLLCalibration</code>	
	<code>MScripts.DataModelBase.DLLMeasurement</code>	
Use	<code>getDllName</code>	The DLL name is the name of the DLL being used in DLL mode.
	<code>getModuleOffset</code>	The module offset is the memory offset where the variable is located in the DLL. For more information, refer to <a href="#">"DLL Mode" on page 31</a> .

## 4.20 Supported Simulink® Block Types

INCA-SIP supports the following Simulink® block types:

Block Type	Constant
Block Type	Gain
Block Type	Lookup
Block Type	Lookup2D
Block Type	Lookup3D
Block Type	<code>sfun_lookupnd</code>
Block Type	Stateflow
Prerequisites	Parameters - Workspace Variables
Block Type	LookupNDDirect
Prerequisites	<code>NumberOfTableDimensions</code> → 1-4

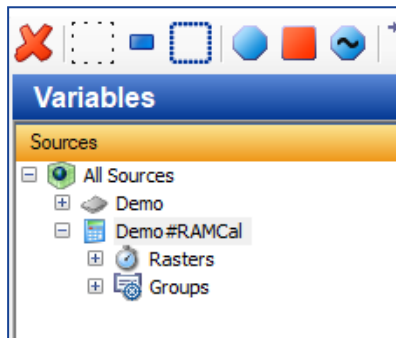
Block Type	S-Function & M-S-Function
Prerequisites	Tunable → on
	Visible → on
	Enabled → on
	ReadOnly → off
Block Type	Interpolation_n-D & S-Function (FunctionName → sfun_kflookupnd)
Prerequisites	TableSource → Dialog
	NumberOfTableDimensions/numDimsPopupSelect → 1-4
Block Type	ManualSwitch
Block Type	PreLookup D & S-Function (FunctionName → sfun_idx-search)
Block Type	Lookup_n-D
Prerequisites	NumberOfTableDimensions → 1-4

## 4.21 Restrictions and Other Behaviors

This section contains a range of tips and information on restrictions and behaviors of INCA-SIP.

### 4.21.1 RAM Calibration

RAM calibration is not supported in INCA-SIP. RAM calibration allows calibration of measurements. You can enable or disable this in the INCA user options. Such calibrations are represented in the **Variable Selection Dialog** as shown in the image below.



### 4.21.2 Parallel Measurements with Other Hardware

When you use INCA-SIP, it is not supported to measure and calibrate with other devices in INCA.

### 4.21.3 Pausing Measurements

If you use the "Pause Recording" function in INCA, you may notice in the subsequent measure data analysis in MDA that there is no data or data has been interpolated over extended areas.

This behavior deviates from that of measure physical hardware in which a paused measurement in the measure file is displayed in the form of breaks of 1s.

## 5 Tutorial

This tutorial presents a simple example in order to show how to operate INCA-SIP. You find the "SIPDemo" model in the following directory:

```
<drive>:\ETASDATA\INCA7.5\Demo\AddOn_INCA-SIP7.5.
```

In this tutorial, you connect a Simulink<sup>®</sup> model to INCA, run the experiment and make changes to the Simulink<sup>®</sup> model.

You can perform the following steps:

- ["To add INCA-SIP to MATLAB<sup>®</sup>" below](#)
- ["To connect INCA-SIP to INCA" below](#)
- ["To calibrate and measure the model in INCA" below](#)
- ["To change the emulation mode" on the next page](#)

### To add INCA-SIP to MATLAB<sup>®</sup>

If you use your MATLAB<sup>®</sup> version with INCA-SIP for the first time, then add INCA-SIP to the MATLAB<sup>®</sup> path. For more information, refer to ["Installation Path of INCA-SIP" on page 15](#).

1. Open MATLAB<sup>®</sup>.
2. Navigate to the "SIPDEMO" folder.
3. To open the Simulink<sup>®</sup> model, double-click **sipdemo.mdl**.

The Simulink<sup>®</sup> editor opens and displays the model.

### To connect INCA-SIP to INCA

1. In the MATLAB<sup>®</sup> **Tools** menu, select **INCA-SIP > Connect to INCA**.

An "ETAS SIP" block is added to the model. INCA is started and a folder within the "INCASIP" database, a workspace, and an ECU project are created.

### To calibrate and measure the model in INCA

1. In the INCA "Database Objects" window area, open the folder `INCASIP/-sipdemo`.

2. Double-click on the workspace  "sipdemo\_wsp".

The INCA experiment environment opens.

3. Select **Variables > Variable Selection**

or


click  or press SHIFT + F4.

The Variable Selection Dialog (VSD) opens.

4. Select the variable "SineWaveOutput".
5. Right-click on the variable and select **Add to... > Layer\_1 > New... > YT-Oscilloscope**.
6. Select the variable "SineWaveAmplitude".

7. Click **OK**.

The calibration and measurement variables are added to a new YT oscilloscope.

8. To start the experiment, click  or press F11.


The simulation in Simulink® and the measurement in INCA is started and displayed in the YT-oscilloscope.

9. To stop the experiment, click  or press F9.

### **Note**

If you change the "SineWaveAmplitude" values in the INCA experiment, the oscilloscope reflects these changes.

#### To change the emulation mode

1. In the INCA experiment environment, select **Variables > Variable Selection** or click  or press SHIFT + F4.

The Variable Selection Dialog (VSD) opens.

2. Select the calibration variable "RealTimeBlock.mode" and "RealTimeBlock.realTimeMultiplier".

The default editor for the scalar variable opens. You can select a default editor via the INCA user options.

3. Change the value of the "RealTimeBlock.realTimeMultiplier" variable to 4.

The simulation runs four times faster than real time.

*or*

Change the "RealTimeBlock.realTimeMultiplier" variable to 0.5.

The simulation runs slower than real time.

*or*

Change the "RealTimeBlock.mode" variable to Fast Emulation Mode.

The simulation runs at its fastest level.

## 6 Contact Information

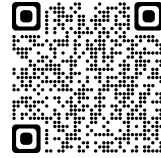
### Technical Support

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

[www.etas.com/hotlines](http://www.etas.com/hotlines)

ETAS offers trainings for its products:

[www.etas.com/academy](http://www.etas.com/academy)



### ETAS Headquarters

ETAS GmbH

Borsigstraße 24	Phone:	+49 711 3423-0
70469 Stuttgart	Fax:	+49 711 3423-2106
Germany	Internet:	<a href="http://www.etas.com">www.etas.com</a>

# Index

<b>B</b>	
Basic Operation .....	10
Block Reduction .....	29
Busy ECU .....	30
<b>C</b>	
C++ Compiler .....	14
Calibrate and Measure the Model .....	60
calibration	
create .....	35
Calibration	
Model .....	10
RAM .....	59
Refusing .....	30
Calibrations .....	35
Calibrations and Measurements .....	28
Connection	
Simulink to INCA .....	9
Contact Information .....	62
Conventions	
Naming .....	28
Create	
calibrations .....	35
Create object .....	35
Create Object .....	35
Creating	
DLL Search Script .....	33
Object .....	35
Custom Hook .....	44
<b>D</b>	
Data .....	8
Directories	
Installation Files .....	15
Directories and Files .....	15
DLL	
Type .....	32
DLL Mode .....	31
DLL Search Script .....	33
<b>E</b>	
ECU	
Adding a new .....	19
Busy .....	30
Emulation Mode .....	28
Fast .....	29
How to change .....	61

Real-Time .....	28
ETAS	
Contact Information .....	62
Experiment	
Start .....	30
Stop .....	30
<b>F</b>	
Fast Emulation Mode .....	29
Files	
MATLAB Script .....	15
Functions	
addGroupPath .....	40
createArrayMeasurement .....	38
createCalibrationAxis .....	36
createCalibrationAxisWith MeasurementKey .....	37
createFixedAxis .....	36
createScalarCalibration .....	35
createScalarMeasurement .....	38
createTableCalibration .....	35
set2AsapMaxValue .....	39
setAsap2Description .....	39
setAsap2MinValue .....	39
setAxis .....	37
<b>G</b>	
Groups .....	28
<b>H</b>	
Handling Calibrations .....	13
Handling Measurements .....	12
Hardware .....	14
<b>I</b>	
INCA-SIP	
Add to MATLAB .....	60
and Block Reduction .....	29
Behaviors .....	59
Emulation Mode .....	28
Installation .....	14
Restrictions .....	59
Installation	
INCA-SIP .....	14
Installation Path .....	15
<b>M</b>	
Macros .....	27
MATLAB	
Add INCA-SIP .....	60
Block Reduction .....	29
Refuse Calibrations .....	30



Search Path .....	17
Simulation Modes .....	29
Script Files .....	15
MDF .....	25
Measure and Calibrate the Model .....	60
Measurement	
Model .....	10
Pausing .....	59
with Other Hardware .....	59
Measurements and Calibrations .....	28
Menu Item	
Configuration .....	23-24,42
Menu Items .....	17
Menu Item	
Connect to INCA .....	17
Mode	
DLL .....	31
Model	
Calibration .....	10
Measurement .....	10
MScripts.AccessInfo .....	56
MScripts.VariableWrapper .....	51
<b>N</b>	
Naming	
Conventions .....	28
<b>O</b>	
Object	
create .....	35
<b>P</b>	
Parallel Measurements with Other Hardware .....	59
<b>R</b>	
RAM Calibration .....	59
Real-Time Emulation Mode .....	28
Referenced Models Support .....	30
Refusing Calibrations .....	30
Registering	
C++ Compiler .....	14
<b>S</b>	
setGroupRenamingFunctionHandle .....	49
setVariableCustomisationFunctionHandle .....	46
setVariableRenamingFunctionHandle .....	48
Simulation Modes .....	29
Simulink	
Connection to INCA .....	9
Referenced Models .....	30
Software .....	14

Support	
Simulink.Parameter and Asap2.Parameter .....	31
Simulink.Signal and Asap2.Signal .....	31
Synchronization .....	12
System Requirements .....	14
Hardware .....	14
Software .....	14
<b>T</b>	
Technical Overview .....	10
Basic Operation .....	10
Components .....	11
Connection .....	11
Handling Calibrations .....	13
Handling Measurements .....	12
Synchronization .....	12
Tutorial .....	60