

**LABCAR**

**-**

**AUTOMATION**

**4**

**.2**

**.2**

How to …? Frequently asked questions

–

Tips & Tricks

**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 2012 - 2015 ETAS GmbH, Stuttgart

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

2

Contents

**Contents**

[1 Introduction 5](#_Toc67324)

[1.1 Conventions 5](#_Toc67325)

[1.2 Installation Paths 6](#_Toc67326)

[2 For Test Case Developers 7](#_Toc67327)

[2.1 How do I write test cases? 7](#_Toc67328)

[2.2 Offline Test Case Design 7](#_Toc67329)

[2.3 Test Case Design and Debugging within Visual Studio 11](#_Toc67330)

[2.4 Test Case Design without Test Management tools (Embeddable Package) 12](#_Toc67331)

[2.5 Automated Label Mapping 13](#_Toc67332)

[2.6 Intermediate close of INCA Experiment 13](#_Toc67333)

[2.7 Building Test Case Definition (.tcd) File 13](#_Toc67334)

[3 For Test Bench Configuration Responsibles 14](#_Toc67335)

[3.1 Configuration Wizard 14](#_Toc67336)

[3.1.1 Typical Use Case 14](#_Toc67337)

[3.1.2 Features of the Configuration Wizard 14](#_Toc67338)

[3.2 Test Bench Configurations can be fully determined by “Unit under Test” 15](#_Toc67339)

[3.3 The SUT Mapping Editor 16](#_Toc67340)

[3.4 Access to INCA with or without LABCAR-OPERATOR 16](#_Toc67341)

[3.4.1 Using INCA with LABCAR-OPERATOR 17](#_Toc67342)

[3.4.2 INCA Standalone 18](#_Toc67343)

[3.4.3 Necessary Installations for the use of INCA standalone 18](#_Toc67344)

[3.5 Flashing with INCA and PROF 20](#_Toc67345)

[3.5.1 Pre-requisitions 20](#_Toc67346)

[3.5.2 Preparing the test bench configuration 20](#_Toc67347)

[3.5.3 Lines in code 24](#_Toc67348)

[3.6 Parameter recording for Sync DL 25](#_Toc67349)

[3.7 Failure Simulation Load Cut off 26](#_Toc67350)

[3.8 Connection to dSpace Test Bench 27](#_Toc67351)

[4 For Testers 29](#_Toc67352)

[4.1 Changing the layout of the report 29](#_Toc67353)

[4.1.1 Changing report layout after creation per report 29](#_Toc67354)

[4.1.2 Changing report layout for all reports 29](#_Toc67355)

[5 Complex features 30](#_Toc67356)

[5.1 Offline Project generation 30](#_Toc67357)

[5.1.1 Typical Use Case 30](#_Toc67358)

[5.1.2 Features of the Project Generator 30](#_Toc67359)

[5.1.3 Feature Handling 30](#_Toc67360)

[5.2 Report Structure ‘Abstract Section’ 31](#_Toc67361)

[5.2.1 Feature Description 31](#_Toc67362)

[5.2.2 Test case development for use of the Abstract Section 31](#_Toc67363)

[5.2.3 Report View with Abstract Section 33](#_Toc67364)

[5.3 Improved Data logging functionality 33](#_Toc67365)

[5.3.1 Typical use case 33](#_Toc67366)

LABCAR-AUTOMATION 4.2.2 How to …? Frequently asked questions – Tips & Tricks 3 Contents

5.3.2 Feature Description .................................................................................................... 34 5.3.3 Feature Handling ....................................................................................................... 34

5.3.4 Test bench configuration for LCO project dataloggers .................................................. 35

5.3.5 Hints ......................................................................................................................... 35

5.4 Maps and Curves ............................................................................................................ 35 5.4.1 Typical use case ......................................................................................................... 35

5.4.2 Feature Description .................................................................................................... 35 5.4.3 Feature Handling ....................................................................................................... 35

5.4.4 Mapping rules ............................................................................................................ 36

5.4.5 Example .................................................................................................................... 36

5.5 Real Time Tests support ................................................................................................. 38

5.5.1 Typical use cases ....................................................................................................... 38

5.5.2 Feature Description .................................................................................................... 38

5.5.3 Feature handling ........................................................................................................ 39

5.5.4 Test case development for Real Time testing ............................................................... 40 5.5.5 Test bench configuration for Real Time testing ............................................................ 40

5.5.6 Hints ......................................................................................................................... 40

5.6 Soft-Stop Function for Test cases .................................................................................... 41

5.6.1 Typical use cases ....................................................................................................... 41

5.6.2 Feature Description .................................................................................................... 41 5.6.3 Feature Handling ....................................................................................................... 41 5.6.4 Test case development for Soft Stop ........................................................................... 41 5.6.5 Test bench configuration for Soft Stop ........................................................................ 42

5.6.6 Hints ......................................................................................................................... 42

5.6.7 Difference between status shown in Test Handler and Verdict reported by test case ...... 42

5.7 Error Manager ................................................................................................................ 43

5.7.1 Typical Use Cases ...................................................................................................... 43 5.7.2 Feature Description .................................................................................................... 43 5.7.3 Feature Handling ....................................................................................................... 43

5.7.4 Test Case development with Error Manager ................................................................. 43

5.7.5 Test Bench Configuration for Error Manager ................................................................ 45

5.7.6 Hints ......................................................................................................................... 45

5.8 Working with a Signal Generator ..................................................................................... 45

5.8.1 Typical use cases ....................................................................................................... 45

5.8.2 Feature Description .................................................................................................... 45 5.8.3 Feature Handling ....................................................................................................... 46

5.8.4 Test case development for the Signal Generator .......................................................... 47 5.8.5 Test bench configuration for the Signal Generator ........................................................ 48

5.8.6 Mappings ................................................................................................................... 50

5.8.7 Working with dSpace .................................................................................................. 50

5.8.8 Example with LABCAR-OPERATOR 3.2.5 ...................................................................... 51

1. General issues ....................................................................................................................... 52
   1. Test Handler Tool Options – across different installations ................................................. 52
   2. Silent Installation ........................................................................................................... 52
   3. License Management ...................................................................................................... 54
2. ETAS Contact Addresses ........................................................................................................ 55

LABCAR-AUTOMATION 4.2.2 How to …? Frequently asked questions – Tips & Tricks 4 Introduction

# 1 Introduction

## Этот документ поможет вам

## 

##  настроить вариант использования,

##  получение ответов на часто задаваемые вопросы,

##  находить полезные подсказки при попадании в беду,

##  описание сложных функций.

## 

## Всякий раз, когда вам нужна информация об использовании LABCAR-AUTOMATION, этот документ должен стать вашей первой отправной точкой для поиска решения вашей проблемы.

## Этот документ сгруппирован в различные разделы, касающиеся функциональных ролей автоматизированного процесса тестирования и общих вопросов, касающихся, например.

## 1.1 Conventions

Formatting of dialog elements

If names of entry fields used by a dialog are used in the documentation they are written in italic letters.

E.g: Folder, Device, Database

Formatting of entries

Entries to be made in dialog fields are written in **bold** letters.

Introduction

## 1.2 Installation Paths

All installation paths in this document refer to a **Windows 7** (64bit) system environment. The standard directories in this environment are:

**Program Files:**

C:\Program Files (x86)\ETAS\LABCAR-AUTOMATION 4.x

**Configuration Files:**

C:\ProgramData\ETAS\LABCAR-AUTOMATION 4.x

**Examples and Default Data:**

C:\Users\Public\Documents\ETAS\LABCAR-AUTOMATION 4.x

When working in a **Windows XP** environment, these directories are located at:

**Program Files:**

C:\Program Files\ETAS\LABCAR-AUTOMATION 4.x

**Configuration Files:**

C:\Documents and Settings\All Users\Application Data\

ETAS\LABCAR-AUTOMATION 4.x

**Examples and Default Data:**

C:\Documents and Settings\All Users\Documents\

ETAS\LABCAR-AUTOMATION 4.x

For a more detailed explanation please have a look into the LABCAR-AUTOMATION User’s guide, chapter: 2.4.1 the “\Users\Public\Documents\ETAS” Folder.

# 2 For Test Case Developers

## 2.1 Прежде всего, прежде чем выполнять тесты, их необходимо создать. Для этого есть большие возможности.

## 

##  Использование построителя последовательности автоматизации

##  Пожалуйста, ознакомьтесь с руководством пользователя в главе 4.1 Конструктор последовательности автоматизации.

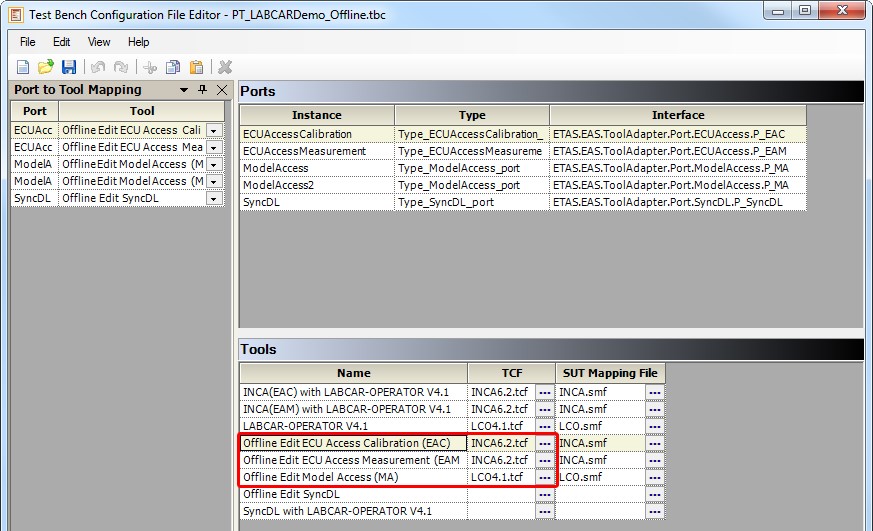
## 2.2 Offline Test Case Design

Чтобы уменьшить дорогостоящее использование тестового стенда, существует возможность разработать код тестового примера с помощью виртуального тестового стенда, который называется Offline Test Bench. Автономные испытательные стенды имитируют поведение реального испытательного стенда. Тестовые примеры или полезные функции, использующие сигнатуры ATCL, можно создавать, запускать и проверять без полноценной тестовой системы.

Поведение полной тестовой системы соответствует реальной. Передается вся цепочка инструментальных адаптеров, например сопоставления SUT, преобразование данных и преобразование единиц измерения. Только окончательные вызовы API к инструментам тестирования подделываются с помощью текстовых файлов, содержащих ожидаемое поведение тестового стенда, или диалогов ввода, возвращающих ожидаемые значения.

 Этот механизм можно использовать для выполнения тестовых последовательностей в автономном режиме из конструктора последовательности автоматизации, а также тестовых случаев из обработчика тестов

«Автономный тестовый стенд» указывается в редакторе файлов TBC, как и любой другой тестовый стенд, путем выбора «Автономных инструментов». Эти инструменты заменяют реальные инструменты.

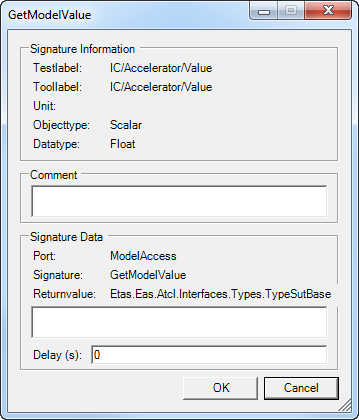


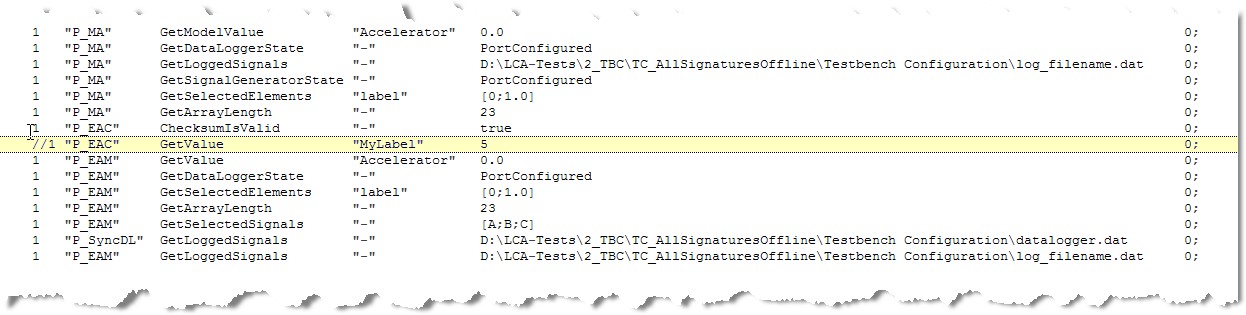
For Test Case Developers

Есть два способа предоставления ожидаемых значений.

 Для одиночных значений наиболее подходящим методом является ввод на консоли тестировщика через диалог.

 Более часто ожидаемые возвращаемые значения предпочтительно описываются в текстовом файле, который называется «автономный файл». В этом файле каждая строка представляет одну запись для одного возвращаемого значения.





Пример с вызовами всех автономных подписей

**Offline file design**

Автономный дизайн файла

 Столбцы в строке (разделенные табуляторами):

Повторение: количество использований строки.

Порт: порт, на котором вызывается подпись

Подпись: подпись, для которой требуется возвращаемое значение.

Метка: метка, для которой должно быть получено значение.

Возвращаемое значение: значение/значения, данные, которые должны быть возвращены.

Задержка возврата: время, которое потребуется, по крайней мере, для возврата

 Комментирование разрешено («//»),

 Допускается включение других автономных файлов («#include»)

 С помощью { } можно группировать строки. Число перед фигурной скобкой — это повтор этой группы

**Possible values in offline file and the dialogue**

* Возможные значения для всех состояний получения подписей:

PortAny

PortLoading

PortError

PortUndefined

PortStarting

PortCompleted

PortCreated

PortPaused

PortToolConfigured

PortConfigured

PortRecording

PortLLConfigured

PortRunning

PortStopped

PortHLConfigured

PortClosed

PortReset

PortListening

PortStateUnknown

*  Значения для сигнатур, извлекающих значения в виде массива, должны быть указаны в следующем формате: [0;1.0]

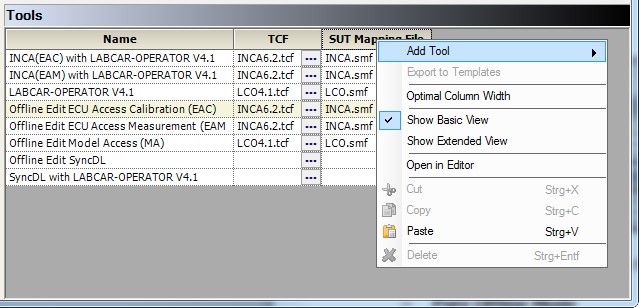
[A;B;C]

*  Сигнатура GetLoggedSignals ожидает путь к файлу регистратора данных «.dat», содержащему значения.

**Test Bench configuration for Offline tests**

Автономный файл указывается дополнительно к автономному инструменту в редакторе TBC для каждого инструмента. Имя файла необходимо ввести в столбец «Файл конфигурации адаптера», выбрав расширенный вид с помощью правого щелчка мыши.

В одном файле может храниться поведение многих инструментов, поэтому на один и тот же файл могут ссылаться разные инструменты..



Создание и использование автономных файлов реализовано в различных режимах:

**Pure Offline Mode**

Адаптеры исключительно автономных инструментов будут воспроизводить автономный файл для каждой подписи. В случае несоответствия файла и тестового примера создается ошибка, похожая на реальную ошибку тестового стенда.

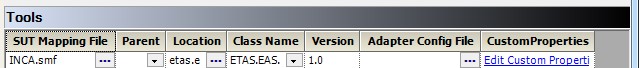
**Offline/Edit Mode**

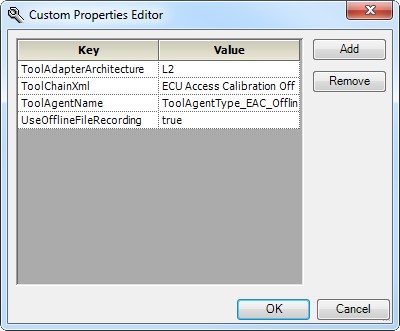
Этот режим используется для создания/изменения новых или существующих автономных файлов с помощью адаптера инструмента «Автономно/Редактировать».

Если возникает несоответствие между вызываемой подписью и файлом, пользователь указывает желаемое поведение в диалоговом окне, и файл расширяется.

**Offline Recording**

Третий режим предназначен для создания автономного файла. В этом режиме необходимые данные записываются посредством тестового прогона в режиме онлайн. Запись должна быть включена для онлайн-инструмента в пользовательских свойствах в редакторе файлов TBC.





Воспроизведение такого созданного автономного файла можно запустить в чистом автономном режиме.

«Offline», «Offline/Edit» и «Offline Recording» — адаптер инструмента доступен со следующими портами:

**Model Access**

GetLoggedSignals, GetValue, GetSelectedElements, GetSelectedElementsLength, GetSignalGeneratorState, GetDataLoggerState

**ECU Access Measurement**

The offline feature of this port is realized for LCO 4.1 and higher only

GetLoggedSignals ,GetValue, GetSelectedElements, GetSelectedElementsLength, GetDataLoggerState

**“ECU Access Calibration”**

The offline feature of this port is realized for LCO 4.1 and higher only

ChecksumIsValid, GetValue

“Offline” and “Offline/Edit” – Tool adapter are available with following port as well:

**“Synchronized Logging”**

P\_SyncDLGetLoggedSignals, getState

All “Set\*” – signatures are assumed to be successful on an Offline Test Bench

## 2.3 Дизайн тестовых случаев и отладка в Visual Studio

Запуск и отладка тестовых случаев из Visual Studio («F5») полностью поддерживается.

Он включает спецификацию конфигураций испытательного стенда и расположение файла отчета.

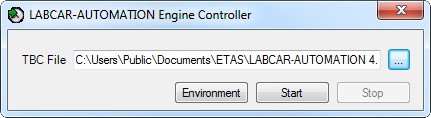
Для этого LABCAR-AUTOMATION Engine Controller работает как служба Windows, к которой подключается разработчик тестового примера.

Чтобы запустить контроллер двигателя LABCAR-AUTOMATION, выберите его в меню «Пуск»:

**Start**  **Programs**  **ETAS**  **LABCAR-AUTOMATION 4.x**  **Test Design (ATCL)** 

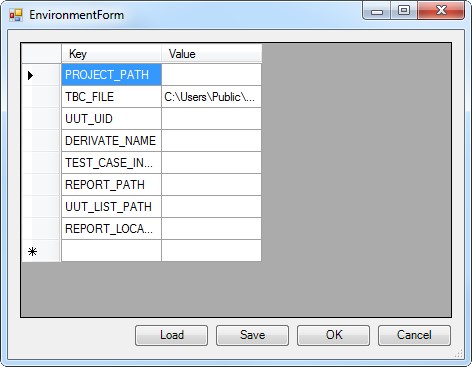
**LABCAR-AUTOMATION Engine Controller**

Контроллер двигателя LABCAR-AUTOMATION размещает значок на панели задач Windows®. С помощью этого значка пользователь должен двойным щелчком указать необходимую Конфигурацию испытательного стенда (TBC), которую ЛАБКАР-АВТОМАТИЗАЦИЯ будет использовать для испытания. Обычно это автономная конфигурация тестового стенда, поскольку у разработчика могут быть установлены не все необходимые инструменты.

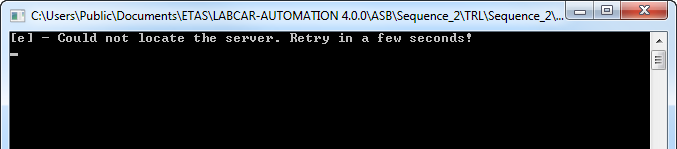


Чтобы узнать, какие значения установлены в выбранном TBC, нажмите кнопку «Среда».

Как только файл TBC выбран, запустите LABCAR-AUTOMATION Engine Controller с помощью соответствующей кнопки. TBC можно переключить, только если контроллер двигателя LABCAR-AUTOMATION остановлен..



Если LABCAR-AUTOMATION Engine Controller не был запущен, вы получите сгенерированную Windows® ошибку, как показано ниже, как только вы запустите свой тестовый пример:



## 2.4 Дизайн тестовых случаев без инструментов управления тестированием (встраиваемый пакет)

 Процедура тестового выпуска упрощена. Генератор TCD может работать в пакетном режиме для автоматического создания файлов библиотеки тестовых случаев.

 Полностью интегрирован Microsoft® .NET Framework версии 2.0.

 Благодаря улучшениям в интерфейсе ATCL «ParameterManager» полностью параметризованные наборы параметров по умолчанию могут быть созданы непосредственно в тестовом сценарии..

## 2.5 Automated Label Mapping

Когда во время выполнения обнаруживается тестовая метка, которая не была сопоставлена ни в одном из файлов сопоставления SuT, соответствующих используемому инструменту, она будет автоматически сопоставлена с той же меткой данного файла a2l. Имя метки, тип и диапазоны заимствованы из a2l.

Если тестовая метка была активно сопоставлена в файле сопоставления, это сопоставление является основным. Благодаря этому исходная функциональность ваших существующих тестовых случаев гарантирована.

## 2.6 Intermediate close of INCA Experiment

В случае длительных тестовых случаев может быть целесообразно временно закрыть эксперимент INCA. Это обеспечивает стабильность таких более длительных тестов. Возобновление эксперимента возможно с помощью сигнатур порта EAM StopCommunication и StartCommunication. Затем необязательный параметр содержит предыдущий закрытый эксперимент.

Эти методы изменяют состояние порта. После вызова StopCommunication состояние «PortToolConfigured», а после StartCommunication состояние «PortConfigured».”.

## 2.7 Building Test Case Definition (.tcd) File

Файл определения тестового примера необходим для использования тестовых примеров (например, .exe или .py) внутри проектов LABCAR-AUTOMATION. Он содержит ссылку на все файлы структурирования, так как есть:

* Test Parameter File (.tpa)
* Test hierarchy definition (.thd)
* Test architecture definition (.tad)

Также в файл .tcd могут быть включены дополнительные используемые библиотеки динамической компоновки.

Файлы .tpa, .thd и .tad создаются при первом выполнении файла .exe тестового примера. Это можно сделать вне любого инструмента (например, обработчика тестов), исполняемый файл можно запустить двойным щелчком мыши напрямую. Файлы создаются автоматически в той же папке, что и исполняемый файл.

Имейте в виду, что исполняемый файл тестового примера также может использовать функции отчета. Если в то же время активен запуск обработчика тестов, эти функции отчета могут привести к перезаписи отчета, который фактически создается при выполнении теста обработчика тестов.

Пожалуйста, не запускайте исполняемый файл тестового примера параллельно с тестовыми запусками обработчика тестов!

С редактором TCD у вас есть графический пользовательский интерфейс для сбора файлов определений для вашего тестового примера и создания файла .tcd, или вы можете использовать приложение TCDCMD.exe, которое представляет собой инструмент командной строки, генерирующий файл TCD без показа мастера.

Для получения более подробной информации обратитесь к ATCL Getting Started.pdf, глава 4.4.1 «Файл определения тестового примера» и далее.

# 3 For Test Bench Configuration Responsibles

## 3.1 Configuration Wizard

### 3.1.1 Typical Use Case

LABCAR-AUTOMATION — это хорошо структурированный и достаточно полный набор инструментов. Многие варианты возможны и поддерживаются этой коллекцией. Однако высокая гибкость требует определенных усилий для настройки набора инструментов, чтобы он соответствовал собственным требованиям.

Настройка LABCAR-AUTOMATION немного сложна. Особенно, когда используется более или менее статическая композиция используемого испытательного стенда, должен быть быстрый и интуитивно понятный метод для достижения полной и правильной конфигурации.

### 3.1.2 Features of the Configuration Wizard

Руководство по всем необходимым действиям перед запуском тестового примера LABCAR-AUTOMATION на ПК является главной задачей этого мастера.

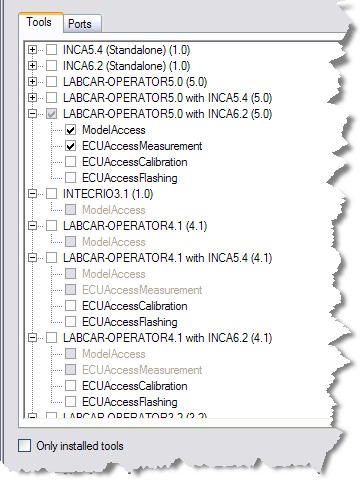
С помощью мастера настройки

 создается полная конфигурация испытательного стенда

 создается тестовый пример по умолчанию для дальнейшего редактирования

 создается тестовый проект по умолчанию с использованием тестового примера по умолчанию и созданного тестового стенда.

Основой для конфигурации тестового стенда, которая обрабатывается Мастером настройки, являются в первую очередь установленные инструменты. В зависимости от лицензируемого пакета ЛАБКАР-АВТОМАТИЗАЦИЯ набор настраиваемых инструментов распространяется на все инструменты, поддерживаемые ЛАБКАР-АВТОМАТИЗАЦИЯ.



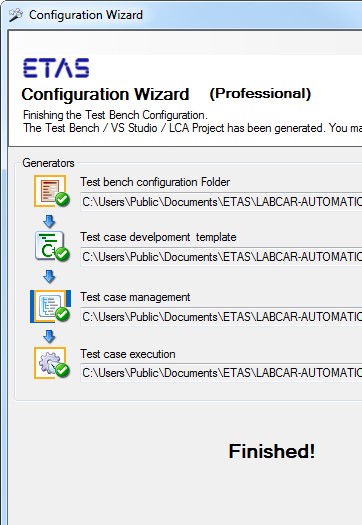
For Test Bench Configuration Responsibles

Большинство записей предварительно сконфигурированы либо с примерами файлов или папок, либо с записями по умолчанию соответствующего инструмента. Кроме того, мастер настройки поддерживает другие записи с несколькими функциями поиска. Все возможные записи фильтруются в соответствии с допустимым расширением. В случае конфигурации соединений INCA он ищет необходимые записи в данной базе данных INCA.

После выбора всех инструментов и предоставления всех необходимых входных данных создается конфигурация испытательного стенда.

Следующий шаг предоставляет тестовый пример по умолчанию и тестовый проект с использованием именно этого тестового стенда. В коде тестового примера доступны все выбранные инструменты тестового стенда для подготовки тестового примера к использованию портов и подписей инструмента. Диспетчер тестирования, запущенный из мастера настройки нажатием кнопки, использует именно этот тестовый пример по умолчанию.

Запуск обработчика тестов из мастера настройки, он также предварительно настроен с помощью созданного тестового стенда.

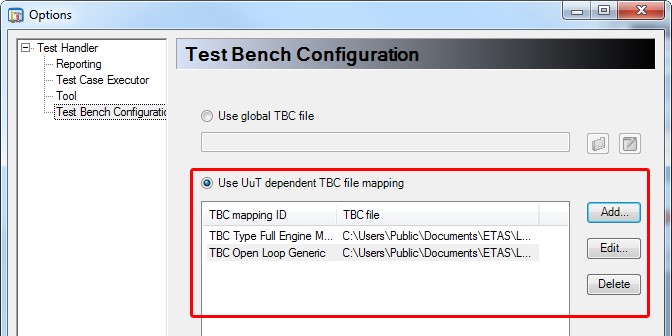


Таким образом, как только мастер настройки завершит свою работу, заказчик сможет запустить свой первый тестовый проект на работоспособном тестовом стенде..

The complete detailed instruction manual is placed in the user’s guide at chapter ‘4.5.1 The Configuration Wizard.

## 3.2 Test Bench Configurations can be fully determined by “Unit under Test”

Пользователи могут указать конфигурацию испытательного стенда в зависимости от UUT, выбранного для тестирования:





«Идентификатор сопоставления TBC» может быть указан в среде каждого UUT.

Если активировано, обработчик тестов сопоставляет «TBC ID» с конкретной конфигурацией тестового стенда, используемой для этого теста.

API сервера UuT расширен, чтобы позволить автоматическую настройку «Идентификатора TBC»

## 3.3 The SUT Mapping Editor

Редактор отображения SUT используется как LABCAR-OPERATOR 5.x, так и LABCAR-AUTOMATION V3.3 ff., и может работать с обоими форматами файлов, текстовым файлом LABCAR-OPERATOR и форматом LABCAR-AUTOMATION.

## 3.4 Access to INCA with or without LABCAR-OPERATOR

К INCA можно обратиться из тестового примера с использованием или без использования LABCAR-OPERATOR.

Используя как INCA, так и LABCAR-OPERATOR, вы можете использовать дополнительные функции, например. функции синхронной регистрации данных, которые синхронизируют метки времени обоих журналов (модели и ЭБУ).

Использование без LABCAR-OPERATOR называется «автономным INCA».

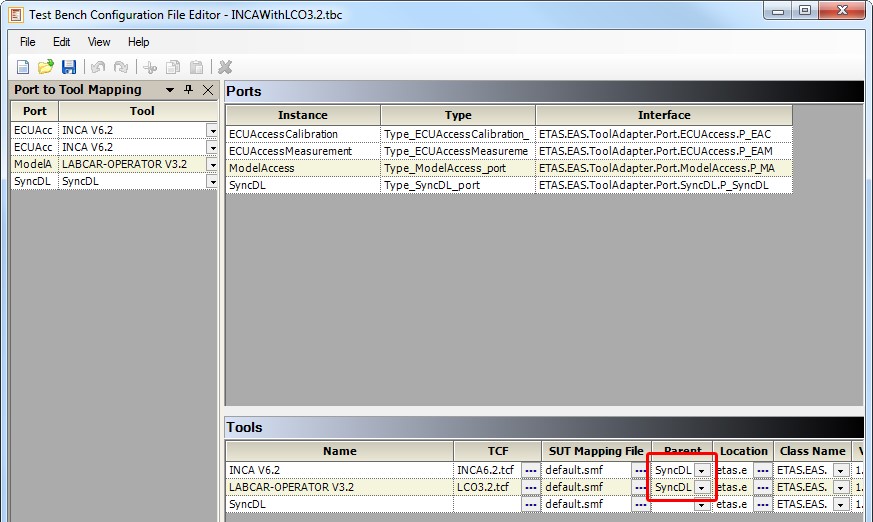
В следующих примерах показаны различные записи и описания конфигурации. Поскольку это имеет доступ только к ECU, синхронизация меток времени не требуется и невозможна.

### 3.4.1 Using INCA with LABCAR-OPERATOR

При использовании INCA вместе с LABCAR-OPERATOR вы должны сначала посмотреть на свою версию LABCAROPERATOR. Обработка до LABCAR-OPERATOR V3.2.5 отличается от более новых версий.

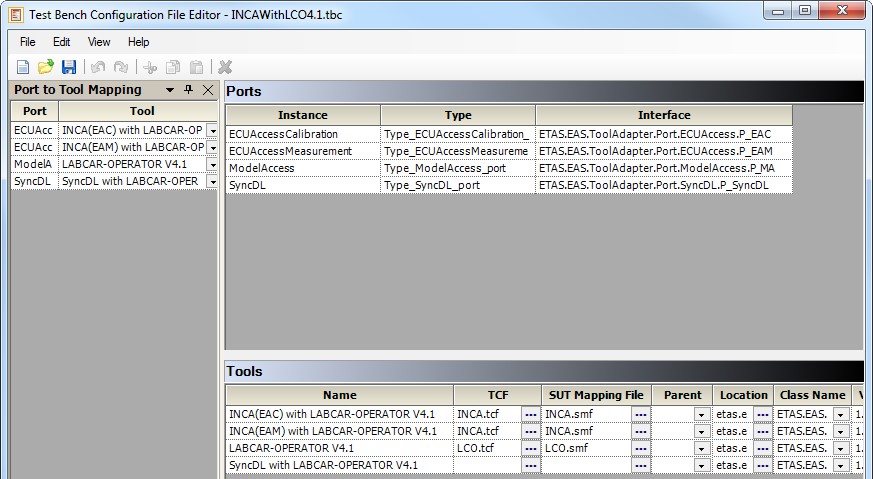
Если оба, INCA и LABCAR-OPERATOR, установлены, и ваш тестовый пример использует порт ModelAccess для LABCAR-OPERATOR и порт ECUAccess для INCA, конфигурации подобны показанным ниже.

* **LCO3.2.5** and **INCA5.4** or **INCA6.2** are installed Here you see a possible Test Bench:



* + If you like to use the synchronous datalogging feature, then for **INCA** and **LCO** Tool in the parent column the **parent** port **P\_SyncDL** have to be selected.
  + The name of the ports (e.g. P\_EAC and P\_MA) have to fit to your test case names.

* **LCO4.1** (or higher) and **INCA5.4** or **INCA6.2** are installed



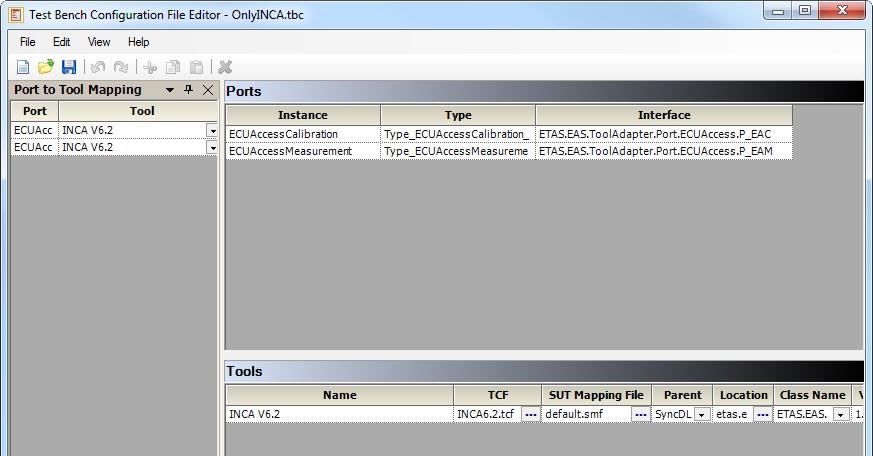
** Порт ModelAccess для LABCAR-OPERATOR должен быть настроен до настройки порта ECUAccess для INCA внутри тестового примера.**

** Если вы используете версию LABCAR-OPERATOR выше V4.1, пожалуйста, соответствующим образом измените номер версии в приведенных выше примерах**.

### 3.4.2 INCA Standalone

Использование INCA без LABCAR-OPERATOR называется «INCA Standalone», так как LABCAROPERATOR не должен быть установлен на вашем ПК.

Тестовый пример только с использованием порта ECUAccess (INCA), здесь возможный тестовый стенд.



* Поскольку доступен только один источник сигнала, синхронная регистрация данных здесь не предусмотрена.

 В случае, если также установлен ЛАБАВАР-ОПЕРАТОР V3.2.5, всплывает окно ЛАБКАВ-ОПЕРАТОР и тут же закрывается. Это не повредит исполнению без LCO.

 Даже если на вашем ПК также установлен LABCAR-OPERATOR V4.1, НЕ используйте шаблон инструмента «INCA(EAx) with LABCAR-OPERATOR» Vx.x, так как это приведет к ошибке.

### 3.4.3 Necessary Installations for the use of INCA standalone

Both, INCA and LABCAR-OPERATOR, are installed

* INCA5.4/INCA6.2. and LCO5.0 ff are installed
* Install INCA Standalone from the CD.

* INCA5.4/INCA6.2. and LCO4.1 are installed
  + - 1. Install INCA Standalone from the CD.
      2. Unregister the old API. Default Installation Path: “C:\Program

Files (x86)\ETAS\LABCAR-OPERATOR4.1\LABCAR-API” (regsvr32

/u /s “<LCO4.1Installation PATH>\LCO3API.dll”)

* + - 1. You will no longer be able to use synchronous data logging, though you still can log both signal sources. But you have to treat them as individual, separate data source.

* INCA5.4/INCA6.2. and LCO 3.2.x are installed on your PC and you want to measure with INCA without LABCAR-OPERATOR:
* No further Installation is necessary.

Use of INCA5.4/INCA6.2 without installed LABCAR-OPERATOR application

* + - 1. Install INCA Standalone from the CD.
      2. Copy all dlls from installation CD folder:

\Data\ThirdParty\\_AVC++7SP1.Redistributables to your ETAS directory of shared components: C:\ETAS\LABCAR-CCIStandalone3.2\System32

Use of INCA7.0/INCA7.1 without installed LABCAR-OPERATOR application

1. Install INCA Standalone from the CD.

(\Data\INCA Standalone\INCAXX\INCAAddOn\_XML4LabCar.exe) depends on the INCA Version.

3.5 Standalone Diagnostic with INCA 7.1

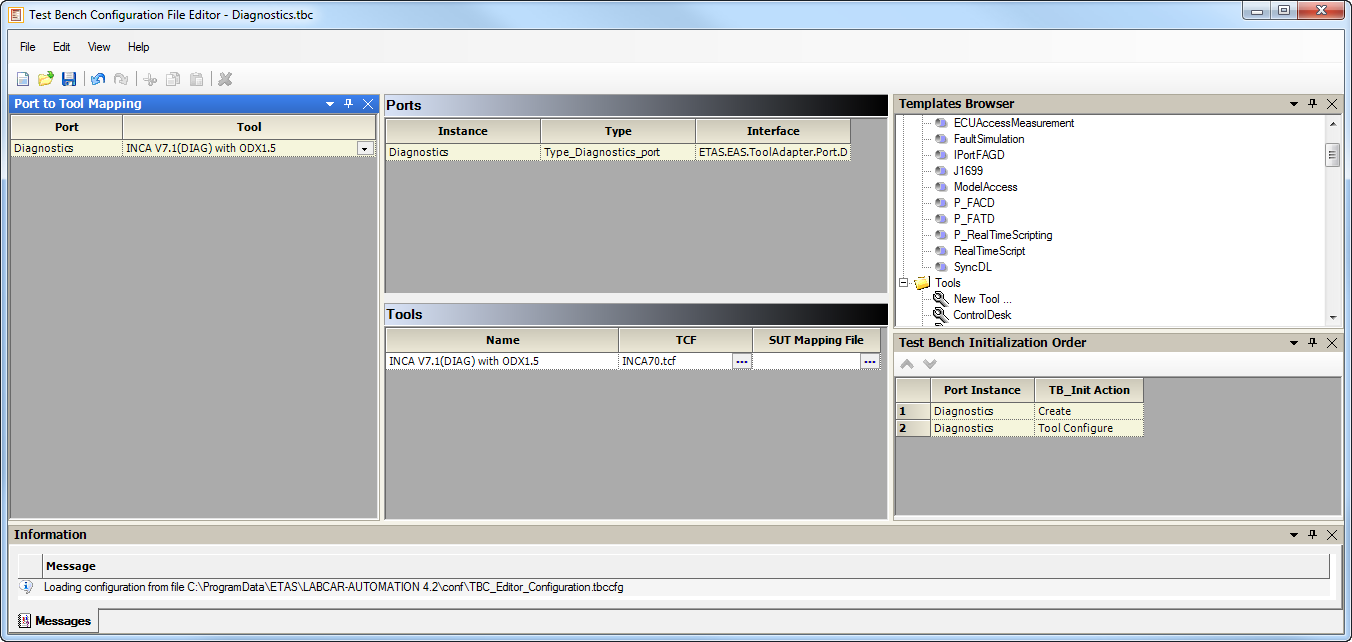
3.5.1 Pre-requisitions

A working installation of INCA and ODX-Link must be available on the computer.

3.5.2 Preparing the test bench configuration

Test bench configuration file

In your test bench configuration file you will need to define an instance of a Diagnostic port and connect this Port to the Tool ‘INCA V7.1(DIAG) with ODX-LINK’



**Hint:**

It is not possible to use the tool ‘ODX (Diagnostics) with LABCAR-OPERATOR V5.x.’. See next chapter.

3.6 Standalone Diagnostic with LABCAR-OPERATOR

In the older LABCAR-AUTOMATION version it was possible to use the tool ‘ODX

(Diagnostics) with LABCAR-OPERATOR V5.x.’ also for a standalone Diagnostic. That means that we are using the Diagnostic Port without a Model Access from LABCAR-OPERATOR. This functionality will be removed in one of the next version. If you still need this feature modify one/all of ODX files:

%ProgramData%\ETAS\LABCAR-AUTOMATION

4.2\conf\TBC\ETAS\Tools\ODX\_DiagnosticsPort\_EE[xx].xml



## 3.7 Flashing with INCA and PROF

### 3.7.1 Pre-requisitions

A working installation of INCA and the PROF Tool must be available on your personal computer. In addition all necessary accessories to flash the ECU must be available, i.e. hex or s19 together with a2l-file, an INCA device to connect to and flash the ECU, like ES59x, and an INCA database providing a fitting workspace. For the PROF-Tool a configuration is necessary.

**Hint:**

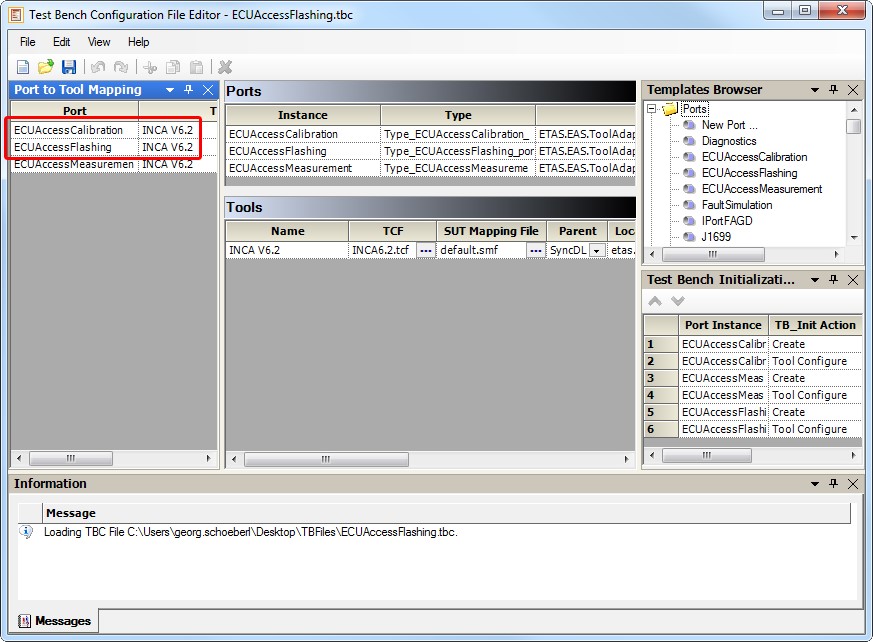
Make sure the ECU can be flashed by hand with the INCA database, workspace and PROFConfiguration you are going to use for your automated flashing.

### 3.7.2 Preparing the test bench configuration

Test bench configuration file

In your test bench configuration file you will need to define an instance of an ECU Access Flash port.

Make sure you choose the right INCA version, which is working with your PROF version smoothly.



Tool configuration file

In your tool configuration file you have to give the same information you provide for any other ECU access. As there are (see Figure 1)

* the INCA Database to open
* Used Measurement Device
* Used Experiment
* Folder in database containing the used experiment
* Used Workspace
* Folder in database containing the used workspace
* SuT Mapping File to be used

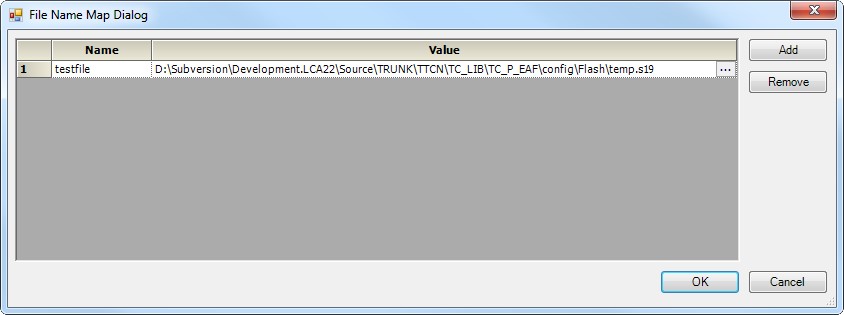
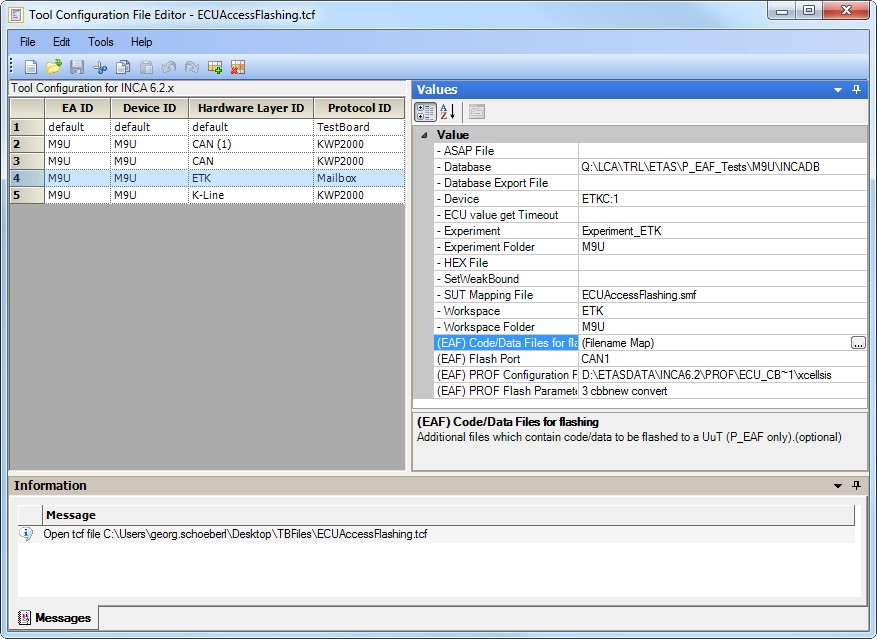


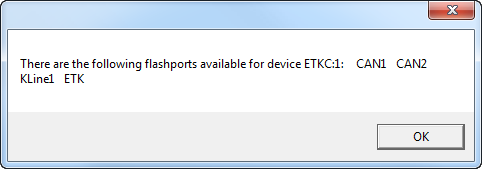
Fig.1: Tool configuration file prepared for the EAF port

In addition you have to give the information necessary for flashing

* file you want to use for flashing is given in (EAF) Code/Data Files for flashing (see

Figure 1 and also “File Name Map Dialog ” )

* The Flash Port , i.e. the port used by INCA to flash the ECU. You can find this information
  + in the INCA hardware configuration in your flash port option. (See Figure 2). But be aware that depending on the device chosen this information might be misleading.
  + by using the visual basic script GetFlashPortNames.vbs. It allows you to determine the flash ports available in the device used by your experiment.



You then have to choose the one used by your PROF tool.

* + If you are using ODX have a look for your physical link device. You can find it in the vehicle information table (VIT). You can use ODX-CONFIG to check.
  + extract the information from the target server log file (TgtSvr.log)

Depending on the INCA version it is stored in

C:\ETAS\LogFiles\TgtSvr

or

C:\ETAS\LogFiles\ProcessLogsV2

Have a look for the line containing the string “StorePermanentlyToTarget”, e.g.

* + Logfile excerpt:

12-05-2006 17:59:52 API: StorePermanentlyToTarget(CAN1,

C:\DOCUME~1\...\Temp\flshtmp1.hex 0 cbbnew convert D:\ETASDATA\INCA5.3\PROF\FCU\_CB~1\xcellsis)

where CAN1 is the Flash Port entry, 0 cbbnew convert is the flash parameter and D:\ETASDATA\INCA5.3\PROF\FCU\_CB~1\xcellsis is the PROF configuration file entry.

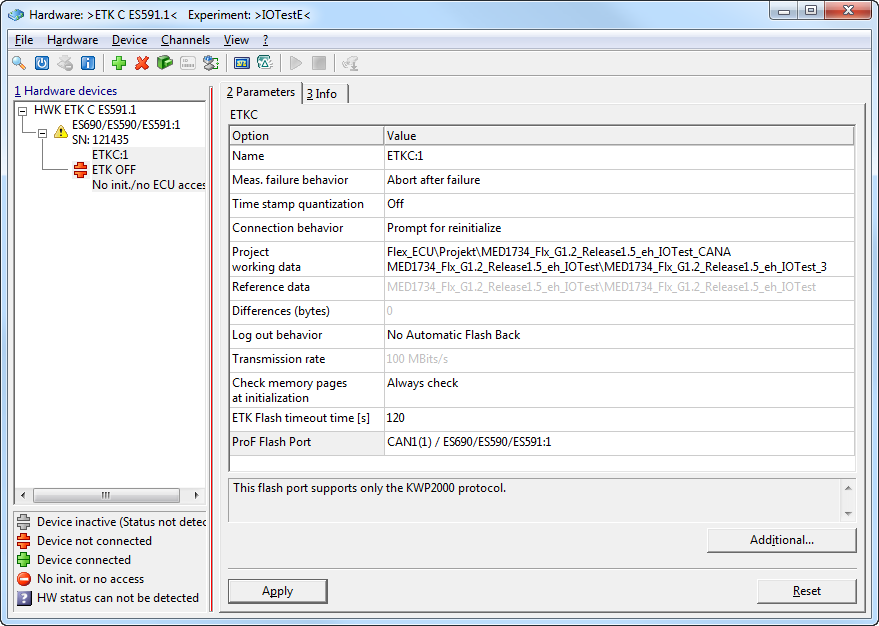


Fig. 2: Hardware dialog in INCA

* The (EAF) PROF Configuration File is holding the path to the PROF layout file (the filename has no extension (see Figure 3) and Figure 1

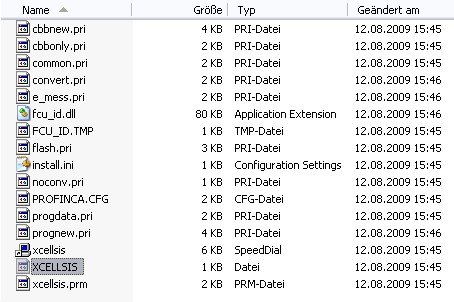


Fig. 3: The layout file is without any extension; therefore the type is Datei or file respectively

* the (EAF) PROF Flash Parameter is holding the parameter string given to the PROF program when it is called by INCA.

The parameters used to flash your ECU can be extracted from the Logfile (see page

27) or you can take it from the layout file directly (see Figure 4).

In both cases your parameters for the here shown example will be: **0 cbbnew convert**

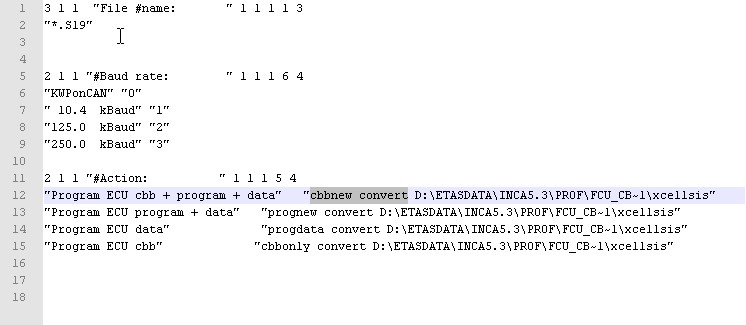


Figure 4 Contents of PROF layout file

SuT mapping file

Is the same you use for your ECU measurement access.

### 3.7.3 Lines in code

Initialization of the port:

private IPortEAF m\_portEAF = null; private IPortEAC m\_portEAC = null; private IPortMA m\_portMA = null;

/// <summary>

/// Registers the ports. /// </summary> private void RegisterPorts()

{ m\_portMA = Factory.GetPortMA("ModelAccess"); m\_portMA.Timeout = -1;

m\_portEAC = Factory.GetPortEAC("ECUAccessCalibration"); m\_portEAC.Timeout = -1;

m\_portEAF = Factory.GetPortEAF("ECUAccessFlashing"); m\_portEAF.Timeout = -1;

}

And to perform some flash action add the following parts to your test performance method:

#region ECU access flashing

Reporting.SectionBegin("Flash access"); m\_portEAF.Configure("device2", "default", "default"); m\_portEAF.DeviceFlash("workingpage");

// workingpage is the test label for the working page in INCA

// the testlabel is resolved by the SuTMapping

if (m\_portEAF.ChecksumIsValid())

{

Reporting.SetInfoText(0L, "Flashed ETk successfully", 6L);

}

// The following flash command flashes the ECU with a given hex or s19 file

//m\_portEAF.UuTFlashFile(@"C:\Users\Public\Documents\ETAS\LABCA

R-AUTOMATION 4.1.0\Examples\Test Bench Configurations\ATCL

Example NEW Test Bench for LCO50\Demo03.hex");

// The following flash command flashes the ECU with the working or reference page

//m\_portEAF.UuTFlash("workingpage");

Reporting.SectionFinished("", new Verdict(VerdictCode.Pass));

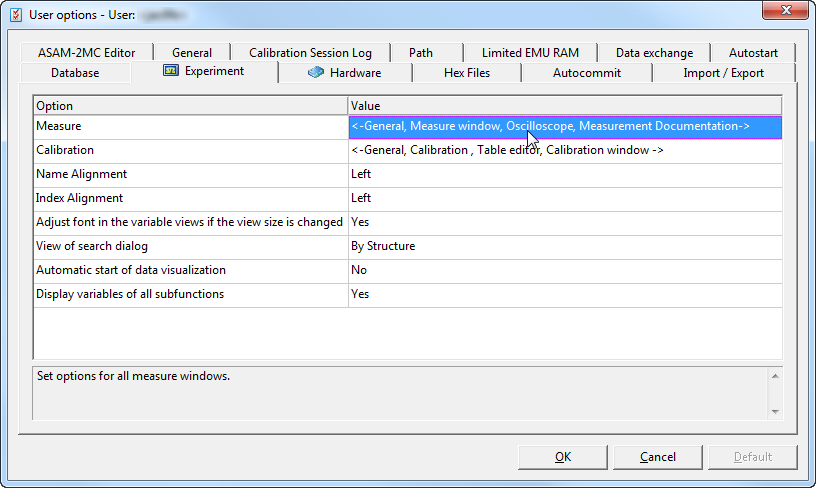
#endregion ECU access flashing

You can find the full test case in the LABCAR-AUTOMATION examples.

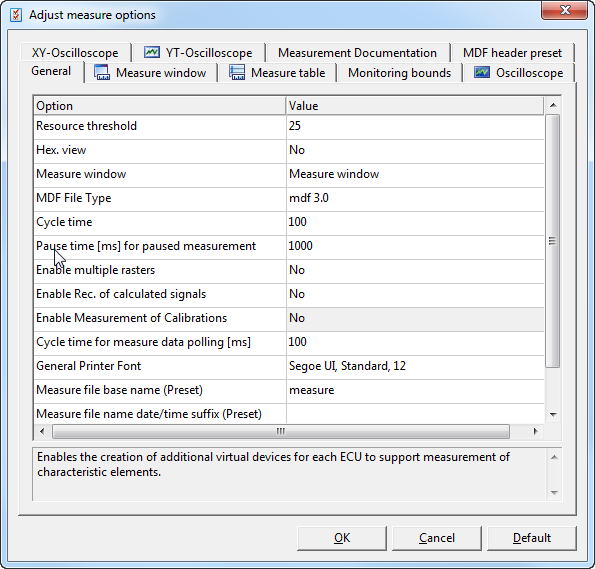
## 3.8 Parameter recording for Sync DL

To use the INCA parameter recording for the SyncDL you have to set an user option inside INCA to enable it.

Open your INCA and select in the Menue Bar: Options -> User Options and activate the tab ‚Experiment’.



After double click at value for measurement in this window the panel for the change of the measurement options open.



In the Change Measurement options – window select the tab ‚General’ and look for the entry ‚Enable Measure Calibration devices’ and set it to ‚yes’.

## 3.9 Failure Simulation Load Cut off

At the FS-Port for failure simulation it is possible to cut off the load via test case for each failure set separately.

To use this functionality you’ve to reference the ATCL (Etas.Eas.Atcl.Interfaces.dll) build version 1.0.0.4 and to use the Load Flag property of the ErrorDefinition.

The load cut off for the complete port by a parameter in the tool configuration file is still available.

This option setting (unless it is empty) is overwriting any settings inside the test case!

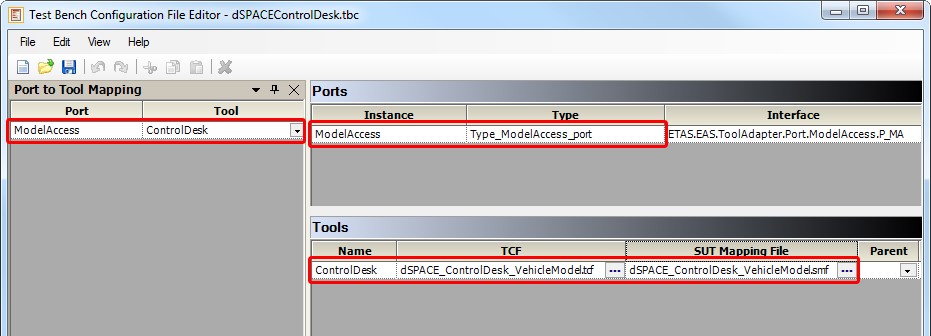
## 3.10 Connection to dSpace Test Bench

Connection to dSpace test benches is very easy as ETAS provides already a Tool Adapter for it. This Tool Adapter enables the connection to dSpace ControlDesk 3.2.1

**Configuration**

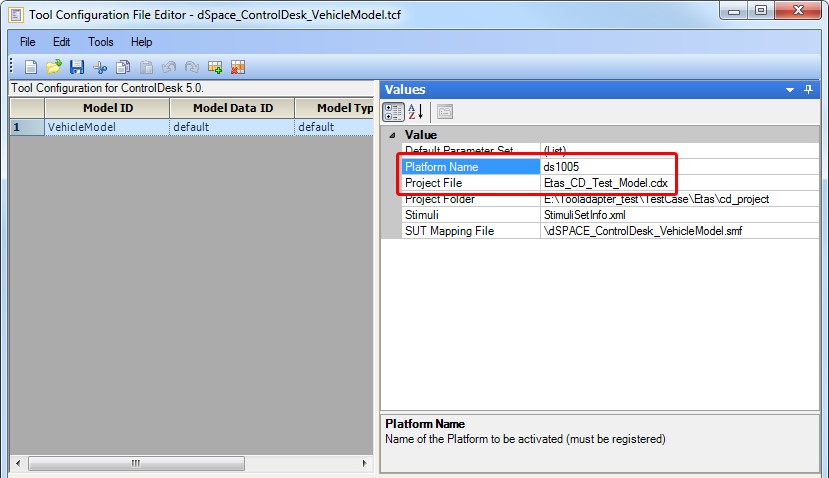
The changes in the test bench are tool configuration related only:

* the Model Access – which is used for the access via LABCAR-OPERATOR as well - has to bind to the Tool Adapter for dSpace ControlDesk like shown in the following picture:



Thus the Model access Port (in this example called ‘ModelAccess’) with all its functionality (“signatures”) is available now for the test cases.

* the Tool Configuration File has to point to the dSpace Control Desk project and to the platform which is used. These are the main differences to the Tool Configuration File of the LABCAR-OPERATOR.



The dSpace ControlDesk Project file has the extension .cdx

The SuT Mapping File is slightly different to the one for LABCAR-OPERATOR due to the different syntax in the dSpace ControlDesk. SUT Mappings naturally differ in tool labels (“Right hand side” of mapping). Additionally not only labels may differ, but also object and data types.



Example of a LABCAR-OPERATOR SuT Mapping File (left) / dSpace ControlDesk related SuT Mapping File (right)

Use the example TBC, TCF or SMF out of the examples folder at C:\

Users\Public\Documents\ETAS\LABCAR-AUTOMATION 4.x\Examples\Test Bench Configurations\Sample Files

**Differences between LABCAR-OPERATOR and dSpace ControlDesk**

* Signal Generation is limited to one object by dSPACE ControlDesk
* Datalogging functionality is limited by dSPACE ControlDesk  Some Data Types are missing in ControlDesk
  + boolean,
  + char, string
  + bitstrings; octetstrings, hexstrings
* Some Variable Objects are missing but exist in ControlDesk
  + “Map”/”2D-Table”
  + “Curve”/”1D-Table”
* Initial variable creation by “ModelValueSelect” is not necessary in dSPACE Test Bench

LABCAR-AUTOMATION 4.2.2

For Testers

# 4 For Testers

## 4.1 Изменение макета отчета

Есть две возможности изменить макет вашего отчета. Просто кратко о технике позади.

Отчеты хранятся в формате xml и форматируются браузером, который вы используете для отображения. Это может быть Microsoft Internet Explorer или наша программа просмотра отчетов LABCAR-AUTOMATION.

Браузеры, использующие таблицы стилей (файлы xslt), ссылаются на форматирование ввода для отображения. Средство просмотра отчетов LABCAR-AUTOMATION копирует эти таблицы стилей в каждый каталог отчетов, основываясь на настройках параметров по умолчанию.

### 4.1.1 Изменение макета отчета после создания для каждого отчета

Изменить макет одного отдельного отчета можно после создания отчета (и только потом). Вам просто нужно изменить таблицы стилей, скопированные в то же место, что и отчет:

<местоположение вашего проекта LCA>\TestRuns\Report <дата и время> \Standard Report\styles Вы снова открыли отчет для просмотра после изменения таблиц стилей.

### 4.1.2 Changing report layout for all reports

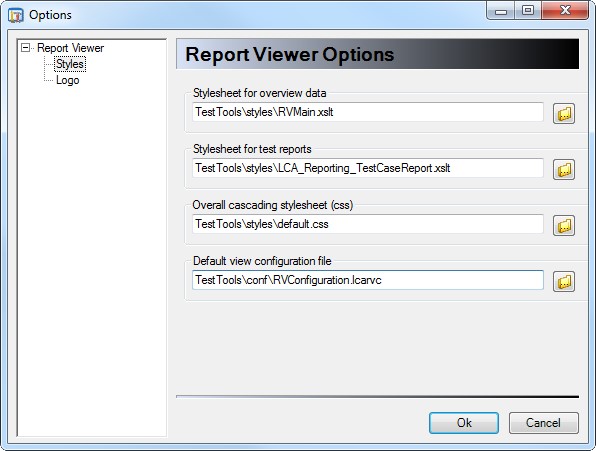
Чтобы изменить макет всех дальнейших отчетов, вы должны изменить таблицу стилей, расположенную в

папка программы ЛАБКАР-АВТОМАТИЗАЦИЯ (C:\Program Files (x86)\ETAS\LABCAR-

AUTOMATION4.x\TestTools\styles) и адаптировать параметр внутри средства просмотра отчетов.

Пожалуйста, перезапустите средство просмотра отчетов после изменения этого параметра.

Для существующих отчетов стили не адаптированы, приходится делать это вручную



# 5 Complex features

## 5.1 Offline Project generation

### 5.1.1 Typical Use Case

Проекты LABCAR-AUTOMATION представляют собой набор нескольких артефактов, работающих вместе во время управления и выполнения тестирования: тестовый пример, набор параметров, тестируемые модули, конфигурация тестового стенда, конфигурации инструментов. На уровне папок они хорошо структурированы, и один файл определяет полное содержание проекта.

Что касается экономии средств и времени, проекты предназначены для повторного применения для различных целей испытаний, испытательных стендов, этапов испытаний или только с другими наборами параметров. Если необходимо изменить только один из артефактов, а остальные следует сохранить и в целом они должны быть непротиворечивыми, целесообразно централизованно управлять артефактами.

Создание проекта отделено от определения отдельных миров, и часто требуется автоматизировать его.

Генератор проектов — это выбор для таких действий..

### 5.1.2 Features of the Project Generator

Имея все артефакты проекта LABCAR-AUTOMATION, доступные в любом месте в структуре папок, Генератор проектов собирает из них проект, управляемый так называемым файлом управления проектом (.pmf).

### 5.1.3 Feature Handling

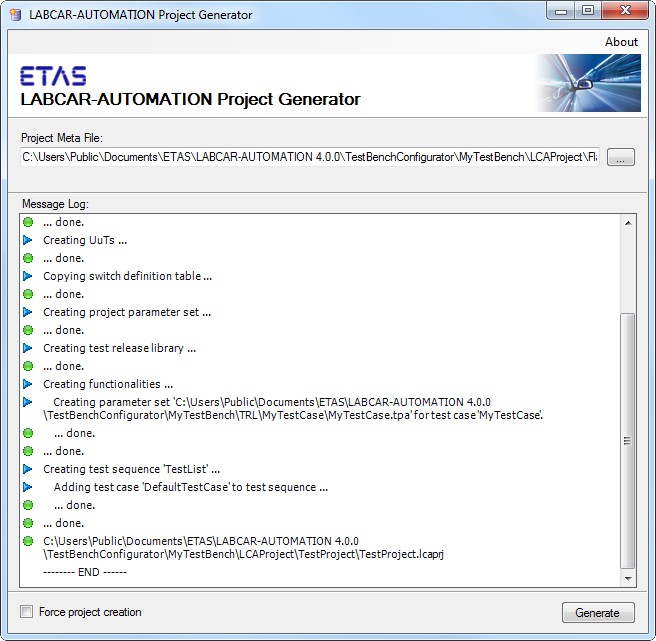
Чтобы предоставить все входные данные Генератору проектов, необходимо создать метафайл проекта (.pmf). Он структурирован в формате XML и содержит ссылки на все артефакты проекта LABCAR-AUTOMATION. Все артефакты должны быть созданы заранее, т.е. путем создания нового или изменения существующего проекта LABCAR-AUTOMATION.

Формат следует заданной схеме. Полная схема находится в каталоге установки по адресу: C:\Program Files (x86)\ETAS\LABCAR-AUTOMATION.

4.x\TestTools\XMLSchema\lca\_pmf\_v1.1.xsd

Этот файл .pmf является единственным вводом, который необходимо ввести в диалоговом окне Генератора проекта, чтобы включить генерацию. С опцией «Принудительное создание проекта» Генератору проектов предписывается перезаписать существующий проект.

Просто нажмите «Создать», и он запустится.



## 5.2 Report Structure ‘Abstract Section’

### 5.2.1 Feature Description

Большая часть информации во время выполнения теста собирается, сообщается и просматривается в соответствии с хронологическим циклом теста.

Тем не менее, сводка результатов — более подробная, чем только вердикт, — интересно рассматривать в обобщенном виде и прямо в верхней части отчета или в специальном разделе, даже если информация частично получена в более поздний момент времени теста. бегать.

Этот абстрактный раздел в отчете позволяет.

### 5.2.2 Test case development for use of the Abstract Section

Резюме — это специальный раздел внутри отчета. Тестовый пример должен ограничивать части отчета для этого специального раздела двумя подписями.

 Reporting.SectionAbstractBegin

 Reporting.SectionAbstractFinished

Инструкции по вводу в раздел «Реферат» можно использовать несколько раз в тестовом примере.

И он также может быть вложен в другие разделы. В этом случае содержимое абстрактного раздела относится к разделу и отображается в верхней части этого раздела. Возможен один абстрактный раздел на раздел отчета.

Реферативная часть должна быть реализована в следующих элементах структуры:

 начало теста/конец теста

 Начало раздела/Завершение раздела

 TestEntityCalled/TestEntityFinished

 ФункцияВызвана/ФункцияЗавершена

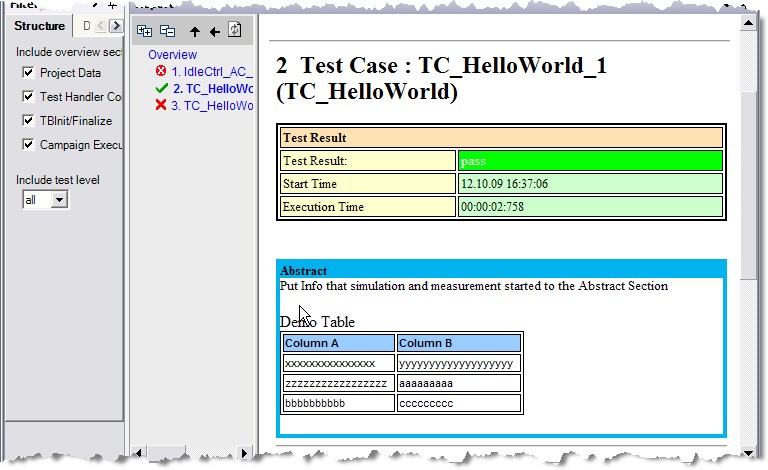
Абстрактная секция позволяет использовать следующие элементы/методы структуры внутри:

|  |  |
| --- | --- |
| Name | Description |
| AddPlot2Report | Overloaded. |
| AddTableToReport | Add the table to the report. |
| FunctionCalled | Overloaded. |
| FunctionFinished | Overloaded. |
| SectionBegin | Adds a new Section to the report. |
| SectionFinished | Finishes a section in the report. |
| SetErrorLevel | Set the error level. |
| SetErrorLink | Sets the error link. |
| SetErrorText | Sets the error text. |
| SetErrorTextAndValue | Sets the error text and value. |
| SetHeaderLink | Sets the header link. |
| SetHeaderText | Sets the header text. |
| SetHeaderTextAndValue | Sets the header text and value. |
| SetInfoLink | Sets the info link. |
| SetInfoText | Sets the info text. |
| SetInfoTextAndValue | Sets the info text and value. |
| SetLink | Set a link. |
| SetResultLink | Sets the result link. |
| SetResultText | Sets the result text. |
| SetResultTextAndValue | Sets the result text and value. |
| SetText | Set some Text. |
| SetTextAndValue | Set some text and some values. |
| SetWarningLink | Sets the warning link. |
| SetWarningText | Sets the warning text. |
| SetWarningTextAndValue | Sets the warning text and value. |
| StateCalled | Overloaded. |
| StateFinished | Overloaded. |
| TestEntityCalled | Overloaded. |
| TestEntityFinished | Overloaded. |

### 5.2.3 Report View with Abstract Section

Раздел «Абстрактный» отображается средством просмотра отчета в верхней части отчета или раздела.

Используя обзорную страницу тестового примера, вы найдете содержимое абстрактного раздела вверху как главу «Аннотация»:

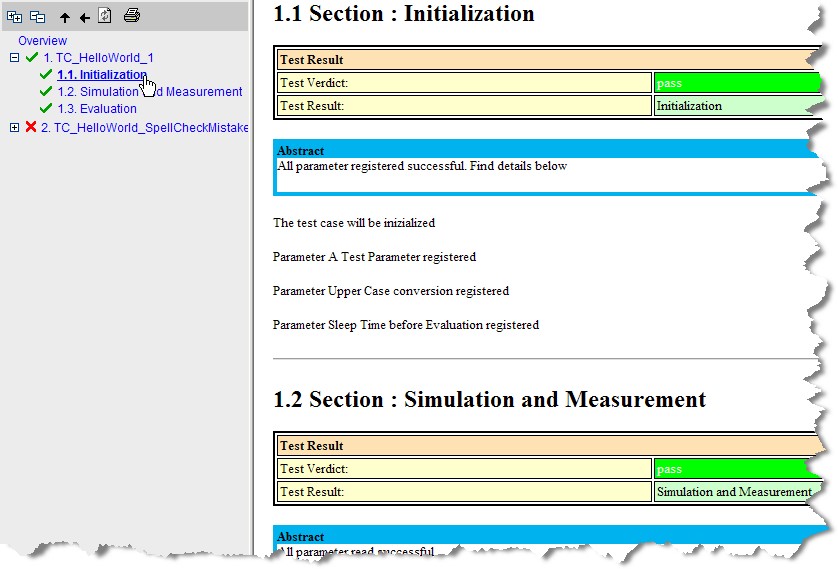


Используя ссылку в дереве на раздел тестового примера, вы найдете содержание раздела, связанного с рефератом, в главе «Реферат» в верхней части этого специального раздела:

## 5.3 Улучшенная функциональность регистрации данных

### 5.3.1 Typical use case

Если заказчик использует стандартные проекты LABCAR-OPERATOR, которые необходимо использовать для моделирования и испытаний, может быть полезно предварительно определить регистраторы данных для значений, которые должны быть зарегистрированы в любом случае. В зависимости от варианта использования тестовый пример или последовательность симуляции могут добавлять некоторые значения для регистрации для более подробного представления о поведении.



Усилия для базовых регистраторов данных не нужно тратить дважды. Однако тестовый пример может регистрировать все необходимые детали, не перегружая проект LABCAR-OPERATOR слишком большим количеством регистраторов данных.

5.3.2 Feature Description

Former LABCAR-AUTOMATION releases create their own Datalogger in the LABCAROPERATOR project when a test runs, independently of the Dataloggers which already exist in the LABCAR-OPERATOR project.

With the newest version of LABCAR-AUTOMATION it is now possible to access Dataloggers that exist in the LABCAR-OPERATOR project already.

This feature is optional, by means that it can be selected, if the LABCAR-OPERATOR

Dataloggers shall be visible and usable inside the LABCAR-AUTOMATION test cases, or not.

5.3.3 Feature Handling

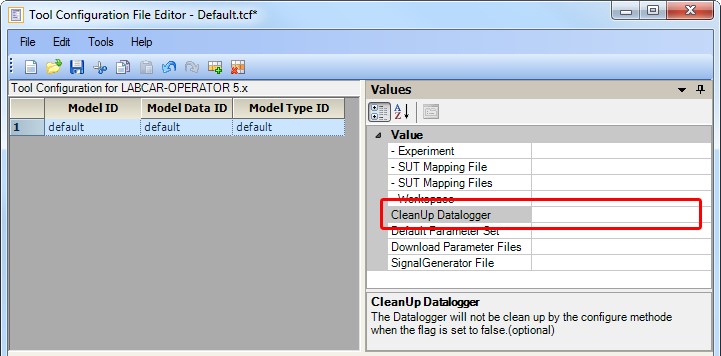
The new feature is provided by LABCAR-AUTOMATION as a new entry inside the tool configuration of the model access. It is only available from LABCAR-OPERATOR V5.0 onwards.

There are no code changes necessary for the option to use the LABCAR-OPERATOR project dataloggers in the LABCAR-AUTOMATION test case as well. It is only controlled by the tool configuration of the LABCAR-OPERATOR.

If the test case uses the same value for logging like already defined by the LABCAROPERATOR project and if the use of LABCAR-OPERATOR project dataloggers is enabled, LABCAR-AUTOMATION merges both dataloggers automatically.

5.3.4 Test bench configuration for LCO project dataloggers

The tool configuration template of the LABCAR-OPERATOR V5.0 contains a new entry ‘CleanUp Datalogger’. It accepts the values ‘True’ and ‘False’.



In case ‘False’ is entered, the Dataloggers defined in the LABCAR-OPERATOR project are visible and usable by LABCAR-AUTOMATION as well.

To ensure the same behavior like in the past, the default value of this entry is ‘True’. Leaving the entry blank has the same meaning like ‘True’.

5.3.5 Hints

The entry field of this new entry is free text. Thus you can enter each string you want. The only value which is evaluated is ‘False’ or ‘FALSE’. All other values are resulting internally to ‘True’.

5.4 Maps and Curves

5.4.1 Typical use case

Signals which are recorded or used for stimulation are usually stored as a digital curve. It uses pairs of dedicated values for the time line (x-axis) and the amplitude (y-axis). A set of signals can be stored together as a map, using the same values for the time line. The synchronous datalogging is a representative use case for these maps.

5.4.2 Feature Description

LABCAR-AUTOMATION is able to work with curves and maps using two dedicated TypeSut – Types, called TypeSut1DFloatTable and TypeSut2DFloatTable.

5.4.3 Feature Handling

Reading measurement values or feeding parameter values as maps or curve these types can be mapped accordingly to the tool labels.

Access to the single values and the time line point are possible via index values.

5.4.4 Mapping rules

|  |  |  |  |
| --- | --- | --- | --- |
| Test case type | | | SuT Mapping type |
| TypeSut1DFloatTable | | | Curve |
| TypeSut2DFloatTable | | | Map |
|  | |  |
| 5.4.5 | | Example |

// A very simple Test Case Example snippet

TypeSut1DFloatTable resultCurve =

(TypeSut1DFloatTable)Ports.ECUAccessCalibration.GetValue(new

TypeSut1DFloatTable("",

"One\_D.STD\_sint16\_sint16.All\_Curves\_STD.Curves\_Raster\_1", "The Test Label is mapped in the SutMapping File", new double[0], new double[0], 0.0, 0, 0, 0, "", ""));

TypeSut2DFloatTable resultMap =

(TypeSut2DFloatTable)Ports.ECUAccessCalibration.GetValue(new TypeSut2DFloatTable("", "Group\_Two\_D\_1.Map\_Mod", "The Test Label is mapped in the SutMapping File", new double[0], new double[0], new double[0], 0, 0, 0, 0, 0, 0, "", "", ""));

// assign the first two amplitude figures to the curve resultCurve.ValueY[0] = 100; resultCurve.ValueY[1] = 200;

// define a map containg values from 0 .. map length for x- and yaxis

// assign the x-axis figures to the map int index = 0; for (int i = 0; i < resultMap.ValueX.Length; i++) { resultMap.ValueX[i] = index; index++;

}

// assign the y-axis figures to the map index = 0; for (int i = 0; i < resultMap.ValueY.Length; i++) { resultMap.ValueY[i] = index; index++;

}

// assign the z-axis figures to the map index = 10; for (int i = 0; i < resultMap.ValueZ.Length; i++) { resultMap.ValueZ[i] = index;

index++;

}

// download curve and map settings onto the ECU

Ports.ECUAccessCalibration.SwitchSectionPage("WorkingPage");

Ports.ECUAccessCalibration.SetValue(resultCurve);

Ports.ECUAccessCalibration.SetValue(resultMap);

// read curve and map back

TypeSut1DFloatTable resultCurve2 =

(TypeSut1DFloatTable)Ports.ECUAccessCalibration.GetValue(new

TypeSut1DFloatTable("",

"One\_D.STD\_sint16\_sint16.All\_Curves\_STD.Curves\_Raster\_1", "The Test Label is mapped in the SutMapping File", new double[0], new double[0], 0.0, 0, 0, 0, "", ""));

TypeSut2DFloatTable resultMap2 =

(TypeSut2DFloatTable)Ports.ECUAccessCalibration.GetValue(new TypeSut2DFloatTable("", "Group\_Two\_D\_1.Map\_Mod", "The Test Label is mapped in the SutMapping File", new double[0], new double[0], new double[0], 0, 0, 0, 0, 0, 0, "", "", ""));

// compare written and read figures of the curve if(resultCurve.ValueY.LongLength == resultCurve2.ValueY.LongLength)

{ for (int i = 0; i < resultCurve.ValueY.Length; i++) { if( resultCurve.ValueY[i] != resultCurve2.ValueY[i]) {

Fail();

} } for (int i = 0; i < resultCurve.ValueX.Length; i++) { if (resultCurve.ValueX[i] != resultCurve2.ValueX[i]) {

Fail();

}

}

Pass(); } else {

Reporting.LogExtension("Length of written and read back curve is different");

Fail();

}

// compare written and read figures of the map

if (resultMap.ValueY.LongLength == resultMap2.ValueY.LongLength)

{ for (int i = 0; i < resultMap.ValueY.Length; i++) { if (resultMap.ValueY[i] != resultMap2.ValueY[i]) { Fail();

} } for (int i = 0; i < resultMap.ValueX.Length; i++) { if (resultMap.ValueX[i] != resultMap2.ValueX[i]) {

Fail();

} } for (int i = 0; i < resultMap.ValueZ.Length; i++) { if (resultMap.ValueZ[i] != resultMap2.ValueZ[i]) {

Fail();

}

}

Pass();

} else {

Reporting.LogExtension("Dimension of written and read back map is different");

Fail();

}

Pass();

5.5 Real Time Tests support

5.5.1 Typical use cases

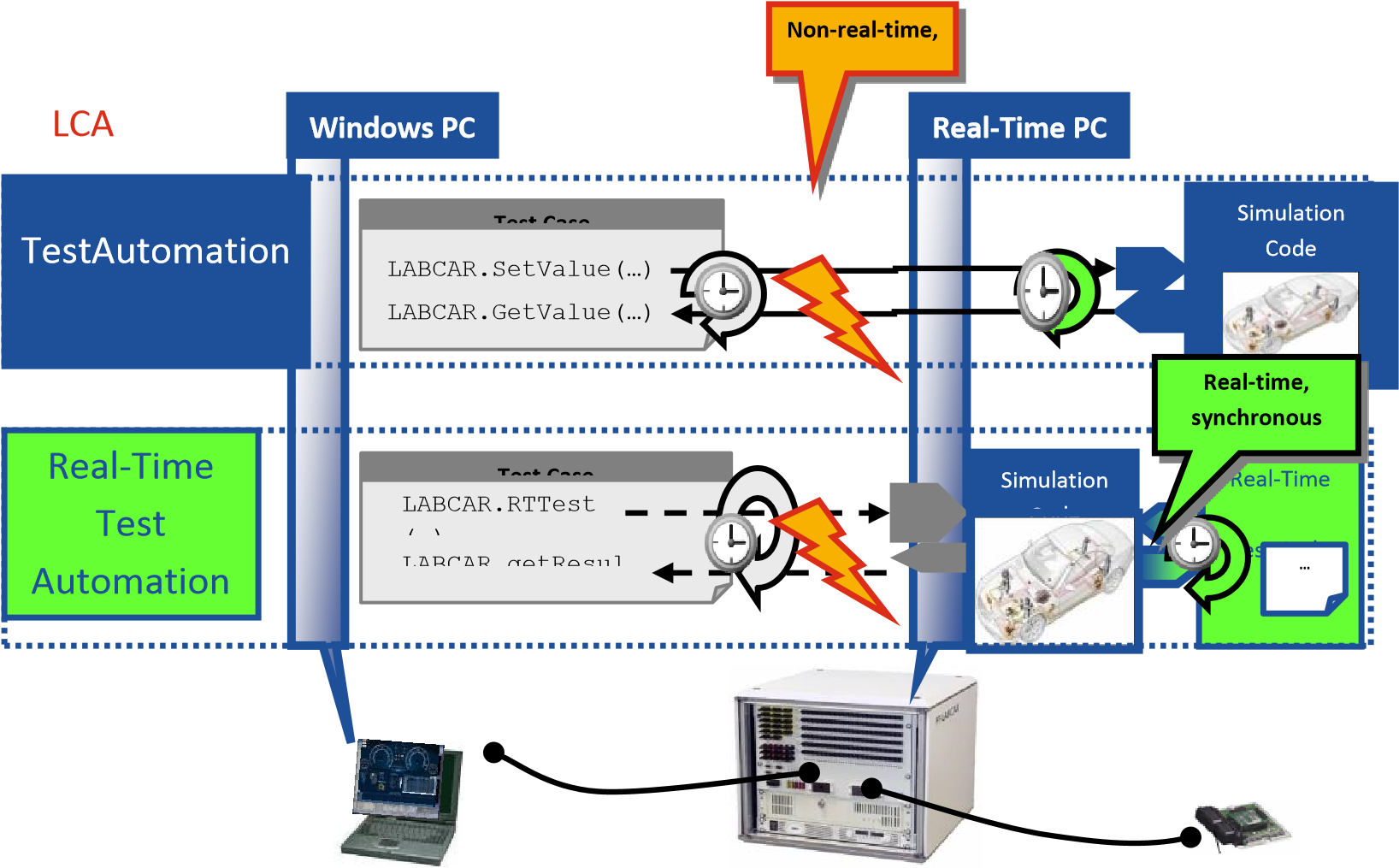
The strong networking of ECUs and other controlling elements in cars leads more and more to a time dependant interconnection. Especially the use case of a bus system where information are exchanged randomly and with a high frequency of messages leads to the necessity to observe the exact point in time as soon as messages are received.

Due to asynchronous design of communication between sensors, actors and different ECU’s, the test of the behavior regarding reaction along the timeline becomes important. To realize such tests the target reaction has to be detected like in the reality – in real time.

The possibility to perform tests at the real time environment (Real Time PC) of a LABCAR covers this demand. Downloading new parts of code onto the Real Time PC during and thus expand the model during test execution is another quite exciting benefit of this feature.

5.5.2 Feature Description

In case action and reaction of several values calculated by the model the point in time of setting or measuring a value might impact the behavior. Without real time test possibility both actions are depending on the performance and signal transmission time, which can lead to unintentional delays. These parts of the test which are time critical in this sense should be performed as real time tests.



The Real Time PC (RTPC) of the LABCAR expects C-code. LABCAR-AUTOMATION is able to trigger the download of such code onto the RTPC, to set it into operation (activate) or to deactivate it. Precondition for usage of this feature are:

* LCO 5.0 as Model access tool
* Prepared Model (hooks introduced)
* LABCAR as the test bench with RTPC for Simulation
* Existing C-code for download

5.5.3 Feature handling

To use the real time feature the model has to be prepared for it. Models use ‘hooks’ to enable docking of alternative code. These hooks are the input and output interfaces for the new code.

First you have to provide the C-code for the download onto the RTPC. One possibility to develop such code is the LABCAR-ASC . LABCAR-ASC is an ASCET Add-On and comes with LABCAR-OPERATOR. It provides among others the ASCET Target “RTPC” and enables compilation on RTPC.

For more information on creating real time C-code and preparing the model please refer to the user guides of RTPC and LABCAR-OPERATOR.

When developing your test case you have to use a new port ‘RealTimeScripting’ to initiate the following actions

* Code Download
* Activation of downloaded code
* Deactivation of downloaded code to use the Real Time Testing feature.

For executing LABCAR-AUTOMATION provides a special Tool Adapter (Test Bench Connector Real Time Test – LABCAR-TBCRT) for access to this feature. It has to be configured in the test bench configuration.

5.5.4 Test case development for Real Time testing

The port providing the access to the real time test functionality is provided by LABCARAUTOMATION with in the ATCL (Automotive Testing Class Library):

* Ports.RealTimeScripting

The creation and closing signatures are like usually

* Create
* ConfigureTool
* Configure
* Close

Special signatures for the real time test handling itself are:

* AddFiles – to pass the initial set of C-files to RTPC for later activation
* Activate – activation of the downloaded code at RTPC  GetRTPCState – receive the current state of the RTPC
* Deactivate – deactivation of the downloaded code at RTPC

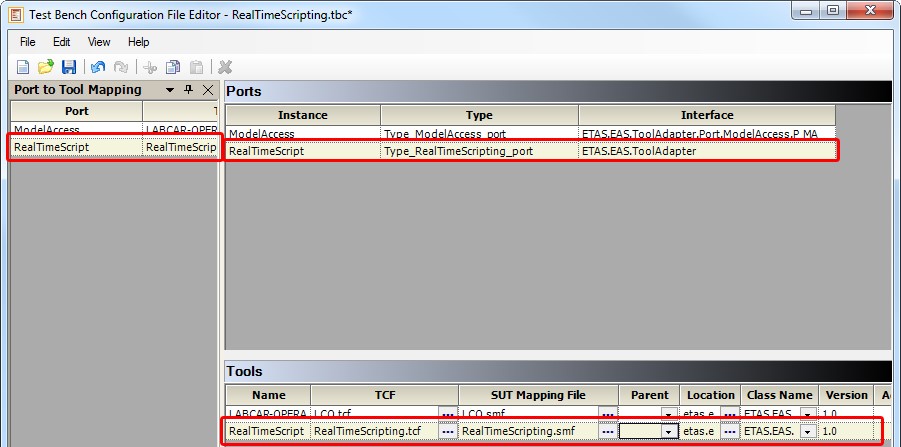
To get detailed information on the test case design find an example, delivered with the installation of LABCAR-AUTOMATION, at:

C:\Users\Public\Documents\ETAS\LABCAR-AUTOMATION 4.x\Examples\Test Case Development\TC\_RealTimeScripting

5.5.5 Test bench configuration for Real Time testing

Within the LABCAR-AUTOMATION installation a tool adapter ‚…’ is provided . To use the feature the test bench has to configure it.

Example of a test bench configuration with LABCAR-OPERATOR 5.0 / Experiment Environment 3.2:



|  |  |
| --- | --- |
| 5.5.6 | Hints |

Datalogging with labels of the new code is not possible directly. The reason is the ignorance of the new code labels – and thus of the mapped test labels – during test bench initialization. A loophole out of this drawback is to ‘handover’ the values to the original downloaded model and to log these model values.

5.6 Soft-Stop Function for Test cases

5.6.1 Typical use cases

**Big test runs**

In the case that the test run contains a lot of test cases and the test cases themselves have a long duration an erroneous behavior of one test case might ‘kill’ the complete test run. In this case it should be possible just to stop this single test case and continue with the next one.

The already performed test cases may not run again and the following test cases can run without any further actions.

**Wrong test case coding**

Test cases may include endless loops by accident. A test run stop would wait until this endless test case is finished, which will never happen. The ‘Stop Test’ is a way out of this.

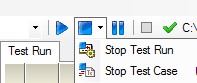
5.6.2 Feature Description

In the past the stop button in Test handler stopped execution of the complete test run, but did not finish the actual running test case before. The process was killed hardly, thus the test bench left in an undefined, unpredictable state.

Now this additional choice is available via button push inside the Test Handler. The test case receives the information about this stop and is able to react on this button push. Thus the developer can decide, what has to be performed when the stop of the test case occurs.

5.6.3 Feature Handling

In the test handler the button ‚Stop’ is extended with a small menu. The tester can select the kind of stop which has to be carried out.



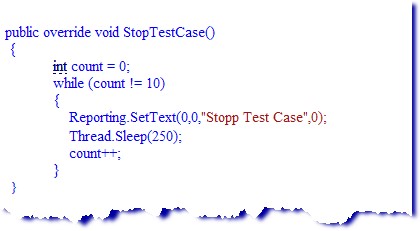
For faster reaction of the tester in front of the desktop the different stop activities can be triggered via short cut key combinations like shown in the picture right beside.

The test case which has been stopped with ‘Stop Test Case’ or Shift+F10 will have the status ‘interrupted’ in the Test Handler’s execution overview:

5.6.4 Test case development for Soft Stop

The LABCAR-AUTOMATION library provides for its class TestCase a new virtual method StopTestCase. This Method is prepopulated with no statements and does simply nothing, except to set the verdict to ‘inconc’.

To enable the test case to react on the event it just has to override this method with the appropriate actions.



This example just writes out ten times the "Stopp Test Case" string in time lags.

Typical Actions within this StopTestCase Method might be:

* Report, that the test case was stopped by an outside request
* Report the status of the test bench, e.g. the status of each port
* Measure and Log some important values
* Report the state of the test case
* Finalize the test bench, close all open ports or reset the test bench

5.6.5 Test bench configuration for Soft Stop

There is no special test bench configuration necessary for this feature.

5.6.6 Hints

To prevent the running test cases or test run directly following the interrupted test case from failure, each test case should provide a StopTestCase behavior by overriding the method. At least a reset of the test bench should be performed to prepare a clean test bench for the next test.

If the overridden method runs endless the Test Case will be not closed!

5.6.7 Difference between status shown in Test Handler and Verdict reported by test case

**Test Handler status calculation and Verdict reporting by Test case**

The Test Handler shows in the overview pane the Status of test case execution. If the test case finishes its operation normally the status is been build of its verdict (pass, fail or inconc). In the case the test case is not working (error in coding), the test bench was set up wrongly or the test case is interrupted this is shown in the Status column of the Test handler as values ‘error’ or ‘interrupted’.

In opposite to this the Report shows the final result of the test case – its verdict. If no verdict could calculated at the end due to interruption or error this status is taken over as result.

**What does this mean for this feature?**

When a test case is stopped by the tester using the new Stop button, the test handler sets the status ‘Interrupted’ immediately as soon as the button was pressed and fires the appropriate event. This is done independently of the test case behavior (as the test case design is completely under developer’s control).

In some cases the report contains a verdict pass or fail, even if the test handler shows ‘Interrupted’.

Following two situations can lead to this rare case:

* The test case is ‘almost’ finished. In this glare situation the finish of the test case and the stop event may occure at the same point in time. They are overlapping each other, the event is fired but does not reach the test case anymore. LCA is not able to set the verdict to ‘inconc’ (by default, defined in the provided virtual method) due to the stop event. The interrupt of the test case is marked in the Test Handler as ‘interrupted’ even if the test case has been finished normally.
* The test case has overridden the virtual method and do not set the verdict hardly to ‘inconc’ but to another one (eg. ‘pass’ or ‘fail’), depending on its state or results before. E.g. if there is a cool down of the engine performed after all test case steps. In this case the verdict-relevant test part has finished and has evaluated a valid verdict already, which should not overwritten by an interrupt anymore.

5.7 Error Manager

* + 1. Typical Use Cases

In some cases you like to check the counter of messages of one of the used tools of your test bench to check it inside the test case. If the tool underneath the LABCAR-AUTOMATION does not report e.g. a warning via it’s API – may be it is not relevant in the automated mode – the test case is able to check if a warning was produced.

* + 1. Feature Description

With the Error Manager it is possible to read the counter of errors, warnings or information messages logged by a tool. Currently only the Experiment Environment version 3.2.1 supports this feature.

* + 1. Feature Handling

The feature is available for programmed test cases, e.g. in C# or another .Net compatible software language. It can be used by referencing the necessary error logging library in your test case development project. This library provides you with the methods to

* Get the counter of Application Error Messages
* Get the counter of Application Warning Messages
* Get the counter of Application Information Messages
* Get the counter of Hardware Error Messages
* Get the counter of Hardware Warning Messages
* Get the counter of Hardware Information Messages
* Reset all Application Message Counters
* Reset all Hardware Message Counters

5.7.4 Test Case development with Error Manager

**Library reference**

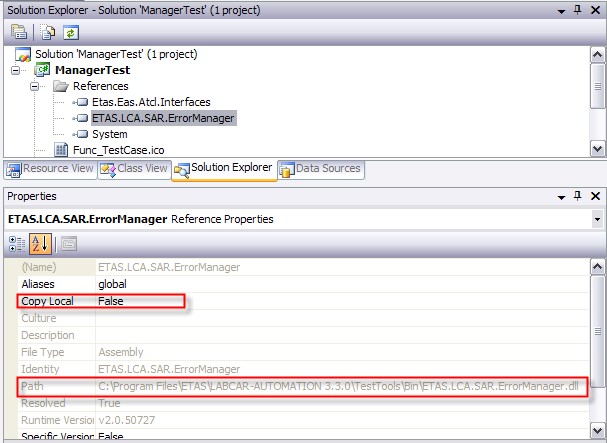
Add the ErrorManager reference to your Test Case.

You find the ETAS.LCA.SAR.ErrorManager.dll to be referenced at C:\Program Files

(x86)\ETAS\LABCAR-AUTOMATION x.y.z\TestTools\Bin, whereby C:\Program Files

(x86)\ETAS\LABCAR-AUTOMATION x.y.z\ is your actual LABCAR-AUTOMATION installation folder.

In the following picture you find as example a Visual Studio Solution with a test project.



**How to Use the ErrorManager in the Test Case**

* Constructor

public IErrorManager errorManager

{ get { return

(IErrorManager)Factory.GetManager("IErrorManager");

}

}

* Available methods

* int applicationerror = errorManager.GetApplicationErrorMessageCount(Ports.ModelAcces s);

* int applicationwarning = errorManager.GetApplicationWarningMessageCount(Ports.ModelAcc ess);

* int applicationinfo = errorManager.GetApplicationInfoMessageCount(Ports.ModelAccess

);

* int Hardwareerror = errorManager.GetHardwareErrorMessageCount(Ports.ModelAccess);
* int Hardwarewarning = errorManager.GetHardwareWarningMessageCount(Ports.ModelAccess

);

* int Hardwareinfo = errorManager.GetHardwareInfoMessageCount(Ports.ModelAccess);
* errorManager.ResetAllApplicationMessageCount(Ports.ModelAcces s);

* errorManager.ResetAllHardwareMessageCount(Ports.ModelAccess);

5.7.5 Test Bench Configuration for Error Manager

There are no configurations necessary in the test bench to use the Error Manager.

As from the Example above (Chapter 4.7.4) the Port Name must fit to the name used in the test case – like usually for all calls to this port.

5.7.6 Hints

Due to the fact that only the Experiment Environment supports this feature, the method will threw an exception in case of any other tool is referenced by the given port (parameter of the methods).

5.8 Working with a Signal Generator

* + 1. Typical use cases

To stimulate the model reproducible with a dedicated signal traces, thus reactions are compareable you need a possibility to store and ‚re-call’ this trace.

In some case it is necessary to stimulate different signals with a dedicated offset in time or started equally at exactly the same point in time. For this you should have a possibility to define these points exactly.

Third use case is the continuous realtime stimulation. The signal feed is created under real time condition.

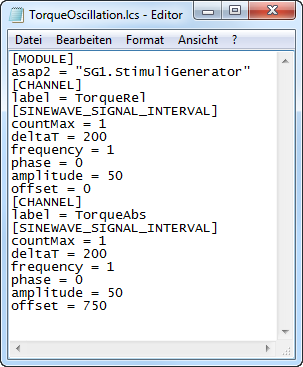
* + 1. Feature Description

The feature uses different files to provide the input for the signal generation.

First of all you need the signal feed itself – the values which have to be set at certain points in time. This input file is a description of the signal. It can be created out of a recorded signal by LABCAR-OPERATOR (extension .lcs) or edited by hand.

The input of this file will be assigned to a signal generator. The Signal Generator, running at the real time environment, is build up the signal and connects it to the signal input at the model. Thus the model uses this as signal feed instead of the output of a device or other model parts.

The generatation can be configured, started and stopped.



* + 1. Feature Handling

The signal generator is part of the model access port. Thus the configuration information are included in the model acces Tool Configuration File.

The set of information regarding signal to be used, acquisition task and file to be used are provided inside a stimuli set file additionally. This file will be automatically created by the configuration wizard at your test bench configuration folder at the subfolder \TBC and called per default StimuliSetInfo.xml. (see 4.8.5Test bench configuration for the Signal Generator)

In case of LABCAR-AUTOMATION Standard Package is installed the name and storage location must not be changed!

In case of LABCAR-AUTOMATION’s Shell Component ‘Editors Package’ is licensed, you can change name and storage location of this file and reference this directly in the tool configuration file.

Following steps are necessary when you like to use this feature:

* Provide configuration information for the signal generator
  + A traced signal – available as file of type .lcs (see 4.8.2 Feature Description)
  + A stimuli set information file, provided by Configuration Wizard or manually created.

If you do not use the Configuration Wizard you find an example at C:\ Users\Public\Documents\ETAS\LABCARAUTOMATION 4.x\Examples\Test Bench Configurations\Demo

Test Bench\LCO 5.0\StimuliSetInfo.xml

* + A configuration information in the Tool Configuration File.

If you use the Configuration Wizard the Tool Configuration File is created and prefilled with the correct values and options.

* + SuT Mapping Information in the SuT Mapping File to map signal name, channel, mode and acquisition task

* Configure the signal generator and connect it to signals to be stimulate
  + Test case steps to configure (defining signal name and stimuli set file)
  + Connection to the correct signal channels  Using LABCAR-OPERATOR V3.2.5:

The connection has to be done in the LABCAR-OPERATOR in the signal list manually before starting the automated test.

* + Using LABCAR-OPERATOR V5.0 or higher:

The connection has to be done inside the test case with the method SetModelValue. (see 4.8.4 Test case development for the Signal Generator). The values have to set the signal mode, the signal name and the channel name.

The separate connection to dedicated signals is possible only with the LABCAR-OPERATOR!

* Start/Stopp the stimualtion
  + Test case steps to configure start, pause (optional) and stop the signal generator. (see 4.8.4 Test case development for the Signal Generator).

5.8.4 Test case development for the Signal Generator

In case you programm your test cases with C# you can find here an excample of the code:

* first part – configure the signal generator

m\_maport.ConfigureSignalGenerator(

"ACTorqueSignal", //SignalGenerator name <SG .../> from

StimulisSetInfo.xml

"pulse"); // SGSet name <SGSet .../> from StimuliSetInfo.xml, selects .lcs file

* second part – connect the signal generator to the correct channels m\_maport.SetModelValue(new TypeSutString("",

SutMapping.SIGNALSIGNALGENERATOR, "", "ACTorqueSignal")); m\_maport.SetModelValue(new TypeSutString("",

SutMapping.SIGNALSIGNALGENERATORCHANNEL, "", "Torque"));

// channel label inside the lcs file // Available modes:

// 0 = constant

// 1 = stimuli

// 2 = model

// 3 = stim + model

// 4 = stim \* model

TypeSutFloat mode = new TypeSutFloat("", SutMapping.SIGNALMODE, "Mode value", 3.0, 0, 0, "");

m\_maport.SetModelValue(mode);

 // third part – start and stop the signal generator m\_maport.Start(); m\_maport.StartSignalGenerator("ACTorqueSignal");

Thread.Sleep(4000);

m\_maport.StopSignalGenerator("ACTorqueSignal"); m\_maport.Stop();

The separate connection to dedicated signals is possible only with the LABCAR-OPERATOR!

With LABCAR-OPERATOR V5.0 and higher it is mandatory to connect the signal generator after configuration

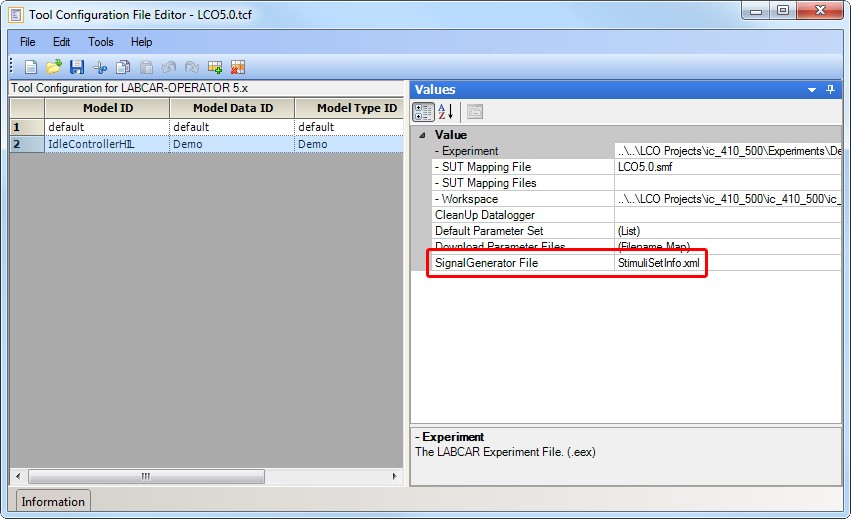
The information regarding the signal (name and signal set name) have to match the information of the stimuli information set. (see 4.8.5 Test bench configuration for the Signal Generator)

The information which channel name, signal generator name and signal mode has to be used for the signal to be stimulated you find in the signal list / signal center of LABCAROPERATOR. Please have a look into the User guide of the appropriate version of LABCAROPERATOR.

5.8.5 Test bench configuration for the Signal Generator

The configuration information is provided by the Configuration Wizard by default.

If you like to change this information or if you do not use the Configuration Wizard you need to install the Tool Configuration File Editor, which is part of the Editors Package.



For the format of the StimuliSetInfo file is like follows:

 StimuliSetInfo.xml (for LCO):

<?xml version="1.0" encoding="utf-8" ?>

<File type="Stimuli Set Info" extension="XML" version="1.0"> <SG name="ACTorqueSignal">

<SGModelName name="SignalGenerator\_0" />

<AcquisitionTask name="TaskDVEModel" />

<SGSet name="pulse" file="TorqueOscillation.lcs"/> </SG>

</File>

5.8.6 Configurable Signal Generator Input Mode Mapping

LABCAR-AUTOMATION 4.2.2 allows choosing between two different mapping behaviors for the Signal Generator input mode.

The mode can be defined by the newly introduced flag “Inport Mode” within the test bench configuration (tcf).

* **Inport Mode = False (Default Mode):** Mapping behavior represents for compatibility reasons the behavior of previous version of LABCAR-AUTOMATION as introduced with older versions LABCAR-OPERATOR Experiment Environment. This is the default behavior of LABCAR-AUTOMATION 4.2.2.
* CONST = 0
* STIMULI = 1
* MODEL = 2
* STIMULI + MODEL = 3
* STIMULI \* MODEL = 4

* **Inport Mode = True (Latest Mode):** Mapping behavior represents the latest behavior of the supported LABCAR-OPERATOR Experiment.
* CONST = 0
* MODEL = 1
* MODEL\_PLUS\_CONST = 2
* MODEL\_MULT\_CONST = 3
* MODEL\_PLUS\_SIGNALGENERATOR = 4
* MODEL\_MULT\_SIGNALGENERATOR = 5
* SIGNALGENERATOR = 6
* SIGNALGENERATOR\_PLUS\_CONST = 7
* SIGNALGENERATOR\_MULT\_CONST = 8

**Hint:** This feature is only supported with the tool adapter for LABCAR-OPERATOR Experiment Environment 3.5 or higher!

5.8.7 Mappings

All information to be used inside the test case for the signal generator have to be provided accordingly in the mapping file. This is independent if you use C# as programming language or the Automation Sequence Builder to set up the test case graphically.

Mappings are neccesary for

* Signal channel = <label>.sgchannel
* Signal mode = <label>.mode
* Signal generator = <label>.sg
* Acquisition task

Whereby <label> is the special signal tool label of LABCAR-OPERATOR.

5.8.8 Working with dSpace

* StimuliSetInfo.xml (for ControlDesk)

<?xml version="1.0" encoding="utf-8"?>

<File type="Stimuli Set Info" extension="XML" version="1.0">

<SG name="SG">

<SGModelName name="SignalGenerator\_0" />

<MPSubSystem name="" />

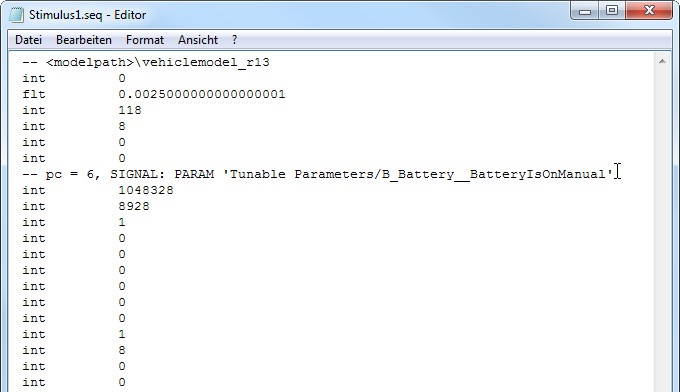
<SGSet name="start" file="Stimulus1.seq" />

<SGSet name="stop" file="Stimulus2.seq" />

</SG>

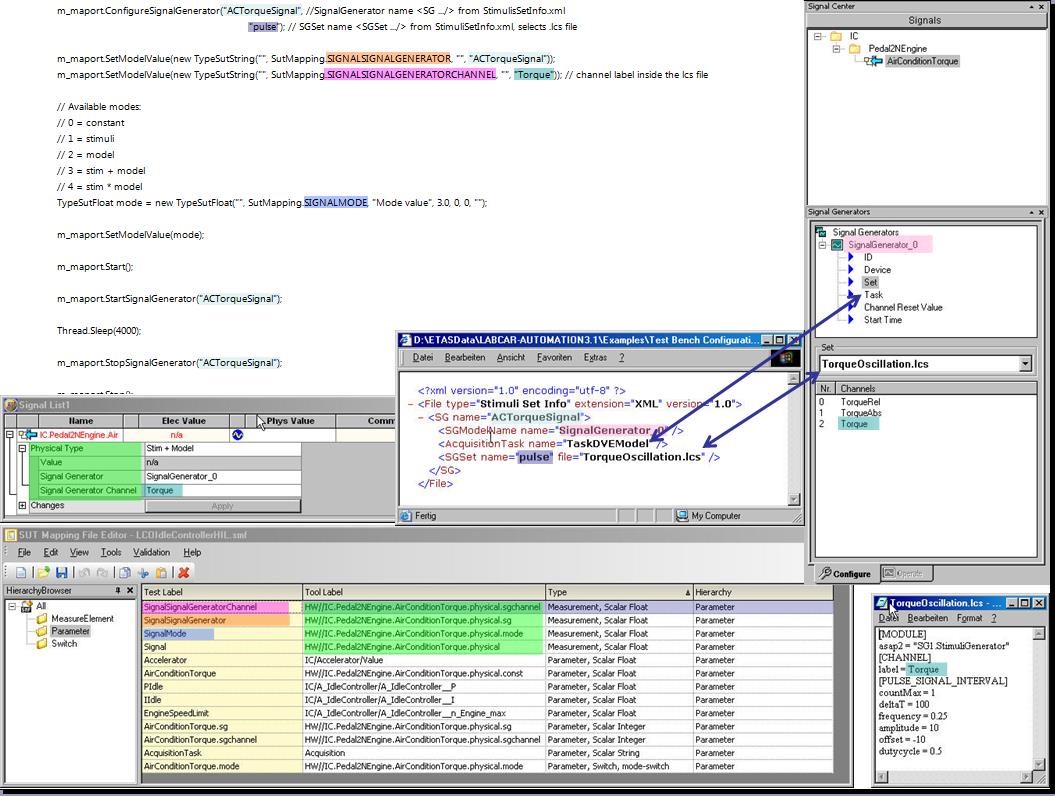
</File>

* Signal trace file - input for signal generator (see example picture right side)



5.8.9 Example with LABCAR-OPERATOR 3.2.5

To get a general idea please find here an overview of the used data in the different tool windows.



1. **General issues**

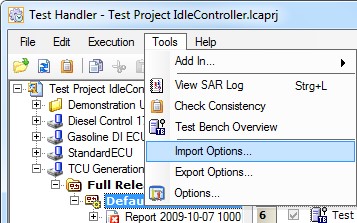
6.1 Test Handler Tool Options – across different installations

You may change your options in the Test Handler, e.g. for Reporting or UuT-TBC mappings.

After installation of a new version it is rather conceivably you like to have these option available for the new version as well. Another requirement might be to save different tool option configurations for further reuse.

The functionality you need, is to import and export these options. You find the new functions a the Tools Menu.

To take over the content of the imported ToolOptions.conf file, a restart of the Test Handler is mandatory.



Additionally the options file ToolOptions.conf will not be deleted during uninstall of LABCARAUTOMATION from Version 3.4 onwards. Completive this file will not be overwritten during re-installation. Thus you have the options available right after re-installation or repair.

An import is not necessary.

If you like to force to overwrite the Test Handler Tool Options during re-installation or repair to the initial default values, please delete the ToolOptions.conf file before starting installation or repair.

For details see the Release Notes.pdf you find on your LABCAR-AUTOMATION Installation CD and at the ETAS program folder.

6.2 Silent Installation

LABCAR-AUTOMATION can be silently installed via the command line. Therefore call ‘setup.exe /silent’.

Examples:

* To specify which feature to be installed use the “feature” option like ‘Setup.exe /features="Core,Doc,Basic,TestDesign,ReportViewer,Examples,Shell,TestCreation,Te stManager,TestHandler,Editors,Misc,ConfigurationWizard" /silent’
* To specify ALL Features to be installed use the below option

Setup.exe /features="ALL" /silent

* To Install ReportViewer alone, use the below option

Setup.exe /features="Core,Basic,ReportViewer" /silent

In the following list, the LCA feature names used for silent installation are listed:

* Please be aware, that every LCA installation must contain the core, otherwise the operation of LCA will fail

|  |  |  |
| --- | --- | --- |
| **LABCAR-AUTOMATION feature name** | | **LABCAR-AUTOMATION components** |
| Core | | All core components of LABCAR-AUTOMATION including Tool Adaptors for INCA, LCO/EE and  INTECRIO |
|  Doc | | User manuals and product documentation. |
|  Basic | |  |
| o TestDesign | | The main LCA modules (required for operation):  Executor (Engine), Automotive Test Case Library (ATCL) |
| o ReportViewer | | The Report Viewer to view the test reports. |
| o Examples | | The examples folder |
| Shell | |  |
|  TestCreation | | Automation Sequence Builder |
|  TestManager | | Test Manager |
|  TestHandler | | Test Handler |
|  Editors | | Editors Package |
|  Misc | | Project Generator |
| o ConfigurationWizard | | Configuration Wizard |
| ToolAdapters | |  |
|  RealTimeScriptingAdapter | | Real Time Testing Tool adaptor |
|  dSpaceAdapter | | dSpace ControlDesk Tool adaptor |
|  | ODXAdapter | Diagnose Tool adaptor for ODX Link (acc. INCA  Version) |
|  | ES4440Adapter | Fault Simulation Tool Adaptor (ES4440) |

For an overview on the LABCAR-AUTOMATION components please refer to the LABCARAUTOMATION release notes, chapter ‘LABCAR-AUTOMATION Packaging’, delivered with this CD and placed in the Programs / ETAS folder after installation.

All the other additional Components, like Tool Adaptors for MLBA4, CANape, Excel etc. are not handled in the LABCAR-AUTOMATION-4.x.msi (resp. Setup.exe). Separate installation is required, the relevant installation files (.msi) are provided in the Installation CD.

* Please be aware, that the LABCAR-AUTOMATION is completely controlled by licenses. This means, although you might have installed a specific component, it would not work in case you do not own an appropriate license. In this case please contact ETAS sales department, to order to correct component license

The different addons (tool adapters) can be silently installed as given below: “msiexec /i <addon\_installer>.msi /passive”

6.3 License Management

Every component of the LABCAR-AUTOMATION is protected by use of a license.

During Installation you are able to select ‘your’ package. If you select another option or more than the available licenses grant, all selected components are installed, however they are usable only in a ‘Grace mode’. This mode allows you to test the full functionality of the components for a time frame of 30 operational days. After expiration of this time the components without valid license do not work any longer.

For further particulars we refer you to the documentation “How to get a license file” listed in the tool section of the Installation CD and the user’s guide to LABCAR-AUTOMATION.

**Note:**

Machine based licenses do not work together with Microsoft’s “Remote Desktop Connection”, but you can use “Remote Desktop Sharing” of Windows NetMeeting, or use VNC.

There is no issue with Remote Desktop with a server based license.

ETAS Contact Addresses

**7 ETAS Contact Addresses**

ETAS HQ

|  |  |  |
| --- | --- | --- |
| ETAS GmbH  Borsigstraße 14 | Phone: | +49 711 89661-0 |
| 70469 Stuttgart | Fax: | +49 711 89661-106 |
| Germany  ETAS Subsidiaries and Technical | WWW:  Support | www.etas.com |

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:

|  |  |  |
| --- | --- | --- |
| ETAS subsidiaries | WWW: | [www.etas.com/en/contact.php](http://www.etas.com/en/contact.php) |
| ETAS technical support | WWW: | [www.etas.com/en/hotlines.php](http://www.etas.com/en/hotlines.php) |

LABCAR-AUTOMATION 4.2.2 56