



DRIVING EMBEDDED EXCELLENCE

# ETAS ASCET-RP V6.4

## 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 2023** ETAS GmbH, Stuttgart

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

ASCET-RP V6.4 – User Guide R08 EN – 03.2023

# Contents

---

<b>1</b>	<b>Safety and Privacy Information</b>	<b>5</b>
1.1	Intended Use	5
1.2	Target Group	5
1.3	Classification of Safety Messages	5
1.4	Safety Information	6
1.5	Privacy Notice	6
1.5.1	Data Processing	7
1.5.2	Data and Data Categories	7
1.5.3	Technical and Organizational Measures	7
1.5.4	Description of Problem Report	8
<b>2</b>	<b>About ASCET-RP</b>	<b>9</b>
2.1	Installation	9
2.1.1	Components	9
2.1.2	Sample Files	9
2.2	Finding Out More	10
2.2.1	ASCET-RP User Guide	10
2.2.2	ASCET-RP Online Help	11
<b>3</b>	<b>Configuring Experimental Targets</b>	<b>12</b>
3.1	The Hardware Options	12
3.1.1	"Hardware Configuration" Subnode	13
3.1.2	"Signal Mapping" Subnode	13
3.1.3	"HWC-Paths" Subnode	13
3.1.4	"Hardware Connection" Subnode	14
3.2	Hardware Connection with the ETAS Network Manager	14
3.3	Selecting a Compiler	14
3.3.1	Using Your Own Compiler	15
3.3.2	Changing Target or Compiler	15
3.4	Hints on Using ASCET-RP	17
3.4.1	Preprocessing Available Data Bases	17
3.4.2	Using dT	17
<b>4</b>	<b>Rapid-Prototyping Experiments</b>	<b>18</b>
4.1	Experimenting with ASCET	18
4.1.1	The User Interface	18
4.1.2	Running Online Experiments	20
4.1.3	Standalone Mode	25
4.2	Experimenting with INCA	26
4.3	Experimenting with INTECRIO	33
<b>5</b>	<b>Hardware Configuration</b>	<b>34</b>
5.1	Hardware Configurator	34

5.2	Migration of Existing Projects .....	37
5.3	HWC Editor (Legacy) .....	37
<b>6</b>	<b>ETK Bypass .....</b>	<b>38</b>
6.1	ETK Bypass: Definition .....	38
6.2	Hardware Configuration of an ETK Bypass .....	38
6.3	ASCET Project for the ETK Bypass .....	39
6.4	How the ETK Bypass Works .....	40
6.5	Data Exchange Between Control Unit and ETAS Experimental System .....	41
6.6	Initially Required Information and Data .....	47
<b>7</b>	<b>Bypass Concept .....</b>	<b>50</b>
7.1	ETK Bypass Concept Description .....	50
7.2	Bypass Input .....	50
7.3	Hook-Based Bypass .....	51
	7.3.1 Classical .....	51
	7.3.2 With Distab17 .....	51
7.4	Service-Based Bypass .....	52
7.5	Safety Considerations .....	54
	7.5.1 Bypass Input Data .....	54
	7.5.2 Bypass Calculation .....	54
	7.5.3 Bypass Output Data .....	54
	7.5.4 Message Copies .....	54
7.6	Service-Based Bypass Specifics .....	55
	7.6.1 Service Processes for the SBB Implemented as Service Functions .....	56
	7.6.2 Controlling the ECU Behavior from ASCET-RP .....	57
	7.6.3 Summary .....	57
<b>8</b>	<b>Troubleshooting General Problems .....</b>	<b>59</b>
8.1	Network Adapter cannot be selected via Network Manager .....	59
8.2	Search for Ethernet Hardware fails .....	59
<b>9</b>	<b>Contact Information .....</b>	<b>64</b>
	<b>Figures .....</b>	<b>65</b>
	<b>Index .....</b>	<b>66</b>

# 1 Safety and Privacy Information

---

In this chapter, you can find information about the intended use, the addressed target group, and information about safety and privacy related topics.

Please adhere to the ETAS Safety Advice (**Help > Safety Advice**) and to the safety information given in the user documentation.

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

## 1.1 Intended Use

---

The ASCET tools support model-based software development. In model-based development, you construct an executable specification – the model – of your system and establish its properties through simulation and testing in early stages of development. When the model behaves as required, it can be converted automatically to production quality code.

The ASCET **Rapid Prototyping** (ASCET-RP) software package is used to integrate the ES900 experimental target (E-Target), together with I/O periphery, in ASCET. With that, ASCET-RP enables rapid prototyping of software functions created in ASCET – both in the laboratory and in the vehicle. This enables early validation of functions in the real world.

## 1.2 Target Group

---

This ASCET-RP User Guide is a supplement to the ASCET documentation (Getting Started and online help). It addresses qualified personnel working in the fields of automobile control unit development and calibration. Specialized knowledge in the areas of measurement and control unit technology is required.

You should be familiar with the basic features and operation of ASCET before installing and using ASCET-RP.

## 1.3 Classification of Safety Messages

---

The safety messages used here 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

Please adhere to the ETAS Safety Advice and to the following safety information to avoid injury to yourself and others as well as damage to property.



### **WARNING**

#### **Harm or property damage due to unpredictable behavior of vehicle or test bench**

Wrongly initialized NVRAM variables (NV variables) can lead to unpredictable behavior of a vehicle or a test bench. This behavior can cause harm or property damage.

ASCET projects that use the NVRAM possibilities of ASCET-RP targets expect a *user-defined* initialization that checks whether all NV variables are valid for the current project, both individually and in combination with other NV variables. If this is not the case, all NV variables have to be initialized with their (reasonable) default values.

Due to the NVRAM saving concept, this is *absolutely necessary* when projects are used in environments where any harm to people and equipment can happen when unsuitable initialization values are used (e.g. in-vehicle-use or at test benches).

In addition, take all information on environmental conditions into consideration before setup and operation (see the documentation of your computer, hardware, etc.).

Further safety advice for this ETAS product is available in the following formats:

- In electronic form on the DVD: Documentation\ETAS Safety Advice.pdf
- The ASCET V6.4 safety manual, available at ETAS upon request

## 1.5 Privacy Notice

Your privacy is important to ETAS so we have created the following Privacy Statement that informs you which data are processed in ASCET, which data categories ASCET uses, and which technical measure you have to take to ensure the users' privacy. Additionally, we provide further instructions where this product stores and where you can delete personal data.

### 1.5.1 Data Processing

Note that personal data or data categories are processed when using this product. As the controller, the purchaser undertakes to ensure the legal conformity of these processing activities in accordance with Art. 4 No. 7 of the General Data Protection Regulation (GDPR). As the manufacturer, ETAS GmbH is not liable for any mishandling of this data.

### 1.5.2 Data and Data Categories

Note that this product creates files containing file names and file paths, e.g. for purposes of error analysis, ensuring correct uninstallation, referencing source libraries, or for communicating with third-party programs.

The same file names and file paths may contain personal data, if they refer to the current user's personal directory or subdirectories (e.g., `C:\Users\<UserId>\Documents\...`).

If you do not want personal information to be included in the generated files, please make sure of the following:

- The workspace of the product points to a directory without personal reference.
- All settings in the product (see the menu option **Tools > Options** in the product) refer to directories and file names without personal reference.
- All project settings in the product (see the menu option **File > Properties** in the ASCET project editor) refer to directories and file names without personal reference.
- Windows environment variables (such as the temporary directory) refer to directories without personal reference because these environment variables are used by the product.

In this case, please also make sure that the users of this product have read and write access to the newly set directories.

When using the ETAS License Manager in combination with user-based licenses, particularly the following personal data or data categories are recorded for the purpose of license management:

- User data: UserID
- Communication data: IP address

As an option, the following personal data or data categories may be recorded for the purpose of assisting development:

- Problem Report, see section 1.5.4 below

When using the ASCET add-on ASCET-DIFF, particularly the following personal data or data categories are recorded for the purposes of user-specific settings and user-specific log files:

- User data: UserID

### 1.5.3 Technical and Organizational Measures

This product does not itself encrypt the personal data or data categories that it records. Ensure that the data recorded is secured by means of suitable technical or organizational measures in your IT system, e.g. by using classic anti-theft and access protection.

Personal data in generated files can be deleted by tools in the operating system.

## 1.5.4 Description of Problem Report

### **Purpose:**

When an error occurs, ASCET offers to send an error report to ETAS for troubleshooting. ETAS uses the personal information to have a contact person in case of system errors.

### **Personal Data:**

The problem report may contain the following personal data or data category:

- user data
  - name and address entered during the installation process
  - UserID
- communication data
  - IP address

Additionally to the problem information that is entered by the users themselves, ASCET-RP collects the available product-related log files in a zip archive to support the bug fixing process at ETAS. The zip file is named by using the following pattern `EtasLogFiles<index number>.zip` and stored in the ETAS-specific log files directory.

This automatically created zip file contains the following:

- product-related log files created at installation time. (Necessary for uninstall action)
- ETAS log files stored in the ETAS log files directory matching the file name pattern `*.log`
- recursive registry export of ETAS (32bit)-key (and sub keys):  
`HKEY_CURRENT_USER\Software\ETAS`
- registry export of ETAS (32bit)-key (and sub keys):  
`HKEY_LOCAL_MACHINE\Software\ETAS`

All ETAS-related log files in the ETAS-specific log files directory and the zip archives created by the Problem Report feature can be removed after closing all ETAS applications if they are no longer needed.



## 2 About ASCET-RP

---

The execution of real-time software requires experimenting hardware that is capable of real-time processing. ASCET-RP is used to integrate the ES900 experimental target (E-Target) in ASCET. Together with I/O periphery, powerful development systems can be built on the basis of these experimental targets.

In addition to the license for the compiler toolset used, the ASCET-RP package also includes extensions to the ASCET development environment, such as for the homogeneous integration of compiler and linker calls and the RTA-OSEK operating system kernel for experimental targets.

ASCET-RP contains, in addition, the basic functions of the ASCET base software. You can display existing models, but you cannot edit them. If you want to use the ASCET modeling functions, you need ASCET-MD, too.

### NOTE

Since V6.4.5, ASCET-RP supports only the ES900 hardware system.

Since ASCET-RP V6.4.6, projects with an ES1000 hardware configuration cannot be created or used.

You can still create and use projects with an RTPRO-PC hardware configuration. To do so, however, you have to use an external INTECRIO V4.7.2 or older as hardware configurator. See "To enable an external Hardware Configurator" on page 34 for details.

## 2.1 Installation

---

The ASCET-RP installation is described in the ASCET installation guide (`ASCET V6.4 Installation.pdf` on the installation disk). Details on licensing are also given in the ASCET installation guide.

### 2.1.1 Components

The ASCET-RP V6.4 installation includes the following components:

- Integration of the ES910 target
- Integration of the I/O interfaces for the ES910 target
- Integration of the RTPRO-PC

### NOTE

To work with the RTPRO-PC target, you have to use an external INTECRIO V4.7.2 or older as hardware configurator.

- QCC compilers
- Documentation and examples

### 2.1.2 Sample Files

After the installation of ASCET-RP V6.4, the sample databases are located in the `EXPORT` directory of your ASCET installation in the `Tutorial RTIO.*1` and `Tutorial INTECRIO.*1` files.

---

1. \* = `exp` (binary export format) or `axl` (xml-based export format)

## 2.2 Finding Out More

If not specified otherwise during installation, the following PDF manuals are available in the `ETAS\ETASManuals` folder after installing ASCET-RP and ASCET-MD:

- ASCET Getting Started (`ASCET V6.4 Getting Started.pdf`)
- ASCET Installation Guide (`ASCET V6.4 Installation.pdf`)
- ASCET Icon Reference Guide (`ASCET V6.4 Icon Reference Guide.pdf`)
- ASCET AUTOSAR User Guide (`ASCET V6.4 AUTOSAR User Guide.pdf`)
- AUTOSAR to ASCET Importer User Guide (`ASCET V6.4 AUTOSAR To ASCET Converter User Guide.pdf`)



### NOTE

The cooperation of ASCET and AUTOSAR requires the installation of the ASCET-SE target `ANSI-C`.

- ASCET-RP user guide (this manual; `ASCET-RP V6.4 User Guide.pdf`)
- ASCET-RP online help; accessible via the **Help** menu and <F1> in the Hardware Configurator

When you ASCET-SE, ASCET-SCM, or ASCET-DIFF, further documentation is available:

- ASCET-SE
  - ASCET-SE user guide (`ASCET-SE V6.4 User Guide.pdf`)
  - EHOOKS Add-On user guide (`ASCET-SE V6.4 EHOOKS Add On User Guide.pdf`)
- ASCET-SCM
  - online help (integrated in the main ASCET online help)
- ASCET-DIFF
  - ASCET-DIFF installation guide
  - separate online help; accessible via the **Help** menu and <F1> in the ASCET-DIFF windows

### 2.2.1 ASCET-RP User Guide

The ASCET-RP V6.4 user guide consists of the following main sections:


- General Section (chapters 1, 2, 3, 4)
- Hardware Configuration (including Bypass Interface, chapters 5, 6, 7)
- Appendix (chapters 8)

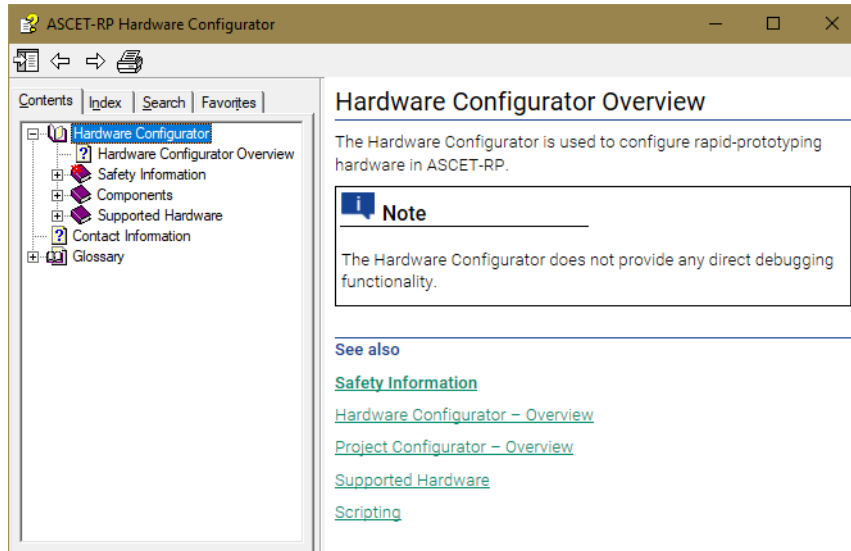
The general section is intended for all users of ASCET-RP V6.4. Here, you find information about the structure, installation and usage of ASCET-RP V6.4.

The subsequent chapters introduce and explain the functionality and operation of ASCET-RP.

Using the index, full-text search, and hypertext links, you can find references fast and conveniently.

## 2.2.2 ASCET-RP Online Help

In the Hardware Configurator, use the **Help > Help** menu option or the  button to invoke the general help function. Press the <F1> function key to call context-sensitive help.



The tabs of the help window provide you with the following options:

- The "Contents" tab allows you to browse the help topics by categories.
- The "Index" tab lists all index entries. Browse the entire list, or enter a search term to limit the scope of listing.
- The "Search" tab allows you to search for individual words or terms included in a help topic. Type a search string and let the help function list the entries it has found related to this term.
- The "Favorites" tab allows you to bookmark topics.

## 3 Configuring Experimental Targets

ASCET Rapid Prototyping V6.4 (ASCET-RP V6.4) contains the compiler and linker tools required for producing executable files for a rapid prototyping target, and an extension of the ASCET development environment for the hardware integration. The target itself can be selected in the target options of the project. That way the targets are fully integrated into ASCET.

The configuration of the compiler and the linker, as well as the description of the interface to the actual target hardware is not described in ASCET directly, but either with the help of the ETAS Network Manager or with the help of \*.ini files. Section 3.1 describes the hardware options of ASCET-RP, hardware connection using the ETAS Network Manager is described in section 3.2.

Section 3.3 describes the selection of the compiler, and section 3.4 gives some hints on using ASCET-RP.



### NOTE

Since V6.4.5, ASCET-RP supports only the ES900 hardware system.

Since ASCET-RP V6.4.6, projects with an ES1000 hardware configuration cannot be created or used.

You can still create and use projects with an RTPRO-PC hardware configuration. To do so, however, you have to use an external INTECRIO V4.7.2 or older as hardware configurator.

### Structure of the E-Target Directories

Installing ASCET-RP V6.4 results in the creation of several e-target subdirectories in the ..\ASCET6.4\Target directory. These subdirectories contain E-target-specific information, configuration and library files.

The following table lists the subdirectories:

ASCET Subdirectory	E-Target	Computer Node
..\Target\ES910	ES910.2/ES910.3	ES910
..\Target\Prototyping	(determined in INTECRIO)	
..\Target\QNx86	RTPRO-PC	RTPRO-PC

ASCET-RP versions prior to V6.4.5 also supported the ES1130 and ES1135 targets. These targets have been completely removed from ASCET-RP.

All makefiles and build scripts support paths with blanks.

- If a path containing blanks is to be used in a makefile, ASCET converts it to short Windows format (for example, c:\Documents and Settings would be converted to c:\DOCUME~1).
- If a path containing blanks is to be used in a batch file, ASCET generates it encapsulated in ", or converts it to short Windows format.

### 3.1 The Hardware Options

ASCET-RP adds the "Hardware" node and its subnodes to the ASCET options window. These nodes contain all tool options available for hardware configuration.

### 3.1.1 "Hardware Configuration" Subnode

In this node, you can specify the hardware configurator you want to use.

- **Use separately installed INTECRIO for hardware configuration**  
Activate this option if you want to use a separately installed version of INTECRIO as Hardware Configurator, instead of the one installed with ASCET-RP V6.4.  
A compatible INTECRIO installation is required for this purpose.
- **HW configurator path**  
Only available if **Use separately installed INTECRIO ...** is activated.  
Path to the INTECRIO installation you want to use as Hardware Configurator.
- **HWC working directory**  
Working directory for internal usage by the Hardware Configurator. Here, temporary files (e.g., \*.hwx) are stored.



#### NOTE

To work with the RTPRO-PC target, you must select an external INTECRIO V4.7.2 or older as hardware configurator.

### 3.1.2 "Signal Mapping" Subnode

In this node, you can specify settings for automatic mapping.

- **Message Creation Target**  
This option determines whether ASCET messages generated during automatic mapping are created in the project itself (`Project`, default) or in one of the included modules (`Module`).
- **Module Instance Name**  
This option is only effective when "Message Creation Target" is set to `Module`.  
Enter the name of the module for the generated messages in the input field. You can use any module included directly or indirectly (in another module) in the project. For modules in modules, the module names are separated by a dot. If, e.g., module A contains module B, you enter `A.B` if you want to create the messages in module B.  
If the selected module is not included in the project, the following error message appears:

```
Can't add Message '<msg name>', because module
'<module name>' doesn't exist in project!
```

### 3.1.3 "HWC-Paths" Subnode

This node shows various paths. It is read-only.

### 3.1.4 "Hardware Connection" Subnode

In this node, you can specify options for the connection between ASCET-RP and the hardware. It is suggested that you select the item you use most frequently.



#### NOTE

The options in this node are described in the ASCET online help.

## 3.2 Hardware Connection with the ETAS Network Manager

Hardware configuration with ETAS Network Manager is described in the ASCET online help, section "INTECRIO Connectivity / ASCET-RP".

The hardware selection window, named "Experimental Target Hardware Selection", is described in the same section of the ASCET online help.

## 3.3 Selecting a Compiler

A GNU compiler (MinGW GNU 4.7.2 in the user interface) is integrated in ASCET-RP for the Prototyping target. The QCC compiler (QCC v6.5.0 in the user interface) is integrated in ASCET-RP for the ES910 and RTPRO-PC targets.

Target	Compiler
ES910	QCC compiler
Prototyping	MinGW GNU compiler
RTPRO-PC <sup>a</sup>	QCC compiler

- a. Since ASCET-RP V6.4.5, this target is no longer supported. To use it, you need INTECRIO V4.7.2 or older as hardware configurator.

**Tab. 3-1** Target/Compiler Overview

For each compiler, version-specific makefiles are provided in the `\Target\ES910\trgmake`, `\Target\Prototyping\trgmake`, or `\Target\QNx86\trgmake` directory. Version-independent settings are stored in files named `settings_<compiler>_common.mk`. In these files, you can set separate compiler options for module code, project code and init code. The following section in the makefiles is provided for that purpose:

```
# Compilation settings for different lists of files
FILES_MODULES_INV = $(CC_INV)
                   #Add specific options here

FILES_PROJECT_INV = $(CC_INV)
                   #Add specific options here

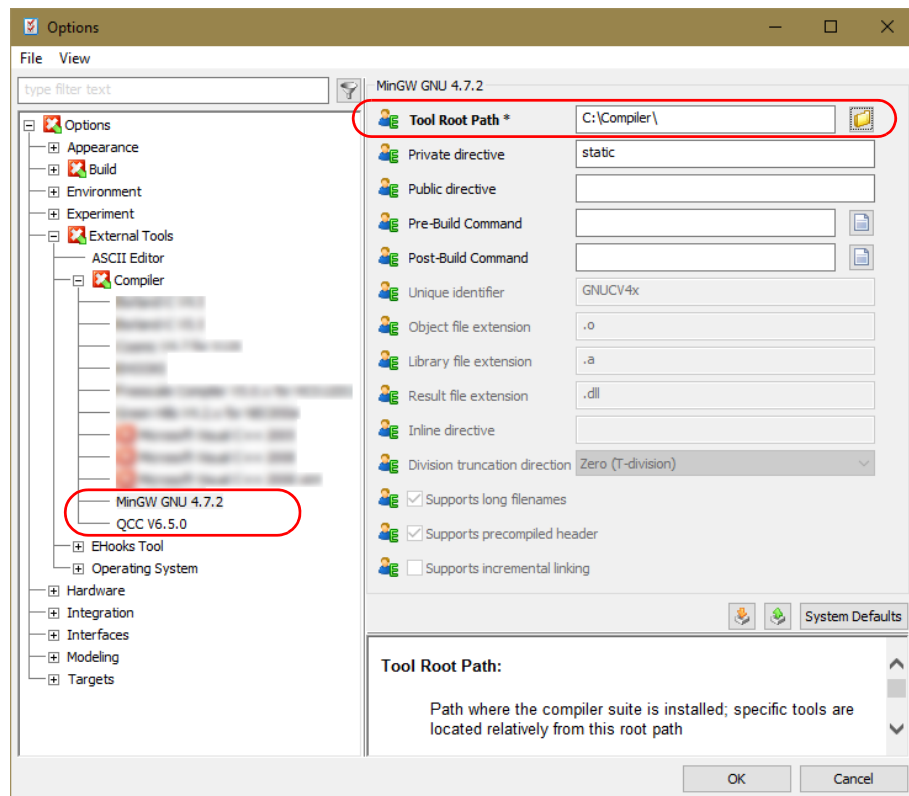
FILES_INIT_INV    = $(CC_INV)
                   #Add specific options here
```

Precompiling C header files used by many C code files can significantly speed up the compilation process. The MinGW GNU compiler supports usage of precompiled headers. Only one precompiled header file can be used per compilation. Therefore, `Project` should be selected in the "Header/C Code Structure" list of the "Build" node of the project settings, to use precompiled headers as

effectively as possible. Using precompiled headers is activated automatically (ASCET options window, "MinGW GNU 4.7.2" node, **Supports precompiled header** option).

### 3.3.1 Using Your Own Compiler

If you want to use your own compiler, you have to reset the path of the MinGW GNU 4.7.2 or QCC V6.5.0 compiler to the new compiler in the ASCET options dialog.



### 3.3.2 Changing Target or Compiler

You have to select a suitable combination of target and compiler for your ASCET-RP project. Proceed as follows.

#### To select a suitable target/compiler combination



1. In the project editor, click on the Project Properties button. The "Project Properties" window opens.
2. In the "Build" node, select the target ES910 or Prototyping.

**NOTE**

You can also select the RTPRO-PC target.

To actually use that target, you have to select an external INTECRIO V4.7.2 or older as hardware configurator. See "To enable an external Hardware Configurator" on page 34 for details.

A suitable compiler is selected automatically.

3. Click **OK**.

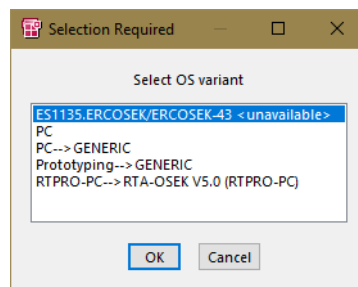
If your project uses C code blocks, and if you used another target before, you should copy the C code to the current target. This procedure is described in section "To copy operating system settings and C Code" below.

4. Finally, select **Build > Touch > Recursive** so that all components of the project are recompiled in the next run.

#### To copy operating system settings and C Code

1. Select the "OS" tab in the project editor.
2. Select **Operating System > Copy From Target** from the project editor.

The "Selection Required" window opens.

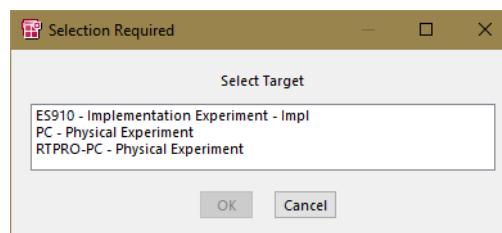


3. From the "Selection Required" window, select the original target of the old project.
4. Click **OK**.

The operating system code is copied from the old target to the current target.

5. In the project editor, select **Extras > Copy C-Code From**.

The "Selection Required" window opens.



6. In the "Selection Required" window, select the old target, experiment and implementation of the project.
7. Click **OK**.

The code is copied from the old target and experiment to the current settings.



## 3.4 Hints on Using ASCET-RP

### 3.4.1 Preprocessing Available Data Bases

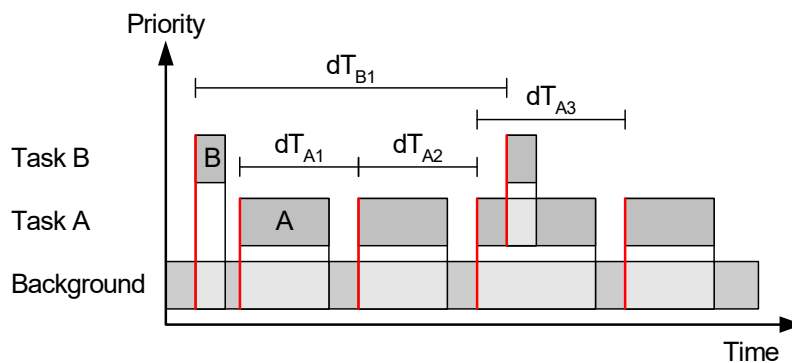
ASCET databases which were created with ASCET versions prior to V4.x must be converted to ASCET V4.x before they can be opened and converted with ASCET V6.4.

#### NOTE

Detailed information on converting very old ASCET projects (with TIPEXP V3.x and older, Target PPC) is given in the ASCET online help.

### 3.4.2 Using $dT$

An RTA-OSEK operating system is implemented for the ES910 and RTPRO-PC targets. The operating system enables access to the time  $dT$  which has elapsed between the last and second-last call of the running task.  $dT$  always refers to the task in which the variable is used (see Fig. 3-1).



**Fig. 3-1**  $dT$  Scheme

$dT$  is a global `uint32` variable. It is declared in one of the ERCOS<sup>EK</sup> or RTA-OSEK header files and contains the value for the current task in units of system ticks.



$dT$  can be accessed from ASCET using the **dT** button in the editors. This enables you to create an element which contains the time in units of seconds.

If users do not generate this element in the C code editor, but still access  $dT$ , no error message appears because  $dT$  is declared in the RTA-OSEK files. But as  $dT$  in RTA-OSEK and  $dT$  in ASCET have different units (system ticks or seconds respectively), the calculations are incorrect. Users should therefore ensure that they generate the corresponding element with the **dT** button.

In the generated code for experimental targets, access to the global element  $dT$  is indirect: A C code macro is generated to access  $dT$ . The name of the macro is created as follows:

```
_<PROJECTNAME>_ <IMPLEMENTATION>_dTAccess
```

A default definition of the macro is generated by ASCET which can be replaced by a user-defined definition.

## 4 Rapid-Prototyping Experiments

This chapter describes the different possibilities of running a Rapid-Prototyping experiment.

### NOTE

Since V6.4.5, ASCET-RP supports only the ES900 hardware system. Beginning with ASCET-RP V6.4.6, projects with an ES1000 hardware configuration cannot be created or used. You can still create and use projects with an RTPRO-PC hardware configuration. To do so, however, you have to use an external INTECRIO V4.7.2 or older as hardware configurator.

Experiments that use an ES910 or RTPRO-PC target contain two memory pages. To make use of both memory pages, you have to use INCA/INCA-EIP as experiment environment (see also section 4.2); the ASCET or ASCET experiment environment does not support multiple memory pages. See the INCA and INCA-EIP documentation for details on using memory pages.

### 4.1 Experimenting with ASCET

If you want to run the Rapid-Prototyping experiment in ASCET, you can choose between an online and an offline experiment in the project editor. For more details, please refer to the ASCET online help.

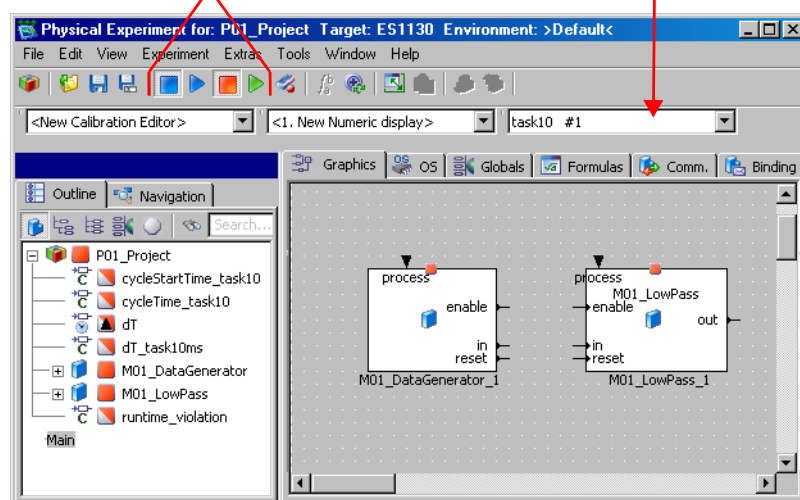
Only the special features of the online experiment are described here.

#### 4.1.1 The User Interface

The user interface of the online experiment is very similar to that of the offline experiment. However, the buttons for controlling the experiment and the NVRAM cockpit, the "Task" combo box and the functions in the **Experiment** and **Tools** menus are different from the offline experiment.

Control buttons (experiment)

"Task" combo box



### 4.1.1.1 Buttons



- A Exit to Component (ends the experiment and invokes the project editor)
- B Load Environment (loads an experiment environment, i.e. predefined measure and calibration windows with assigned variables)
- C Save Environment (saves the current experiment environment)
- D Save Environment As (saves the current experiment environment under a freely definable name)
- E Stop OS (stops the operating system and thus the experiment)
- F Start OS (starts the operating system and thus the experiment)
- G Stop Measurement (stops measurement, i.e. the data display)
- H Start Measurement (starts measurement, i.e. the data display)
- I Open Data Logger
- J Open CT Solver (opens a window in which you can configure the integration method)  
This button is *only* available if you are experimenting with a CT block or a hybrid project.
- K Update Dependent Parameters (updates the values of dependent parameters)
- L Expand / Collapse Window (shows/hides the component display)
- M Always on top (keeps the experiment window on top)  
This button is only available if the window is collapsed.
- N Navigate down to selected component of element tree (shows the included component selected in the "Outline" tab.)
- O Navigate up to parent component (shows the parent component)

If you are experimenting with the RTPRO-PC or ES910 target, the toolbar contains another button **Open NVRAM Cockpit**.



#### NOTE

To work with the RTPRO-PC target, you have to use an external INTECRIO V4.7.2 or older as hardware configurator.

### 4.1.1.2 Experiment Menu

- *Stop OS (<F6>)*  
Stops the operating system.
- *Start OS (<F7>)*  
Starts the operating system.
- *Stop Measurement (<F8>)*  
Stops the measurement.
- *Start Measurement (<F9>)*  
Starts the measurement.

### 4.1.1.3 Tools Menu

- *Data Logger*  
Opens the Data Logger.
- *Target Debugger*  
Opens the debugger window for C code components.
- *NVRAM Cockpit*  
(Only available when the target ES1135, ES910 or RTPRO-PC is selected in the code generation options.)  
Opens the NVRAM Cockpit.

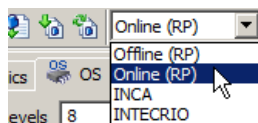
The other elements of the user interface correspond to those of the offline experiment; they are described in the "Experimentation" section of the ASCET online help.

## 4.1.2 Running Online Experiments

Start the online experiment environment for a project from the project editor.

### To start the online experiment

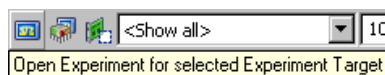
1. Open the project or component you require.
2. If you want to experiment with a component, open the relevant default project.
3. In the project properties, "Build" node, select the target and an appropriate compiler.
4. In the "Experiment Target" combo box, select `Online (RP)`.



`Offline (RP)` is intended for offline experiments on the target, `INCA` and `INTECRIO` are intended for transfer to the respective tools.

The buttons **Open Experiment for selected Experiment Target** and **Reconnect to Experiment of selected Experiment Target** are now available.

5. Do one of the following:
  - Select **Build > Experiment**.
  - Click the **Open Experiment for selected Experiment Target** button.



### To select hardware

When you activated the **Use ETAS Network Manager (enables 'Select Hardware')** option in the hardware options, the hardware selection window opens under certain conditions. The hardware options and the hardware selection window are described in the ASCET online help.

1. In the "Select simulation board of type <type>" field, select the hardware you want to use.

The **OK** button becomes available.

2. If required, perform other settings.
3. Close the window with **OK**.

The system checks whether the selected hardware is available and agrees with the target you selected in the code generation options of the project. If this the case, the experiment environment opens.

#### What to do in case of an error

If no agreement is found between selected and available hardware, you have to close the hardware selection window with **Cancel**. An error message opens:

```
Target connection check failed. No hardware selected for
target <target>. Check target connection again before
starting the build process?
```

1. Do one of the following:
  - Click **Yes** to repeat the search for a hardware connection.
  - Click **No** to start the build process without hardware connection.

The hardware selection window opens again after the Build process.

Or

2. Click **Cancel** to abort the experiment.

After successful hardware selection (cf. page 20), the default experiment environment for the component opens immediately after starting the experiment (cf. page 20), provided that no other experiment environment is available.

Continue reading at "To assign a new element to a measure window" on page 22.

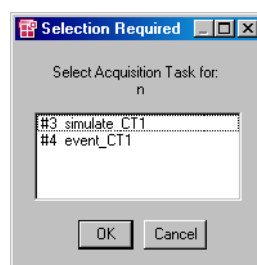
If several environments are available, the "Environment Browser" dialog window opens. Proceed as described in the following instruction.

#### To open the experiment environment for the online experiment

1. In the "Environment Browser" dialog window, select the environment you want to use, and click **Load**.

For more details, refer to the section "Experimentation" in the ASCET online help.

If an existing experiment is opened, but a previous acquisition task is no longer available in the OS, you are prompted to select a replacement for each element that used the old acquisition task.



- In the "Selection Required" window, select one task and click **OK**.

#### To assign a new element to a measure window

- In the "Task" combo box, select the task you want to use as acquisition task for this element.

When the experiment environment is saved, the elements' acquisition tasks are referred to by name.

- In the "Outline" tab, select an element you want to measure.

You can select several elements at the same time. These are assigned the same acquisition task and measure window.

- Assign the selected element(s) to a measure window (see online help for details).

You can assign elements with different acquisition tasks to the same measure window.

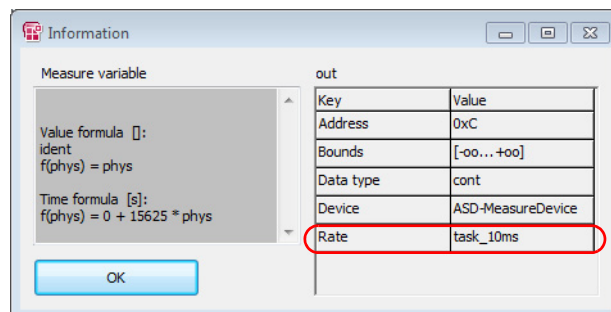
If an element is assigned to several measure windows, all measure windows use the same acquisition task for this element.

#### To check an element's acquisition task

You can check the acquisition task used by an element at a later time.

- Go to the measurement window that contains the element you want to check.
- Right-click the element and select **About Variable** from the context menu.

The "Information" window opens. The acquisition task name is shown in the "out" table on the right, row "Rate".

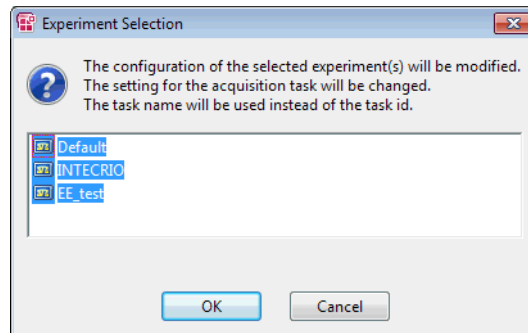


In ASCET-RP versions prior to V6.3, acquisition tasks were referenced by task number instead of task name. Experiments saved in ASCET-RP V6.2 or earlier are updated automatically when you open and re-save them in V6.3 or higher. Alternatively, you can use the **Extras > Bound acquisition task by name** menu option in the project editor to update all experiments saved for a given project.

**To convert acquisition task references**

1. In the project editor, select **Extras > Bound acquisition task by name**.

The "Experiment Selection" window opens. It lists all experiments saved for the project.



2. Select the experiments you want to convert.
3. Click **OK**.

The references to acquisition tasks are converted.

Setting up an online experiment only entails the setting up of the measure and calibration windows. The measure and calibration windows in the online experiment are the same as those in the offline experiment. The "Experimentation" section in the ASCET online help explains how to use them.

Once the experiment has been set up, you can start it. While the online experiment is running, you can modify the display options in all measure and calibration windows, open and close measure/calibration windows and modify data values with the calibration system.

**To start an experiment and measurement**

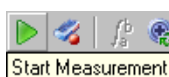
1. Open the experiment environment for the project you want to experiment with.
2. Do one of the following:
  - Select **Experiment > Start OS**.
  - Click the **Start OS** button.



If you set the relevant options in the ASCET options window, "Experiment" node, variables and parameters are initialized each time the OS is started.

The operating system and the experiment are started. Measure data is not displayed yet.

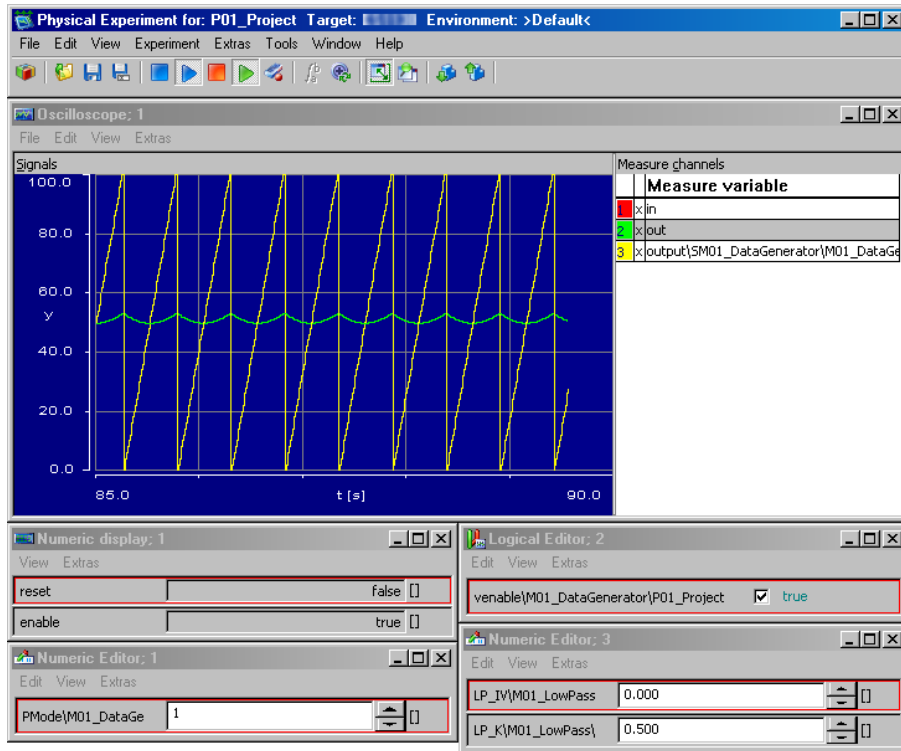
3. Do one of the following:
  - Select **Experiment > Start Measurement**.
  - Click the **Start Measurement** button.



**NOTE**

The measurement may affect the real-time behavior of the model.

Measurement starts. All values set up in the measure system are shown in the respective windows.



**To stop measurement**

1. Do one of the following:
  - Select **Experiment > Stop Measurement**.



- Click the **Stop Measurement** button.

Measurement is stopped, but the experiment continues. When measurement is restarted, the time axis is set to the current value.

All settings remain active. The measure data is retained in the oscilloscope window; you can analyze the data.

**To stop the experiment**

1. Do one of the following:
  - Select **Experiment > Stop OS**.



Any measurement which is currently running is stopped. The operating system, and thus the experiment, is stopped and enters the inactive mode. The inactive mode of the operating system may contain one task with trigger mode `init` that is executed when the operating system is stopped. This can be used to reset external hardware, for instance.

When the operating system is restarted, it goes through the start mode and the corresponding `init` task again. There is no pause function for the operating system. If you set the relevant options in the ASCET options window, "Experiment" node, variables and parameters are initialized.



2. Click the **Exit to Component** button to exit the experiment environment and return to the project editor.



3. Select **File > Exit** to exit the experiment environment and close the project.

As with the offline experiment, you can take a look at the implementation of the project with which you are experimenting from the experiment environment at any time. It does not matter whether the experiment is running or whether it has been stopped.

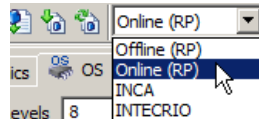
Also as with the offline experiment, components specified in C Code offer additional opportunities for displaying debugger information or error messages during experimenting. You can embed debugger or error messages in your C code. Debugger information is displayed in the Debugger window which can be opened during experimenting. Error messages are displayed in the ASCET monitor window. The debugger in the online experiment works like the debugger in the offline experiment (see the section "Experimentation", subsection "Running Experiments" in the ASCET online help).

### 4.1.3 Standalone Mode

If you have a suitable experimental target, you can run ASCET experiments in standalone mode, without the experimentation environment. For this purpose the experimental target has to be equipped with flash memory and it must be possible to boot from it.

#### To load the experiment in the Flash memory

1. From the "Experiment Target" combo box, select `Online (RP)`.



#### **NOTE**

You must select `Online (RP)` to enable the menu option **Build > Flash Target**.

2. Select **Build > Flash Target**.

When you are working with the ES910 target, you are asked if you want to continue.

3. Click **Yes** to flash the target.

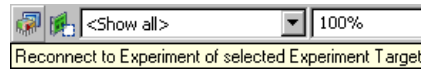
When you activated the **Use ETAS Network Manager (enables 'Select Hardware')** option in the hardware options, the hardware selection window may open.

Section "To select hardware" describes the required actions in this window.

The code generated by ASCET is written to the flash memory of the experimental target instead of to the RAM. A startup routine for booting from the flash memory is integrated into the code. The target hardware will now execute the ASCET model after each reset.

**To experiment in standalone mode**

1. Do one of the following:
  - Select **Build > Reconnect**.
  - Click on **Reconnect to Experiment of selected Experiment Target** to switch to a running online experiment.



The hardware selection window opens under certain conditions.

Section "To select hardware" describes the required actions in this window.

The online experimentation environment will start up as if the experiment had been started from scratch.

You can also reconnect to online experiments running in non-standalone mode that you disconnected from earlier. To disconnect from a running experiment simply exit the experimentation environment, without first stopping the online experiment running on the experimental target.

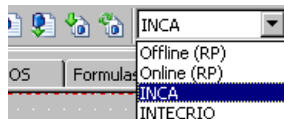
## 4.2 Experimenting with INCA

---

You can not only experiment in ASCET with your project but also in INCA (from Version 4.0.4), together with the add-on INCA-EIP. The project editor offers a function for this purpose which allows convenient transfer of the experiment.

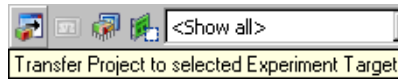
**To initiate a transfer**

1. Open the project with which you want to experiment.
2. From the project properties in the project editor, "Build" node, select the target and a suitable compiler.
3. From the "Experiment Target" combo box, select INCA.



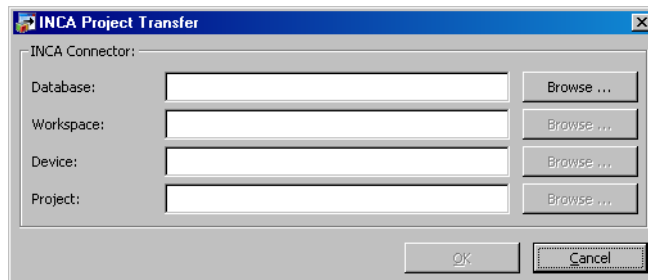
The **Transfer Project to selected Experiment Target** and **Reconnect to Experiment of selected Experiment Target** buttons are now available.

4. Do one of the following:
- Click on **Transfer Project to selected Experiment Target**



- Select **Build > Transfer**.

The "INCA Project Transfer" window opens.



In this window, you define the INCA database, the workspace, and the project within the INCA database you want to use.

If you click one of the **Browse** buttons or the **OK** button, INCA will be launched, if it is not already used. If you have installed several INCA versions, the version that was installed last will be launched – even if this is not the version with the highest version number.

**NOTE**

If your INCA version is too old (i.e. V3.x or older), or if INCA is not installed on your PC, an error message is displayed which cancels the transfer.

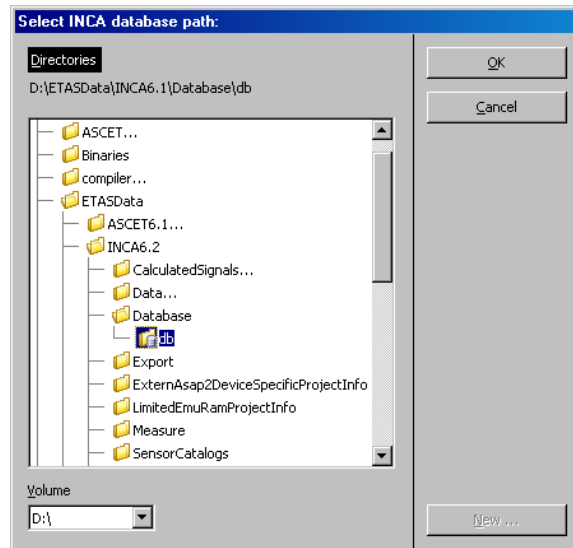
### To set the INCA database path

1. In the "INCA Project Transfer" window, enter an existing database path in the "INCA Database" field.

Or

2. Click on the **Browse** button to search for a database.

The "Select INCA database path" window opens. The databases are shown as folders with an overlay icon (🗄️).



3. Do one of the following:

- Select an INCA database path and click **OK**.

#### **NOTE**

In this window, ASCET databases are marked in exactly the same way as INCA databases. Make sure you really select an **INCA database**.

- Click the **New** button to create a new directory, in which a new INCA database for the project transfer will be created. The new, empty directory is shown as a normal folder.

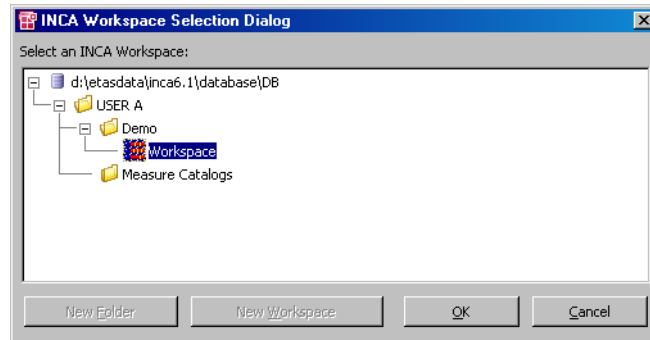
### To select an INCA workspace

1. In the "INCA Project Transfer" window, enter the path and name of an existing workspace in the "INCA Workspace" field

Or

2. Click on the **Browse** button to search for a workspace.

The "INCA Workspace Selection Dialog" window opens. It displays the workspaces in the selected INCA database.



3. Do one of the following:
  - Select an existing workspace.
  - Create a new workspace in a new or existing folder. The procedure is described on page 29.
4. Once you have selected a workspace, click **OK**.

### To create a folder/workspace in the "Workspace Selection Dialog" window

#### A Creating a folder

1. In the "INCA Workspace Selection Dialog" window, select the database name or a folder.

The **New Folder** button is activated.

2. Click on **New Folder**.

3. In the "Create New INCA Folder" window, enter a name and click on **OK**.

The new folder is created.

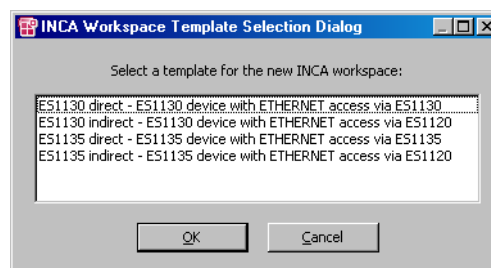
#### B Creating a workspace

1. Select a folder.

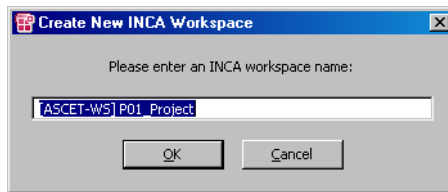
The **New Workspace** button is activated.

2. Click on **New Workspace**.

The "INCA Workspace Template Selection Dialog" window opens.



3. Select a template for the workspace and click **OK**.  
The dialog window "Create New INCA Workspace" opens.



The name [ASCET-WS] <ascet project name> is assigned.

4. Enter a name for the workspace and click on **OK**.  
The new workspace is created in the selected folder and selected.
5. Click **OK** to close the "INCA Workspace Selection Dialog".

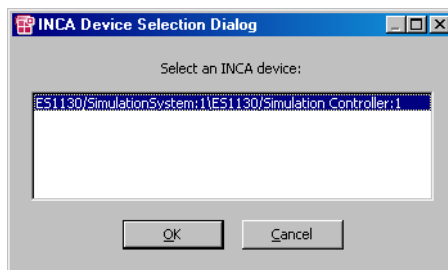
**To select an INCA device**

1. In the "INCA Project Transfer" window, enter a device in the "INCA Device" field.

Or

2. Click on the **Browse** button to search for a device.

The "INCA Device Selection Dialog" window opens. It contains all suitable devices.



3. Select a device and click **OK**.

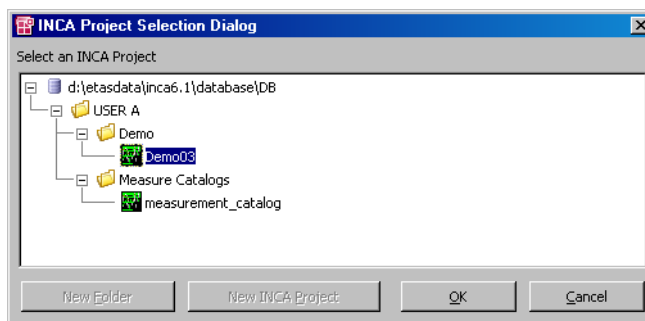
**To select an INCA project**

1. In the "INCA Project Transfer" window, enter the path and name of an existing project in the "INCA Project" field.


Or

2. Click on the **Browse** button to search for a project.

The "INCA Project Selection Dialog" window opens. It displays the projects in the selected INCA database.



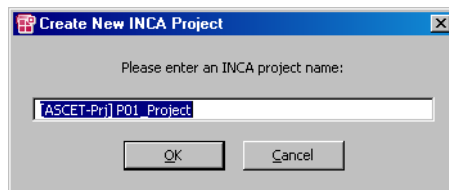
3. Do one of the following:
  - Select an existing project.
  - Create a new project in a new or existing folder. The procedure is described on page 31.
4. Once you have selected a project, click on **OK**.

 **NOTE**

The selected project is replaced by the transferred project without any warning when the transfer starts.

#### To create a folder/project in the "INCA Project Selection Dialog" window

- A Creating a folder
  1. To create a new folder, proceed as described under "Creating a folder" on page 29.
- B Creating a project
  1. Select a folder.  
The **New INCA Project** button is activated.
  2. Click on **New INCA Project**.  
The "Create New INCA Project" input window opens. The name [ASCET-Prj] <ascet project name> is assigned.



3. Enter a name for the project and click on **OK**.  
The new project is created in the selected folder and selected.

#### To start a transfer

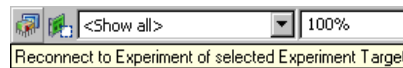
1. Once you have made all settings in the "INCA Project Transfer" window, click on **OK**.  
The transfer of the ASCET project to INCA is executed. First of all the ASAM-MCD-2MC code is generated and the project is refreshed if necessary (i.e. code generated, compiled and linked).  
If you selected an existing project in the "INCA Project" field, it is overwritten.

Once transfer is complete, you can execute the experiment in INCA. For more details on how to do this, refer to the INCA and INCA-EIP documentation.

In addition, the Back-Animation in ASCET provides you with a special experiment environment in which you can calibrate values in the standard manner. The measure system of this experiment environment works in the standard way but is reduced in function in comparison to offline and online experiments in ASCET: oscilloscope, Recorder and Data Logger are not available. These need synchronous measuring which is not given for Back-Animation when experimenting with INCA. Instead, use the relevant instruments of INCA.

**To use Back-Animation**

1. Start the INCA experiment with your project.
2. In the ASCET project editor, make sure that INCA is selected in the "Experiment Target" combo box.
3. Do one of the following:
  - Click the **Reconnect to Experiment of Selected Experiment Target** button

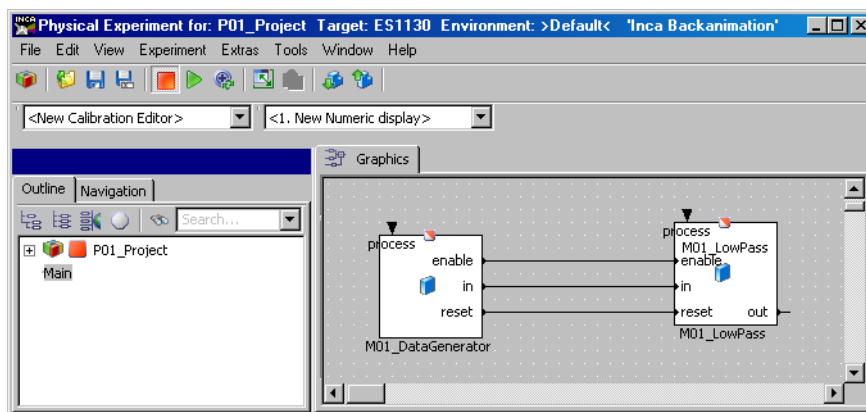


- Select **Build > Reconnect**.

The hardware selection window opens under certain conditions.

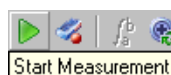
Section "To select hardware" describes the required actions in this window.

The connection to the running INCA experiment is established. The "Physical Experiment ..." window opens. "INCA Backanimation" indicates the special experiment environment.



Unlike with the online and offline experiment, this window only contains the "Graphics" tab.

4. Create the necessary measure windows (see ASCET online help, "Experimentation" section) and set these up.
5. Create the necessary calibration windows (see ASCET online help, "Experimentation" section) and set these up.
6. Do one of the following:
  - Select **Experiment > Start Measurement**.



- Click the **Start Measurement** button to start measurement.

The displays of the measure and calibration windows are updated cyclically.

You can load, save and export environments as described in the section "Experimentation" of the ASCET online help. When you load an environment which contains unavailable elements (e.g. an oscilloscope), these are ignored.

The monitor function (see the ASCET online help) for monitoring numeric and logical variables is available. You can activate the function for individual or all variables of a component. The setting of the monitor function is saved in the environment.



You can navigate between the components of your project (see the ASCET online help).

If your project contains state machines, you can use the animation function for state machines (see the ASCET online help).

You can write data from the experiment into the ASCET model or onto the hard disk; you can also read in data from the hard disk. This is described in the ASCET online help, "Experimentation" section.

#### To end Back-Animation

1. Do one of the following:

- Select **File > Exit**.
- Click the **Exit to Component** button.



The Back-Animation is ended and the experiment environment closed. The INCA experiment, however, continues running.

## 4.3 Experimenting with INTECRIO

If you have installed both ASCET-RP and INTECRIO, you can also experiment with your Rapid-Prototyping project in INTECRIO.

General instructions are given in the ASCET online help, section "INTECRIO Connectivity / ASCET-RP". A specific example can be found in the ASCET Getting Started manual<sup>1</sup>, section "Using INTECRIO Connectivity".

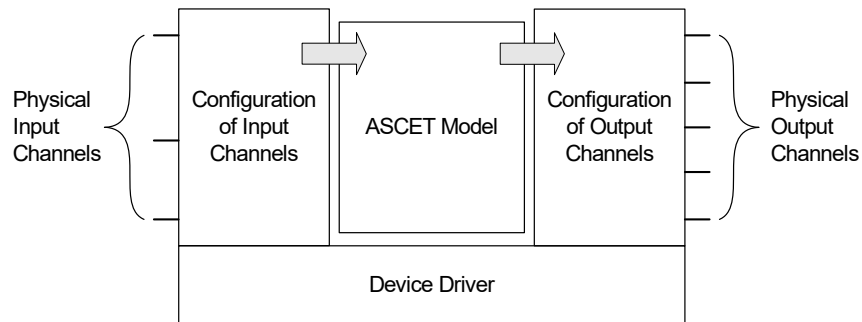
---

1. ASCET V6.4 GettingStarted.pdf in the ETAS\ETAS Manuals folder or on the ASCET installation disk

## 5 Hardware Configuration

ASCET-RP allows easy prototype development for your ASCET model in a real-time environment on the ES900 hardware.

For that purpose, the model has to be embedded in your ES900 environment. On the logical level, the ASCET model has to be connected to the I/O channels.



This connection is performed in two steps. The Hardware Configurator (section 5.1 and online help) is used to describe and configure the I/O boards, i.e. to configure the physical input and output channels. Each of these channels is then mapped onto one ASCET message, which serves as an input or output to the ASCET model.

The configuration is saved to an XML description file (extension \*.hwx) and stored with the ASCET project.

### 5.1 Hardware Configurator

The Hardware Configurator (available since ASCET-RP V6.1) is installed with ASCET-RP.

INTECRIO uses the same editor for hardware configuration; if a compatible INTECRIO version (i.e. V4.3 or higher) is installed on your PC, you can use that tool for hardware configuration.

#### To enable an external Hardware Configurator

1. In the Component Manager (or in the project editor), select **Tools > Options** to open the ASCET options window.
2. In the options window, open the "Hardware\Hardware Configuration" node.
3. To use a separately installed INTECRIO, proceed as follows.
  - i. Activate the **Use separately installed INTECRIO for hardware configuration** option.
  - ii. In the "HW configurator path" field, enter or select the installation directory of the INTECRIO version you want to use.

#### **NOTE**

To create and use projects with an RTPRO-PC hardware configuration, you have to select INTECRIO V4.7.2 or lower.

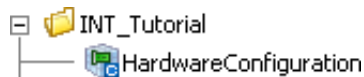
4. In the "HWC working directory" field, enter or select a working directory for the Hardware Configurator.

5. Open the "Hardware\Hardware Connection" node.
6. Activate the **Use ETAS Network Manager (enables 'Select Hardware')** option.

#### To create a hardware configuration for the Hardware Configurator

You have to create an empty hardware configuration in the Component Manager.

1. In the Component Manager, select **Insert > Hardware Configuration**.  
A new hardware configuration appears in the "1 Database" or "1 Workspace" list. Its name is highlighted for in-place editing.
2. If desired, edit the name.
3. Press <ENTER> to confirm the name.  
The hardware configuration is displayed as follows.



#### To create a hardware description file component

If you want to perform an ETK bypass or use a daisy chain, you *have to* create, at first, a hardware description file component in the Component Manager.

CAN databases, LIN description files, and FlexRay configuration files can optionally be added as hardware description file components, too. However this is not mandatory to use them for hardware configuration purposes.

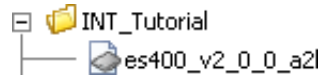
#### **NOTE**

A2L files imported with **File > Import** or the **Import** button *cannot* be used with the Hardware Configurator.

1. In the Component Manager, select **Insert > Hardware Description File**.  
The "Hardware Description File" dialog window opens. The following selections are available:

ASAM-2MC (*.a2l)	for ETK/XETK/XCP bypass
LDF (*.ldf)	LIN configuration
Daisy Chain,	daisy chain configuration
FIBEX (*.xml)	FlexRay configuration
CANdb (*.dbc)	CAN database
2. Select the file you want to import.
3. Click on **OK**.  
A new hardware description file appears in the "1 Database" or "1 Workspace" list. The name of the imported file is used as default name; it is highlighted for in-place editing.
4. If desired, edit the name.

- Press <ENTER> to confirm the name.  
The hardware configuration is displayed as follows.



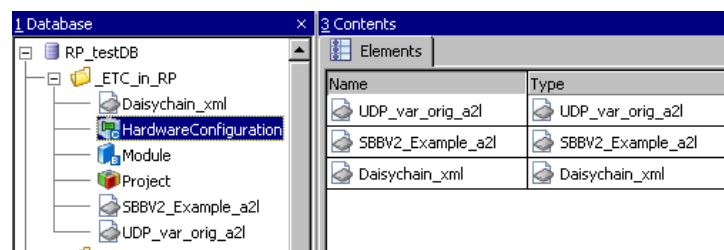
### To add the hardware description to the hardware configuration

After you created the hardware description file component, you have to add it to your hardware configuration.

#### **NOTE**

Only hardware description files added to a hardware configuration are available in the Hardware Configurator for this hardware configuration.

- In the Component Manager, select the hardware configuration you want to use.  
The "3 Contents" field shows the "Elements" tab.
- Right-click in the "Elements" tab and select **Add** from the context menu.  
The "Select Item" window opens.
- In the "1 Database" or "1 Workspace" list, select the hardware description files you want to add.
- Click on **OK**.  
The hardware description file is added to the container and displayed in the "Elements" tab.



For other editing possibilities, see the ASCET online help.

### To set up the project and open the Hardware Configurator

The Hardware Configurator can be opened only via the project editor.

- Create and set up the project as usual.
- In the project editor, use **Insert > Component** to add the hardware configuration to the project.

#### **NOTE**

A project can contain only **one** hardware configuration.


- Do one of the following:
  - Double-click the hardware configuration.
  - Right-click the hardware configuration and select **Open Component** from the context menu.

- Select **Edit > Open Component**.

The Hardware Configurator opens.



#### NOTE

Working with the Hardware Configurator is described in the Hardware Configurator online help, accessible via <F1> or the **Help** menu or the  button.

If OS or compiler settings of the project do not match the hardware configuration (if, e.g., the target PC or an ASCET-SE target is selected), a message of the following kind opens:

```
Current target <target name> does not support specification of hardware configuration. Do you want to edit the project properties?
```

**OK** opens the "Project Properties" dialog window. Select a suitable target and open the Hardware Configurator.

## 5.2 Migration of Existing Projects

---

Since ASCET-RP V6.4.8 does not support the ES1000 hardware anymore, you cannot migrate existing projects that contain configurations created with the legacy HWC editor.

## 5.3 HWC Editor (Legacy)

---



#### NOTE

Since ASCET-RP V6.3, the legacy HWC editor is no longer available.

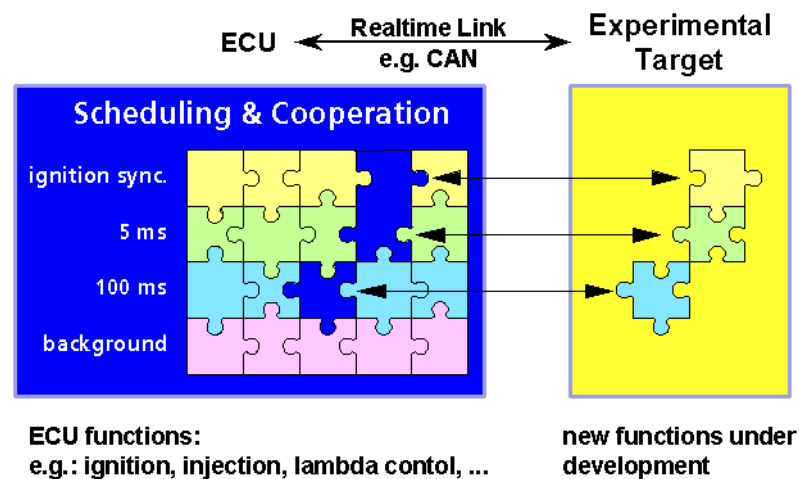
## 6 ETK Bypass

This chapter describes the RTIO Package for ETK bypass. It assumes that the user is familiar with the necessary ASCET techniques; the use and operation of ASCET is not explained.

### 6.1 ETK Bypass: Definition

In an ETK bypass, certain functions of the control unit (ECU) are outsourced to a simulation computer, i.e. the PowerPC processing node of the ETAS experimental system. In doing so, data is transferred from the control unit to the experimental hardware. Based on these data, individual ECU functions are computed on the experimental hardware. Thus, outsourced data can be easily modified and tested. The results are transferred back to the ECU.

Modifications in the control unit software, referred to as "bypass hooks", determine the functions to be outsourced.



The connection to the ECU is implemented via an ETK (emulator test probe). A dual ported RAM is used for the communication between ECU and ETK.

### 6.2 Hardware Configuration of an ETK Bypass

In an ETK bypass application, the outsourced functions are specified in ASCET; code is then generated from the specification that can be executed on the PPC module of the ETAS experimental system. The generated code is downloaded by the host PC to the ETAS experimental system. The ETAS experimental system is connected to the ETK of the control unit via an ETK Bypass device.

Additionally, interface parameters can be set in the control unit program that are required for the configuration of the software interface between the simulation computer and the control unit. Furthermore, new software versions can be downloaded to the control unit. Measurement and calibration tasks in the control unit can be performed while running the bypass.

### 6.3 ASCET Project for the ETK Bypass

Each ASCET RTIO project for ETK bypass needs to consider the following special features:

- A hardware configuration (cf. section 5.1) must be added to the ASCET project.
- The task list in the OS Editor must include an init task (Type: Init / Application Mode: active) and an exit tasks (Type: Init / Application Mode: inactive).

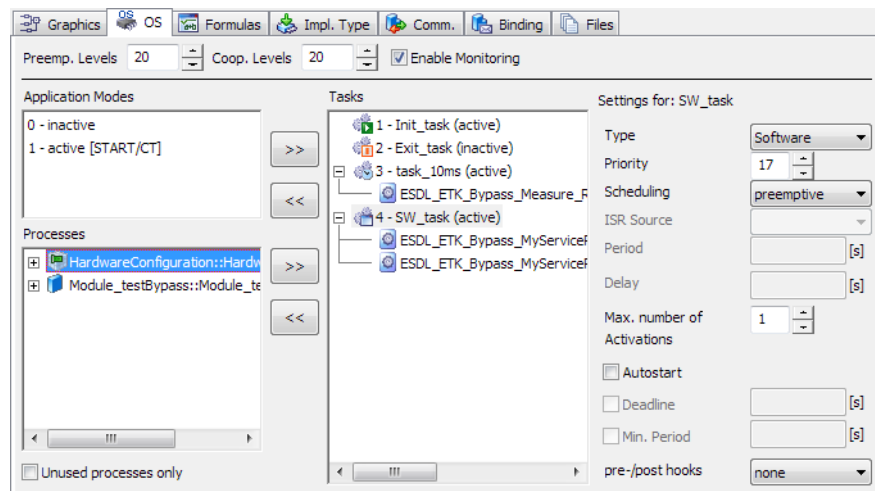


**NOTE**

*Init* and *Exit* are reserved keywords; they must not be used as task names.

- The task list in the OS Editor must contain the necessary tasks that can be assigned to the signal groups which provide the bypass data received from the driver or the bypass data to be sent to the driver. These tasks must have the type *alarm* or *software*.  
The task assignment must be specified manually in the Hardware Configurator.
- Global messages are available for the RTIO communication.

The following illustration shows an example configuration in the "OS" tab:



The following table shows the tasks settings:

Task Name	App. Mode	Trigger Mode	Prio.	Group	Max. No. of Act.	Period
Init_Task	active	Init	-	-	-	-
Exit_Task	inactive	Init	-	-	-	-
task_10ms	active	Alarm	16	cooperative	1	-
SW_task	active	Software	17	preemptive	1	-

## 6.4 How the ETK Bypass Works

To conduct a bypass project, a control unit with bypass hooks, i.e. software modified for the bypass, is required. The bypass hooks enable you to switch between the functions running in the control unit and those running on the simulation computer. The bypass hooks include all information required to transmit the bypass input data to the simulation computer and to process the bypass output data in the control unit.

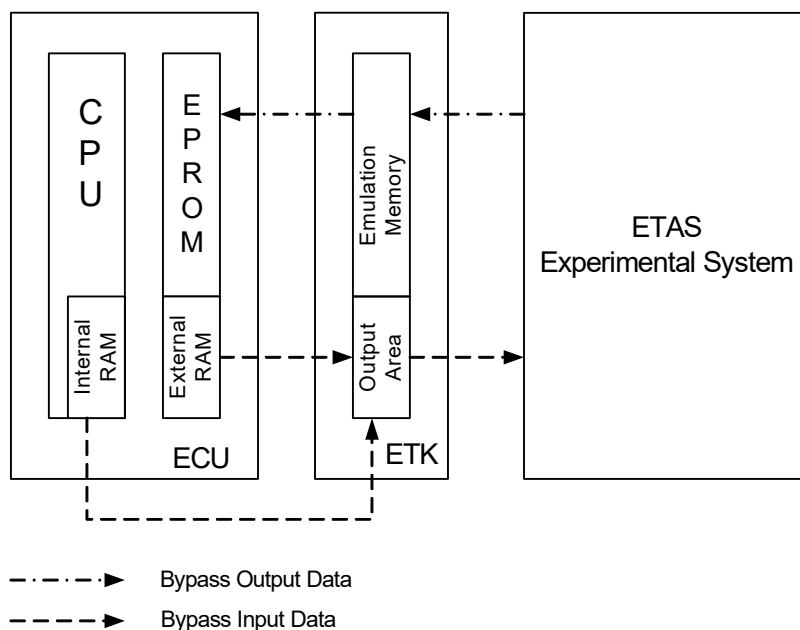
The bypassed functions in the control unit are usually also calculated if the bypass is enabled. But instead of the results of the bypassed functions, the results obtained by the simulation computer, i.e., the bypass output data, are used.

The bypass input data is the data that the experimental target reads from the control unit. This data is used to calculate the outsourced functions. The calculation results are returned to the control unit program as bypass output data.

The ETK memory is divided into an emulation memory area and an output area. The control unit program resides in the emulation area; this area replaces the control unit ROM. The bypass output data is also stored in this area because the control unit cannot read directly from the output area.

The output area is used only for transferring the bypass input data from the control unit to the simulation computer. This area contains a copy of the control unit RAM. Depending on the ETK model and control unit, the CPU-internal RAM of the control unit can also be located in this area.

The following figure illustrates the data flow within an ETK bypass project:



The data flows cyclically within the bypass project, i.e. the bypass input data is first read and then processed by the hardware system. Then the bypass output data calculated in the hardware system is written back to the control unit where it is processed further. Reading the bypass input data is synchronized by a time- or angle-synchronous grid in the control unit. Writing the data back to the control unit is not synchronized.



## 6.5 Data Exchange Between Control Unit and ETAS Experimental System

The DISTAB data exchange method is used for data transfer between the ETAS experimental system and the control unit.

For several years, the DISTAB 12 method has been used for the interaction between the control unit and the ETK. DISTAB 12 supports measured data of up to two bytes in length.

Four-byte integer variables or four-byte or eight-byte real or float variables require the use of DISTAB 13 or DISTAB16. These supports capturing 1, 2, 4 and 8 byte long measured data from the control unit regardless of its format (signed / unsigned integer or float).

For handling DISTAB, the ETK bypass project requires various parameters that can be passed to ASCET in a complete ASAM-MCD-2MC project description file matching the current software version of the control unit. DISTAB is defined in the DISTAB\_CFG section of the TP\_BLOB:

```

/begin DISTAB_CFG
0xC          /* type of display table:          */
              /* 0xC =DISTAB12, 0xD =DISTAB13      */
0x1          /* Data type of display table:          */
              /* 1=byte 2=word (ECU Data Mode)        */
              /* additional code table for          */
              /* distab13 depending on bus        */
              /* width/bus access (see distab13   */
              /* spec. for more information)      */
MSB_LAST     /* Byte Order: MSB_FIRST/MSB_LAST      */
0x383000     /* Trigger Segment Address              */
0x0          /* Trigger Configuration                */
TRG_MOD 0xB7 /* Dyn. length for TRG_MOD              */
              /* (special code)                       */
/end DISTAB_CFG

```

### NOTE

The comments are usually not included in the ASAM-MCD-2MC file. They are added here only for clarity.

ASAM-MCD-2MC is an established standard in the automotive industry for describing a control unit project (calibration parameters, measured variables, conversion rules, addresses, etc.). The ASAM-MCD-2MC file needs to be created for each new program version and should therefore be the result of the software development process. Detailed knowledge of the DISTAB method is not required for the ETK process because all necessary settings are part of the bypass hooks. Nevertheless, we will briefly look at the working principle to facilitate examining the ETK memory for diagnostic purposes.

The communication mechanisms for the data transfer between the control unit and the simulation computer are referred to as "bypass channels".

For the number of channels, see the online help Hardware Configurator.

ASCET-RP supports several AML versions (see the following sections for more details). ASAM-MCD-2MC files of all supported versions may contain XETK\_AML V1.0 blocks; the content of such a block is of no consequence for working with ASCET-RP.

### Bypass Communication (AML V1.1)

The bypass communication is explained here using a bypass channel named A. The process is the same for both channels. Tab. 6-1 contains the names and description of the parameters used in the ASAM-MCD-2MC file (\*.a21, AML V1.1) to define both channels.



#### NOTE

The parameter names are not part of the ASAM-MCD-2MC standard; they are not necessarily included in the \*.a21 file. They are added manually to the examples of this manual to provide clarity.

Parameter	Description
BYPASS_S	Start address of pointer list for bypass input data (bypass channel A)
BYPASS_X	Start address of pointer list for bypass input data (bypass channel B)
CHNL_S	Start address of data buffer for bypass input data (bypass channel A)
CHNL_X	Start address of data buffer for bypass input data (bypass channel B)
CHNL_T	Start address of data buffer for bypass output data (bypass channel A)
CHNL_Y	Start address of data buffer for bypass output data (bypass channel B)
TRGID_S	Trigger ID address for bypass input data (bypass channel A)
TRGID_X	Trigger ID address for bypass input data (bypass channel B)
BPMAX_S	Size of data buffer for bypass input data (bypass channel A)
BPMAX_X	Size of data buffer for bypass input data (bypass channel B)
BPMAX_T	Size of data buffer for bypass output data (bypass channel A)
BPMAX_Y	Size of data buffer for bypass output data (bypass channel B)
TRGSEG_A	Trigger address for bypass channel A
TRGSEG_B	Trigger address for bypass channel B

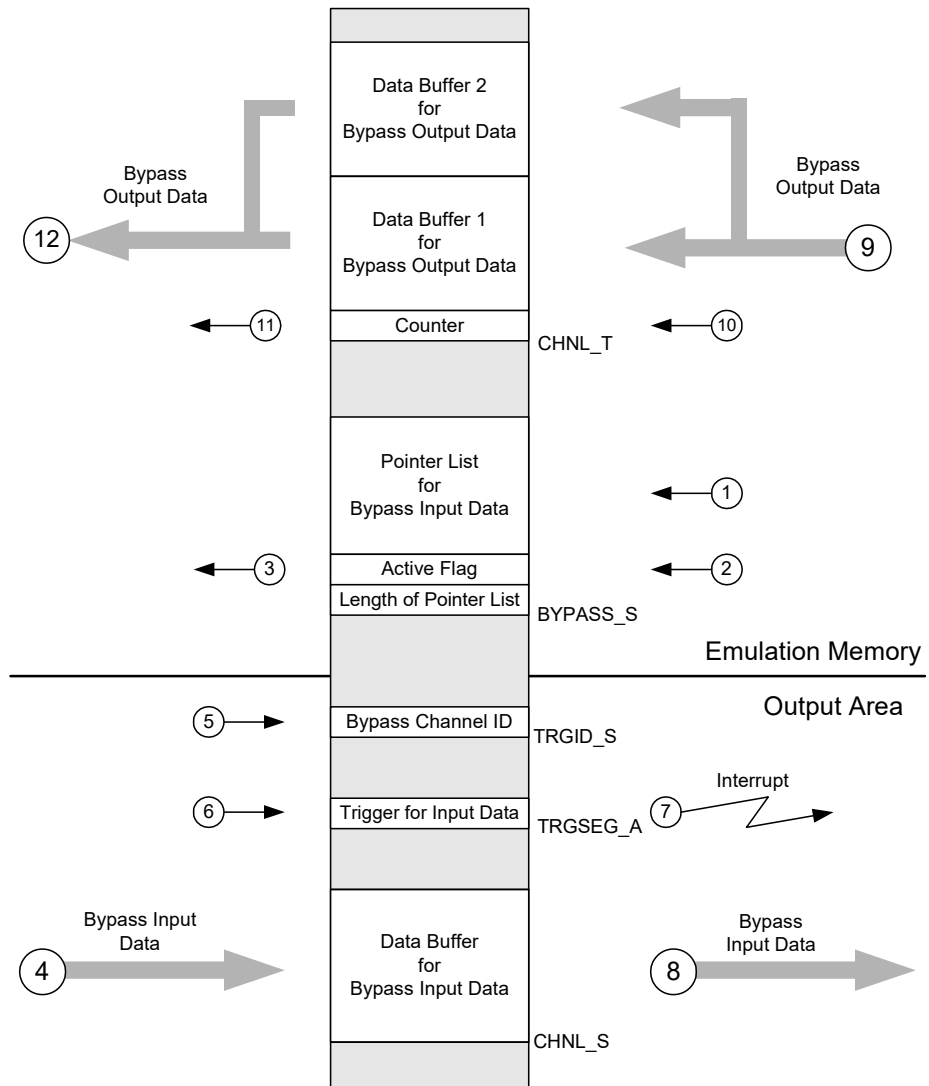
**Tab. 6-1** Bypass communication parameters (AML V1.1)

The parameters for channel A are provided in a QP\_BLOB in the IF\_DATA ETK section of the ASAM-MCD-2MC file.

```
/begin IF_DATA ETK
  /begin SOURCE "BYPASS A"
    0
    0
  /begin
```

```
QP_BLOB
4    /* Acquisition raster;                */
    /* 1=A (typ. angle synchronous)        */
    /* 2=B (typ. time synch. 10ms)        */
    /* 3=C (typ. time synch. 100ms)       */
    /* 4=S/T angle synch. (bypass only)   */
    /* 5=X/Y time synch. (bypass only)    */
100   /* BPMAX_S                          */
0x81025E /* BYPASS_S                      */
0x3801E0 /* CHNL_S                        */
0x38302E /* TRGID_S                      */
2     /* trigger repetition rate          */
      /*(worst case)                     */
100   /* BPMAX_T                          */
0x8103F2 /* CHNL_T                        */
/end QP_BLOB
/end SOURCE
...
/end IF_DATA
```

Fig. 6-1 schematically shows the process of the bypass cycle with the DISTAB data exchange method using the example of bypass channel A. The numbers represent the individual steps of the bypass cycle:



**Fig. 6-1** Schematic process of a bypass cycle (DISTAB method)

- A When starting the bypass experiment on the simulation computer, the pointer list for the bypass input data is filled.

DISTAB12	DISTAB13 and DISTAB16
For each byte to be sent to the simulation computer, there is a pointer pointing to the address of the byte in the control unit memory. With the DISTAB 12 method, data is always transmitted in bytes; therefore, two pointers are required to send a word. After filling the pointer list, its length is written into the first byte of the list.	Bytes 4 to 7 of the pointer list contain the number of 8-byte, 4-byte, 2-byte, and 1-byte signals. The following bytes contain the signal addresses, beginning with the first 8-byte signal, and ending with the last 1-byte signal. Each address covers 4 bytes; the byte order of the addresses is determined by the <code>Byte Order</code> parameter (cf. page 41).

The bypass offsets (cf. step 11, page 46) are transferred in this step, too.

- B After the pointer list has been filled, the active-flag is set to 1. This initiates communication between the control unit and the simulation computer. Steps 1 and 2 are executed only once, at the beginning of a bypass experiment.

DISTAB12	DISTAB13 and DISTAB16
active-flag: the last bit of the second byte of the pointer list	active-flag: bit 0 of the first byte (byte 0) of the pointer list

- C The control unit continuously checks the active-flag byte of the pointer list. Once the last bit is set to 1, the bypass is activated and the data transmission begins.
- D The bypass input data is written by the control unit into the appropriate data buffer, `CHNL_s`.

DISTAB12	DISTAB13 and DISTAB16
This is done byte by byte; for each entry in the pointer list, the byte is written to the corresponding address of the data buffer. The transmission finishes when the number of bytes has been sent that is specified by the contents of the first byte in the list.	The data is written signal-wise, in the order provided by the pointer list. First, the bytes of the first 8-byte signal are written; the <code>Byte Order</code> parameter again defines which bit is written first. The other signals follow.

- E The control unit writes the ID of the current bypass channel (i.e., 4 for bypass channel A) to the address `TRGID_s`. This is necessary because only two addresses are available for the channel IDs, while there are a total of five channels. Two channels are used as bypass channels; the other three are available for calibration purposes. The ES1232 has up to 32 channels, 16 bypass channels and 16 measurement channels.
- F The trigger address `TRGSEG_A` is loaded by the control unit with a random value.
- G Writing to the trigger address triggers an interrupt in the simulation computer.

- H By reading the channel ID from the address `TRGID_S`, the simulation computer determines which channel was initialized and then reads the corresponding data buffer. From the information in the ASAM-MCD-2MC file, such as the conversion formula, the simulation computer is able to convert the raw data from the control unit into physical model variables that can then be processed by the ASCET model.
- I The simulation computer computes the functions outsourced to the bypass model on the basis of the bypass input data. The results are then written back to the appropriate data buffer as bypass output data. For each bypass channel there are two data buffers that are alternately written to. The simulation computer first internally (i.e. in the simulation computer) increments the counter that is later written to the `CHNL_T` address. The counter is incremented when the interrupt in step 7 occurs. Depending on the contents of the counter, one of the data buffers is used for writing. If the count is odd, the first data buffer is used; if it is even, the second buffer is used.
- J After writing to the data buffer, the internal count is written to the `CHNL_T` address.
- K The value of the counter is read by the control unit. The control unit decides on the basis of this value which data buffer is used for reading. If the value is odd, the first data buffer is read from; if it is even, the second buffer is read from. This ensures that the data consistency of the bypass output data is maintained.

The layout of these two buffers depends on the selected bypass output signal. Thus, the ECU must know where in the buffer the signal value actually is located. For that purpose, the signal position relative to the buffer start address—the *bypass offset*—is determined. For safety reasons, no pointer list exists for the bypass signals; instead, the ECU software contains a special bypass offset parameter for each signal. The bypass offsets are filled into the bypass offset parameters in step 1 (cf. page 45). These parameters can be directly accessed because they are located in an ECU application data area allocated in the ETK RAM.

- L The bypass output data is read by the control unit. In general, a safety mechanism is implemented in the control unit that determines the behavior of the control unit in case of a failure in the bypass communication. For details, please consult your control unit programmer.

Two data buffers are required for writing the bypass output data back, because this process is not synchronized. The results of the calculations in the simulation computer are written back as soon as the calculations are finished. The control unit may, therefore, request the results before or while the data is being written back. For this reason, a copy of the data is always held in a data buffer until the new data has been fully written back to ensure consistency of the bypass output data.

### Other AML versions

A `QP_BLOB` of version AML V1.1.1 does not differ from the above AML V1.1 example (see page 42).

In AML V1.2, several changes have been made. Some parameters have been removed, and new parameters have been added. The (optional) names have changed, too. The `QP_BLOB` reads as follows:

```
/begin IF_DATA ETK
```

```

/begin SOURCE "BYPASS A4"
0
0
/begin
  QP_BLOB
    0x0100 /* version 1.0 */
    15 /* hardware trigger */
    INDIRECT /* indirect/direct triggering */
    5 /* raster number/priority */
    /* (32..1) */
    BYPASS /* raster type */
    /* (BYPASS/MEASUREMENT) */
    0x38302E /* trigger id/flag address for */
    /* indirect triggering */
    100 /* Max. length of display table */
    /* in bytes */
    0x81025E /* address of the display table */
    /* in the memory */
    0x3801E0 /* address, where the ECU */
    /* writes the display values */
    100 /* Max. size of bypass receive */
    /* table */
    0x0 /* StartAddress of the Address */
    /* table for bypass output */
    0x8103F2 /* Output address of the */
    /* bypass table */
    2 /* worst case raster timing */
  /end QP_BLOB
/end SOURCE
...
/end IF_DATA

```

AML V1.3, AML V1.4, AML V1.6 and AML V1.7 can be used, too. The differences between those versions and AML V1.2 are of no consequence for working with ASCET-RP.



#### NOTE

AML V1.1.x describes ETK data for 3+2 measurement rasters. AML V1.2 and higher describe ETK data for multirasters.

## 6.6 Initially Required Information and Data

Before you can begin creating an ETK bypass project, you need some information and data about your control unit. Normally you obtain this from the control unit programmer who implemented the bypass hooks. The items listed in this section may be used as a checklist to ensure a smooth information transfer.

## Control Unit Program

The bypass hooks control unit program must be loaded in the control unit.

## ASAM-MCD-2MC File

In addition to other tasks, the ASAM-MCD-2MC file describes the data structures of the control unit. The ETK bypass package uses the ASAM-MCD-2MC file to access the input and output variables of the bypass project. Therefore, it is crucial to have the proper ASAM-MCD-2MC file matching the current software version of the control unit. The supported AML version depends on the hardware. The descriptions of the various boards include the necessary information.

## Data Format of the Control Unit Processor

Different processor families use different word data storage formats that are known in general as *Big Endian* or *Little Endian*. When setting up the ETK bypass, it must be known whether the processor of the control unit uses the *Little Endian* or *Big Endian* storage format, or whether the word data storage format is specified in the ASAM-MCD-2MC file and can be read by the Hardware Configurator.

In the *Big Endian* format (e.g., used by Motorola processors), the first byte represents the *most significant byte*. In the *Little Endian* format (e.g., used by Intel processors), the first byte represents the *least significant byte*.

## Base Addresses

The start addresses of the transfer channels and the trigger addresses must be specified in the ETK bypass package, or the start addresses are specified in the ASAM-MCD-2MC file and can be read by the HWC Editor. The significance of the base addresses is explained in section 6.5 "Data Exchange Between Control Unit and ETAS Experimental System" on page 41.

## Parameters for Activating the Bypass

The bypass must be enabled in the control unit program. If the bypass is enabled, the appropriate results in the simulation computer are used instead of the function results in the control unit program. Each bypass function is activated by setting a parameter with INCA.

## Bypass Output Variables

The ETK bypass project can only modify certain variables in the control unit. These so-called bypass output variables are specified by the control unit programmer in the bypass hooks. The bypass output variables have to be known when creating the project, or the relevant information is stored in the ASAM-MCD-2MC file from where it can be read by the HWC Editor.

## Bit Masks for the Transformation in the NEAR Region

The DISTAB 12 method supports only 16-bit memory addresses for transfer between the control unit and the bypass computer. For control units using memory addresses larger than 16 bits, these have to be transformed to 16 bits



by means of a data page pointer. The `longAdrANDMask` and `longAdrORMask` bit masks are used for this purpose. The bit masks need to be specified in the Hardware Configurator.

### **What to do in case of an Error / Safety Mechanism**

In general, a safety mechanism is implemented in the control unit that defines the behavior of the control unit in case of a failure in the bypass communication. Sometimes the results of the control unit functions are used; at other times the control unit is switched into *Reset* or *Emergency* mode (depending on the control unit). It is crucial to know the behavior of the control unit if an error occurs, particularly for bypass experiments in the vehicle.

## 7 Bypass Concept

### 7.1 ETK Bypass Concept Description

With an ETK equipped ECU, the ECU code must be prepared to set up data structures and communication with the ETK to enable communication between the rapid prototyping system used for the ETK bypass and the ECU itself.

Independent of the ECU implementation of these drivers, some safety issues common to the concept of the bypass must be considered.

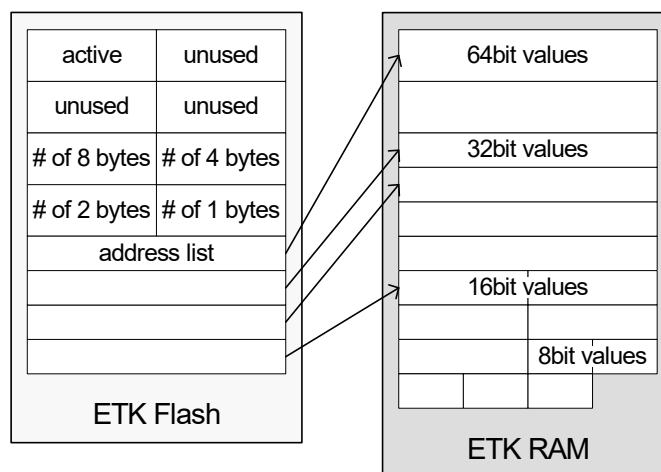
### 7.2 Bypass Input

Like for measurement, the Distab13 (and Distab 12 for hook-based bypass) mechanism is used to provide ECU variables as inputs for the bypass.

The DISplay TABLE for the Distab13 contains a sorted list of addresses of variables in the ETK Flash. 8, 4, 2 and 1 byte values are supported. The addresses are ordered corresponding to the size of the values they point to. The ECU driver parses the table and writes the contents of the addresses to a table of return values in the ETK RAM. This table is ordered in the same manner as the address list: first, all 8 byte values, then the 4 byte values, etc.

With this approach, INCA and ASCET-RP are given access to values of the internal memory of the microcontroller. Also, collecting the data in a table allows using block modes for transferring the data to the PC.

Fig. 7-1 gives an overview on the data layout for Distab13.



**Fig. 7-1** Distab13 data structure

For each bypass raster that contains hooks for the hook-based bypass, an instance of the Distab is created and a Distab process is called. For the service-based bypass, one instance of the Distab data structure exists for each trigger where a service is provided and configured to read ECU values as input for the bypass calculation.

For the hook-based bypass, the number and name of Distabs implemented in the ECU code, and their size, e.g. the number of bytes per channel, is set by the ECU software setup and documented in the A2L description.

For service-based bypass, all tables are allocated dynamically in a given working memory.

## 7.3 Hook-Based Bypass

### 7.3.1 Classical

For the classical *hook-based bypass*, the input values of the bypass are gathered with the same Distab13 mechanism as the measurements. After the bypass input data is written to the ETK RAM, the bypass calculation is triggered. In addition, a channel for writing back bypass results to the ETK where they can be retrieved by the ECU is introduced.

For each bypass input channel, an output channel is provided. The size of these channels, as well as their names, also have to be documented in the A2L description. The prototyping tool (ASCET or ASCET-RP) can define the number of variables written back to the ECU, depending on the bypass experiment setup. Each variable written to by the bypass must be prepared in the ECU software by applying a hook to prevent the ECU from writing to this value if the bypass is active. The hook code is specific to the ECU and the variable it is applied to. No service is provided within this sample implementation for this task. The values prepared have to be documented in the A2L file by an IF\_DATA ASAP1B\_BYPASS description.

The following figure describes the hook-based bypass principle. The hook indicates the possibility to toggle between the results of the original function (Fn) and the bypass function (Fn\*).

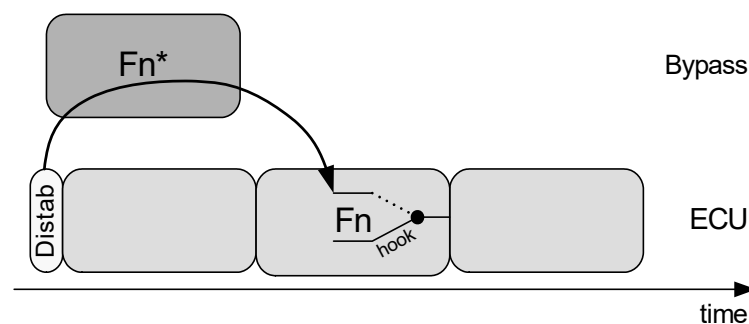
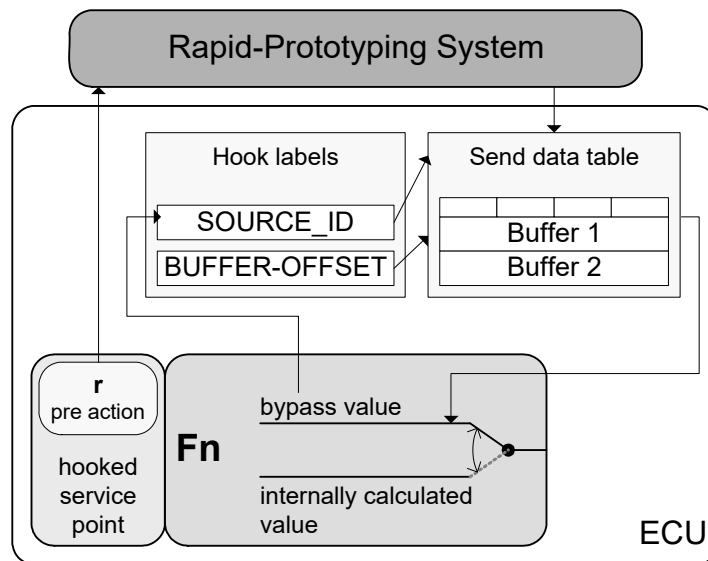


Fig. 7-2 Hook-Based Bypass: Principle

### 7.3.2 With Distab17

In connection with Distab17, the concept of service point configuration extends to the hook-based bypass. In this case, the hook-based bypass (HBB) is integrated with the service-based bypass (SBB): For hooked service points, the new signal values calculated in the rapid prototyping system are transferred to the ECU software by dedicated hook codes in the ECU functions instead of generically in a service point write action. Hooked service points are supported as of Distab17.

- In the following illustration, **Fn** denotes the original function that runs on the ECU.



**Fig. 7-3** Bypass with hooked service points

- Prior to the execution of the original function that runs on the ECU, hooked service points can be used to receive data from the ECU to the rapid prototyping system via a read or receive pre-action. The associated hook codes are usually implemented at the end of the original function. The hooks receive their source (i.e. SOURCE\_ID) and offset (i.e. BUFFER\_OFFSET) information from the associated hook labels.

During the execution of the original function in the ECU, the rapid prototyping system writes data to a double-buffered send data table that can be accessed by these hooks in the original function. The two buffers are used alternately. Either the resulting bypass value or the value calculated internally by the original function is used.

## 7.4 Service-Based Bypass



### NOTE

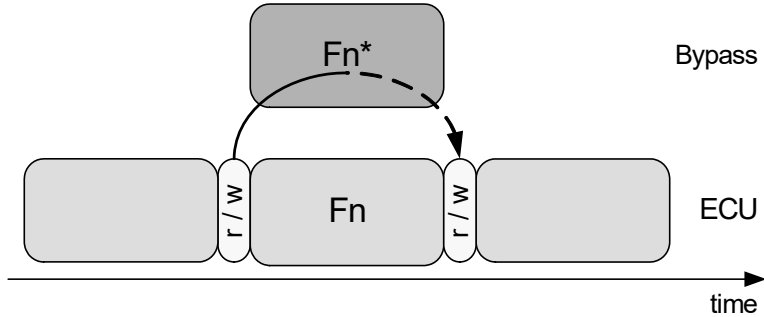
Service-based bypass on an ETK with 8 Mbit/s is not supported.

For the *service-based bypass*, both the input values and the output values of the bypass function are transmitted with the same Distab13 mechanism. After the bypass input data is written to the ETK RAM, the bypass calculation is triggered. In addition, a channel for writing back bypass results to the ETK where they can be retrieved by the ECU is introduced. Here, each bypassed ECU process uses and calls its own Distab.

This service also contains an inverted Distab mechanism to write back bypass outputs to the ECU. The ECU does not need to apply hooks to the variables written to, since the service simply overwrites the values with the bypass outputs. ASCET-RP sets up a Distab-like sorted address table with the addresses of the ECU variables to be written to, and writes the corresponding values in a

table of the ETK Flash. The part of the ECU service that writes the bypass outputs to the ECU parses the address table and gets the corresponding values from the data table and writes the values to the ECU addresses.

The following figure describes the service-based bypass principle.



**Fig. 7-4** Service-Based Bypass: Principle (The dashed line indicates that bypass data can be written back at a later time as well.)

The current version of ASCET-RP supports several versions of service-based bypass (SBB). The following table lists the supported SBB versions for each target (+: supported, -: not supported).

	SBB V2.0	SBB V2.1	SBB V3.*
ES 910.2 / supported ETKs	+	+	-
ES 910.3 / supported ETKs	+	+	+

**NOTE**

Service-based bypass on an ETK with 8 Mbit/s is not supported.

**Tab. 7-1** Supported SBB versions

SBB version	supported AML versions	supported Distab types
V2.0 + V3.0	ETK AML 1.2.0 – 1.7.0	Distab13 - Distab16
	XETK AML 1.0.0 – 2.0.0	
	SBB AML ≥ 2.0.0	
V2.1	ETK AML not supported	Distab17
	XETK AML ≥ 2.0.0	
	SBB AML ≥ 3.1.1	
V3.1	ETK AML not supported	Distab17
	XETK AML ≥ 2.0.0	
	SBB AML ≥ 2.0.0	

**Tab. 7-2** SBB versions and AML versions

## 7.5 Safety Considerations

---

With calculating a bypass function on a rapid prototyping system and feeding data back into the ECU, the same care needs to be taken for development and use of bypass software as for ECU software. The bypass output may directly or indirectly influence the output channels of the ECU. The same applies, of course, if the rapid prototyping system uses own output channels.

Thus, it is highly recommended that the bypass functions include range checks and validations algorithms for the bypass outputs.

### 7.5.1 Bypass Input Data

To perform proper calculations, the bypass obviously needs consistent and valid input data. The ECU software must ensure this and prevent activation of the bypass if the ECU software detects incorrect or invalid inputs. This also includes ECU states like initialization and afterrun, or error modes the ECU might run in. In other words, activation of and data transfer to the bypass must be covered by the safety mechanisms of the ECU.

### 7.5.2 Bypass Calculation

The ECU software must be aware whether the bypass is active and must provide measures to react on bypass failures, for example missing calculations or unexpected shut off of the bypass system.

Failsafe measures might be using the alternative output values of the bypassed ECU functions, using constant fallback values, or even resetting the ECU, depending on the bypassed ECU function. This is entirely under responsibility of the ECU provider who integrates the bypass drivers.

Some implementations of the ECU bypass drivers, as for the service-based bypass, allow the deactivation of the bypassed ECU function by the bypass user. In this case, the results of the bypassed ECU function obviously cannot be used as fallback. This must be considered when setting up a safety strategy.

### 7.5.3 Bypass Output Data

The ECU provider must guarantee that any value sent back by the bypass system leads to a predictable behavior of the ECU – the bypass output values must undergo the same range check and validation as the values calculated within the bypass.

As said before, the implementation of the ECU drivers must, in any case, ensure that bypass failures can be detected by the ECU and valid and safe fallback values are available at any time.

### 7.5.4 Message Copies

If the ECU software contains message copies, the bypass must be aware of them.

The usual implementations of the *hook-based bypass* are an exception to that rule. Here, the hooks (and thus the messages written to) are known before compilation, so that the hook code can take care of and use message copies if needed—individual code and addresses are used at each hook.

With the *service-based bypass*, users cannot choose variables and decide which message to write to before they set up the bypass experiment in ASCET-RP. Thus, the services in the ECU are generic code and do not know about specific message copies.

This requires two steps:

- A The ECU software provider must provide this message copy information in the A2L file (usually in encrypted and password-protected form).
- B If the message copy information is encrypted, the user of the bypass system receives a password from the ECU software provider. He must enter this password to decrypt the information and use it for system configuration.

If one of these two steps is missing (usually a wrong password is entered), the bypass system has no knowledge of message copies and reads from / writes to the original variable address, as it is declared in the `MEASUREMENT` declaration of the A2L file. For receive variables, this yields old data values. For send variables, the data value written by the bypass model may be overwritten by other parts of the ECU software. Both cases may cause bypass malfunction!

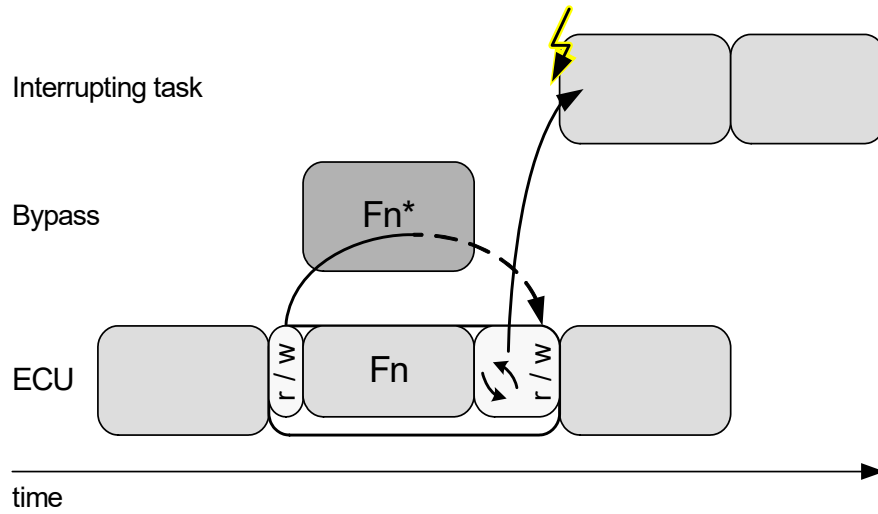
Another principal problem can arise with ECU software that contains message copies. The original code to create the message copies for an ECU task was based on the given message usage and generated appropriate code. By writing variable values into the ECU via bypass methods, the data flow changes, and a new or different message copy may become necessary. This can result in wrong variable values in the ECU software even at locations which are not directly related to the bypassed functions. Whether writing to a certain variable at a certain location (e.g. service point) may be dangerous or not, can be answered only by the ECU software supplier. This information cannot be declared in the A2L file.

## 7.6 Service-Based Bypass Specifics

---

Unlike the hook-based bypass where either the ECU or the bypass writes to the bypassed variable, the service-based bypass has an inherent possibility of data inconsistency, since both the ECU and the bypass write to the same value consecutively. Due to ECU real-time constraints, interrupts cannot be disabled to protect the sequence of writing the results of the ECU function and then writing the variable values of the bypass.

So, if a preemptive task of higher priority interrupts the tasks containing the bypass service, it will see the ECU value instead of the bypass value. The probability of this inconsistency depends on the distance between the two writes.



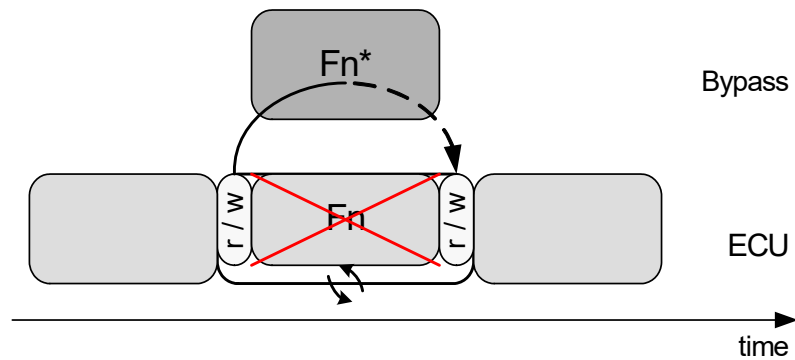
**Fig. 7-5** Possible data inconsistency (the small arrows (↻) indicate a waiting time for bypass results)

A countermeasure to this problem is disabling the ECU function, so that only the bypass is writing to the bypassed ECU values (see below). Be aware that disabling the ECU function implies other safety constraints in case of bypass failures as discussed below.

### 7.6.1 Service Processes for the SBB Implemented as Service Functions

For SBB, the way of ECU implementation is to replace the ECU process to be bypassed by a container process that contains service function calls before and after it calls the original ECU process. This allows to call the ECU process under certain conditions only, e.g. to deactivate it in case of possible data consistency problems. To simulate the timing behavior of the disabled ECU process, a delay time can be configured.

Therefore, the suggested ECU implementation looks like this:



**Fig. 7-6** Suggested SBB implementation



For setting up the bypass experiment in the Hardware Configurator and implementing the service point in the ECU software as container process, a service point is defined as


- A receiving data from the ECU (pre read action),
- B waiting for data to be sent (a timeout is defined),
- C sending data to the ECU (pre write action),
- D conditionally executing the original ECU process,
- E receiving data from the ECU (post read action),
- F waiting for data to be sent (a timeout is defined),
- G sending data to the ECU (post write action).

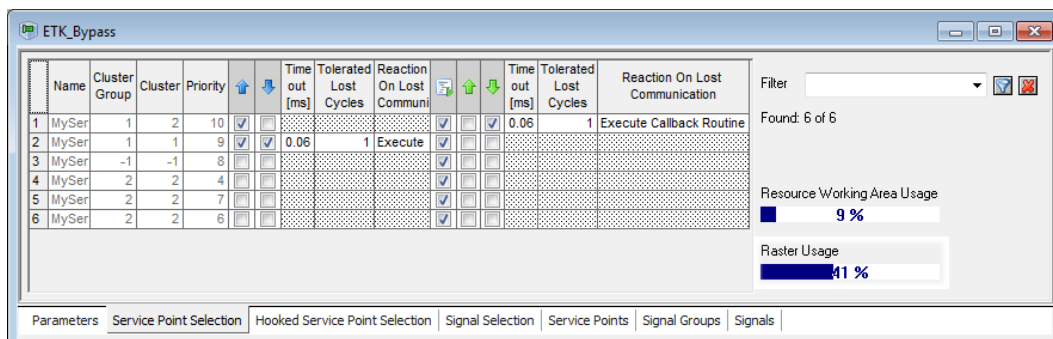
Each pre and post action can be freely configured or activated / deactivated.

### 7.6.2 Controlling the ECU Behavior from ASCET-RP

Upon setting up the ASCET-RP experiment, an initial setting of the control variables can be done in ASCET-RP. These values are written to the ETK on experiment initialization.

The ECU function can be controlled by the user in several ways (if the ECU drivers also provide this functionality):

- The ECU function can be deactivated in the service point editor of the Hardware Configurator (  column, see Fig. 7-7),
- The detection of a bypass error can be defined
- The bypass error behavior of the ECU code can be influenced



**Fig. 7-7** Controlling ECU function execution from the Hardware Configurator  
(The columns "Cluster Group" and "Cluster" are only available for SBB V3.  
The "Raster Usage" display is only available for SBB V2.)

### 7.6.3 Summary

The implementation and integration of the ECU drivers must take care of the following:

- As the service-based bypass only overwrites ECU values with bypass values without preventing the ECU from writing these values, there is a possibility of data inconsistency which can lead to unpredictable behavior of the ECU.

- The Hardware Configurator provides a configuration variable to set the maximal number of tolerated lost cycles, e.g. how many ECU calculation cycles without receiving bypass values are tolerated before this is regarded as an error. But it is up to the provider of the ECU software to make sure missing bypass output values are detected.
- The Hardware Configurator provides a configuration variable to set a specific error behavior (if also supported by the ECU implementation) But it is up to the provider of the ECU software to make sure bypass failures or bypass deactivation can be detected by the ECU software! The configuration setting in the ASCET-RP GUI can then be used to choose between different provided error behaviors in the ECU.
- The Hardware Configurator allows disabling the bypassed ECU process. In this case, no ECU values can be used as fallback values for bypass failures! It is up to the provider of the ECU software to make sure sensible data is written to the variables if both the ECU process and the bypass is disabled.

## 8 Troubleshooting General Problems

---

This chapter gives some information of what you can do when problems arise that are not specific to an individual software or hardware product.

### 8.1 Network Adapter cannot be selected via Network Manager

---

#### *Cause: APIPA is disabled*

The alternative mechanism for IP addressing (APIPA) is usually enabled on all Windows systems. Network security policies, however, may request the APIPA mechanism to be disabled. In this case, you cannot use a network adapter which is configured for DHCP to access ETAS hardware. The ETAS Network Manager displays a warning message.

The APIPA mechanism can be enabled by editing the Windows registry. This is permitted only to users who have administrator privileges. It should be done only in coordination with your network administrator.

#### To enable the APIPA mechanism

1. Open the Registry Editor:
  - i. Press <WINDOWS LOGO> + <R>.
  - ii. Enter `regedit` and click on **OK**.  
The registry editor is displayed.
2. Open the folder `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\`
3. Select **Edit** > **Find** to search for the key `IPAutoconfigurationEnabled`.

If you cannot find any instances of the registry key mentioned, the APIPA mechanism has not been disabled on your system, i.e. there is no need to enable it. Otherwise proceed with the following steps.

4. Set the value of the key `IPAutoconfigurationEnabled` to 1 to enable the APIPA mechanism.  
You may find several instances of this key in the Windows registry which either apply to the TCP/IP service in general or to a specific network adapter. You only need to change the value for the corresponding network adapter.
5. Close the registry editor.
6. Restart your workstation in order to make your changes take effect.

### 8.2 Search for Ethernet Hardware fails

---

There is a number of different causes which might lead to connection problems, most of them being based on inappropriate Windows or hardware settings. Usually, these can easily be modified, once they have been identified.

The following list of causes shall help you in finding the root cause of the problem and fixing it.

***Cause: The versions of the Hardware and the ETAS Software are not compatible***

If you are using ETAS hardware with ETAS software, you can use the ETAS HSP Update Tool to check the firmware version of your hardware:

- Make sure you use the ETAS HSP Update Tool with the latest HSP (Hardware Service Pack) version.
- Also use the HSP Update Tool to check whether the hardware is compatible with the MC software used.
- Make sure any additional drivers for that hardware are installed correctly.

You can get the required HSP from the ETAS internet pages at [www.etas.com](http://www.etas.com).

If you still cannot find the hardware using the HSP Update Tool, check whether the hardware offers a Web interface and whether you can find using this interface. Otherwise check whether one of the following causes and solutions might apply.

***Cause: Personal Firewall blocks Communication***

Personal firewalls may interfere with access to ETAS Ethernet hardware. The automatic search for hardware typically cannot find any Ethernet hardware at all, although the configuration parameters are correct.

Certain actions in ETAS products may lead to some trouble if the firewall is not properly parameterized, e.g. upon opening an RP experiment in the ETAS experiment environment or searching for hardware from within INCA or HSP.

If a firewall is blocking communication to ETAS hardware, you must either disable the firewall software while working with ETAS software, or the firewall must be configured to give the following permissions.

**Permissions given through the firewall block ETAS hardware:**

- Outgoing limited IP broadcasts via UDP (destination address 255.255.255.255) for destination ports 17099 or 18001
- Incoming limited IP broadcasts via UDP (destination IP 255.255.255.255, originating from source IP 0.0.0.0) for destination port 18001
- Directed IP broadcasts via UDP to the network configured for the ETAS application, destination ports 17099 or 18001
- Outgoing IP unicasts via UDP to any IP in network configured for the ETAS application, destination ports 17099 through 18020
- Incoming IP unicasts via UDP originating from any IP in the network configured for the ETAS application, source ports 17099 through 18020, destination ports 17099 through 18020

- Outgoing TCP/IP connections to the network configured for the ETAS application, destination ports 18001 through 18020

 **NOTE**

The ports that have to be used in concrete use cases depend on the hardware you use. For more precise information on the port numbers that can be used please refer to your hardware documentation.

#### Permissions given through the firewall block XCP on Ethernet:

- Outgoing IP multicasts for XCP Slave Detection via UDP to any IP in network, destination IP 239.255.0.0, port 5556.
- Incoming IP multicasts for XCP Slave Detection via UDP from any IP in network, destination IP 239.255.37.45, port 3745.

Contact your IT responsible to clarify whether the required permissions are, or can be, given by the firewall.

#### **NOTICE**

##### **Changes to your firewall configuration can make your system insecure.**

Always consult your IT responsible and/or check the IT security policies of your company before changing your firewall configuration and reconnecting the computer to the network.

#### ***Cause: Client Software for Remote Access blocks Communication***

PCs or notebooks which are used outside the ETAS hardware network sometimes use a client software for remote access which might block communication to the ETAS hardware. This can have the following causes:

- A firewall which is blocking Ethernet messages is being used (see "Cause: Personal Firewall blocks Communication" on page 60).
- By mistake, the VPN client software used for tunneling filters messages. As an example, Cisco VPN clients with versions before V4.0.x in some cases erroneously filtered certain UDP broadcasts.

If this might be the case, please update the software of your VPN client.

#### ***Cause: ETAS Hardware hangs***

Occasionally the ETAS hardware might hang. In this case switch the hardware off, then switch it on again to re-initialize it.

#### ***Cause: ETAS Hardware went into Sleep Mode***

In order to save power, some ETAS devices will go to sleep mode if they do not see that they are connected to another device/computer.

To solve that, connect your Ethernet cable from your computer to the "HOST"/"Sync In" port on the device. After the device turns on, connect to the device using the web interface and change the settings so that the device stays always on. Consult the device's manual for details on how to do that.

***Cause: Network Adapter temporarily has no IP Address***

Whenever you switch from a DHCP company LAN to the ETAS hardware network, it takes at least 60 seconds until ETAS hardware can be found. This is caused by the operating system's switching from the DHCP protocol to APIPA, which is being used by the ETAS hardware.

***Cause: ETAS Hardware had been connected to another Logical Network***

If you use more than one PC or notebook for accessing the same ETAS hardware, the network adapters used must be configured to use the same logical network. If this is not possible, it is necessary to switch the ETAS hardware off and on again between different sessions (repowering).

***Cause: Device driver for network card not in operation***

It is possible that the device driver of a network card is not running. In this case you will have to deactivate and then reactivate the network card.

**To deactivate and reactivate the network card**

1. To deactivate the network card, open the Control Panel.
2. Go to the "Network and Sharing Center", then click on the "Change adapter settings" link.
3. In the "Network Connections" window, right-click on the used network adapter and select **Disable** in the context menu.
4. In order to reactivate the network adapter, right-click on it again and select **Enable**.

***Cause: Laptop power management deactivates the network card***

The power management of a laptop computer can deactivate the network card. Therefore you should turn off power monitoring on the laptop.

**To switch off power monitoring on the laptop**

1. Press <WINDOWS LOGO> + <x> to open the power user menu. Select **Device Manager** from that menu.
2. In the Device Manager open the tree structure of the entry **Network Adapters**.
3. Right-click on the used network adapter and select **Properties** in the context menu.
4. Select the **Power Management** tab and deactivate the **Allow the computer to turn off this device to save power** option.
5. Select the **Advanced** tab. If the property **Autosense** is included, deactivate it also.
6. Click **OK** to apply the settings.

***Cause: Automatic disruption of network connection***

It is possible after a certain period of time without data traffic that the network card automatically interrupts the Ethernet connection. This can be prevented by setting the registry key `autodisconnect`.

**To set the registry key autodisconnect**

1. Open the Registry Editor.
2. Select under HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Services\lanmanserver\parameters the Registry Key autodisconnect and change its value to 0xffffffff.

## 9 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 Headquarters

ETAS GmbH

Borsigstraße 24

70469 Stuttgart

Germany

Phone: +49 711 3423-0

Fax: +49 711 3423-2106

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



## Figures

---

Fig. 3-1	dT Scheme .....	17
Fig. 6-1	Schematic process of a bypass cycle (DISTAB method) .....	44
Fig. 7-1	Distab13 data structure .....	50
Fig. 7-2	Hook-Based Bypass: Principle .....	51
Fig. 7-3	Bypass with hooked service points .....	52
Fig. 7-4	Service-Based Bypass: Principle (The dashed line indicates that bypass data can be written back at a later time as well.) .....	53
Fig. 7-5	Possible data inconsistency (the small arrows () indicate a waiting time for bypass results) .....	56
Fig. 7-6	Suggested SBB implementation .....	56
Fig. 7-7	Controlling ECU function execution from the Hardware Configurator (The columns "Cluster Group" and "Cluster" are only available for SBB V3. The "Raster Usage" display is only available for SBB V2.) .....	57

# Index

---

## A

ASCET options  
  "Hardware Configuration" node . . . . .13  
  "Hardware Connection" node . . . . .14  
  "Hardware" node . . . . .12  
  "Signal Mapping" node . . . . .13  
ASCET Rapid Prototyping . . . . .12  
Automatic Mapping . . . . .13

## B

bypass  
  concept . . . . .50–58  
  ETK (see also *ETK Bypass*) . . . . .38  
  hook-based . . . . .51  
  safety . . . . .54  
  service-based . . . . .52, 55  
bypass offset . . . . .46

## C

CAN database  
  import . . . . .35  
Compiler  
  precompiled header . . . . .14  
  select . . . . .14  
  use own ~ . . . . .15  
Contact information . . . . .64

## D

daisy chain  
  import configuration . . . . .35  
DISTAB method . . . . .41  
dT . . . . .17

## E

ES910  
  dT . . . . .17  
  memory pages . . . . .18  
ETAS contact information . . . . .64  
ETAS network  
  Hardware Connection . . . . .14  
ETAS Safety Advice . . . . .6  
ETK Bypass . . . . .38  
  ASCET project . . . . .39  
  data exchange . . . . .41  
  DISTAB method . . . . .41  
  Hardware Configuration . . . . .38  
  how it works . . . . .40  
Experimental target . . . . .9  
Experimenting with ASCET . . . . .18–26  
  acquisition task . . . . .22  
  assign element to measurement  
    window . . . . .22  
  C code Debugger . . . . .25  
  open environment . . . . .21

run online~ . . . . .20–25  
  setting up an experiment . . . . .23  
  standalone mode . . . . .25  
  start experiment . . . . .23  
  start measurement . . . . .23  
  stop experiment . . . . .24  
  stop measurement . . . . .24  
Experimenting with INCA . . . . .26–33  
  Back-Animation . . . . .31  
  INCA database path . . . . .28  
  initiating a transfer . . . . .26  
  selecting a device . . . . .30  
  selecting a project . . . . .30  
  selecting a workspace . . . . .29  
  starting a transfer . . . . .31  
EXPORT Subdirectory . . . . .9  
External hardware configurator . . . . .34

## F

FlexRay  
  import configuration . . . . .35

## H

hardware configuration  
  add hardware description . . . . .36  
  create (Hardware configurator) . . . . .35  
Hardware Configurator . . . . .34  
  add hardware description to hardware  
  configuration . . . . .36  
  automatic mapping options . . . . .13  
  create configuration . . . . .35  
  create hardware description file . . . . .35  
  external . . . . .34  
  open . . . . .36  
  set up project . . . . .36  
  use INTECRIO . . . . .34  
Hardware Connection  
  with ETAS Network Manager . . . . .14  
hardware description file  
  add to hardware configuration . . . . .36  
  create . . . . .35  
hardware options . . . . .12  
  "Hardware Configuration" node . . . . .13  
  "Hardware Connection" node . . . . .14  
  "Signal Mapping" node . . . . .13  
hook-based bypass . . . . .51  
  classical . . . . .51  
  with Distab17 . . . . .51

## I

INCA  
  see *Experimenting with INCA*

INTECRIO	
Experimenting with ~	33
use as Hardware Configurator	34
<b>L</b>	
LIN	
import configuration	35
<b>O</b>	
Online experiment	
acquisition task	22
assign element to measurement	
window	22
open experiment environment	21
running	20–25
standalone	25
start	20
<b>P</b>	
precompiled header	14
privacy	6
Product liability disclaimer	6
project	
add hardware configuration	36
<b>R</b>	
RTA-OSEK	9
RTPRO-PC	
dT	17
memory pages	18
<b>S</b>	
Safety	5
intended use	5
Safety information	6
service-based bypass	52
specifics	55
<b>T</b>	
target	
set up interfaces (with ETAS Network Manager)	14
Target directory	12