

INTECRIO-ASC V6.2

User's 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 2013** ETAS GmbH, Stuttgart

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

Document EC011101 V6.2 R01 EN - 05.2013

Contents

1	Introduction	7
1.1	Safety Advice	7
1.1.1	Correct Use	7
1.1.2	Labeling of Safety Instructions	7
1.1.3	Demands on the Technical State of the Product	8
1.2	Components	8
1.3	Installation	9
1.4	Manual Structure	9
1.5	Conventions	10
1.5.1	Documentation Conventions	10
1.5.2	Typographic Conventions	10
2	Configuring Experimental Targets	11
2.1	The Hardware Options	11
2.2	Hardware Connection with the ETAS Network Manager	13
2.2.1	The Hardware Selection Window	14
2.3	Interface Setup Without ETAS Network Manager (ES1000 Only)	17
2.4	Selecting a Compiler	18
2.4.1	Using Your Own Compiler	18
2.4.2	Changing to the GNU Cross Compiler or QCC Compiler	19
2.5	Hints on Using INTECRIO-ASC	21
2.5.1	Preprocessing Available Data Bases	21
2.5.2	Converting Projects for ES1000.1 to a Supported Target	21
2.5.3	Using dT	22
3	Hardware Systems	25
3.1	Hardware – ES1000.x Experimental System	25
3.2	ES900 Experimental System	26
3.3	RTPRO-PC Experimental System	29

3.3.1	Startup	30
3.3.2	Configuring RTPRO-PC	30
3.4	Special Features of the ES1135	32
3.4.1	Watchdog	32
3.4.2	LEDs	34
3.4.3	Cache-Locking	34
3.5	Non-Volatile RAM (NVRAM)	38
3.5.1	Basics	38
3.5.2	Hardware Support	39
3.5.3	NV Variable Initialization and Update	41
3.5.4	Data Consistency	41
3.5.5	NVRAM Cockpit	43
3.5.6	Tips	45
4	Experimenting with INTECRIO	47
4.1	Project Transfer to INTECRIO	47
4.2	The INTECRIO Experiment	55
4.3	The Back-Animation	56
4.4	ASCET and SCOOP-IX	61
5	Tutorial – Experimenting with INTECRIO	67
5.1	Preparations	67
5.2	Transferring the Project	70
5.3	Experimenting in INTECRIO	71
5.4	Using Back-Animation	73
6	Appendix A: Compiler Switches and API Functions	77
6.1	Compiler Switches for External C Code	77
6.2	API Functions (ERCOS ^{EK})	77
6.2.1	Application Modes	79
6.2.2	Tasks	80
6.2.3	System Time	81
6.2.4	Interrupt Handling	82
6.2.5	dT Query	83
6.3	API Functions (NVRAM)	84
6.4	API Functions (Watchdog)	90
6.4.1	Watchdog Configuration	91
6.4.2	Watchdog Service	94
6.4.3	Interrupt Control	95
6.4.4	Watchdog Status	96
6.5	API Functions (ES1135 LEDs)	97
6.6	API Functions (Miscellaneous)	98
7	ETAS Network Manager	99
7.1	Overview	99
7.2	ETAS Hardware Addressing	99
7.3	Network Adapter Addressing	100
7.3.1	Type of Network Adapter Addressing	100
7.3.2	Addressing the Network Adapter Manually	100
7.3.3	Addressing the Network Adapter via DHCP	100

7.4	User Interface	101
7.4.1	"Network settings for ETAS hardware (Page 1)" Dialog Window .	101
7.4.2	"Network settings for ETAS hardware (Page 2)" Dialog Window .	103
7.4.3	"Network settings for ETAS hardware (Page 4)" Dialog Window .	104
7.5	Configuring Network Addresses for ETAS Hardware	104
7.5.1	Adapter with Fixed IP Address	104
7.5.2	Adapter in DHCP Environment.	106
7.6	Troubleshooting Ethernet Hardware Access	107
8	Troubleshooting General Problems	109
8.1	Problems and Solutions	109
8.1.1	Network Adapter cannot be selected via Network Manager.	109
8.1.2	Search for Ethernet Hardware fails.	110
8.1.3	Personal Firewall blocks Communication	112
9	ETAS Contact Addresses	117
	Index	119

1 Introduction

The INTECRIO ASCET Connector (INTECRIO-ASC) is an add-on product to ASCET and INTECRIO or higher. It contains everything required for a successful linking of ASCET models in INTECRIO for integration and rapid prototyping.

During the installation of INTECRIO-ASC, the required files and settings are automatically inserted in the existing ASCET installation. These files and settings have the following tasks:

- Selection of INTECRIO-ASC as rapid prototyping environment in the ASCET project editor.
- Provision of a special code generation for INTECRIO in which all required files (C code, ASAM-MCD-2MC file, SCOOP-IX description file) are created.
- If the ASCET code generation for INTECRIO is used, automatic import of the project in INTECRIO.
- Adjusting the handling of non-resolved global variables and messages to the needs of INTECRIO.
- Providing options for the migration of existing projects.
- Providing the option of performing the integration in an INTECRIO system directly from ASCET.
- Providing the option for back animation of an ASCET model during the experiment.

Back animation means that the model variables can be measured and calibrated directly from within the ASCET model.

INTECRIO-ASC V6.2 supports ASCET V6.2 and INTECRIO V4.0 or higher.

1.1 Safety Advice

Please adhere to the Product Liability Disclaimer (ETAS Safety Advice) and to the following safety instructions to avoid injury to yourself and others as well as damage to the device.

1.1.1 Correct Use

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

1.1.2 Labeling of Safety Instructions

The safety instructions contained in this manual are shown with the standard danger symbol shown below:



The following safety instructions are used. They provide extremely important information. Read this information carefully.



WARNING!

Indicates a possible medium-risk danger which could lead to serious or even fatal injuries if not avoided.



CAUTION!

Indicates a low-risk danger which could result in minor or less serious injury or damage if not avoided.



NOTICE

Indicates behavior which could result in damage to property.

1.1.3 Demands on the Technical State of the Product

The following special requirements are made to ensure safe operation:

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



WARNING!

Wrongly initialized NVRAM variables can lead to unpredictable behavior of a vehicle or a test bench, and thus to safety-critical situations.

*Projects that use the NVRAM possibilities expect a **user-defined** INIT process 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).*

Further safety advice is given in the ASCET V6.2 safety manual (ASCET_Safety_Manual.pdf) available on your installation disk, in the ETASManuals\ASCET_V6.2 folder on your computer or in the download center of the ETAS web site.

1.2 Components

The INTECRIO-ASC installation includes the following components:

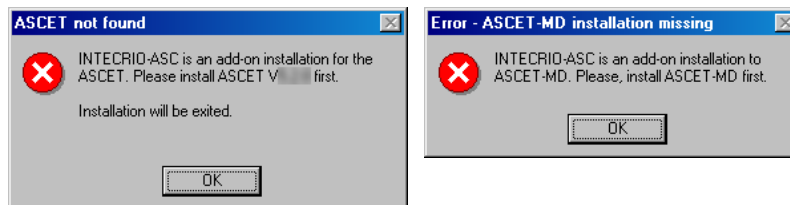
- Integration of the targets Prototyping, ES1130, ES1135¹, ES910 and RTPRO-PC;
- GNU and QCC compilers;
- Documentation and example.

1.3 Installation

The INTECRIO-ASC software is supplied on an installation disk. Before installing INTECRIO-ASC V6.2 on your PC you have to install ASCET V6.2. You can use INTECRIO-ASC to generate code for INTECRIO, even if INTECRIO is not installed on your computer.

First, start the installation programs `ASCET.exe` and `ASCET-MD.exe`. Continue with the installation program `INTECRIO-ASC.exe` from the CD-ROM. Finally, obtain a license file.

When you try to install INTECRIO-ASC before the ASCET base system or ASCET-MD, the following error messages are displayed.



The installation is aborted. The installation is also aborted when ASCET-RP is found on your PC, because ASCET-RP covers the complete functionality of INTECRIO-ASC V6.2.

Further details on the INTECRIO-ASC V6.2 installation can be found in the release note.

Details on licensing are given in chapter 3 "Licensing" of the the ASCET installation guide.

Sample Files

After the installation of INTECRIO-ASC V6.2, the sample database exported from ASCET is located in the `EXPORT` directory of your ASCET installation in the Tutorial `INTECRIO.*` (* = `.exp` or `.axl`) file.

1.4 Manual Structure

The INTECRIO-ASC V6.2 user's guide consists of two main sections:

- General Section
- Tutorial

The general section is intended for all users of INTECRIO-ASC V6.2. Here, the users find information about the structure, installation and usage of INTECRIO-ASC V6.2.

The tutorial contains a lesson about experimenting with INTECRIO.

¹. The term *ES113x* is used throughout this manual for an arbitrary ES1000 system controller.

1.5 Conventions

1.5.1 Documentation Conventions

Instructions are phrased in a task-oriented format as shown in the following example:

To reach a goal:

- Execute operation 1.
Explanations are given below an operation.
- Execute operation 2.
- Execute operation 3.

In this manual, an *action* is a sequence of operations that need to be executed in order to reach a certain goal. The title of an action usually expresses the result of the operations, such as "To create a new component" or "To rename an item". The action descriptions often include screenshots of the corresponding ASCET window or dialog window related to the action.

1.5.2 Typographic Conventions

The following typographic conventions are used throughout this manual:

Select File → Open .	Menu options are shown in boldface/blue.
Click OK .	Buttons are shown in boldface/blue.
Press <ENTER>.	Keyboard commands are shown in angled brackets, in block capitals.
The "Open File" dialog window opens.	Names of program windows, dialog windows, fields, etc. are enclosed in double quotes.
Select the <code>setup.exe</code> file.	Text in combo boxes on the screen, program code, as well as path and file names are shown in <code>Courier</code> font.
A <i>distribution</i> is always a one-dimensional table of sample points.	Emphasized text portions and newly introduced terms are printed in <i>italic</i> font face.
The OSEK group (see http://www.osekvd.org/) has developed certain standards.	Links to internet documents are set in <u>blue</u> font.

Important notes for the users are presented as follows:

Note

Important note for the user.

2 Configuring Experimental Targets

INTECRIO-ASC offers code generation for the Prototyping, ES113x, ES910 and RTPRO-PC targets incl. SCOOP-IX and ASAM-2MC files, transfer to ASCET and back-animation of the ASCET model (after INTECRIO download).

INTECRIO-ASC V6.2 contains the tools required for producing the files required for the integration of the project with INTECRIO. 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. Chapter 2.1 describes the hardware options of INTECRIO-ASC, hardware connection via the ETAS Network Manager is described in chapter 2.2. Chapter 2.3 describes the configuration of the host interface via changes in target.ini, as well as the configuration of the ethernet interface.

Chapter 2.4 describes the selection of the compiler, and chapter 2.5 gives some hints on using INTECRIO-ASC.

Structure of the Target Directories

Installing INTECRIO-ASC V6.2 results in a target directory with three subdirectories being created in the ASCET directory. These subdirectories contain target-specific information, configuration and library files. The following table shows the both subdirectories:

ASCET Subdirectory	E-Target	Computer Node
..\Target\ES1130	ES1000.2/ES1000.3 ^a	ES1130
..\Target\ES1135	ES1000.2/ ES1000.3	ES1135
..\Target\ES910	ES910.2/ES910.3	ES910
..\Target\QNX86	RTPRO-PC	RTPRO-PC
..\Target\Prototyping	(determined in INTECRIO)	

a: The terms *ES1000* or *ES1000.x* are used in this manual, unless a particular target is meant.

The `...\Target\ES113x` directory contains files used by both ES1000 simulation nodes, e.g. compiler-specific make files.

All makefiles and build scripts support paths with blanks.

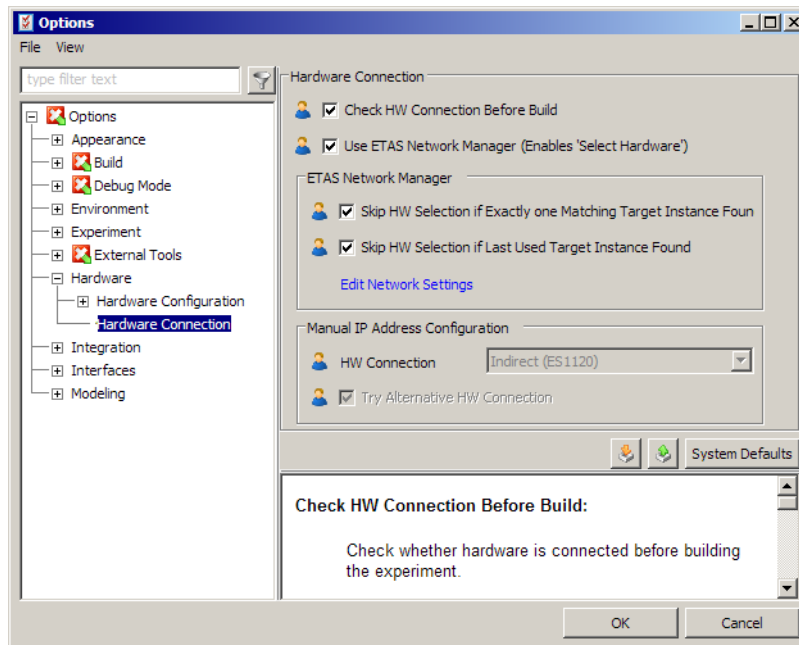
- If a path containing blanks is to be used in a makefile, ASCET converts it to short Windows format (for example, `c:\Documents and Settings` would be converted to `c:\DOCUME~1`).
- If a path containing blanks is to be used in a batch file, ASCET generates it encapsulated in `"`, or converts it to short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

It is not necessary to change the target root path in the ASCET options window to correspond to the target.

2.1 The Hardware Options

INTECRIO-ASC adds the "Hardware" node and its subnodes to the ASCET option window for easy setting of the interface.

For INTECRIO-ASC, only the "Hardware Connection" node is relevant. In that node, you can specify the following options. It is suggested that you select the item you use most frequently.



- **Check HW connection before Build**

Use this option to determine whether, upon starting an experiment (**Open Experiment**), the hardware search is performed *before and after* (activated, default) or *only after* the build process. If no suitable hardware is detected, an error message occurs.

When the option is activated, you can correct the error by adding a suitable hardware without losing the time for the build process.

When the option is deactivated, you can perform the build process without an error message, despite missing hardware.

- **Use ETAS Network Manager (enables 'Select Hardware')**

Use this option to determine whether the ETAS Network Manager (chapter 7) is used (activated, default) or not.

When the option is activated, the **Select Hardware** button and the **Tools → Select Hardware** menu option in the project editor become available.

Note

*To work with the ES910 or RTPRO-PC, you **must** use the ETAS Network Manager.*

- **Skip HW selection if exactly one matching target instance found**

Available only when you are using the ETAS Network Manager.

When this option is activated (default), the "Experimental Target Hardware Selection" window does not open when only one hardware, which matches the project, is found upon experiment start.

When this option is deactivated, the next option determines whether the "Experimental Target Hardware Selection" window opens each time you start an experiment. This window offers all experimental targets connected to your PC for selection.

- **Skip HW selection if last used target instance found**

Available only when you are using the ETAS Network Manager.

When this option is activated (default), "Experimental Target Hardware Selection" window does not open when only that hardware which was last used with the project is found upon experiment start.

When this option is deactivated, the previous option determines whether the "Experimental Target Hardware Selection" window opens each time you start an experiment.

- **Edit Network Settings**

This link opens the ETAS Network Manager (see chapter 7).

- **HW connection**

Available only when you are not using the ETAS Network Manager.

In this combo box, you select whether the ES1000 and your PC are, by default, connected via the ES1120 control unit (`Indirect (ES1120)`, default) or via the ES113x simulation computer (`Direct (ES113x)`).

- **Try alternative HW connection**

Available only when you are not using the ETAS Network Manager.

Use this option to determine whether a connection to both the device selected in the "HW connection" combo box and the other device (activated, default) or only to the selected device (deactivated) is to be searched.

2.2 Hardware Connection with the ETAS Network Manager

The ETAS Network Manager offers several advantages for the hardware connection.

- You can use a single network adapter for the ETAS hardware and our company network.
- You can assign individual network addresses.

Working with the ETAS Network Manager is described in chapter 7. Here, you find information regarding hardware connection using the EAS Network Manager.

To activate ETAS Network Manager usage:

- In the component manager or in the project editor, select **Tools → Options**.
The "Options" dialog window opens.
- Open the "Hardware\Hardware Connection" node.

- Activate the **Use ETAS Network Manager (enables 'Select Hardware')** option.
If *only* this option is activated, the hardware selection window "Experimental Target Hardware Selection" window opens at each experiment start.
- Activate the **Skip HW selection if *** options to skip the hardware selection window under the respective conditions.
- Activate the **Check HW connection before Build** if the hardware search is to be performed before the build process.

In addition to this automatic, you can open the "Experimental Target Hardware Selection" window from the project editor at any time.

To open the hardware selection window manually:

Note

The **Select Hardware** button and the **Tools → Select Hardware** menu option are only available when the **Use ETAS Network Manager (enable 'Select Hardware')** option is activated.

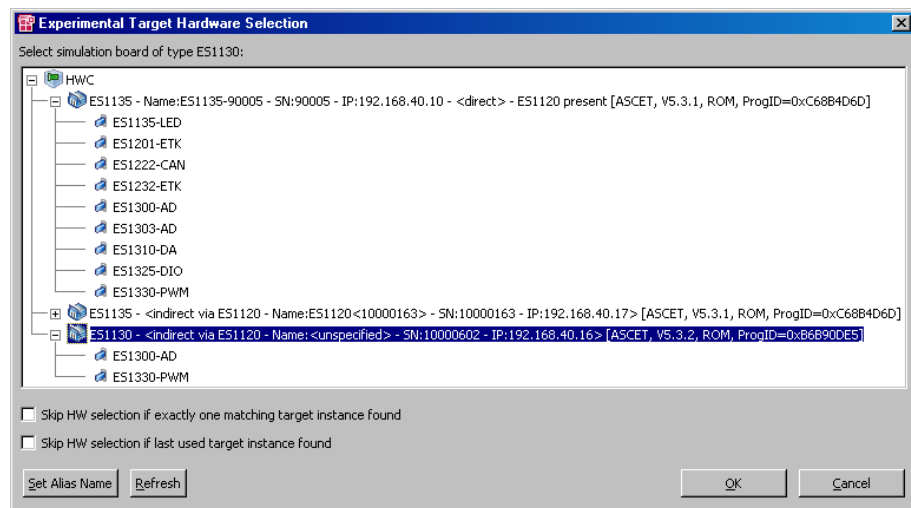
- Select **Tools → Select Hardware**

or






- click on the **Select Hardware** button.
The hardware selection window opens.

2.2.1 The Hardware Selection Window



The hardware selection window, named "Experimental Target Hardware Selection", contains the following elements:

- Field "Select simulation board of type *<type>*"¹
 This field displays, below the main entry HWC (symbol ) , all simulation controllers (ES113x, ES910 or RTPRO-PC – symbol ) connected with the PC.
 The simulation controller label contains, in addition to the controller name, further information; see page 15. Available ES1000 boards (symbol ) are displayed below the simulation controller; the ES910 and RTPRO-PC interfaces are not visible in this window.
 For the experiment, select the simulation controller you have entered in the code generation options of your project.
- Options **Skip HW selection if exactly one matching target instance found** and **Skip HW selection if last used target instance found**
 These options offer the same functionality as the identical options in the "Hardware" tab of the ASCET option window (see chapter 2.1).
 The settings performed here are transferred to the "Hardware" node and vice versa.
- Button **Set Alias Name**
 You can use this button to assign an arbitrary name to the ES113x or ES1120 or ES910 or RTPRO-PC.
- Button **Refresh**
 This button updates the "Select simulation board of type *<type>*" field. Hardware newly connected or switched on is displayed afterwards, hardware that was removed or switched off, disappears from the display.
- Buttons **OK** and **Cancel**
 Click **OK** to accept the selection, or **Cancel** to close the hardware selection window without accepting the selection.

If an ES1000 simulation controller is directly connected to the PC, its entry in the "Select simulation board of type *<type>*" field looks as follows:

```

ES113x - Name:<alias> - SN:<serial number> -           ↵
  IP:<IP address> - <direct> - ES1120 present         ↵
  [<SW>, <syslib version>, <boot mode>,             ↵
  ProgID=<ID>]
  
```

- ES113x is the simulation controller label.
- Name:<alias> is the optional name you can assign to the simulation controller.
 If you do not specify a name, this part is absent.
- SN:<serial number> is the serial number of the ES113x.
- IP:<IP address> is the IP address of the ES113x.
- <direct> indicates that the ES113x is connected directly to the PC.
- ES1120 present indicates that the ES1000 contains an unconnected ES1120.
 If the ES1000 contains no ES1120, this part is absent.

¹ The name element *<type>* is determined by the target selected in the respective project.

- *<SW>* is the software you used to load a program to the ES1000 (e.g., ASCET or INTECRIO).
- *<syslib version>* is the version of the hardware system library in use.
- *<boot mode>* indicates whether the project was started from the Flash memory when the ES1000 was switched on (ROM), or whether a download followed power-on (RAM).
- ProgID=*<ID>* is the identifier *<ID>* assigned to the project by the software *<SW>*.

If an ES1000 simulation controller is indirectly connected to the PC, i.e. via ES1120, its entry in the "Select simulation board of type *<type>*" field looks as follows:

```
ES113x - <indirect via ES1120 - Name:<alias> -           ↵
      SN:<serial number> - IP:<IP address>>             ↵
      [<SW>, <syslib version>, <boot mode>, ProgID=<ID>]
```

- ES113x is the simulation controller label.
- *<indirect via ES1120 ...>* indicates that the ES113x is connected indirectly to the PC.
- Name:*<alias>* is the optional name you can assign to the ES1120.
- SN:*<serial number>* is the serial number of the ES1120.
- IP:*<IP address>* is the IP address of the ES1120.
- *<SW>*, *<syslib version>*, *<boot mode>* and ProgID=*<ID>* have the same meaning as the identical parts in a direct connection.

If an ES900 simulation controller is connected to the PC, its entry in the "Select simulation board of type *<type>*" field looks as follows:

```
ES910 - Name:<alias> - SN:<serial number> -           ↵
      IP:<IP address> [<SW>, <syslib version>,           ↵
      <boot mode>, ProgID=<ID>]
```

- ES910 is the simulation controller label.
- Name:*<alias>* is the optional name you can assign to the ES910.
- SN:*<serial number>* is the serial number of the ES910.
- IP:*<IP address>* is the IP address of the ES910.
- *<SW>*, *<syslib version>*, *<boot mode>* and ProgID=*<ID>* have the same meaning as the identical parts in an ES1000 direct connection (see page 16).

If RTPRO-PC is used as simulation controller, its entry in the "Select simulation board of type *<type>*" field looks as follows:

```
RTPRO-PC - Name:<alias> - SN:<serial number> -         ↵
      IP:<IP address> - <direct> -                       ↵
      [<SW>, <syslib version>, <boot mode>, ProgID=<ID>]
```

- RTPRO-PC is the simulation controller label.
- Name:*<alias>* is the optional name you can assign to RTPRO-PC.
- SN:*<serial number>* is the serial number of RTPRO-PC.
- IP:*<IP address>* is the IP address of RTPRO-PC.
- *<direct>* indicates that RTPRO-PC is connected directly to the PC.

- *<SW>*, *<syslib version>*, *<boot mode>* and *ProgID=<ID>* have the same meaning as the identical parts in an ES1000 direct connection (see page 16).

2.3 Interface Setup Without ETAS Network Manager (ES1000 Only)

For special ES1000 use cases, INTECRIO-ASC offers the possibility to work without the ETAS Network Manager, in accordance with previous versions. The ethernet interface is set up for ASCET in the `target.ini` file of the target you are using. The files are located in the `..\Target\ES1130` or `..\Target\ES1135` directory.

To determine the ES1000 connection:

- In the Component Manager, select **Tools** → **Options**.
The "Options" dialog window opens.
- Open the "Hardware\Hardware Connection" node.
The options are described in chapter 2.1.
- Deactivate the **Use ETAS Network Manager (enables 'Select Hardware')** option.
- In the "HW connection" combo box, select the appropriate entry for your ES1000.
- Activate the **Try alternative HW connection** option when a connection to both the device selected in the "HW connection" combo box and the other device is to be searched.
When the option is deactivated, only a connection to the selected device is searched.
- Activate the **Check HW connection before Build** if the hardware search is to be performed before and after the build process.
When the option is deactivated, the hardware search is performed only *after* the build process.
- Finally, click **OK** to accept your settings.

Depending on your selection, a particular IP address variable from the respective `target.ini` file in the target subdirectory (`..\Target\ES1130` or `..\Target\ES1135`) is used for the ASCET experiment environment

- If the ASCET host PC is connected to the control unit (ES1120) of the ES1000.x, the following variable is used:
 - **ES1130**

```
IndirectIpAddress=192.168.40.10
;Default IP-Address for ES1120.x
```
 - **ES1135**

```
IndirectIpAddress=192.168.40.10
;Default IP-Address for ES1120.x
```
- If the ASCET host PC is connected to the computer node (ES1130) of the ES1000.x, the following variable is used:

- **ES1130**
`DirectIpAddress=192.168.40.11`
`;Default IP-Address for ES1130.x`
- **ES1135**
`DirectIpAddress=192.168.40.15`
`;Default IP-Address for ES1135.1`

2.4 Selecting a Compiler

The GNU Cross Compiler (GNU-C V3.4.4 (Power PC) in the user interface) is integrated in INTECRIO-ASC for the Prototyping and ES113x targets. The QCC compiler (QCC V6.5.0 in the user interface) is integrated in INTECRIO-ASC for the ES910 and RTPRO-PC target.

Target	Compiler
Prototyping, ES1130, ES1135	GNU Cross Compiler
ES910, RTPRO-PC	QCC compiler

Tab. 2-1 Target/Compiler Overview

For each compiler, version-specific make files are provided in the `..\Target\Prototyping\trgmake` or `..\Target\ES113x\trgmake` or `..\Target\ES910\trgmake` or `..\Target\QNx86\trgmake` directory. Version-independent settings, which are included in the specific make files, are stored in the 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 make files 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

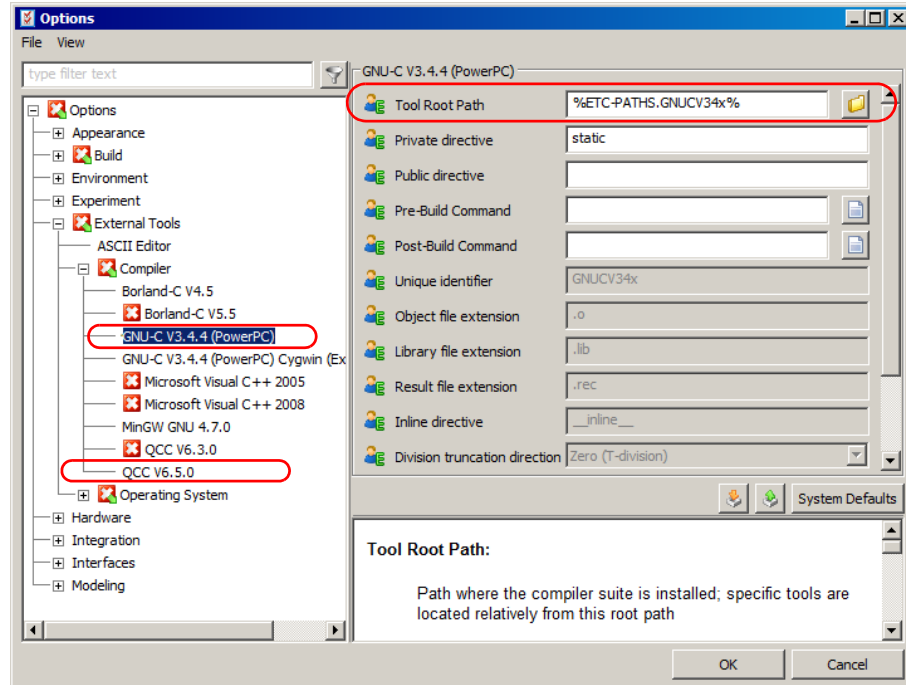
FILES_INIT_INV    = $(CC_INV)           ↵
                  #Add specific options here
```

Precompiling C header files used by many C code files can significantly speed up the compilation process. The GNU Cross Compiler, V3.4 and higher, supports usage of precompiled headers. Only one precompiled header file can be used per compilation. Therefore, `Project` should be selected in the "Build" node of the project settings, to use precompiled headers as effectively as possible. Using precompiled headers is activated automatically (ASCET options window, "GNU-C V3.4.4 (PowerPC)" node, [Supports precompiled header](#) option).

2.4.1 Using Your Own Compiler

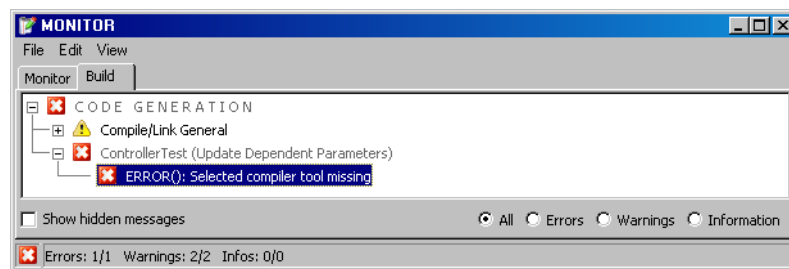
The MS-DOS version of the GNU Cross Compiler and the QCC compiler are supplied as a standard parts of the INTECRIO-ASC V6.2 package.

If you want to use your own compiler, you have to reset the path of the GNU-C V3.4.4 (PowerPC) or QCC V6.5.0 compiler to the new compiler in the ASCET options dialog.

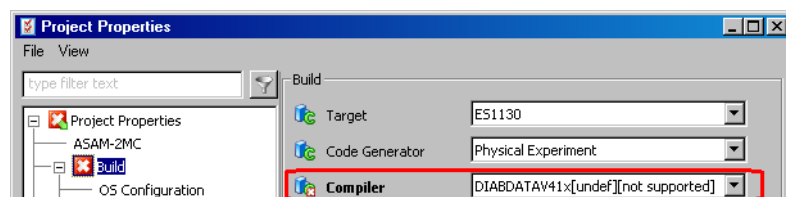


2.4.2 Changing to the GNU Cross Compiler or QCC Compiler

If you are working with an older project that uses the ES1130 target with the Diab Data compiler, *no* error message is shown when you open the project. Only the generation of executable code produces an error message in the ASCET monitor window.



When you double-click the error message, the ASCET options window opens in the "Build" node. The compiler is marked as undefined and not supported.



If you are working with an older project that uses the ES1112 target, the following error message is displayed when you open the project, and the missing target is replaced by the target PC.

```
The required Target >ES1112< is not available! Changed
Target to >PC<!
```

In both cases, you have to select a suitable combination of target and compiler for the project. Proceed as follows.

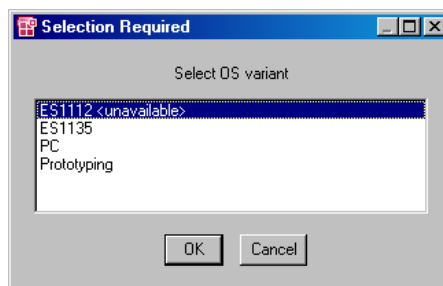
To change to a suitable target/compiler combination:



- In the project editor, click on the **Project Properties** button.
The "Project Properties" window opens.
- In the "Build" node, select the target **Prototyping** or **ES1130** or **ES1135** or **ES910** or **RTPRO-PC**.
A suitable compiler is selected automatically.
- Click **OK**.
Depending on which target the project used, you now have to copy the C code, if your project uses C code blocks. The procedure is described in "To copy operating system settings and C Code:" below.
- 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:

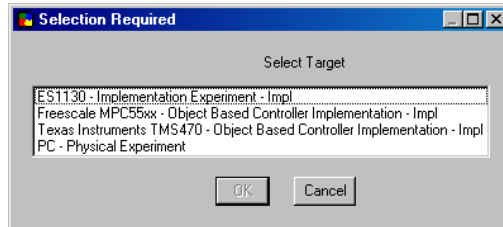
- Select the "OS" tab in the project editor.
- Select **Operating System** → **Copy From Target** from the project editor.
The "Selection Required" window opens.



- From the "Selection Required" window, select the original target of the old project.
- Click **OK**.
The operating system code is copied from the old target to the **Prototyping** or **ES1130** or **ES1135** or **ES910** or **RTPRO-PC** target.

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

The "Selection Required" window opens.



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

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

2.5 Hints on Using INTECRIO-ASC

2.5.1 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 V6.2.

Note

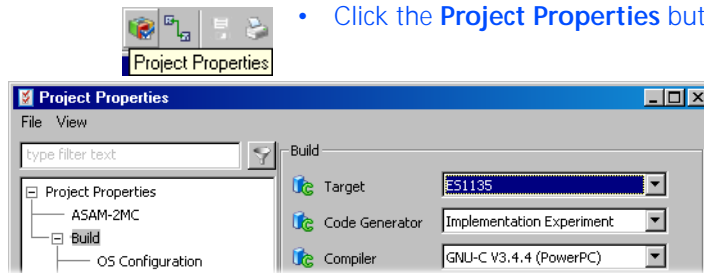
Detailed information on converting very old ASCET projects (with TIPExp V3.x and older, Target PPC) is given in the ASCET online help.

2.5.2 Converting Projects for ES1000.1 to a Supported Target

ASCET projects which were created for the ES1000.1 E-target have to be converted to a supported E-target, i.e. ES1000.2, ES1000.3, ES900 or RTPRO-PC. The following steps have to be executed for the conversion.

To convert an ASCET project for ES1000.1 to a supported target:

- Load the ASCET project.
A message is displayed that the ES1112 target is no longer available.
- Click **OK** to confirm the message.
The project opens; the target >PC< is selected instead of the unavailable target.



- Click the **Project Properties** button.

- In the "Project Properties" window, "Build" node, select the target **Prototyping** or **ES1130** or **ES1135** or **ES910** or **RTPRO-PC**.
A suitable compiler is selected automatically.
- Click **OK** to close the "Project Properties" window.
- Copy the operating system settings and C code to the selected target, as described in "To copy operating system settings and C Code:" on page 20.

The project can now be edited for the ES1000.2/ ES1000.3, ES900 or RTPRO-PC system.

2.5.3 Using dT

The ERCS^{EK} operating system is implemented for the ES113x target. An RTA-OSEK operating system is implemented for the ES910 and RTPRO-PC targets. Both operating systems enable 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. 2-1).

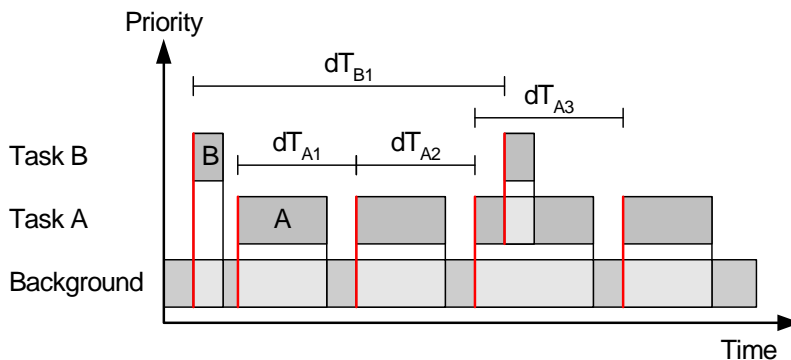


Fig. 2-1 dT Scheme

dT is a global `uint32` variable. It is declared in one of the ERCS^{EK} 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 (`rea164`) which contains the time in units of seconds.

If users do not generate this element in the C code editor, but still accesses dT , no error message appears because dT is declared in the ERCS^{EK}/RTA-OSEK files. But as dT in ERCS^{EK}/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.

3 Hardware Systems

The operation of a rapid-prototyping experiment requires an ETAS experimental system, i.e. one of the following systems:

- ES1000.2
- ES1000.3
- ES900 system (with ES910.2 or ES910.3)
- RTPRO-PC

3.1 Hardware – ES1000.x Experimental System

All ES1000 system controller boards currently available are supported. Fig. 3-1 shows the standard configurations; special configurations are of course possible.

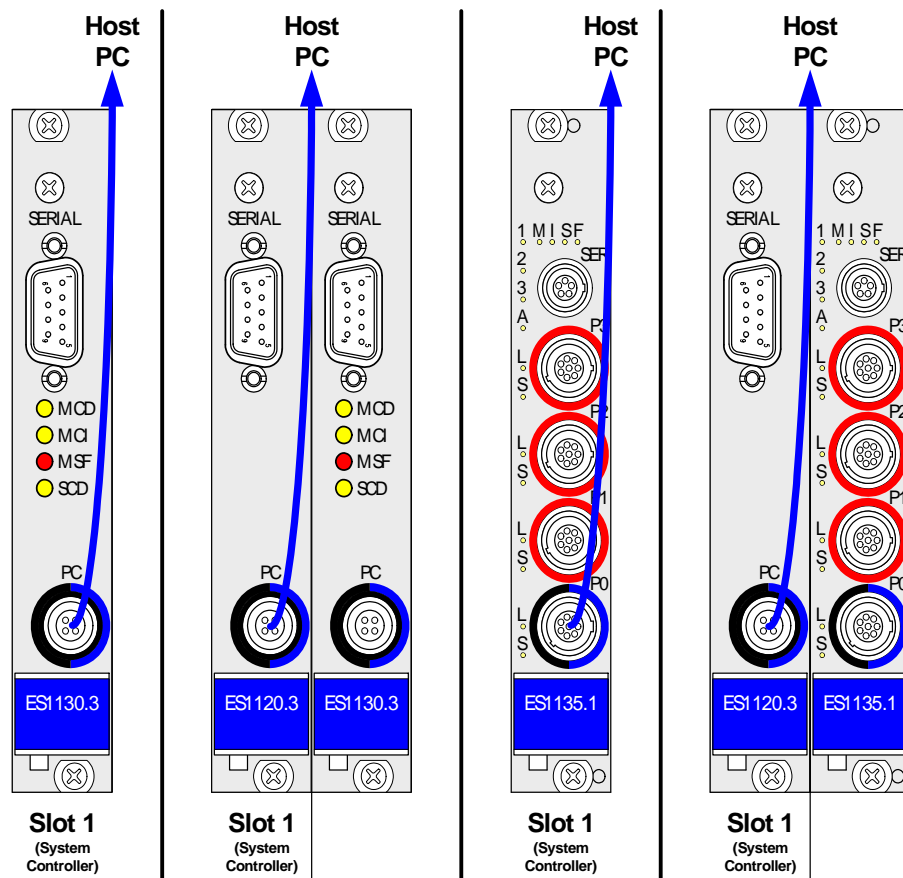


Fig. 3-1 Permissible and Supported System Controller Configurations

Control Unit ES1120 and Simulation Computer ES1130/ES1135

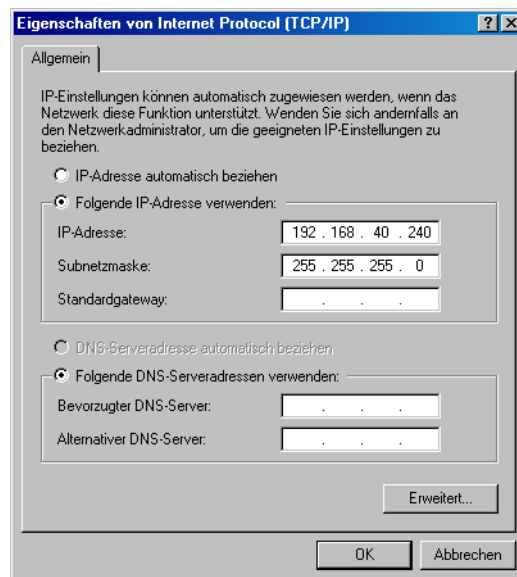
If the ES1000.x is used for application and rapid prototyping simultaneously, the host PC is connected to the control unit ES1120 via ethernet cable. The functions developed with ASCET are loaded via the control unit ES1120 onto the PPC module ES1130 or ES1135, and then executed. Data can be measured and calibrated with ASCET while the experiment runs.

TCP/IP Protocol Options

To avoid conflicts with a second network card that might be used for the LAN, the following TCP/IP settings should be selected.

To configure the TCP/IP protocol options:

- Disable the DHCP service.
- Enter the IP address 192 . 168 . 40 . 240.
- Enter the subnet mask 255 . 255 . 255 . 0.



- For the DNS service, use the local settings of your internal network.
- Disable the WINS service.
- Make sure that the "IP Forwarding" option is **not** activated.

3.2 ES900 Experimental System

ES910.2 and ES910.3 devices are supported.

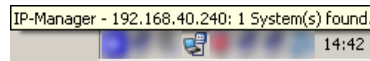
Configuring the ES910

In order to use the ES910, you must adjust the port settings of ES910 via its graphic user interface. A web browser application is used for that purpose; the ES910 must be connected to the PC.

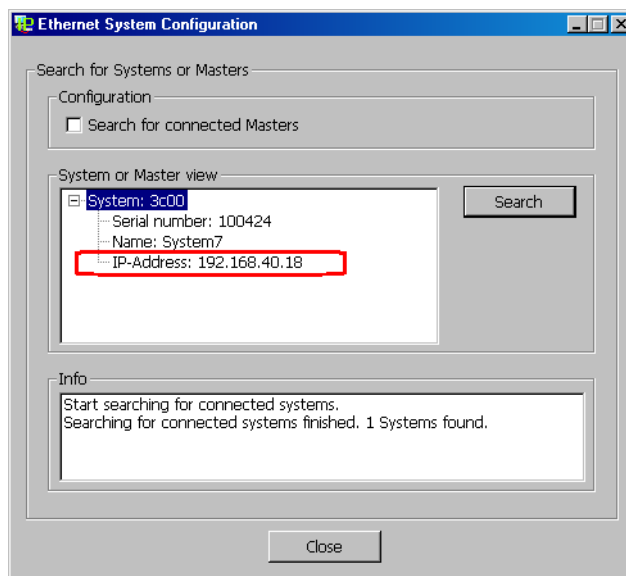
To determine the IP address and open the configuration interface:

An alternative way to open the ES910 user interface is described in the ES910 user's guide.

- In the Windows system tray, right-click the **IP-Manager** icon and select **Ethernet System Configuration** from the context menu.

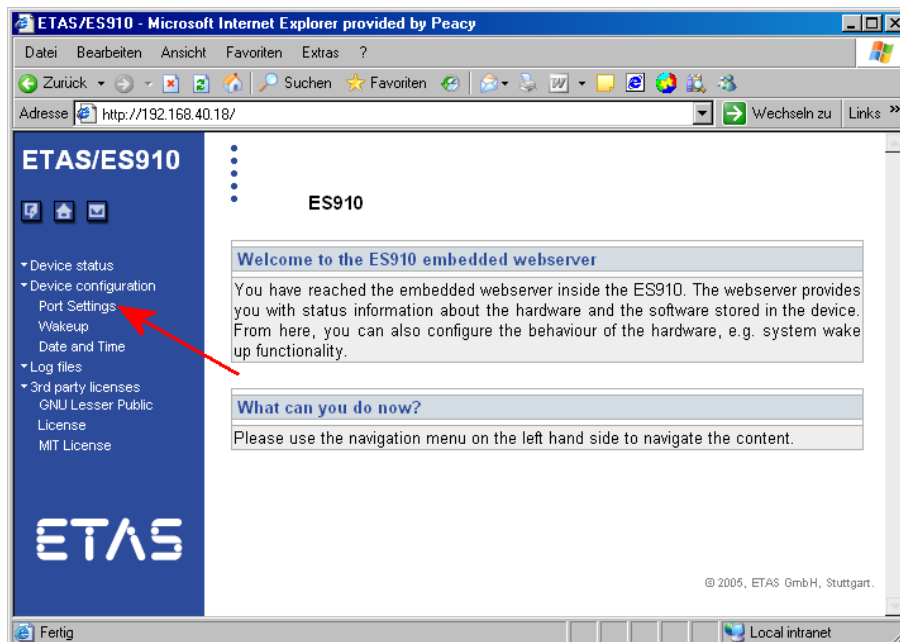


The "Ethernet System Configuration" window opens.



The IP address is shown in the "System or Master view" field.

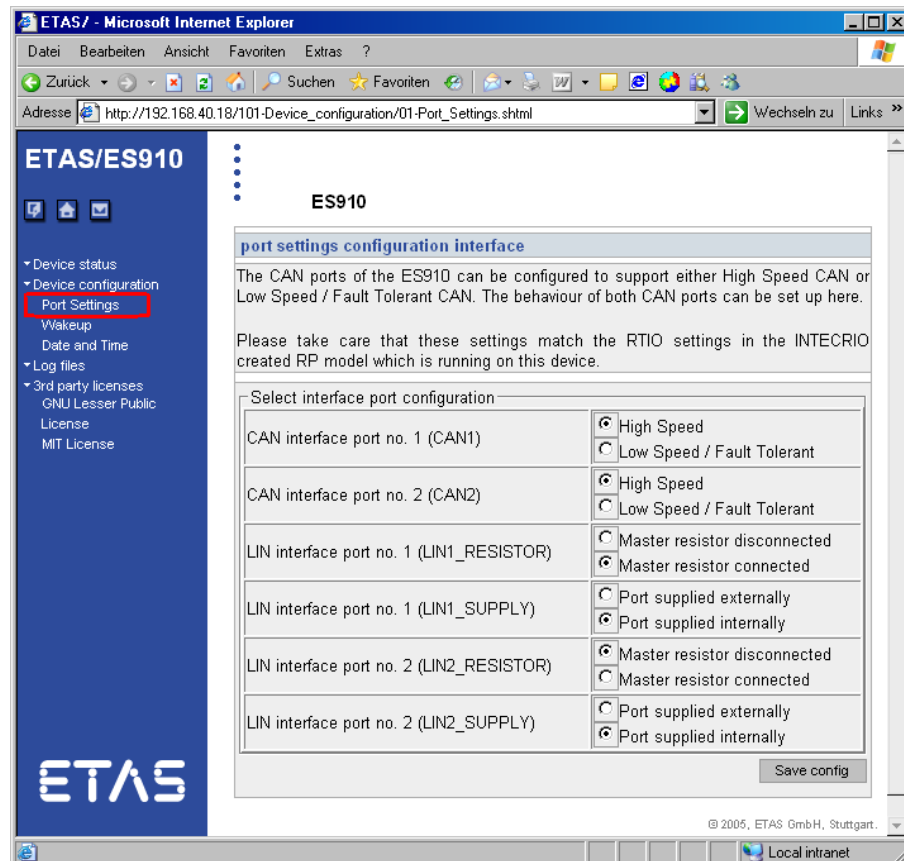
- Enter this IP address in your web browser to open the ES910 user interface.



To configure the ES910:

Master resistor and power supply settings must be adjusted.

- On the start page of the ES910 user interface, follow the link [Port Settings](#) below the "Device configuration" caption.



- Configure the ports according to your needs.
- Save the configuration.

3.3 RTPRO-PC Experimental System

RTPRO-PC allows the real-time execution of prototyping models on an off-the-shelf notebook. After installation of RTPRO-PC, the system can still be used as a standard Windows®-only computer, but it has got a second boot option to work as a combined Windows / prototyping (= real-time) computer.

On the Windows node of the combined mode, standard Windows applications can be used with slightly less performance than in Windows-only mode.

The real-time node is supported as standard ETAS experimental target with the following features:

- The RTPRO-PC target is configured and built in INTECRIO-ASC, ASCET-RP V6.1.3 and higher or INTECRIO V4.1 and higher.

- Experiments can be performed using the ASCET experiment environment, the INTECRIO experiment environment, or INCA with INCA-EIP V7.0.1 and higher.
- An internal switch on the real-time node allows ECU access via RTPRO-PC from INCA.
- Up to four CAN interfaces (via two ES581.3) can be added. Each CAN interface supports either XCP on CAN or CAN I/O.
- One Ethernet controller is supported. This can be used for XCP bypass on UDP and XETK. The ethernet controller supports up to four XCP on UDP interfaces.

The two nodes communicate via a virtual network.

To use the RTPRO-PC experimental system with ASCET-RP, you need a notebook with ASCET-RP and RRTPRO-PC installations. RTPRO-PC is a separate product available at ETAS; please contact your local sales representative.

RTPRO-PC is available for the following notebooks:

- HP EliteBook 8540w with Intel® Core™ i7-7xx or -8xx processor and Mobile Intel® QM57 Express chipset
- HP EliteBook 8560w with Intel® Core™ i7-7xx, -8xx, -27xx, or -28xx processor and Mobile Intel® QM67 Express chipset
- HP EliteBook 8570w with Intel® Core™ i7-* or i5-* (3rd generation) quad-core processor (Intel® Hyper-Threading must be supported) and Mobile Intel® QM77 Express chipset
- Lenovo Thinkpad T530 (with Intel® Core™ i7-* or i5-* (3rd generation) dual-core processor (Intel® Hyper-Threading must be supported) and Mobile Intel QM77 Express chipset

In addition, you need an USB stick to serve as persistent memory and to store the license file and the NVRAM of the experimental system.

3.3.1 Startup

The Realtime Prototyping for PC platform is started with the POWER ON switch of your notebook. After that, BIOS takes control and starts the GRUB bootloader.

The bootloader offers the following boot configurations:

- Windows
- ETAS RTPRO-PC and Windows
- ETAS RTPRO-PC (standalone hypervisor mode)

You must select ETAS RTPRO-PC and Windows or ETAS RTPRO-PC to use RTPRO-PC.

For more details, see the RTPRO-PC user's guide.

3.3.2 Configuring RTPRO-PC

In order to use RTPRO-PC and ES581 (CAN), you must adjust the ES581 device settings of RTPRO-PC. A web interface is used for that purpose.

To launch the web interface:

- In the Windows Start menu, open the **ETAS** program folder, then open the **RTPRO-PC** subfolder and select **RTPRO-PC Control Panel**.
- In the RTPRO-PC control panel, click on **Open Web Interface**.

Or

- In the context menu of the RTPRO-PC system tray icon, select **Open Web Interface**.

Or

- Open a web browser and enter the IP address of RTPRO-PC in the address bar.
- Press <RETURN>.

Or

- Start HSP.
- In HSP, click **Hardware Search**.
- In the "Hardware" list, right-click RTPRO-PC and select **System configuration** from the context menu.

The RTPRO-PC embedded web server opens.

ETAS - RTPRO-PC - Embedded web server - Mozilla Firefox

ETAS 192.168.40.16

ETAS - RTPRO-PC - Embedded web server

DRIVING EMBEDDED

RTPRO-PC ETAS (Internet)

- + Device status
- + Device configuration
- + Log files
- + 3rd party licenses
- + ETAS Licensing

Welcome to the RTPRO-PC embedded web server

Installed Licenses	
Software	License
RTPRO-PC	Evaluation License (83 days left)

Device status	
Firmware	Operational
Flashdisk	Operational
Flashed RP Model	Not present
RP Model Process	Not running

The embedded web server allows you to [view the status](#) of your device and perform [hardware configuration tasks](#), as well as to [upload log files](#) in case of trouble.

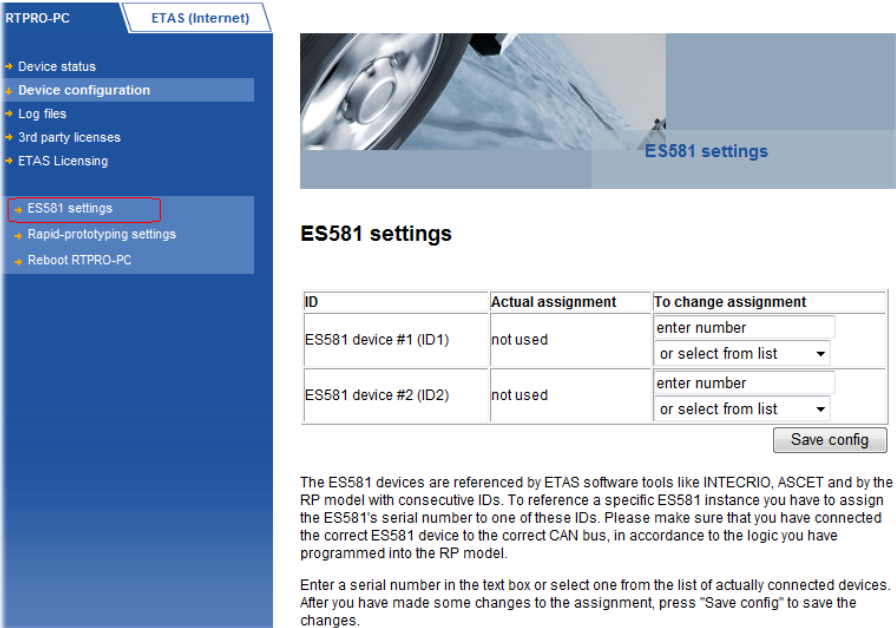
To operate this device a valid ETAS license is required. To manage the license of this device go to [ETAS Licensing](#).

The software of this device uses some 3rd party libraries, programs and scripts. Please refer to the [License information](#) section for details.

Some settings are stored persistently in non-volatile device memory. Make sure the settings of this device match your project's requirements in INTECRIO, ASCET or INCA respectively!

To make settings for ES581:

- On the start page of the RTPRO-PC web interface, follow the [ES581 Device Settings](#) link.
The "ES581 Device Settings" page opens. The combo box for each device lists the serial numbers of the ES581 hardware connected to the simulation node.



The screenshot shows the RTPRO-PC web interface. On the left, a navigation menu lists various options, with 'ES581 settings' highlighted. The main content area is titled 'ES581 settings' and contains a table with the following data:

ID	Actual assignment	To change assignment
ES581 device #1 (ID1)	not used	enter number or select from list
ES581 device #2 (ID2)	not used	enter number or select from list

Below the table is a 'Save config' button. A note explains that ES581 devices are referenced by ETAS software tools and that users should assign the correct serial number to the correct CAN bus.

- Open the combo box of each ES581 device and select a serial number.
The number is displayed in the field above the combo box.
- Click **Save config**, to store your settings.

Note

Make sure that the settings here match the ES581 settings in INTECRIO-ASC.

3.4 Special Features of the ES1135

The ES1135 is a further development of the ES1130, and offers the user several new functions. These functions are described in the sections of this chapter.

3.4.1 Watchdog

To integrate a safety concept for the rapid prototyping system, the ES1135 offers a hardware watchdog function. The watchdog is an independent control unit that monitors the main ES1135 processor. For that purpose, a predefined data

sequence is written periodically to a memory cell (watchdog service register). After a maximum time (watchdog period) without successful write access (watchdog service) to the watchdog service register, an exception handling (event) is triggered in the processor.

The ES1135 HW watchdog can be operated in two modes:

1. Safety-oriented mode (safety mode)
2. Flexible mode with more functions (Reduced Safety Mode Enhanced Function, RSEF Mode)

In the RSEF mode, the following watchdog settings can be re-configured at runtime:

- Event configuration
Defines the exception handling in case of watchdog expiration. The watchdog can also be disabled via event configuration.
- Watchdog period
Defines the time until the watchdog expires if no new watchdog service occurs.
- Switching modes
The safety mode is switched on with an arbitrary watchdog period and event configuration. After that, this mode cannot be reconfigured or left. Therefore, the watchdog service should be set up in advance in a way that no undesired watchdog event occurs.

After the supply voltage is switched on, the watchdog is set to RSEF mode and switched off.

Watchdog Service

The Watchdog must be serviced before it expires. Otherwise, the selected watchdog event occurs. It is the task of the model designer to put the call of the service function at a place, where a malfunction of the model can be detected.

The Simulation Controller firmware provides an automatic watchdog servicing mechanism, which services the watchdog every 30 ms if interrupts are not disabled by the model. Thus, assumed that operating system is running correctly, the watchdog will be serviced regularly (if feature is enabled). This will be sufficient in many use cases.

Interrupt Control

For debug and supervision purposes in particular, it is possible to configure the watchdog to trigger a simulation processor interrupt on watchdog timer expiration.

The interrupt may either be polled or routed to the internal interrupt controller. A watchdog interrupt is latched and needs explicit acknowledging. Functions for fast disabling and enabling of the interrupt source are available. These functions only have effects on the interrupt propagation.

The watchdog interrupt is mapped to a HW Task inside ASCET. The watchdog handler is running below the ERCOS^{EK} level but above other HW interrupts (e.g. from VME Bus), thus the watchdog interrupt is handled even if another HW interrupt is currently handled. Interrupt acknowledgement is done inside the ES1135 firmware, it should therefore not be done inside the handler task. The available set of ERCOS^{EK} calls in the handler task is not restricted.

When a watchdog interrupt occurs, the watchdog is automatically restored from the overrun situation after 250 μ s, restarting a new cycle with the previously selected period. This restoration time may be shortened, by performing a normal watchdog service with `wdService()`.

A detailed description of the watchdog API is given in chapter 6.4 "API Functions (Watchdog)" on page 90.

3.4.2 LEDs

The LEDs on the ES1135 front panel are divided into system LEDs (M, I, S, F, A, L, S) and freely programmable LEDs (1, 2, 3).

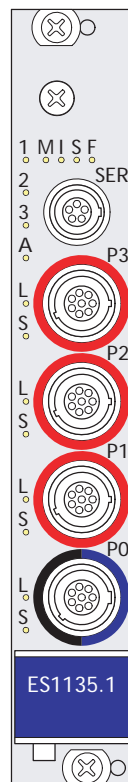


Fig. 3-2 ES1135 – Front Panel

The system LEDs are described in the hardware manual.

The programmable LEDs can be accessed via the program interface (see chapter 6.5 "API Functions (ES1135 LEDs)")

3.4.3 Cache-Locking

Depending on whether the respective function are placed in the main memory or the cache of the ES1135, noticeable runtime differences occur for short-period tasks. For highly time-critical applications, these differences can be intolerable.

As a remedy, INTCRIO-ASC V6.2 provides the possibility to mark highly time-critical parts of the model and thus make sure that they are always available in the cache. This procedure is called *cache locking*. The second-level cache, or *L2 cache*, which is divided into four separate units (fourfold associative cache, 4-ways), is used for that purpose.

You can mark individual variables or parameters, as well as entire methods or processes (cf. page 35). Components can be marked, too; this mark is adopted by the elements of the component, if applicable (cf. page 35, 36, 36, 37).

The mark is part of a particular implementation. In another implementation, the same object can have a different mark.

The following restrictions exist for cache locking:

- Up to 3 units of the L2 cache can be reserved for cache locking (cf page 37). At least one unit remains free for the unmarked parts of the model.
The associativity of the cache can make it impossible to keep all marked model parts permanently in the cache, even though cache space is available. In that case, This problem occurs rarely for code, but frequently for variables and parameters. It is thus recommended to mark code for cache locking.
- A cache unit cannot be further divided. If the model parts marked for cache locking occupy 2.5 units, the free half unit cannot be used otherwise.
- If parts of the cache are reserved for cache locking, less cache is available for the unmarked model parts. Runtime losses can occur.

Three settings are available for cache locking:

<code>Automatic</code>	Variables, parameters, methods and processes adopt the setting of the parent component. This is the default setting. For components, <code>Automatic</code> is the same as <code>OFF</code> .
<code>On</code>	Cache locking is switched on.
<code>Off</code>	Cache locking is switched off.

Set cache locking for variables, parameters and methods/processes:

- [Open the implementation editor for the element.](#)
- [In the "Cache Locking" combo box, select the desired setting.](#)
[Settings for individual elements, methods and processes overwrite the settings for components.](#)

Note

For characteristic lines/maps, the setting in one tab ("Value", "X Distribution", "Y Distribution") is adopted for the other tabs, too.

Set cache locking for a component:

- [Open the implementation editor for the component.](#)
- [Open the "Settings" tab.](#)

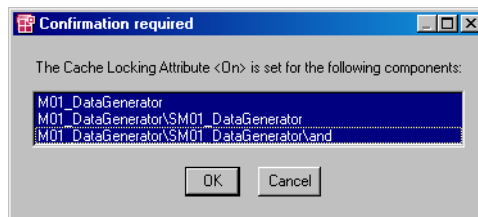
- In the "Cache Locking" combo box, select the desired setting.
The selected setting is adopted by all elements, methods and processes with the `Automatic` setting.
The setting is *not* recursive, however, it does *not* apply to included components.

Set cache locking for an included component:

Note

This procedure is recursive, i.e. it affects included components.

- In the "Outline" tab of a project or component editor, select an included component.
- Select **Edit** → **Set Cache Locking** → `<setting>`
or
- select **Set Cache Locking** → `<setting>` from the context menu of the included component.
A confirmation window opens that lists the selected component and its included components.



- Select the components to which you want to assign `<setting>`.
- Click **OK**.
The setting is assigned to the selected components. In the components, it is adopted by all elements, methods and processes with the `Automatic` setting.

Set cache locking in the OS editor:

In the OS editor ("OS" tab of the project editor), you can select a task and set cache locking for the modules whose processes are included in the task.

Note

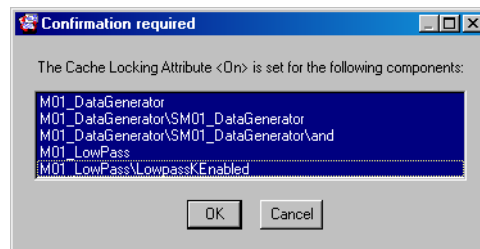
*You **cannot**, in this way, set cache locking for individual processes. If the processes of a module are assigned to different tasks, you can still make only one setting for all of them in the OS editor. Setting cache locking for individual processes is explained on page 35.*

- Open the OS editor.
- In the "Tasks" pane, select a task.

- Select **Task** → **Set Cache Locking** → *<setting>*
or

- select **Set Cache Locking** → *<setting>* from the context menu of the task.

A confirmation window opens that lists the parent components of the processes in the selected tasks, as well as the included components.



- Select the components to which you want to assign *<setting>*.
- Click **OK**.

The setting is assigned to the selected components. In the components, it is adopted by all elements, methods and processes with the **Automatic** setting.

Set cache locking globally for a project:

In the project properties, "Experiment Code" node, you can switch on/off cache locking for the entire project.



- In the project editor, click on **Project Properties** to open the "Project Properties" window.
- In the "Experiment Code" node, activate the **Cache Locking** option to switch on cache locking globally.
- In the "Experiment Code" node, deactivate the **Cache Locking** option to switch off cache locking globally.

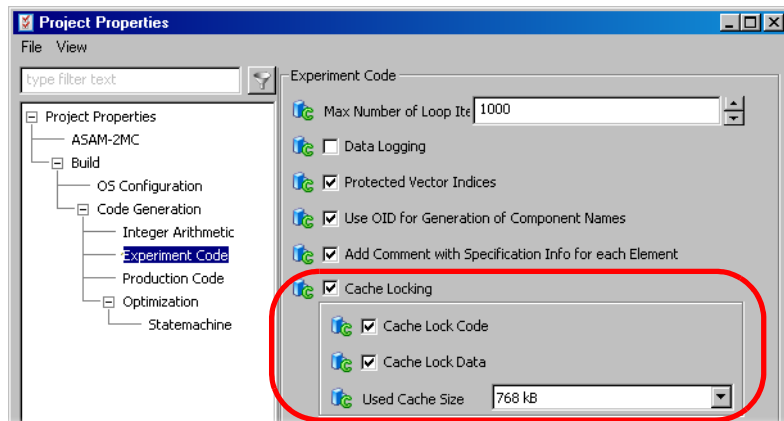
Note

The following options are only available if cache locking is activated.

- Activate/deactivate the **Cache Lock Code** option to switch on/off cache locking for code.
- Activate/deactivate the **Cache Lock Data** option to switch on/off cache locking for data.

- In the "Used Cache Size" combo box, select the number of cache units reserved for cache locking.

0 kB – 0 units
 256 kB – 1 unit
 512 kB – 2 units
 768 kB – 3 units



- Click **OK** to close the window and accept your settings.

The setting is adopted by all elements, methods and processes anywhere in the project with the **Automatic** setting.

3.5 Non-Volatile RAM (NVRAM)

3.5.1 Basics

A non-volatile (NV) variable is a variable which can be used like any other ASCET variable. Particularly, it can be written and read by the model, calibrated via a calibration window and measured/logged with the data acquisition. The special feature of an NV variable is that, in case of a simulation interruption, the current value of the NV variable is available when the simulation with the same model is restarted. This is especially useful for adaptive characteristics, commonly used inside the ECU code for self-learning algorithms, and storage of diagnostic results.

The optional attribute *non-volatile* (NV) is supported for all primitive data types of ASCET (scalars, arrays, matrices, characteristic lines/maps). Only ASCET variables can be configured for NVRAM, C-Code variables are not supported.

An NV variable can be created inside a class or module editor. This is done by activating the **Non-volatile** option in the element editor of a variable.



WARNING!

Wrongly initialized NVRAM variables can lead to unpredictable behavior of a vehicle or a test bench, and thus to safety-critical situations.

*Projects that use the NVRAM possibilities expect a **user-defined** INIT process 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).*

3.5.2 Hardware Support

64 kByte of non-volatile RAM (NVRAM) are available in the address space of the ES1135 main processor (IBM750GX), and 32 kByte of NVRAM are available on the ES910. The amount of NVRAM available on RTPRO-PC depends on the USB stick that is used as persistent memory. In this memory range, data can be stored which are to be available after a power failure or longer than one power-on cycle.

Due to performance reasons (accesses to the NVRAM are significantly slower than accesses to the normal, volatile RAM and, in addition, cannot be cached), NV variables are not directly allocated to the NVRAM. Instead, they are allocated like normal, volatile variables to normal RAM, and periodically saved to the NVRAM (auto-update mode). The period is by default set to 10 seconds; it can be configured with the API method

```
uint32 nvramSetUpdateInterval(uint32 interval_sec)
```

(see Chapter 6.3) in the range from 1 second to 30 seconds. Saving to the NVRAM is done within the idle task, it does not affect the real-time behavior of the model.

For reasons of data consistency, the NVRAM is organized as alternation buffer which halves the available capacity. Furthermore, there is some overhead involved which reduces the NVRAM capacity available to the ASCET model to a little less than 32 kByte (ES1135), 16 kByte (ES910) or half of the available capacity (RTPRO-PC).

The ES1135 or ES910 or RTPRO-PC firmware ensures that the capacity limit is respected. If the cumulative size of NV variables exceeds the NVRAM capacity, an "NVRAM overflow" error message appears in the ASCET monitor window at the start of the experiment. In this case, the NVRAM is not used, i.e. no values are written to it. In order to be able to use the NVRAM, the user needs to reduce number and/or size of NV-variables in the model.

The NVRAM data contain neither address information nor the names of the variables. Therefore, they correspond to the model only if the structure of the NV variables in the model did not change after the last update. To check this, a special *NV identifier* is created during code generation.

This NV identifier changes upon the following actions:

- Changing the instance name of any NV element on the project, module, class, or sub-class level.
- Changing the implementation of any NV element on the project, module, class, or sub-class level, namely:
 - Code generation option `Physical Experiment` – switching the implementation data type between `cont` and `sdisc/udisc`
 - Code generation option `Implementation Experiment` – changes of the implementation interval
- Changing the formula parameters of any NV element.
The NV identifier does not change if the name or the comment of a formula is changed.
- For an NV enumeration:
 - Changing the sequence of the enumerators
 - Changing the names (values) of the enumerators
 - Removing/adding enumerators
 - Selecting a different enumeration, even if it contains the same enumerators
- Changing the maximal size of multidimensional NV elements (array, matrix, characteristic line/map).
- Changing the settings for the implementation limitation in the implementation experiment (code generation option `Implementation Experiment`).
- Deleting/adding NV elements on the project, module, class, or sub-class level.
- Deleting/adding states in a state machine.

The NV identifier does *not* change upon the following actions:

- Renaming the project.
- Renaming the modules or classes within the project.
- Renaming the instances of any normal (volatile) element on the project, module, class, or sub-class level.
- Changing the formula name or comment of any NV element.
When the formula parameters are changed, the NV identifier changes, too.
- Renaming an enumeration.
- Changing the data of NV elements and normal elements.
- Changing the actual size of multidimensional elements (array, matrix, characteristic line/map).
- Changing the memory area of NV elements and normal elements.

3.5.3 NV Variable Initialization and Update

Starting the simulation: After the model code was downloaded to the target, NV variables are initialized with their default values if no matching data are available in the NVRAM. No matching data means that the NV memory is empty, inconsistent (verified via a checksum) or the NV data does not match with the downloaded model (verification via NV identifier).

In case matching data is available in the NV memory, the variables are initialized accordingly before the experiment can be started (**Start OS**).

Stopping the simulation : When the simulation is stopped (**Stop OS**), the most recently saved values of the NV variables are persistently stored inside the NVRAM. Even if the target is powered off or if the code is downloaded again, the simulation can proceed with the most recently saved values of the NV variables.

As mentioned before, the NV variables are periodically saved to the NVRAM in auto-update mode. To make sure that the current values—not the values from the last cyclic update—are available in the NVRAM, the API function

```
void nvramUpdateMemoryExit(void)
```

should be called at the end of the Exit task (task with application mode `inactive`).

If there are no NV variables inside the current model, the NVRAM content remains unchanged.

Model with NV variables inside the FLASH memory: A simulation model with NV variables inside the FLASH memory of the simulation controller is booted when power on occurs. The potential matching NV data is used for initializing the NV variables before starting the simulation.

Display whether model is running on default NV variable values:

Whether the model is running on default NV variable values (as specified in the ASCET data editor), or whether the variables are initialized out of the NVRAM, can be determined in two ways. In the experiment environment, an info message is written in the Target Debugger window. From within the model, the API function

```
uint8 nvramCheckForInitializedVars(void)
```

provides the same information.

Clearing the NVRAM content: To prevent the initialization of a model with the NV content (if the program identifier is matching), it is possible to clear the NV memory content. This enforces the initialization of the NV variables with their default values. This feature is currently not supported by the GUI. However, the API function

```
uint32 nvramClear(void)
```

allows to reset the NVRAM from within the model (see chapter 6.3 "API Functions (NVRAM)").

3.5.4 Data Consistency

When the simulation is interrupted by power off or a system crash, the NVRAM contains values of the NV variables. The relevance of these values depends on the time of the last automatically or manually (from within the model) triggered saving.

To guarantee the consistency of the NVRAM content in case of an unexpected termination of the simulation, different strategies can be used as introduced below.

The consistency level can be set with the following API function:

```
uint32 nvramSetConsistencyLevel(T_consistencyLevel
level)
```

No consistency: The NVRAM update is done without respect to consistency within NV variables and between individual NV variables.

Low level consistency (single variables): Low-level consistency means that the data consistency within NV variables (scalars, arrays and matrices, but not characteristic lines/maps) is guaranteed.

It must be noted here that the update of characteristic lines/maps cannot be done atomically (in the sense of low-level consistency)

High level consistency (among variables, after task completion): High-level consistency means that all NV variables are updated in the idle task, without interruption by the model.

The update for a set of NV variables should be atomic. A self-learning algorithm, for example, works on several variables, and it must be guaranteed that data for different variables in the NVRAM come from the same calculation cycle.

Model-controlled consistency (among variables and over multiple tasks cycles): The high level consistency mechanism guarantees consistency only if manipulations of NV variables are done within one task cycle. There may be cases where this manipulation lasts several task cycles (e.g. the update of an adaptive characteristic, done in several tasks). The ES1135 / ES910 / RTPRO-PC firmware cannot be aware of this, and therefore the model must control the NVRAM update.

For this purpose, the automatic update can be disabled by the following function:

```
uint32 nvramDisableAutoUpdate(void)
```

The manual update of the complete set of NV variables can be started with this command:

```
uint32 nvramManualUpdateBackground(void)
```

Even the manual update it is not allowed to block the whole system, and thus change the real-time behavior, until the update is finished. Therefore, the function for manual update returns immediately and the update is done in the background. The model can poll the status of the manual update with the following function:

```
uint8 nvramCheckRunningUpdate(void)
```

The function returns `true` if the update is running. The user, or the model, is responsible that the NV variables are not modified during the update.

The manual update mode can be disabled with

```
uint32 nvramEnableAutoUpdate(void)
```

A selective update of NV variables is not supported.

If there are no NV variables inside the current model, the NVRAM content remains unchanged.

Defective NVRAM content: In case of defective NVRAM content, e.g. if the checksum test failed, the user is warned. This is done textually in the experiment environment (or the ASCET monitor window).

3.5.5 NVRAM Cockpit

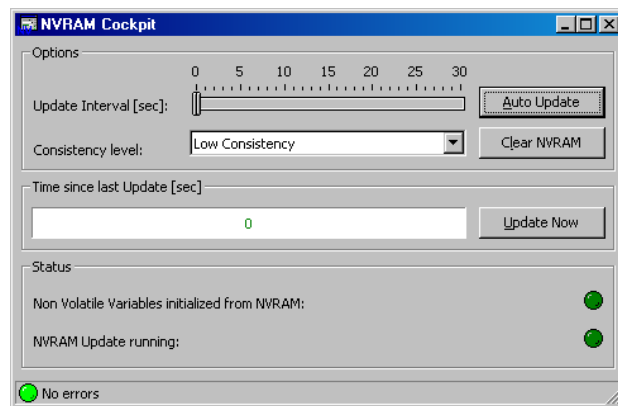
The NVRAM API (see Chapter 6.3) offers several functionalities to control the NVRAM update. During experiment, these functionalities are available in a special window, the *NVRAM cockpit*. Changes applied via the API functions are transferred to the NVRAM cockpit, too.

To work with the NVRAM cockpit:

- In the experiment, select **Tools** → **NVRAM Cockpit** or



- click on the **Open NVRAM Cockpit** button. The NVRAM cockpit opens.



- Use the control elements according to your needs.
- Close the NVRAM cockpit with **Close**.

The NVRAM cockpit contains the following control elements:

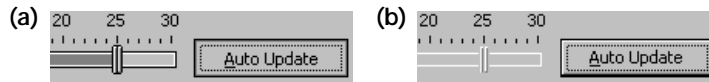
- **Update Interval [sec]**

Use this slider to adjust the interval (in seconds) for the automatic update of the NVRAM content. You can set up to 30 seconds; an interval of 10 seconds is predefined.

The slider is activated *only* when automatic update is switched on.

- **Auto Update**

This button switches the automatic update on and off. Automatic update is switched on if the button appears impressed (a), and switched off if the button appears upraised (b).



Note

*The automatic NVRAM content update works only while the experiment is running. Once you have stopped the experiment with **Experiment → Stop OS** or with the **Stop OS** button, the NVRAM content can no longer be updated automatically.*

- **Consistency level**

Use this combo box to select the consistency level of the update; see also "Data Consistency" on page 41.

- **Clear NVRAM**

Use this button to delete the NVRAM content.

When you click **Clear NVRAM** while the automatic update is running, the NVRAM content is deleted, but it will be written again after the next update interval at the latest.

- **Update Now**

Use this button to start the NVRAM content update manually.

This button is only available when automatic update is switched off.

Note

*The manual NVRAM content update works even if you stopped the experiment with **Experiment → Stop ERCOS** or with the **Stop ERCOS** button.*

The NVRAM cockpit contains the following displays:

- **Time since last Update [sec]**

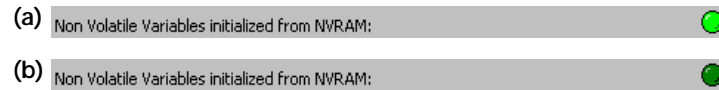
This bar display shows the time elapsed since the last update. The entire bar corresponds to 30 seconds; if this time is exceeded because automatic update is switched off, only the number is increased.

The counting of seconds continues even if the experiment is stopped, because time continues. Only a manual update after stopping the experiment resets the counter, which starts anew.

The bar is green as long as the time since the last update is less than 30 s, and red if this time is exceeded. Exception: The experiment was stopped (**Stop ERCOS**) prior to the overflow; in that case, the bar turns yellow upon overflow.

- **Non Volatile Variables initialized from NVRAM**

This display appears light-green if the NV variables are initialized with the NVRAM content (a), and dark-green if the NV variables are initialized with their default values (b).



- **NVRAM Update running**

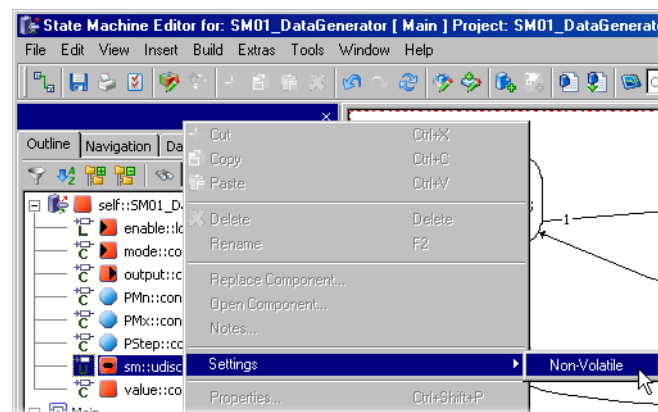
This display appears light-green if an NVRAM content update is currently running.



3.5.6 Tips

The following tips are useful for working with NVRAM.

State variable as non-volatile: The enhanced NVRAM support includes the possibility to assign the *non-volatile* attribute to the `sm` state variable of a state machine. To do so, right-click on the `sm` variable in the "Elements" list of the state machine editor and select **Settings** → **Non-Volatile** from the context menu.



Actual size of multi-dimensional elements: The actual size ("current size" attribute) of multi-dimensional elements (array, matrix, characteristic line/map) is saved in the NVRAM.

Changes of the actual size in the model (offline) become valid in the downloaded program only after the NVRAM content is deleted (e.g. via the NVRAM cockpit, or if the NVRAM content is inconsistent with the program).

Flash project with NVRAM on the ES1000: Bear in mind that the program in the Flash memory is launched each time the ES1000 or ES900 or RTPRO-PC is started. If this project uses NV variables, it uses the NVRAM. A subsequent download of any other program containing NV variables leads to an (unexpected) reset of the NVRAM content.

4 Experimenting with INTECRIO

If you have installed both INTECRIO-ASC and INTECRIO, you can experiment with your Rapid-Prototyping project in INTECRIO. The project editor offers a function for this purpose which allows convenient transfer of the experiment.

This chapter contains general instructions for experimenting with INTECRIO. A specific sample task can be found in chapter 5 "Tutorial – Experimenting with INTECRIO".

4.1 Project Transfer to INTECRIO

First of all, as usual, you create the ASCET project. You have all possibilities available to you which are possible in ASCET. But note the following points:

- By default, messages that are only read in ASCET (i.e. receive messages without relevant send messages) are the signal sinks in INTECRIO. Messages that are only written to (i.e. send messages without the relevant receive message) are the signal sources in INTECRIO. Messages that are both read and written in ASCET are excluded from integration.

If required, you can make the latter appear as signal sources in INTECRIO, see page 50.

- When your project contains unresolved messages (imported messages without corresponding export), the code generation for INTECRIO displays an error message.

You can either resolve the messages automatically or cancel the code generation and manually resolve the messages.

- It is possible to use global variables and parameters, but this is explicitly *not* recommended.
- Enumerations and formulas in your project must have different names. If the project contains an enumeration and a formula with identical names, the code generation for INTECRIO displays an error message.
- You have to select the target `Prototyping`, `ES1130`, `ES1135`, `ES910` or `RTPRO-PC` in the build options of the project. `INTECRIO` is pre-selected in the "Experiment Target" combo box with any of these targets. `INTECRIO` code generated with this target can be used with each experimental target supported by `INTECRIO`.

The target `Prototyping` is strongly recommended for transfer to `INTECRIO`.

- After selecting the `Prototyping` target, the following setup options are deactivated:
 - "Preemp. Levels" and "Coop. Levels" fields (all tasks)
 - **Enable Monitoring** option, "pre-/post hooks" combo box (all tasks)
 - "ISR Source" and "Min. Period" fields (interrupt tasks)
 - "Max. Number of Activations" field (alarm/software tasks)

- **Autostart** option (alarm/software tasks)

In some cases, the number of preemptive levels is set to 0. In that case, you can use **Operating System → Copy From Target** to copy the operating system settings (e.g., from an ES113x) to the **Prototyping** target. The deactivated settings are copied and the "Preemp. Levels" field is made available. You can now enter a suitable value, i.e. a number ≥ 8 .

- See chapter 4.4 "ASCET and SCOOP-IX" for information on how some ASCET settings appear in the SCOOP-IX file generated for use with INTECRIO-ASC.

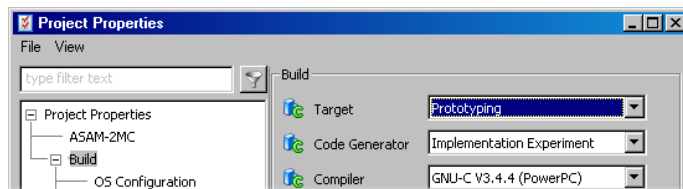
Once you have completely specified the project, you invoke the transfer of the project to INTECRIO as the first step in code generation.

To prepare the project:

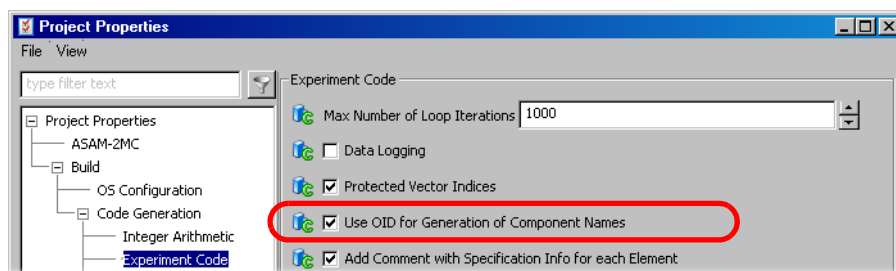
- Open the project you want to transfer to INTECRIO.
- Click the **Project Properties** button.



The "Project Properties" window opens in the "Build" node.



- Select the target **Prototyping** or **ES1130** or **ES1135** or **ES910** or **RTPRO-PC** and an appropriate compiler.
- In the "Experiment Code" node, activate the **Use OID for Generation of Component Names** option.



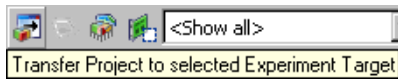
This setting is highly recommended.

To call transfer:



- Make sure that **INTECRIO** is selected in the "Experiment Target" combo box.

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

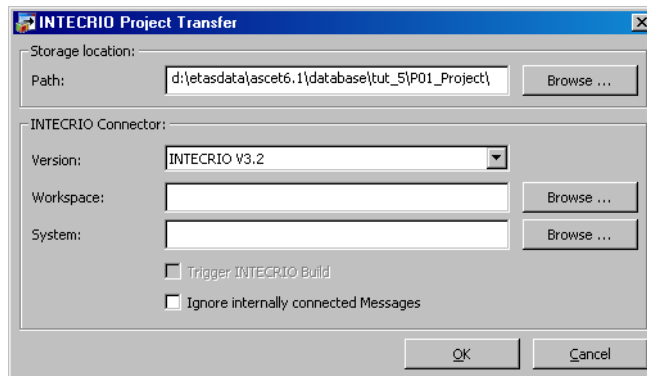


- Click the **Transfer Project to selected Experiment Target** button

or

- select **Build → Transfer**.

The "INTECRIO Project Transfer" window opens. The default transfer path (containing the project name) is entered in the "Path" field. It is located in the database directory. Thus, it is not overwritten, even if the **Keep files in Code generation Directory** option is deactivated.



The second step is to enter the path under which the generated files are stored in the "INTECRIO Project Transfer" window. You then have four choices:

- If you only want to *generate the code* required for INTECRIO, the "Paths" field is the only one that must contain a value. The "Workspace" and "System" fields must be empty.

This might be the case when the generated code is intended for transfer.

Note

Mere code generation for INTECRIO is possible even if no INTECRIO version is installed on your computer.

- If you want to *generate code and import it into INTECRIO*, you must also select the INTECRIO version and the INTECRIO workspace. The "Systems" field remains empty.

By default, the version of INTECRIO *last* installed is selected in the "Version" field. If only one INTECRIO version is installed, this is selected automatically; the field is disabled.

If the workspace does not exist, it is created automatically.

- If you want to *generate code, import it and integrate it into INTECRIO* (i.e. add it to an INTECRIO system project), enter the INTECRIO system project you want to work with.

In this case, both workspace and system project already have to exist.

- If you want to *generate code, import and integrate it into INTECRIO and start the Build process in INTECRIO*, complete all fields and activate **Trigger INTECRIO Build**.

To ensure the Build process can run, and generates a usable prototype, a hardware system and the operating system configuration have to be completely specified in INTECRIO.

Note

*ASCET does **not** check whether an existing workspace was created with the selected INTECRIO version.
If you select another INTECRIO version than the one used to create the workspace, the transfer can fail.*

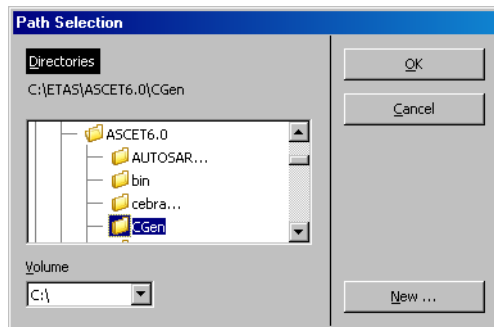
If you want messages that are read and written in the ASCET model to appear as signal sources/sinks, deactivate the **ignore internally connected messages** option. This option works with all of the four choices. Tab. 4-1 summarizes the message-to-interface-conversion for the activated and deactivated option.

Message Access in			INTECRIO Interface	
Project	Module A	Module B	option activated	option deactivated
S/R			—	—
S/R	S		signal source	signal source
S/R	R		signal sink	signal sink
S/R	S/R		signal source	signal source
	S		signal source	signal source
	S	S	signal source	signal source
	S	R	—	signal source
	S	S/R	—	signal source
	R		signal sink	signal sink
	R	R	signal sink	signal sink
	R	S/R	—	signal source

Tab. 4-1 Message conversion summary. *S*: messages sent by the respective component, *R*: messages received by the respective component.

To set the path for the generated files:

- In the "INTECRIO Project Transfer" window, click the **Browse** button next to the "Path" field.
The "Path Selection" window opens.



- If necessary, select a volume in the "Volume" combo box.
- Select an existing directory from the "Directories" list

or

- create a new directory using the **New** button.
- Click **OK**.

The directory is displayed in the "Path" field. Your selection is saved with the project; it is preselected at the next transfer.

To select the INTECRIO version:

If only one INTECRIO version is installed on your computer, that version is selected automatically, and the manual selection is disabled.

- In the "INTECRIO Project Transfer" window, open the "Version" combo box.
The list contains all INTECRIO versions installed on your PC.
- Select an INTECRIO version.
Your selection is saved with the project.

To select the INTECRIO workspace:

- In the "Workspace" field of the "INTECRIO Project Transfer" window, enter name and path of the INTECRIO workspace you want to use.

Or

- Click the **Browse** button next to the "Workspace" field.
The Windows file selection window opens.
- Select the directory which contains the workspace.
- Select the workspace (*.iow).

- Click **Open**.

The workspace is displayed in the "Workspace" field.

To select the INTECRIO system project:

A workspace has to be selected and INTECRIO has to be running for the successful integration of the ASCET project into an INTECRIO system project.

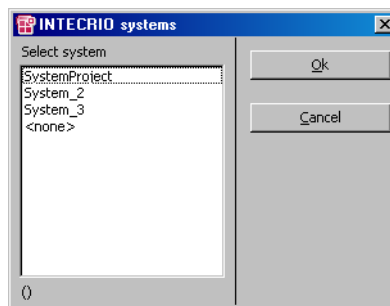
- In the "Systems" field of the "INTECRIO Project Transfer" window, enter the name of the INTECRIO system project you want to use.

Or

- Click the **Browse** button next to the "System" field.

When INTECRIO is not running, it is started now.

The "INTECRIO systems" window opens. It displays all system projects contained in the workspace.



- Select the system project into which you want to add the ASCET project.

- Click **OK**.

The system project is displayed in the "System" field.

To select the INTECRIO Build process:

If the INTECRIO system contains only one ASCET project, and if a hardware system and the OS configuration have been created in INTECRIO, the INTECRIO build process can be automatically started with the transfer.

- In the "INTECRIO Project Transfer" window, activate the **Trigger INTECRIO Build** option.

This is reasonable only when both an INTECRIO workspace and a system project have been selected.

The last step is the transfer to INTECRIO.

To execute transfer:

- Once you have made all the entries you need in the "INTECRIO Project Transfer" window, click **OK**.

The transfer of the ASCET project to INTECRIO is started.

1. Problem: unresolved messages

When your project contains unresolved messages, the following message opens.

There are imported elements without a matching export. Do you want to resolve global elements automatically? Select <OK> to resolve globals automatically or <Cancel> to go back and resolve them manually.

- Click **OK** to automatically resolve the messages. If the automatic procedure works, the transfer to INTECRIO continues.

Or

- Click **Cancel** if you want to abort the code generation and resolve the messages manually. In that case, you have to start the transfer anew.

2. Problem: another INTECRIO workspace is open

When INTECRIO is running, and another workspace is open, at the start of the transfer, the following message opens.

Active workspace in INTECRIO is different from the one you have selected. Close active workspace and start transfer anyway?

- Click **OK** to close the open workspace and continue the transfer of the ASCET project to INTECRIO.

Or

- Click **Cancel** to abort the transfer.

3. Problem: workspace requires higher INTECRIO version than selected

When the selected workspace requires a higher INTECRIO version than the one selected in "Version", the following message opens:

INTECRIO workspace file <path>\<filename>.iow does not fit for the selected INTECRIO version.

- Click **OK** to confirm the message and return to the "INTECRIO Project Transfer" window.
- Select another workspace and/or another INTECRIO version.

4. Problem: path for generated files is not empty

If the folder selected for the generated files (cf. page 51) is not empty, the following message opens:

The folder "<folder path and name>" already exists! If you continue, existing files may be overwritten. Do you want to proceed anyway?

- Click **OK** to proceed.

Or

- Click **Cancel** to abort the transfer.

During the transfer, all files necessary for working with INTECRIO are generated and stored in the specified directory.

If you have made the relevant entries, INTECRIO is started, the project is imported into INTECRIO and integrated into the system project, and the INTECRIO build process is started.

The following generated files are significant for working with INTECRIO:

- <project name>.six

This file contains the description of the project interfaces in the XML-based language, SCOOP-IX.

The interfaces of a possible HWC module are *not* included in the SCOOP-IX file because hardware configuration is done in INTECRIO.

A SCOOP-IX interface description basically consists of the following information:

- Name, type and size of C variables
- Name, return value and signature of C functions
- File origin of the C elements

For more details, refer to the "SCOOP and SCOOP-IX" section of the INTECRIO User's Guide.

If you configured the operating system in the OS editor, the SCOOP-IX file (except version V1.0) also contains some OS information but none from the deactivated options (see page 47). However, this information is currently not used by INTECRIO.

The *.six file can be validated against the schema of the respective SCOOP-IX version with a validating XML parser, e.g. XML SPY 2007 SP1. The SCOOP-IX schema files are stored in the `Formats\SCOOP-IX\<x>.<y>\Schemas` subdirectories of your ASCET installation, <x>.<y> being the SCOOP-IX version number.

- <project name>.a21

The ASAM-MCD-2MC file generated for working with INTECRIO.

A possible HWC module is not included in this file, either.

- `<project name>.oil`

This file contains the description of the operating system which can be used in INTECRIO.

Here, too, the HWC module is ignored because hardware configuration and OS configuration are done in INTECRIO.

Note

*This file is **never** imported automatically into INTECRIO. You either have to configure the operating system in INTECRIO manually or import the *.oil file manually.*

*The format of this *.oil file does not correspond to the OSEK standard; it is an XML-based description of the operating system configuration.*

- *.c and *.h

The C code and header files for the project and its different components. Exactly which *.c and *.h files are used by INTECRIO is contained in the following block of the *.six file:

```
<fileContainer complete="false">
  <pathBase path="{codeDir}" />
  <!-- model specific C files -->
  ... *.c- and *.h files ...
</fileContainer>
```

In addition, further files are created during code generation. These files are, however, irrelevant for working with INTECRIO.

4.2 The INTECRIO Experiment

Once transfer has been completed, you can experiment with the project in INTECRIO. Depending on what specifications you have made for the transfer, you have to carry out different steps.

To start an experiment:

The INTECRIO documentation describes how to execute the individual steps.

- [Import the code manually into INTECRIO.](#)
This step is not necessary if you imported the code automatically.
- [Add the model into the INTECRIO system project.](#)
This step is not necessary if you integrated the code automatically.
- [Complete the system project.](#)
This includes the creation of a hardware system, the configuration of the operating system, the connections between the hardware and the software.
- [Configure the operating system either manually or by importing the *.oil file.](#)
- [Generate the executable.](#)
- [Start the experiment.](#)

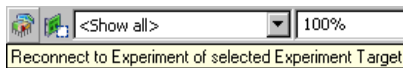
Working with the INTECRIO experiment environment is described in the online help of the experiment environment.

4.3 The Back-Animation

In addition to the INTECRIO experiment environment, the Back-Animation of INTECRIO 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 INTECRIO. Instead, use the relevant instruments of INTECRIO.

To start the Back-Animation:

- Start the INTECRIO experiment with your project.
- In the ASCET project editor, make sure that **INTECRIO** is selected in the "Experiment Target" combo box.
- Click the **Reconnect to Experiment of Selected Experiment Target** button



or

- select **Build** → **Reconnect**.

To select hardware (with ETAS Network Manager):

Note

If you are working without ETAS Network manager, skip this section and continue reading with section "What to do in case of an error:" on page 57.

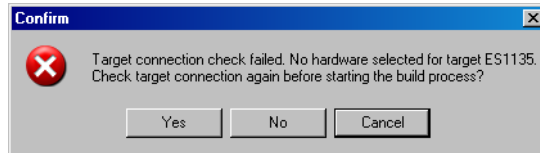
When you activated the **Use ETAS Network Manager (enables 'Select Hardware')** option in the hardware options (Chapter 2.1), the hardware selection window (Chapter 2.2.1) opens under certain conditions.

- In the "Select simulation board of type <type>" field, select the hardware you want to use.
The **OK** button becomes available.
- If required, perform other settings.
- 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; continue reading with section "To open the Back-Animation experiment environment:" on page 57.

What to do in case of an error:

If no agreement is found between selected and available hardware, the following error message opens.



- Click **Yes** to repeat the search for a hardware connection.

Or

- Click **No** to start the build process without hardware connection.

When you are using the ETAS network Manager, the hardware selection window opens again after the Build process.

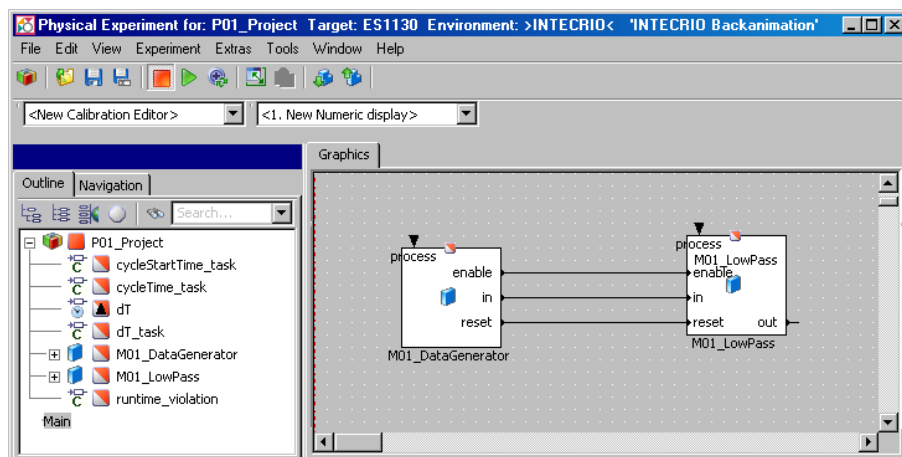
When you are not using the ETAS network Manager, you are asked, after the Build process, whether you want to repeat the search for a suitable hardware or not.

Or

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

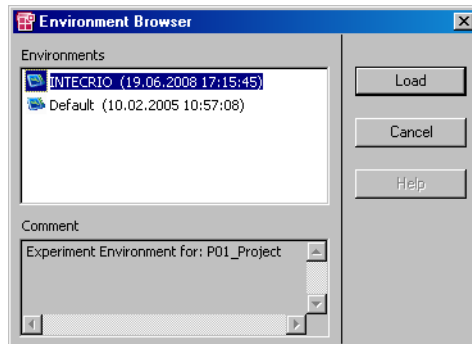
To open the Back-Animation experiment environment:

The connection to the running INTECRIO experiment is established. The "Physical Experiment..." window opens immediately after starting the back-animation (cf. page 56) when you are working without ETAS Network Manager, or after successful hardware selection (cf. page 56) when you are working with the ETAS Network Manager. "INTECRIO Backanimation" indicates the special experiment environment.



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

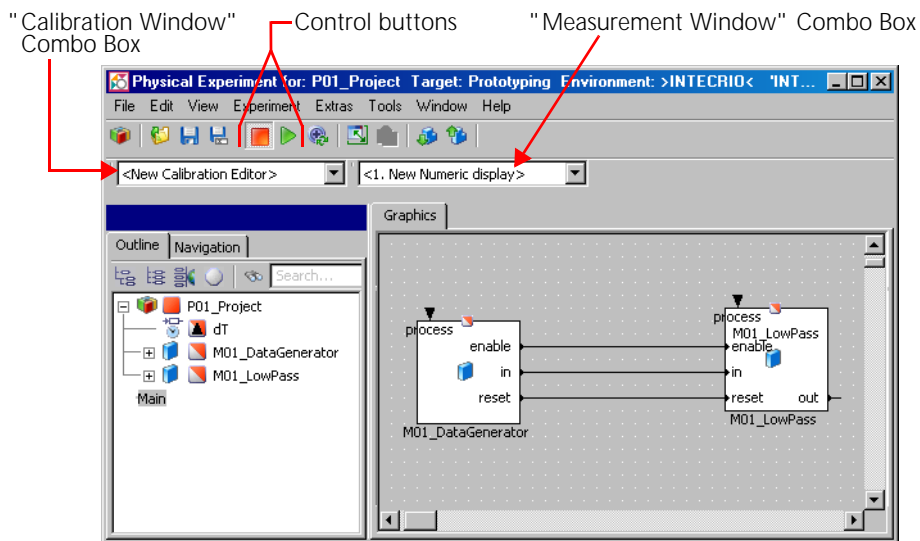
When several environments have been defined, the "Environment Browser" opens.



- In the "Environments" pane, select one entry.
- Click **OK**.

The predefined measurement and calibration windows open.

The User Interface: The user interface of the back-animation experiment environment is very similar to that of the offline experiment. However, the buttons controlling the experiment and the NVRAM cockpit (ES1135, ES910, RTPRO-PC) and the functions in the **Experiment** and **Tools** menus are different from the offline experiment.



Buttons:



1. Exit to Component (ends the experiment and invokes the project editor)
2. Load Environment (loads an experiment environment, i.e. predefined measure and calibration windows with assigned variables)
3. Save Environment (saves the current experiment environment)

4. Save Environment As (saves the current experiment environment under a freely definable name)
5. Stop Measurement (stops measurement, i.e. the data display)
6. Start Measurement (starts measurement, i.e. the data display)
7. Update Dependent Parameters (updates the values of dependent parameters)
8. Expand / Collapse Window (shows/hides the component display)
9. Always on top (keeps the experiment window on top)
10. Navigate down to selected component (shows the selected included component)
11. Navigate up to parent component (shows the parent component)

Experiment Menu:

- *Stop Measurement*
Stops the measurement.
- *Start Measurement*
Starts the measurement.

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 was selected in the code generation options.)
Opens the NVRAM Cockpit.

When the experiment environment for back-animation is opened, the system checks the hardware used by the INTECRIO experiment. If the experiment runs on an ES1135 or ES910 or RTPRO-PC, the NVRAM cockpit (cf. "NVRAM Cockpit" on page 43) is available via the **NVRAM Cockpit** button or the **Tools → NVRAM Cockpit** menu option.

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.

To set up the Back-Animation experiment:

1. Measurement windows



- In the "Measurement Window" combo box, select the type of measurement window that you want to assign an element to.

The available types are listed in brackets, e.g. **<2. Numeric display>**.

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

You can select more than one element.

- Select **Extras** → **Measure**

or

- drag the elements from the "Outline" tab or from the block diagram display to the "Measurement Window" combo box.

A new measurement window opens and the selected elements are added to that window.

The "Measurement Window" combo box also shows the titles of all existing measurement windows. The entries for existing windows are displayed without brackets, e.g. `Numeric Display; 1`.

2. Calibration windows



- In the "Calibration Window" combo box, select the calibration window that you want to assign an element to.

The available windows are listed in brackets, e.g. `<New Calibration Editor>`.

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

- Select **Extras** → **Calibrate**

or

- drag the element from the "Outline" tab pane or the block diagram display and drop it into the "Calibration Window" combo box.

A data editor for the selected element opens.

The "Calibration Window" combo box also shows the titles of all existing calibration windows. The entries for existing windows are displayed without brackets, e.g. `Logical Editor; 2`.

3. Monitors (see also the ASCET online help)

- Right-click an occurrence of an element in the block diagram display.
- Select **Monitor** from the context menu.

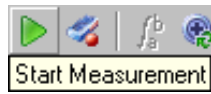
When you start the measurement, the current value of the element is shown in the block diagram display.

- Select the **Monitor** command again to de-assign the monitor.
- Select **View** → **Monitor All** to assign monitors to all elements.
- Select **View** → **Delete Monitors** to delete all the monitors in the diagram.

To start the measurement:

- Select **Experiment** → **Start Measurement**

or



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

The measurement is started and all values set up in the measurement system are displayed in the relevant windows. The displays of the measurement 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 elements 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 the project via the **Navigate ...** buttons ( and ; see also the ASCET online help).

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

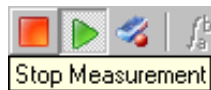
You can write data from the experiment into the ASCET model. This is described in the ASCET online help.

With **Edit → Implementation → Show**, you can look at the implementation of the project with which you are measuring. It does not matter whether the experiment is running or whether it has been stopped.

To end the measurement:

- Select **Experiment → Stop Measurement**

or



- 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.

To end Back-Animation:

- Select **File → Exit**

or



- click the **Exit to Component** button.

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

4.4 ASCET and SCOOP-IX

SCOOP-IX is short for *SCOOP Interface Exchange Language*. This language is based on XML and, therefore, well suited for use in INTECRIO-ASC, ASCET or similar tools.

Exactly one SCOOP-IX file is generated for one ASCET project, regardless of the number of modules and classes in the project.

This section shows some correspondences between ASCET settings and the resulting SCOOP-IX code. More details on SCOOP-IX are given in the INTECRIO user's guide.

General Information:

The following information is part of the SCOOP-IX file:

- The ASCET component in which the equivalent of the respective interface element is embedded
- The type of component (class, module or project; `<pathNode>` block with `kind="asd:module"` option, see page 64)
- The type of equivalent of the interface element (element, message, resource, method, process, task; `<pathNode>` block with `kind="asd:element"` option, see page 64)

Implementation Information:

The settings for Implementation Interval Adaptation are written to the `<saturation>` block in the SCOOP-IX file, see page 64).

The `<saturation>` block contains the options `value`, `resolution` and `assignment`. Depending on the settings in the ASCET implementation editor, the options are set as follows.

- `value` is set to `true` (`false`) if **Limit to maximum bit length** is activated (deactivated).
- `resolution` is set to `automatic`, `keep`, or `reduce`, depending on the selection in the combo box next to **Limit to maximum bit length**.
- `assignment` is set to `true` (`false`) if **Limit Assignments** is activated (deactivated).

The state of the **Zero not Included** option is written to the `value` option of the `<zeroExcluded>` block in SCOOP-IX, see page 64).

- **Zero not Included** activated → `value="true"`
- **Zero not Included** deactivated → `value="false"`

Element Properties:

Information on measurement or calibration is given in the `<usage>` block in SCOOP-IX, see page 64 (measurement) or page 65 (calibration).

- For parameters, the `<usage>` block contains the `calibration` option. Its value is set to `true` (`false`) if the **Calibration** option in the Properties editor is activated (deactivated).
- For variables and messages, the `<usage>` block contains the `measurement` option. Its value is set to `true`.
- For constants and system constants, the `<usage>` block contains the `measurement` option. Its value is set to `false`.

The scope of an element is set in the "Scope" area of the Properties editor. The selected scope is written to the `visibility` option of the `<modelKind>` block in SCOOP-IX, see page 64.

- **Exported** scope → `visibility="public"`
- **Local** scope → `visibility="private"`

Elements with scope **Imported** do not appear in the SCOOP-IX file.

SCOOP-IX Example

An extract of a simple SCOOP-IX file created with ASCET can be found below. The example is used exclusively to show usage of the SCOOP-IX format and possible file contents, it does not claim to be meaningful or correct.

```

...
<module
  xmlns="http://www.etas.de/scoop-ix/1.2"
  xmlns:ix="http://www.etas.de/scoop-ix/1.2"
  xmlns:asd="http://www.etas.de/scoop-ix/1.2/          ↵
    modelDomain/ascet "
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.etas.de/scoop-ix/1.2  ↵
    c:\ETAS\ASCET6.2\Formats\SCOOP-IX\1.2\Schemas\    ↵
      scoop-ix-domain-asd.xsd"
  xmlns:html="http://www.w3.org/1999/xhtml" >
...
<interface>
  <modelLinkBase href="asd://
    {{modelDir}}?INTECRIO-ASC/ASDSimpleModel/" >    ↵
    </modelLinkBase>

  <pathBase path="{{codeDir}}" ></pathBase>
  <headerFile name="asdsmpm.h" ></headerFile>
  <headerFile name="conf.h" ></headerFile>
  <headerFile name="modulem.h" ></headerFile>
  <usage layoutFamily="asd:standardLayout" ></usage>
  &baseTypes-asd;

  <definitions>
    <conversion name="ident">
      <rationalFunction>
        <numerator bx="1" ></numerator>
        <denominator f="1" ></denominator>
      </rationalFunction>
    </conversion>
  </definitions>

  <dataElement interfaceKind="export">
    <dataCInterface identifier=                ↵
      "MODULE_IMPL_ClassObj.Out1->val">
      <type><typeRef name="real64" ></typeRef></type>
      <fileOrigin name="MODULEM.c" ></fileOrigin>
      <initValue value="0.0" ></initValue>
    </dataCInterface>

```

```

<modelOrigin identifier="ASDSimpleModel.      ↵
Module.Out1">
  <name>Out1</name>
  <modelLink href="Module.Out1" ></modelLink>
  <modelLocation>
    <pathNode name="Module" kind="asd:module">
      <pathParameter name="asd:implementation" ↵
        value="Impl" ></pathParameter>
      <pathParameter name="asd:dataSet"      ↵
        value="Data" ></pathParameter>
    </pathNode>
    <pathNode name="Out1" kind="asd:element" > ↵
      </pathNode>
    </modelLocation>
    <modelKind kind="message" visibility="public">
      <flowDirection in="false" out="true" > ↵
        </flowDirection>
    </modelKind>
    <modelType type="continuous" ></modelType>
    <annotation>
      <ix:documentation xmlns=                ↵
        "http://www.w3.org/1999/xhtml"      ↵
        lang="en-US">
        This is output message <i>Out1</i> of ↵
          continuous type.
      </ix:documentation>
    </annotation>
  </modelOrigin>
  <implementation>
    <conversionRef name="ident" ></conversionRef>
    <valueRange min="-2147483648"           ↵
      max="2147483647" ></valueRange>
    <saturation value="true" resolution="reduce" ↵
      assignment="true" ></saturation>
    <zeroExcluded value="false" ></zeroExcluded>
  </implementation>
  <usage measurement="true" virtual="false" ↵
    variant="false" >
    <address kind="pseudo" >
      <BLOB kind="KP_BLOB" device="E_TARGET" > ↵
        <![CDATA[2 1001 1 1001 1]]></BLOB>
    </address>
  </usage>
</dataElement>

<dataElement interfaceKind="export">

```

```

<dataCInterface identifier=
  "ASDSIMPLEMODEL_IMPL_ClassObj.Module->
  myPar->val">
  <type><typeRef name="real64" ></typeRef></type>
  <fileOrigin name="MODULEM.c" >
    </fileOrigin>
  <initValue value="3.2" />
</dataCInterface>
<modelOrigin identifier="ASDSimpleModel.
Module.myPar">
  <name>myPar</name>
  <modelLink href="ASDSimpleModel.Module.myPar"
  > </modelLink>
  <modelLocation>
    <pathNode name="Module" kind="asd:module">
      <pathParameter name=
        "asd:implementation" value="Impl" >
        </pathParameter>
      <pathParameter name=
        "asd:dataSet" value="Data" >
        </pathParameter>
    </pathNode>
    <pathNode name="myPar"
      kind="asd:element" ></pathNode>
  </modelLocation>
  <modelKind kind="parameter"
  visibility="private" ></modelKind>
  <modelType type="continuous" ></modelType>
</modelOrigin>
<implementation>
  <conversionRef name="ident" ></conversionRef>
  <valueRange min="-1.e+037" max="1.e+037" >
    </valueRange>
  <zeroExcluded value="false" ></zeroExcluded>
</implementation>
<usage calibration="true" virtual="false"
variant="false" >
  <address kind="pseudo" >
    <BLOB kind="KP_BLOB" device="E_TARGET"
    ><![CDATA[2 1001 1 1000 1]]></BLOB>
  </address>
</usage>
</dataElement>

<functionElement interfaceKind="export">
  <functionCInterface identifier=
    "MODULE_IMPL_compute">

```

```

<signature>
  <return>
    <type><void /></type>
  </return>
  <void />
</signature>
<fileOrigin name="MODULEM.c" ></fileOrigin>
</functionCInterface>
<modelOrigin identifier="Module.compute">
  <name>compute</name>
  <modelLink href="Module.compute" />
  <modelLocation>
    <pathNode name="Module" kind="asd:module">
      <pathParameter name="asd:implementation" ↵
        value="Impl" ></pathParameter>
      <pathParameter name="asd:dataSet"
        value="Data" ></pathParameter>
    </pathNode>
    <pathNode name="compute" kind="asd:process" >
      </pathNode>
    </modelLocation>
    <modelKind kind="process" ↵
      visibility="public" > </modelKind>
    <runTimeInfo>
      <FPUUsage value="true" ></FPUUsage>
      <TerminateTaskUsage value="false" > ↵
        </TerminateTaskUsage>
      <messageAccess>
        <message identifier= ↵
          "MODULE_IMPL_ClassObj.Out1->val" ↵
          send="true" ></message>
      </messageAccess>
      <resourceAccess ></resourceAccess>
      <constraint>
        <period value="0.01" ></period>
        <execution trigger="timer" priority="0" > ↵
          </execution>
        <scheduling mode="preemptive" > ↵
          <scheduling>
        </scheduling>
      </constraint>
    </runTimeInfo>
  </modelOrigin>
</functionElement>
</interface>
</module>

```

5 Tutorial – Experimenting with INTECRIO

The tutorial describes an INTECRIO experiment, using a supplied example. INTECRIO-ASC V6.2 includes the sample file `Tutorial INTECRIO.*`¹. During the installation, the sample file is stored in the `ASCET6.2\export` subdirectory of your ASCET installation.

The tutorial explains how to transfer ASCET projects to INTECRIO as well as how to use Back-Animation (see page 56) when experimenting with INTECRIO. Creating a project in ASCET or a workspace in INTECRIO is not part of this chapter; all files you need are supplied.

- The export file `Tutorial INTECRIO.*` (* = `exp` or `ax1`) contains the ASCET project with all relevant components.

The ASCET project `P01_Project` consists of a data generator (`M01_DataGenerator` module) which is specified as a state machine (`SM01_DataGenerator`). Use the `PMode` parameter to determine whether the data generator is running as a sawtooth (1) or a triangular signal (2). The generated data represents the input signal for a low-pass filter (`M01_LowPass` module) which is also part of the project.

- The `INTECRIO_Tutorial_Workspace` folder contains an INTECRIO workspace which was prepared for the tutorial. This workspace contains the INTECRIO system projects `SystemProject_ES1130`, `SystemProject_ES1135`, and `SystemProject_ES910`; you use the system project corresponding to your hardware.

The sample file can be imported into a new or an existing database.

Note

This tutorial is written under the assumptions that

- a) an ES1000 system is used,*
- b) a database is used,*
- c) the ETAS Network Manager is **not** used.*

If you are using the ETAS Network Manager, slightly different behavior may result;; see Chapter 2.2 and Chapter 4.

5.1 Preparations

First of all, make the necessary preparations.

To configure the TCP/IP protocol options:

Note

If you are using the ETAS Network Manager, the instruction is obsolete.

To avoid conflicts with a second network card that might be used for the LAN, the following TCP/IP settings should be selected:

- [Disable the DHCP service.](#)
- [Enter the IP address 192.168.40.240.](#)

¹. * = `exp` or `ax1`

- Enter the subnet mask 255 . 255 . 255 . 0.



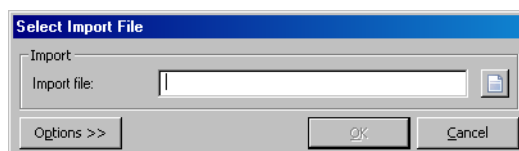
- For the DNS service, use the local settings of your internal network.
- Disable the WINS service.
- Make sure that the "IP Forwarding" option is **not** activated.


To create a new ASCET database:

- In the Component Manager, select the **File** → **New Database** menu option.
The "New Database" window opens.
- Enter the name for the new database.
- Click **OK**.
The database is created and opened.

To import the exercise example:

- In the Component Manager, select **File** → **Import**.
The "Select Import File" window opens.

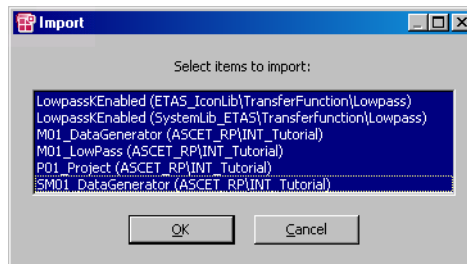


- Use the  button to select the Tutorial INTECRIO.* file from the ASCET6.2\export subdirectory of your ASCET installation.

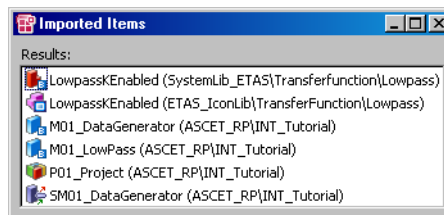
The **OK** button is now available.

You can use the default import options, so that no further action is required.

- Click **OK** to start the import.
- Make sure all components have been selected for the import in the "Import" window..



- Click **OK**.
The items are imported. The "Imported Items" window shows a list of the imported items.



- Close the "Imported Items" window.

To select a target:

- Open the project P01_Project in the project editor.
- Click on **Project Properties** to open the "Project Properties" window.
- In the "Build" node, select the target **Prototyping** and the compiler **GNU-C V3.4.4 (PowerPC)**.



The defined operating system settings for this target are loaded.

To copy the INTECRIO workspace:

The prepared INTECRIO workspace is located in the `ETAS\ASCET6.2\export` directory of your ASCET installation.

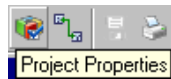
- Copy the `INTECRIO_Tutorial_workspace` directory to your hard disk, e.g. to
`ETASData\INTECRIO4.0\
INTECRIO_Tutorial_Workspace.`

5.2 Transferring the Project

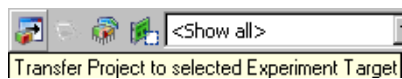
The next step is to transfer the project to INTECRIO.

To transfer the project to INTECRIO:

- Open the project `P01_Project`.
- Click the **Project Properties** button to open the "Project Properties" window.
- Make sure that the target `Prototyping` and the GNU compiler are selected in the "Build" node, and close the "Project Properties" window.

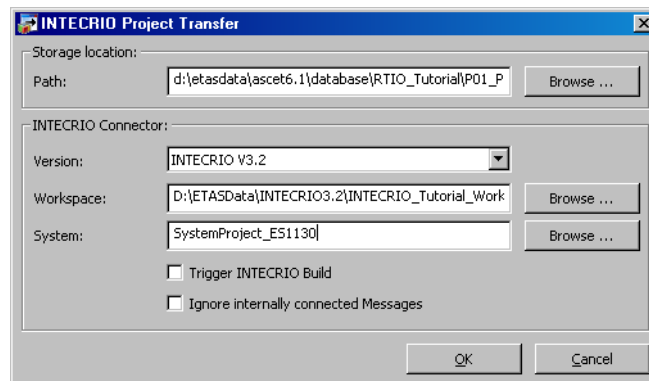


- `INTECRIO` is preselected in the "Experiment Target" combo box; the buttons **Transfer Project to selected Experiment Target** and **Reconnect to Experiment of selected Experiment Target** are now available.



- Click the **Transfer Project to selected Experiment Target** button.

The "INTECRIO Project Transfer" window opens.



- In the "Path" field, enter a path for the generated files.
- In the "Version" combo box, select the INTECRIO version you will use.
- Use the **Browse** button next to the "Workspace" field to enter the supplied workspace.

- Use the **Browse** button next to the "System" field to specify the suitable system project for your hardware.

If INTECRIO is not yet running, it is started now.

- Click **OK** to start transfer.

If the folder for the generated files is not empty, the following message opens:

The folder "<folder path and name>" already exists! If you continue, existing files may be overwritten. Do you want to proceed anyway?

- Click **OK** to proceed.

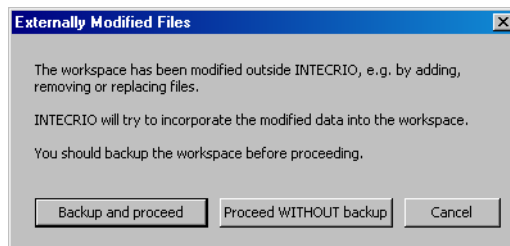
Or

- Click **Cancel** to abort the transfer.

The code necessary for working with INTECRIO is generated and stored in the specified directory.

The ASCET project is imported into INTECRIO and stored as a module under the name `P01_Project`. It is automatically added to the selected system project.

Since you copied the workspace, the following error message can occur:



Click **Proceed without Backup** to continue.

5.3 Experimenting in INTECRIO

Now configure the operating system in INTECRIO, start the Build process and finally the INTECRIO experiment.

To configure the INTECRIO operating system:

- Change to the INTECRIO window.
- In the Systems folder, select the system project you are using, and select **Set As Active Project** from its context menu.
- Select **System** → **OS Configuration**.

The OSC operating system editor opens. As the example is very easy, you can use the automatic configuration.

- Select **System** → **OS Auto mapping**.

The `auto_10msTask` task is created in `UserApp-Mode` operating mode. The two processes of the ASCET project are assigned to this task.

You do not need to make any further settings.

To start the INTECRIO Build process:

- Select **Integration** → **Build** from the INTECRIO window

or

- if you are not starting the Build process for the first time, select **Integration** → **Rebuild**.

The Build process is started. The "Log Window" box at the bottom of the INTECRIO window indicates progress.

The following message is displayed in the last lines after a successful Build process:

```
Action succeeded
```

```
The active system project has been set into the "Build" mode.
```

To start an INTECRIO experiment:

1. Opening an experiment environment

- In the INTECRIO window, select **Experiment** → **Open Experiment**.

The experiment environment opens in its own window. The experiment is loaded into the experiment environment.

2. Starting an experiment

- In the INTECRIO experiment environment, select **Experiment** → **Download**.

The executable file (the prototype) is loaded to the hardware.

- Select **Experiment** → **Start OS**.

The simulation is started.

- Select **Experiment** → **Start Measurement**.

The measurement is started.

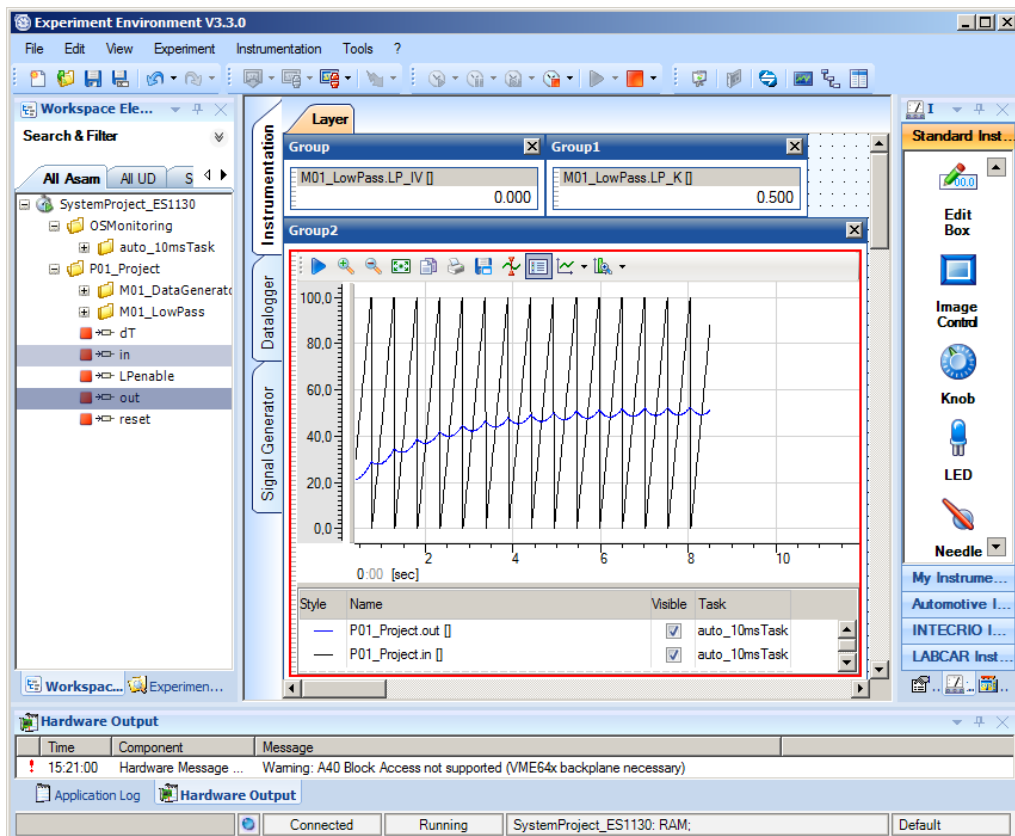
To use Back-Animation, you do not have to open any measure and calibration windows in INTECRIO. But as Back-Animation with ASCET does not provide an oscilloscope, the INTECRIO experiment environment contains a predefined oscilloscope. The experiment environment also contains two calibration instruments. If these instruments do not open automatically, open the prepared experiment manually.

To open the experiment environment:

- In the INTECRIO experiment environment, select **File** → **Open Experiment**.

- Confirm the security question.
A file selection window opens.
- Open the `INTECRIO_Tutorial.exe` file from the `EE\Experiments\INTECRIO_Tutorial` subdirectory of your INTECRIO workspace.
As you have already started the simulation, values are displayed immediately.

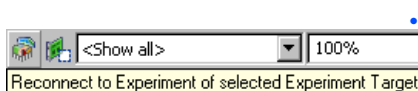
Your INTECRIO experiment environment then looks like this:



5.4 Using Back-Animation

Start Back-Animation from ASCET. The experiment has to continue running in INTECRIO.

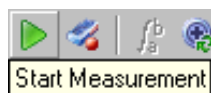
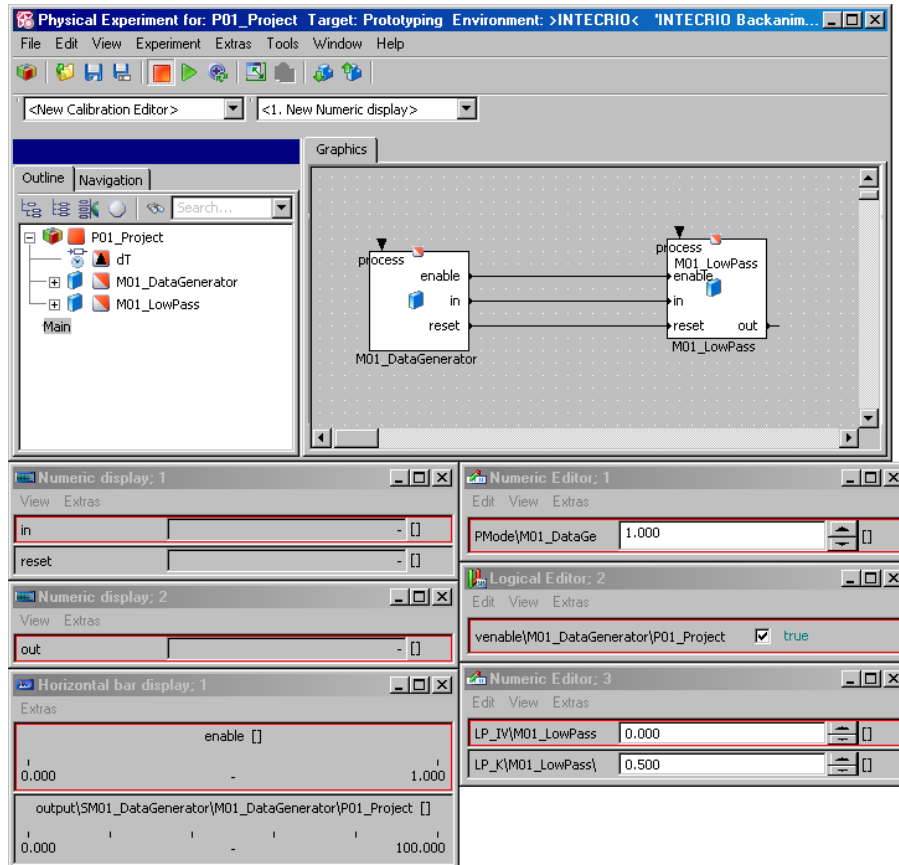
To start Back-Animation:



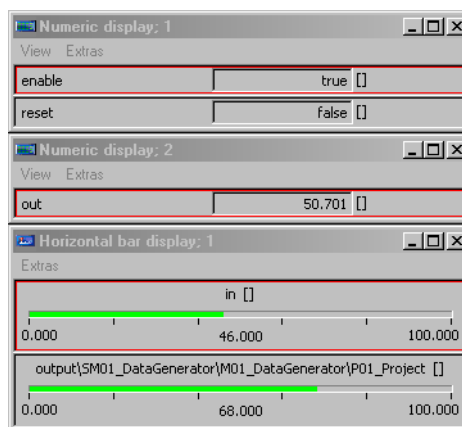
- Click the **Reconnect to Experiment of Selected Experiment Target** button in the ASCET project editor.

The connection is established to the running INTECRIO experiment. The "Physical Experiment ..." window opens.

- In the "Environment Browser" window, select INTECRIO as environment.
The predefined arrangement of measurement and calibration windows opens.



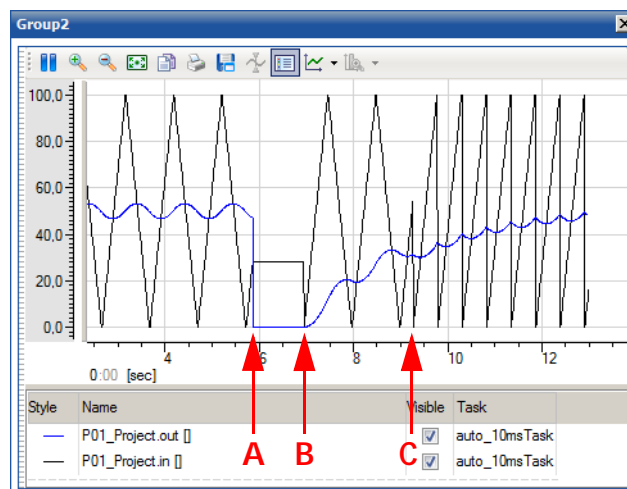
- Click the **Start Measurement** button.
Measuring is started in the ASCET experiment; values are displayed in the measure windows.



You can now calibrate values either in the ASCET experiment or in the INTECRIO experiment. The modified values are transferred to the INTECRIO experiment and displayed and used there.

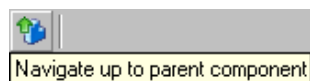
To calibrate values:

- In the ASCET experiment, enter a value for the variable `LP_IV` in the "Numeric Editor; 3" window. The value in the left-hand calibration instrument in the INTECRIO experiment ("Group1") is updated.
- In the INTECRIO experiment, enter another value for the variable `LP_IV` in the "Group1" window. The value in the "Numeric Editor; 3" window of the ASCET experiment is updated.
- In the "Logical Editor; 2" window, set the variable parameter to `false`. The value of the signal generator stays at the last value; the low-pass filter is set to the initialization value `LP_IV` (A in the screenshot).
- Set `enable` back to `true` and then `PMODE` to 2 to select the other signal generator. The display in the INTECRIO oscilloscope changes accordingly (B and C in the screenshot).



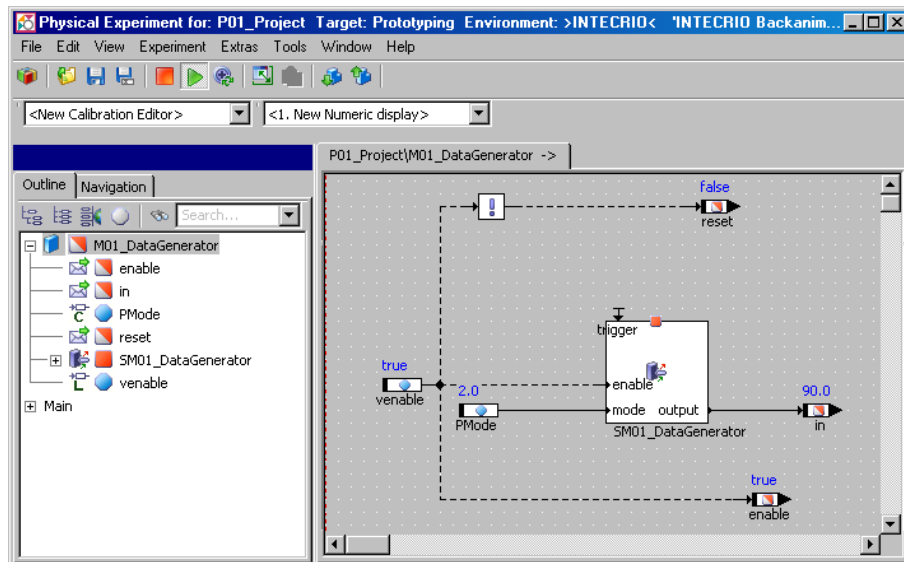
To view the ASCET components:

- In the ASCET window "Physical Experiment ...", "Graphics" tab, double-click a component to view it in detail. The component is displayed in the "Physical Experiment ..." window. You can navigate through the entire hierarchy in this way; the **Navigate up to parent component** button or double-clicking the empty space gets you back to the next highest level.



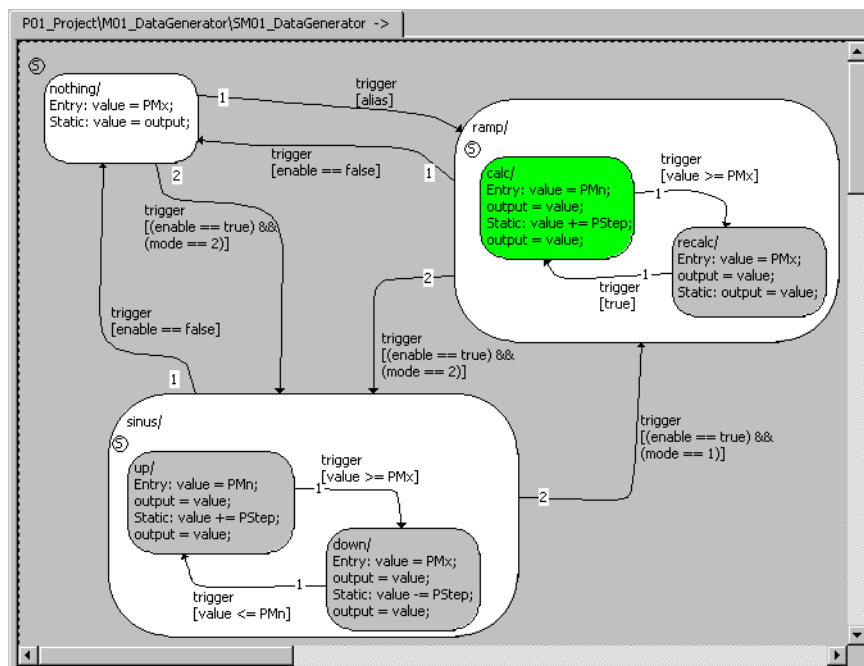
- Select **View** → **Monitor All**.

The current values of the elements are shown above the elements.



- Navigate through the model specification to the state machine **SM01_DataGenerator**.
- Right-click one of the states and select **Animate States** from the context menu.

The current state is shown in color in the state diagram.



6 Appendix A: Compiler Switches and API Functions

This annex contains remarks to target-specific external C code (chapter 6.1 "Compiler Switches for External C Code"), as well as the API functions INTECRIO-ASC provides for the ES113x experimental target. These functions define the interfaces between INTECRIO-ASC and the following applications:

- ERCOS^{EK} (Chapter 6.2)
- NVRAM (Chapter 6.3)
- Watchdog (Chapter 6.4)
- LEDs (Chapter 6.5)
- Miscellaneous (Chapter 6.6)

6.1 Compiler Switches for External C Code

It is sometimes necessary to bracket parts of external C code in target-specific compiler switches. For that purpose, INTECRIO-ASC V6.2 provides the following switches:

- ES910
- ES1130
- ES1135
- ES113x (for ES1130 and ES1135)

Note

If the Prototyping target is selected, the (test) compilation is performed with the ES1135 switch.

The syntax is as follows:

```
#ifdef ES1135
...
/* ES1135-specific code */
...
#endif
```

6.2 API Functions (ERCOS^{EK})

This chapter gives a detailed description of all existing API-functions (**A**pplication **P**rogramming **I**nterface). These service routines define the interface between the application and ERCOS^{EK}.

Each section deals with a group of service routines that are functionally related to one another. The description structure of each service routine is as follows:

exampleRoutine

Function	A short description of the service's functionality.
Syntax	The syntax is specified here in the form of a C function prototype. The C types used are described in the following chapter.

Description	This section contains a detailed description of the service routine, a description of the parameters as well as further details and notes that the user should be aware of or take into consideration when using the service routine.
Return code	Type and value range of the return code (if available) and its significance are specified here.
Example	The Example demonstrates a typical usage of the described function.
See also	List of related functions.
Hint	Some of the function descriptions include a hint providing additional useful information.

The following list provides a short overview of all existing ERCOS^{EK} commands supported by INTECRIO-ASC for the experimental target ES113x. More detailed information (syntax, examples, etc.) can be found in the subsequent chapters.

Command	Function	Page
Application Modes		
DeclareAppMode	Serves as an external declaration of an application mode.	79
SetNextAppMode	Switches to the specified application mode after processing all active tasks.	79
Tasks		
DeclareTask	Serves as an external declaration of a task.	80
ActivateTask	Activates a SW task.	80
System Time		
GetSystemTime	Gets the current system time.	81
GetSystemTimeLow	Gets the low-order part of the current system time.	81
GetSystemTimeHigh	Gets the high-order part of the current system time.	82
Interrupt Handling		
EnableAllInterrupts	Globally enables all interrupts.	82
DisableAllInterrupts	Globally disables all interrupts.	82
dT Query		
GetDeltaT	Returns the value of dT.	83

6.2.1 Application Modes

The concept of application modes allows the efficient management of different processing states in the application software. An application mode is defined by a set of tasks which are active in this mode and one or more optional timetables. Application modes for an engine control unit can be, for example: normal operation (control of the technical process), auto-diagnostics, flash EPROM programming. Only one application mode can be active at a time.

An application mode consists of two phases: the first phase is the initialization phase. This is where the initialization routines of the application are processed. Interrupts are disabled. After initialization, the interrupts are enabled and the execution phase begins. Here the activated tasks of the application are processed according to their priorities (scheduled).

DeclareAppMode

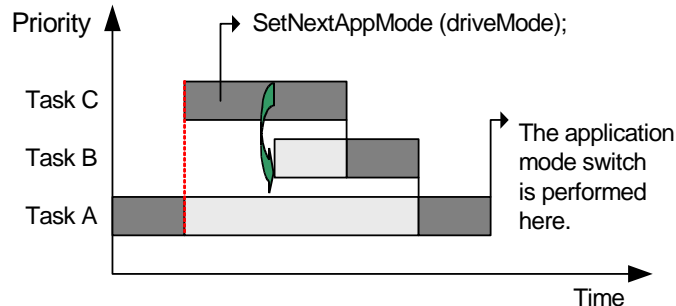
Function	Serves as an external declaration of an application mode.
Syntax	<pre>#define DeclareAppMode(AppID) extern AppModeType AppID</pre>
Description	If an application mode switch is performed within a module, but the application mode descriptor is defined in another module, the usage of the application mode descriptor must be disclosed by DeclareAppMode() . The function and use of this service are similar to that of the external declaration of variables.
Example	<pre>extern uint excCtr; extern uint randx; DeclareAppMode(idleMode);</pre>
See also	DeclareTask

SetNextAppMode

Function	Switches to the specified application mode after processing all active tasks.
Syntax	<pre>StatusType SetNextAppMode(AppModeType appMode)</pre>
Description	SetNextAppMode() requests a change to the application mode referenced by pointer appMode . The operating system executes the change as soon as no further task is running, i.e. when the operating system is in the idle state. However, subsequent task activations via ChainTask() or RestartTask() (not supported for Rapid Prototyping use case) will be ignored. In case hardware tasks are initialized during startup (initialization phase), they will be reinitialized for the next application mode.

Return code `E_OK` Request successfully processed.

Example `SetNextAppMode(driveMode);`



See also -

6.2.2 Tasks

There are two types of tasks in ERCOS^{EK}: firstly software tasks (SW tasks) which are activated by **ActivateTask()**; the processing is coordinated by the ERCOS^{EK} scheduler and secondly hardware tasks (HW tasks) which are activated by an interrupt. In this case scheduling is carried out by the interrupt control logic of the processor, i.e. by the hardware.

DeclareTask

Function Serves as an external declaration of a task.

Syntax `#define DeclareTask(TaskID)`
`extern TaskType TaskID`

Description If a task is used by a module, but is defined in another module, its usage must be disclosed by **DeclareTask()**. The function and use of this service are similar to that of the external declaration of variables.

Example `extern uint excCtr;`
`extern uint randx;`
DeclareTask(synchroSeq);

See also `DeclareAppMode`

ActivateTask

Function Activates a SW task.

Syntax `StatusType ActivateTask(TaskType task)`

Description **ActivateTask()** requires the operating system to process the SW task specified by **task**. If this task activation is successful (cf. return code), the processing of the task is planned according to its priority by the ERCOS^{EK} scheduler. If several activations of a task are allowed (according to the BCC2 definition) and the current number of activations of a task is > 1 , this task is temporarily stored in the FIFO buffer. If **ActivateTask()** cannot be executed successfully, the system switches to the user-specific error function.

Return code `E_OK` Activation successful.

E_OS_LIMIT No activation, as maximum number of task activations for the task specified has already been reached or because the maximum number of tasks in the task FIFO buffer at the specified priority level has already been reached.

Example **ActivateTask**(synchroSeq);

See also –

6.2.3 System Time

A discrete system time is the time base of ER^{ECOS}. For those targets which do not offer a hardware-based system time, the system time is set to 0 with the start of the operating system. The system time, which is normally counted with a width of two machine words, is used as the reference time for alarm services and the ER^{ECOS} timetable. The time until an overflow of the system time occurs depends on CPU and the frequency of the hardware timer used. The system time is not interrupted or reset by an application mode change.

The system time is counted in ticks of the underlying timer register. The macro **SYSTEM_TICK_DURATION** returns the duration of such a tick in nanoseconds.

GetSystemTime

Function Gets the current system time.

Syntax `TimeType GetSystemTime(void)`

Description **GetSystemTime()** returns the system time in ticks. The width is system dependent (32 bit on 16-bit wide and 64 bit on 32-bit wide systems).

Return code Current system time.

Example `TimeType now;
now = GetSystemTime();`

See also `GetSystemTimeLow`, `GetSystemTimeHigh`

GetSystemTimeLow

Function Gets the low-order part of the current system time.

Syntax `TickType GetSystemTimeLow(void)`

Description **GetSystemTimeLow()** returns the low-order part of the current system time in ticks. These are the lower 16 bit for an ER^{ECOS} implementation with a 32 bit wide system time; for an implementation with a 64 bit wide system time, the lower 32 bit.

Return code Low-order part of the current system time.

Example `TickType lowPartOfNow;
lowPartOfNow = GetSystemTimeLow();`

See also `GetSystemTime`, `GetSystemTimeHigh`

GetSystemTimeHigh

Function	Gets the high-order part of the current system time.
Syntax	<code>TickType GetSystemTimeHigh(void)</code>
Description	GetSystemTimeHigh() returns the high-order part of the current system time in ticks. These are the upper 16 bit for an ERCOS ^{EK} implementation on a 32-bit wide system time; for an implementation on a 64-bit wide system time, the upper 32 bit.
Return code	High-order part of the current system time.
Example	<pre>TickType highPartOfNow; highPartOfNow = GetSystemTimeHigh();</pre>
See also	<code>GetSystemTime</code> , <code>GetSystemTimeLow</code>

6.2.4 Interrupt Handling

ERCOS^{EK} provides a routine to save and restore context relevant data in the frame of an interrupt service routine. Furthermore, the certain valid interrupt descriptor can be accessed by an ERCOS^{EK} API-function.

EnableAllInterrupts

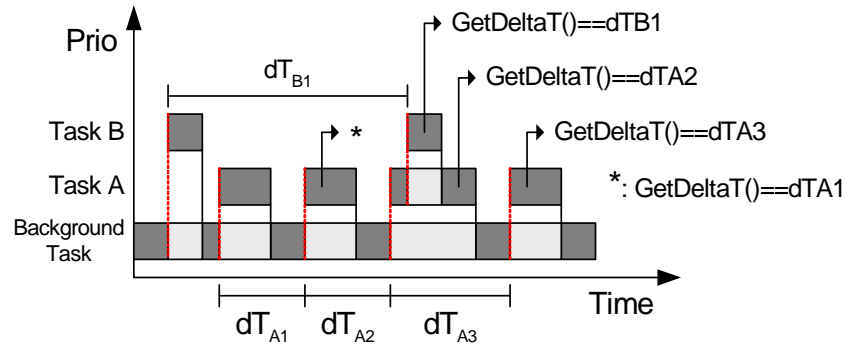
Function	Enables all interrupts globally.
Syntax	<code>void EnableAllInterrupts(void)</code>
Description	EnableAllInterrupts() enables the interrupts for the controller-core globally without manipulating interrupt masks. If multiple calls of DisableAllInterrupts() preceded the interrupts are only enabled if the corresponding number of EnableAllInterrupts() calls have been reached. Hence, a safe realization of nested interrupt disabling is supported.
Return code	None
See also	<code>DisableAllInterrupts</code>

DisableAllInterrupts

Function	Disables all interrupts globally.
Syntax	<code>void DisableAllInterrupts(void)</code>
Description	DisableAllInterrupts() disables all interrupts globally and stores the state of nested calls.
Return code	None
See also	<code>EnableAllInterrupts</code>

6.2.5 dT Query

ERCOS^{EK} provides a service routine for querying the time elapsed between the last start of the currently running task and the start of the currently running task (see figure below). The time returned always concerns the task from which the service was called.



The dT returned by GetDeltaT() is very useful for mathematical calculations e.g. an integration:

$$F(x) = \int_0^x f(T) dT$$

GetDeltaT	
Function	Returns the value of dT.
Syntax	TickType GetDeltaT(void)
Description	<p>GetDeltaT() returns the time expired between two subsequent task executions.</p> <p>Note: If this time exceeds half the width of the hardware timer, the return value can not be relied on.</p> <p>This function is only supported in ERCOS^{EK} debug mode. See chapter "Debug information within the task monitor" in the ERCOS^{EK} manual for detailed information about debugging an application based on ERCOS^{EK}.</p>
Return code	Value of dT in ticks.
Example	<pre>TickType deltaT; deltaT = GetDeltaT();</pre>
See also	-

6.3 API Functions (NVRAM)

The default behavior of the NVRAM manager described in Chapter 3.5 can be altered from within an ASCET model (C code component) via the following interfaces:

nvrRamInitModelVars

Function	Initializes the NV variables.	
Syntax	<code>uint32 nvrRamInitModelVars(void)</code>	
Description	This function initializes the NV variables with the content of the NVRAM if this content is valid and matching. The initialization may be triggered only once (via C code, L1 or automatic flag) and only before any update of the NVRAM occurred.	
Return Value	<code>EC_NVRAM_SUCCESS</code>	Success
	<code>EC_NVRAM_NO_NV_VARIABLES</code>	No NV variables inside the model
	<code>EC_NVRAM_INADMISSIBLE_USE</code>	Function has been already called
	<code>EC_NVRAM_INTERNAL_ERROR</code>	An internal error occurred
	<code>EC_NVRAM_NO_MATCH</code>	The NVRAM content does not match the current model
Example	–	
See also	<code>nvrRamCheckForInitializedVars</code>	

nvrRamSetUpdateInterval

Function	Sets the automatic NVRAM update interval.	
Syntax	<code>uint32 nvrRamSetUpdateInterval(uint32 interval_sec)</code>	
Description	Sets the automatic NVRAM update interval. This is the desired time between two updates. If system load is high, the actual time interval might be larger (depends significantly from the requested consistency level). If the actual update interval exceeds the requested interval for 10 times, a warning is issued inside the experiment environment.	
Return Value	<code>EC_NVRAM_SUCCESS</code>	Success
	<code>EC_NVRAM_INVALID_ARG</code>	<code>Interval_sec > 30</code>
Parameter	<code>interval_sec</code>	Update interval in seconds. Must be a value between 0 and 30 (0: no periodical update).
Example	–	
See also	<code>nvrRamGetUpdateInterval</code>	

nvrAmGetUpdateInterval

Function	Gets the automatic NVRAM update interval.	
Syntax	<code>uint32 nvrAmGetUpdateInterval(void)</code>	
Description	Gets the automatic NVRAM update interval. This is the desired time between two updates. If system load is high, the actual time interval might be larger (depends significantly from the requested consistency level). If the actual update interval exceeds the requested interval for 10 times, a warning is issued inside the experiment environment.	
Return Value	<code>interval_sec</code>	Update interval in seconds.
Example	–	
See also	<code>nvrAmSetUpdateInterval</code>	

nvrAmSetConsistencyLevel

Function	Sets the level of NV variable data consistency.	
Syntax	<code>uint32 nvrAmSetConsistencyLevel(T_consistencyLevel level)</code>	
Description	Sets the level of NV variable data consistency. <i>No consistency:</i> NVRAM update is done without respect to consistency inside NV variables and between individual NV variables. <i>Low level consistency:</i> data consistency within NV variables (scalars, vectors and matrices but not characteristics) is guaranteed. <i>High level consistency:</i> all NV variables are updated without interruption by the model, out of the idle task.	
Return Value	<code>EC_NVRAM_SUCCESS</code>	Success
	<code>EC_NVRAM_INVALID_ARG</code>	Invalid level argument
Parameter	<code>level</code>	<code>NVRAM_NO_CONSISTENCY</code> <code>NVRAM_LOW_CONSISTENCY</code> <code>NVRAM_HIGH_CONSISTENCY</code>
Example	–	
See also	<code>nvrAmGetConsistencyLevel</code>	

nvrAmGetConsistencyLevel

Function	Gets the level of NV variable data consistency.	
Syntax	T_consistencyLevel nvrAmGetConsistencyLevel(void)	
Description	Gets the level of NV variable data consistency. <i>No consistency</i> : NVRAM update is done without respect to consistency inside NV variables and between individual NV variables. <i>Low level consistency</i> : data consistency within NV variables (scalars, vectors and matrices but not characteristics) is guaranteed. <i>High level consistency</i> : all NV variables are updated without interruption by the model, out of the idle task.	
Return Value	NVRAM_NO_CONSISTENCY	No consistency
	NVRAM_LOW_CONSISTENCY	Low level consistency
	NVRAM_HIGH_CONSISTENCY	High level consistency
Example	–	
See also	nvrAmSetConsistencyLevel	

nvrAmEnableAutoUpdate

Function	Enables automatic update of the NVRAM content.	
Syntax	uint32 nvrAmEnableAutoUpdate(void)	
Description	Enables automatic update of the NVRAM content. This comprises periodical update as well as updates initiated by the Exit Task.	
Return Value	EC_NVRAM_SUCCESS	Success
Example	–	
See also	nvrAmDisableAutoUpdate nvrAmCheckForAutoUpdate	

nvrAmDisableAutoUpdate

Function	Disables automatic update of the NVRAM content.	
Syntax	uint32 nvrAmDisableAutoUpdate(void)	
Description	Disables automatic update of the NVRAM content. This comprises periodical update as well as updates initiated by the Exit Task.	
Return Value	EC_NVRAM_SUCCESS	Success
Example	–	
See also	nvrAmEnableAutoUpdate, nvrAmCheckForAutoUpdate	

nvrAmCheckForAutoUpdate

Function	This function checks if auto update mode is enabled.	
Syntax	<code>uint8 nvrAmCheckForAutoUpdate(void)</code>	
Return Value	<code>true</code>	Auto update mode is enabled
	<code>false</code>	Auto update mode is disabled
Example	-	
See also	<code>nvrAmEnableAutoUpdate</code> <code>nvrAmDisableAutoUpdate</code>	

nvrAmManualUpdateExit

Function	Ensures a final update of the NVRAM content.	
Syntax	<code>void nvrAmManualUpdateExit (void)</code>	
Description	This function should be placed inside the Exit Task after the last user process, to ensure a final update of the NVRAM content when the user application mode is left (Stop ERCOS Button). Error messages are posted inside the experiment environment if an error occurs.	
Example	-	
See also	<code>nvrAmManualUpdateBackground</code> <code>nvrAmManualUpdateBlocked</code>	

nvrAmManualUpdateBackground

Function	Starts a manual update of the NVRAM content.	
Syntax	<code>uint32 nvrAmManualUpdateBackground(void)</code>	
Description	This function starts a manual update of the NVRAM content. Manual update has precedence over the automatic periodical update. Thus, a potentially running periodical update is aborted. But if cyclic update is on the way (the Idle task is interrupted by a preemptive task with the call of this function), start of manual update is impossible. This function returns immediately, because the update is running in the background (Idle Task). The completion of this process can be tested via the function <code>nvrAmCheckRunningUpdate()</code> . Note: It is not recommended to use this function when automatic update is enabled.	
Return Value	<code>EC_NVRAM_SUCCESS</code>	Success
	<code>EC_NVRAM_NO_NV_VARIABLES</code>	No NV variables in model
	<code>EC_NVRAM_FATAL_ERROR</code>	Fatal error occurred before
	<code>EC_NVRAM_OVERFLOW</code>	Overflow of NVRAM. Reduce Number / Size of NV variables.

nvrAmManualUpdateBackground

EC_NVRAM_UPDATE_RUNNING	Other Update process (manual or cyclic) is currently running. Start of manual update failed.
-------------------------	--

Example –

See also `nvrAmManualUpdateBlocked`,
`nvrAmManualUpdateExit`

nvrAmManualUpdateBlocked

Function Starts a manual update of the NVRAM content (blocking on the current priority).

Syntax `uint32 nvrAmManualUpdateBlocked`
(`uint32 timeoutUs`)

Description This function starts a manual update of the NVRAM content. Manual update has precedence over the automatic periodical update. Thus, a potentially running periodical update is aborted. But if cyclic update is on the way (the Idle task is interrupted by a preemptive task with the call of this function), start of manual update is impossible. This function blocks on the current priority until all NV variable contents have been written to the local buffer or until a time-out occurred. After the function has returned, the update process (writing from local buffer into the NVRAM) is continued in the Idle task (even if a time-out occurred). The completion of the update process can be tested via the function `nvrAmCheckRunningUpdate()`.
Because interrupts are not suspended during this process, a preemptive task with higher priority might interrupt the update process. This could lead to data inconsistencies if this task modifies any NV variable contents.

Note: It is **not** recommended to use this function when automatic update is enabled.

Return Value	EC_NVRAM_SUCCESS	Success
	EC_NVRAM_NO_NV_VARIABLES	No NV variables in model
	EC_NVRAM_FATAL_ERROR	Fatal error occurred before
	EC_NVRAM_OVERFLOW	Overflow of NVRAM. Reduce Number / Size of NV variables.
	EC_NVRAM_UPDATE_RUNNING	Other Update process (manual or cyclic) is currently running. Start of manual update failed.

nvrAmManualUpdateBlocked

Parameter	timeoutUs	Time-out period in μ s
Example	–	
See also	nvrAmManualUpdateBlocked, nvrAmManualUpdateExit, nvrAmCheckRunningUpdate	

nvrAmCheckRunningUpdate

Function	Checks if an manual NVRAM update started.	
Syntax	uint8 nvrAmCheckRunningUpdate(void)	
Description	This function checks if an manual NVRAM update started by nvrAmStartManualUpdateBackground or nvrAmStartManualUpdateBlocked is still running in the background.	
Return Value	false	Update is finished or has not been started successfully
	true	Update is still running
Example	–	
See also	nvrAmManualUpdateBackground nvrAmManualUpdateBlocked	

nvrAmCheckForInitializedVars

Function	Checks if the NV variables have been initialized.	
Syntax	uint8 nvrAmCheckForInitializedVars(void)	
Description	This function checks if the NV variables inside the model have been initialized with the NVRAM content. This might be triggered by automatic update via the experiment environment or initialization via C code API.	
Return Value	true	NV variables have been initialized with the NVRAM content
	false	NV variables have not been initialized with the NVRAM content but with their default values.
Example	–	
See also	nvrAmInitModelVars	

nvrAmGetUpdateAgeMs

Function	Returns the elapsed time since the last finish of an update.	
Syntax	uint32 nvrAmGetUpdateAgeMs(void)	
Return Value	updateAge	Time in milliseconds

nvrAmGetUpdateAgeMs

Description	This function returns the elapsed time since the last finish of an update (manual or automatic update).
Example	–
See also	–

nvrAmClear

Function	Erases the NVRAM contents.
Syntax	<code>uint32 nvrAmClear(void)</code>
Return Value	<code>EC_NVRAM_SUCCESS</code> Success
Description	This function erases the NVRAM contents. The memory is initialized with zeros.
Example	–
See also	–

6.4 API Functions (Watchdog)

The ES1135 Simulation Controller has a hardware watchdog. the watchdog functionality is summarized in Chapter 3.4.1. The following interfaces are provided by the firmware.

6.4.1 Watchdog Configuration

wdSetSafetyMode

Function	Sets the Safety Mode.	
Syntax	<code>uint32 wdSetSafetyMode (uint32 event, uint32 period)</code>	
Description	<p>This function switches from the pre-operational mode or the RSEF mode to the safety critical mode. This cannot be undone afterwards except by switching power off.</p> <p>The parameter event selects the action which is to be done when the watchdog expires.</p> <p><code>WD_EVENT_DISABLE</code> disables the watchdog. <code>WD_EVENT_PPC750_RESET</code> resets the IBM 750GX simulation processor. <code>WD_EVENT_PPC750_INT</code> triggers an interrupt to the simulation processor.</p> <p>The parameter period (time period after that the watchdog expires) can be configured in the range from 0.25 ms up to 4096 ms.</p>	
Return Value	<code>EC_CFW_SUCCESS</code>	Success
	<code>EC_CFW_WD_SAFETY_MODE</code>	Watchdog is already in safety mode
	<code>EC_CFW_INVALID_ARG</code>	Invalid event or period value
Parameter	<code>event</code>	<code>WD_EVENT_DISABLE</code> <code>WD_EVENT_PPC750_RESET</code> <code>WD_EVENT_PPC750_INT</code>
	<code>period</code>	<code>WD_PERIOD_4096MS</code> <code>WD_PERIOD_1024MS</code> <code>WD_PERIOD_256MS</code> <code>WD_PERIOD_64MS</code> <code>WD_PERIOD_16MS</code> <code>WD_PERIOD_4MS</code> <code>WD_PERIOD_1MS</code> <code>WD_PERIOD_0_25MS</code>
Example	<pre>uint32 period; uint32 event; uint32 retVal; event = WD_EVENT_DISABLE; period = WD_PERIOD_4096MS; retVal = wdSetSafetyMode(event, period);</pre>	
See also	<code>wdSetPeriod</code> , <code>wdSetEvent</code>	

wdSetReducedSafetyMode

Function	Sets the Reduced Safety Enhanced Function Mode.	
Syntax	<code>uint32 wdSetReducedSafetyMode(void)</code>	
Description	This function switches from the pre-operational mode to the reduced safety enhanced function mode (RSEF).	
	Note: This function is already called inside the boot loader. Thus, this API function has no impact for INTECRIO-ASC use, because the model starts with the watchdog in RSEF mode. The loader disables also the watchdog events. Afterwards, watchdog period and event can be modified via <code>wdSetPeriod</code> and <code>wdSetEvent</code> .	
Return Value	<code>EC_CFW_SUCCESS</code>	Success
	<code>EC_CFW_WD_SAFETY_MODE</code>	Watchdog is in safety mode. This cannot be undone.
	<code>EC_CFW_WD_RSEF_MODE</code>	Watchdog is already in RSEF mode.
Example	-	
See also	<code>wdSetPeriod</code> , <code>wdSetEvent</code>	

wdSetPeriod

Function	Sets the Watchdog Period.	
Syntax	<code>uint32 wdSetPeriod(uint32 period)</code>	
Description	This function switches the watchdog period (time period after that the watchdog expires) which can be configured in the range from 0.25 ms up to 4096 ms.	
Return Value	<code>EC_CFW_SUCCESS</code>	Success
	<code>EC_CFW_WD_SAFETY_MODE</code>	Watchdog is in safety mode. No period modification possible.
	<code>EC_CFW_INVALID_ARG</code>	Invalid period value
	<code>EC_CFW_WD_PRE_OP_MODE</code>	Watchdog is in pre-operational mode. Switch first to RSEF mode.

wdSetPeriod

Parameter	period	WD_PERIOD_4096MS WD_PERIOD_1024MS WD_PERIOD_256MS WD_PERIOD_64MS WD_PERIOD_16MS WD_PERIOD_4MS WD_PERIOD_1MS WD_PERIOD_0_25MS
-----------	--------	---

Example `uint32 period;`
 `uint32 retVal;`
 `period = WD_PERIOD_4096MS;`
 `retVal = wdSetPeriod(period);`

See also `wdSetSafetyMode, wdSetEvent`

wdSetEvent

Function	Sets the event to be handled, if the watchdog expires.	
Syntax	<code>uint32 wdSetEvent(uint32 event)</code>	
Description	The function selects the action which should be done when the watchdog expires. WD_EVENT_DISABLE disables the watchdog. WD_EVENT_PPC750_RESET resets the IBM 750GX simulation processor. WD_EVENT_PPC750_INT triggers an interrupt to the simulation processor.	
Return Value	EC_CFW_SUCCESS	Success
	EC_CFW_WD_SAFETY_MODE	Watchdog is in safety mode. No event modification possible.
	EC_CFW_INVALID_ARG	Invalid event value
	EC_CFW_WD_PRE_OP_MODE	Watchdog is in pre-operational mode. Switch first to RSEF mode.
Parameter	event	WD_EVENT_DISABLE WD_EVENT_PPC750_RESET WD_EVENT_PPC750_INT

Example `uint32 event;`
 `uint32 retVal;`
 `event = WD_EVENT_DISABLE;`
 `retVal = wdSetEvent(event);`

See also `wdSetSafetyMode, wdSetPeriod`

6.4.2 Watchdog Service

wdService

Function	Services the Watchdog.
Syntax	<code>Void wdSetEvent(void)</code>
Description	This function services the watchdog. That means, it initializes the watchdog timer to the value set by <code>wdSetPeriod()</code> .
Example	<code>wdService();</code>
See also	<code>wdEnableAutoService</code> , <code>wdDisableAutoService</code>

wdEnableAutoService

Function	Enables automatic servicing.
Syntax	<code>void wdEnableAutoService (void)</code>
Description	This function enables the watchdog automatic servicing feature. It services the watchdog in 30 ms intervals, if interrupts are enabled. Additional servicing may be done by RTIO device drivers. The servicing is enabled by default.
Example	<code>wdEnableAutoService();</code>
See also	<code>wdService</code> , <code>wdDisableAutoService</code>

wdDisableAutoService

Function	Disables automatic servicing.
Syntax	<code>void wdDisableAutoService(void)</code>
Description	This function disables the watchdog automatic servicing feature.
	Note: It is up to the model to service the watchdog accordingly. Please keep in mind, that disabling automatic servicing disables also RTIO internal servicing calls. Because RTIO driver calls (especially driver Init and Exit) potentially block for longer times, automatic servicing should be enabled inside the Init and Exit task.
Example	<code>wdDisableAutoService();</code>
See also	<code>wdService</code> , <code>wdEnableAutoService</code>

6.4.3 Interrupt Control

wdIntEnable

Function	Enables Watchdog interrupt handling.
Syntax	<code>void wdIntEnable(void)</code>
Description	This function enables the watchdog interrupt handling. Use <code>wdSetEvent()</code> in advance to map the watchdog event accordingly. The <code>wdIntEnable()</code> call has only influence on the interrupt propagation. <code>wdIntPend()</code> can be used even if the watchdog interrupt is disabled.
Example	<code>wdIntEnable();</code>
See also	<code>wdSetEvent</code> , <code>wdIntPend</code> , <code>wdIntDisable</code> , <code>wdIntAck</code>

wdIntDisable

Function	Disables Watchdog interrupt handling.
Syntax	<code>void wdIntDisable(void)</code>
Description	This function disables the watchdog interrupt handling.
Example	<code>wdIntDisable();</code>
See also	<code>wdIntEnable</code>

wdIntPend

Function	Checks if interrupt pending.
Syntax	<code>uint8 wdIntPend(void)</code>
Return Value	<code>false</code> No watchdog interrupt is pending <code>true</code> Watchdog interrupt is pending
Description	This function checks, if a watchdog interrupt is pending. Use <code>wdSetEvent()</code> in advance to map the watchdog event accordingly.
Example	<pre>if(wdIntPend() == true) { intPollCount++; /* Reset Interrupt */ wdIntAck(); }</pre>
See also	<code>wdSetEvent</code> , <code>wdIntDisable</code> , <code>wdIntAck</code>

wdIntAck	
Function	Acknowledges Watchdog interrupt.
Syntax	<code>void wdIntAck(void)</code>
Description	This function acknowledges a Watchdog interrupt. The Watchdog counter (automatic restart after triggering an event) is not influenced by this call. If the Watchdog counter should be initialized, use <code>wdService()</code> before.
Example	<pre>if(wdIntPend() == true) { intPollCount++; /* Reset Interrupt */ wdIntAck(); }</pre>
See also	<code>wdSetEvent</code> <code>wdIntDisable</code> <code>wdIntPend</code>

6.4.4 Watchdog Status

wdCheckReducedSafetyMode	
Function	Checks if Watchdog is in RSEF mode.
Syntax	<code>uint8 wdCheckReducedSafetyMode(void)</code>
Description	This function checks, if the watchdog is running in reduced-safety-enhanced-function (RSEF) mode. If so, the watchdog settings can be modified at runtime.
Return Value	<code>false</code> Watchdog is running in safety mode <code>true</code> Watchdog is running in RSEF mode
Example	<pre>asdWriteUserDebug("Active = %u ↵ ReducedSafety = %u \n", wdCheckActive(), ↵ wdCheckReducedSafetyMode()); (asdWriteUserDebug is described in Chapter 6.6.)</pre>
See also	<code>wdSetSafetyMode</code> , <code>wdCheckActive</code>

wdCheckActive	
Function	Checks if Watchdog is active.
Syntax	<code>uint8 wdCheckActive(void)</code>
Description	This function checks if the watchdog is currently active. This depends on the event setting and if a debugger is connected to the ES1135 board.
Return Value	<code>false</code> Watchdog is currently disabled

wdCheckActive

	true	Watchdog is currently enabled
Example	<pre>asdWriteUserDebug("Active = %u ↵ ReducedSafety = %u \n", wdCheckActive(), ↵ wdCheckReducedSafetyMode()); (asdWriteUserDebug is described in Chapter 6.6.)</pre>	
See also	wdSetSafetyMode, wdSetEvent, wdCheckReducedSafetyMode	

6.5 API Functions (ES1135 LEDs)

The ES1135 Simulation Controller has three configurable LEDs. They are briefly described in Chapter 3.4.2. The following interfaces to the LEDs are provided.

userLed[n]On

Function	Switches LED [n] on.
Syntax	<pre>void userLed1On(void) void userLed2On(void) void userLed3On(void)</pre>
Description	These functions switch the respective LEDs on.
Example	<code>userLed1On();</code>
See also	<code>userLed[n]Off</code> , <code>userLed[n]Toggle</code>

userLed[n]Off

Function	Switches LED [n] off.
Syntax	<pre>void userLed1Off(void) void userLed2Off(void) void userLed3Off(void)</pre>
Description	These functions switch the respective LEDs off.
Example	<code>userLed1Off();</code>
See also	<code>userLed[n]On</code> <code>userLed[n]Toggle</code>

userLed[n]Toggle

Function	Toggles LED [n].
Syntax	<pre>void userLed1Toggle(void) void userLed2Toggle(void) void userLed3Toggle(void)</pre>
Description	These functions toggle the respective LEDs.
Example	<code>userLed1Toggle();</code>
See also	<code>userLed[n]Off</code> <code>userLed[n]On</code>

6.6 API Functions (Miscellaneous)

A few more API functions are available.

asdWriteUserError

Function	Writes comment to ASCET monitor window.
Syntax	Equivalent to the ANSI-C function <code>printf</code>
Description	This function displays user messages in the ASCET monitor window.
Example	<pre>uint8 number = 1; asdWriteUserError("Example %u \n", number);</pre>
See also	<code>asdWriteUserDebug</code>

asdWriteUserDebug

Function	Writes comment to ASCET Target Debugger window.
Syntax	Equivalent to the ANSI-C function <code>printf</code>
Description	This function displays user messages in the ASCET Target debugger window.
Example	<pre>uint8 number = 1; asdWriteUserDebug("Example %u \n", number);</pre>
See also	<code>asdWriteUserError</code>

7 **Appendix B: ETAS Network Manager**

The ETAS Network Manager is used for creating a configuration that will be used by the ETAS IP Manager. The IP Manager is responsible for dynamic IP addressing of the ETAS hardware used in your network.

7.1 **Overview**

ETAS software supports different configurations for hardware access via Ethernet:

- Using multiple network adapters:
 - one network adapter for the company network,
 - one network adapter for the ETAS hardware.
- Using one network adapter:
 - automatic toggling between the company network and the ETAS hardware.

Note

You do not require a separate network adapter to connect the ETAS hardware to your PC. You can use the same network adapter both for the company network and the ETAS network.

The ETAS Network Manager supports you in selecting the network adapter for the ETAS hardware.

The ETAS Network Manager gives you an overview of the network adapters available for your PC and the type of IP address assignment. If more than one network adapter is available in the system, you can select the network adapter to use for connecting the ETAS hardware to your PC. You can also specify the address range for the IP assignment for the ETAS hardware.

You do not need administrator rights to select the network adapter and the network environment configuration for the ETAS hardware. You can toggle between the ETAS network and the company network without rebooting your PC.

Note

With Network Manager, you cannot create or modify the configuration for the network adapter. Instead modify the network settings of your PC via the Control Panel (see the documentation for your operating system). Please note that this requires administrator rights.

7.2 **ETAS Hardware Addressing**

The ETAS network allows you to connect several devices (including those that are the same type) to your PC. The connected devices are identified in the local ETAS network by their unique IP address.

An IP Manager integrated in the ETAS software looks up which IP addresses are available in a pre-configured address pool and assigns available IP addresses to the connected ETAS hardware.

The address range for the address pool is specified using the ETAS Network Manager.

7.3 Network Adapter Addressing

7.3.1 Type of Network Adapter Addressing

The type of network adapter addressing done within the company network depends on the operating system being used and the network adapter configuration:

Operating System	Type of Network Adapter Addressing			
	Manual	DHCP	DHCP+APIPA	DHCP+ alternative IP address
Windows Vista	yes	yes	yes	yes
Windows 7	yes	yes	yes	yes

The ETAS network supports the following types of network adapter addressing:

Operating System	Type of Network Adapter Addressing			
	Manual	DHCP	DHCP+APIPA	DHCP+ alternative IP address
Windows Vista	yes	no	yes	yes
Windows 7	yes	no	yes	yes

If you wish to use the network adapters both for the company network and the ETAS network, you cannot use the network adapters that exclusively support DHCP addressing for this dual operation.

Note

DHCP can be used only in combination with APIPA or an alternative IP address!

7.3.2 Addressing the Network Adapter Manually

Addressing a network adapter depends on the operating system.

For instructions on addressing your PC's network adapter, see the documentation for your operating system.

To address the network adapter manually, you need administrator rights. Please contact your system administrator, if necessary.

If the network adapter is addressed manually, i.e., it has a static IP address, it may happen that you accidentally end up searching for or initialize ETAS hardware, although the PC is connected to the company network. The Network Manager allows you to stipulate that if this happens, you are to receive a warning before an IP address is assigned to an ETAS hardware.

7.3.3 Addressing the Network Adapter via DHCP

Addressing via DHCP requires that the DHCP server be available. Should the DHCP server not be available, or if there is no DHCP server (as in the ETAS network), the network adapter has not been configured.

In this instance, each operating system has a feature that automatically assigns the network adapter an IP address:

Windows Vista / Windows 7

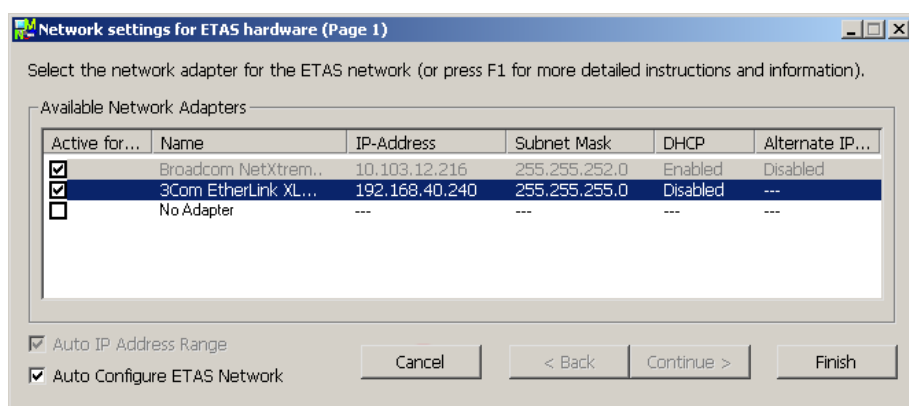
Windows Vista and Windows 7 automatically check whether there is a connection to the DHCP server. If there is none, it either assigns the IP address automatically via APIPA, or it uses the user-specified alternative IP address. The ETAS network always uses either the APIPA address or the alternative IP address.

When toggling between the DHCP network and ETAS hardware, make sure that the operating system is able to detect a connection failure because only then will reconfiguration be initiated. This may take up to 10 seconds. It takes the operating system 60 seconds to entirely reconfigure from a DHCP address to an APIPA address or to the alternative address. If the network adapter is once again connected to the DHCP network, configuring to a DHCP address takes place right after the connection has been detected.

Addressing a network adapter via DHCP without alternative addressing is not supported.

7.4 User Interface

7.4.1 "Network settings for ETAS hardware (Page 1)" Dialog Window



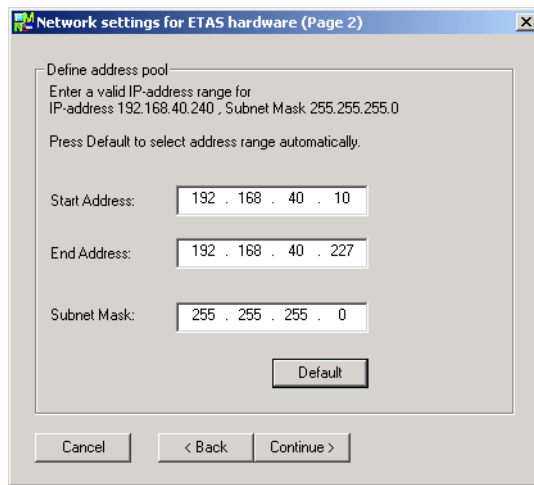
The following information on the available network adapters is displayed:

- **Active for ETAS Network** column
This column is only visible if the "Auto Configure ETAS network" checkbox is ticked. In the checkbox in this column you can determine which network adapters shall be enabled for autoconfiguration by the ETAS Network Manager.
- **Name** column
Name of the network adapter. This entry cannot be edited in this window.
- **IP Address** column
IP address of the network adapter. This entry cannot be edited in this window.
- **Subnet Mask** column
Setting for the subnet mask. This entry cannot be edited in this window.
- **DHCP** column
Shows whether the network adapter is configured for DHCP:

- Enabled
The network adapter is configured for DHCP.
- Disabled
The network adapter is configured with a fixed IP address.
- **Alternate IP Configuration** column
Shows the alternative IP address of the network adapter if it is configured for DHCP. This indication depends on the operating system being used.
 - APIPA
Automatic Private IP Addressing: method for automating the IP configuration for network connections
 - ---
An alternative IP address does not exist.
 - User defined
The user can define a user-specific alternative IP address (Windows Vista / Windows 7).
- **Auto IP Address Range** checkbox
If you tick this checkbox, the next configuration step is skipped, and the ETAS Network Manager automatically assigns default IP address ranges that will be used by the selected network adapter for addressing the ETAS hardware. If the IP address range is automatically changed by the IP Manager, a message is displayed in the system tray.
- **Auto Configure ETAS Network** checkbox
If you tick this checkbox, you can enable or disable several network adapters at once for auto-configuration through the IP Manager.
When you tick the checkbox, the "Auto IP Address Range" checkbox is activated. In the list of available network adapters, the "Active for ETAS Network" column is inserted, where you can determine which network adapters shall be available for auto-configuration through the ETAS Network Manager.
The ETAS Network Manager will go through the list (top-down) and use the first adapter which has a valid IP configuration for ETAS¹ and configure the IP address range automatically. If a configured network adapter fails, e.g. because the network adapter is being disabled or physically not available, the IP Manager will configure the next available network adapter automatically, and a message is displayed in the system tray, indicating the new configuration.

¹. An IP configuration is valid if the network adapter either uses a fix IP address, or if DHCP and APIPA are enabled.

7.4.2 "Network settings for ETAS hardware (Page 2)" Dialog Window



In general, all values can be modified by directly typing them in the corresponding field, or by selecting the default setting from a list box.

The following network parameters can be set:

- **Start Address**
The first IP address in the IP address range for the ETAS hardware.
- **End Address**
The last IP address in the IP address range for the ETAS hardware.
- **Subnet Mask**
Associated Subnet Mask.

Reserved IP Addresses

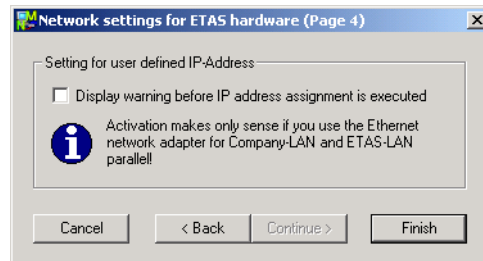
The following IP addresses are reserved for certain ETAS hardware in the IP address range that the ETAS hardware (192.168.40.1 - 192.168.40.254 with Subnet Mask 255.255.255.0) is currently using:

IP_Address	ETAS Hardware
192.168.40.10	ES1120
192.168.40.11	ES1130
192.168.40.12	ES780
192.168.40.13	Reserved
192.168.40.14	LABCAR-RTPC
192.168.40.15	ES1135

These addresses are assigned exclusively to these devices and thus may not be used for other ETAS hardware. This has to be taken into consideration when defining the address pool.

7.4.3 "Network settings for ETAS hardware (Page 4)" Dialog Window

This dialog window appears only if the selected network adapter is addressed manually.



The following parameters can be set:

- **Display warning before IP address assignment is executed**

Use this check box to specify that a warning be displayed before an IP address is assigned to an ETAS hardware device.

Note

Enabling this warning is useful only if you want to run the PC both in the company network or on an ETAS measurement module in the ETAS network using this network adapter.

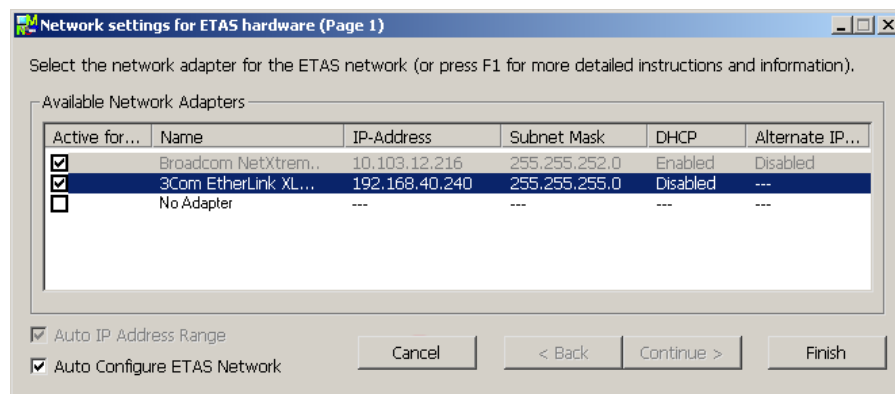
7.5 Configuring Network Addresses for ETAS Hardware

7.5.1 Adapter with Fixed IP Address

To start the Network Manager:

- In the Windows Start menu, go into the program folder of your ETAS software (below **Start** → **Programs** → **ETAS**), and select **ETAS Network Settings**.

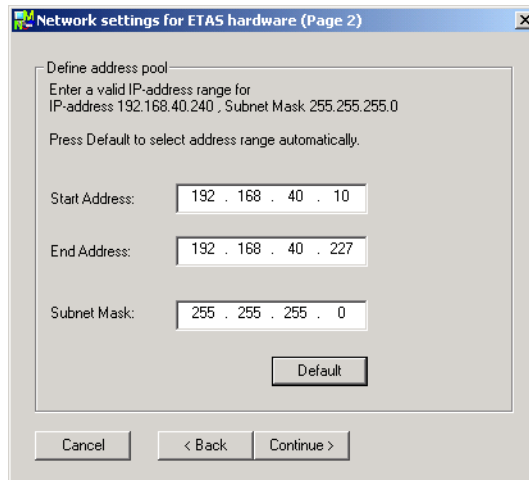
The "Network settings for ETAS hardware (Page 1)" dialog window opens.



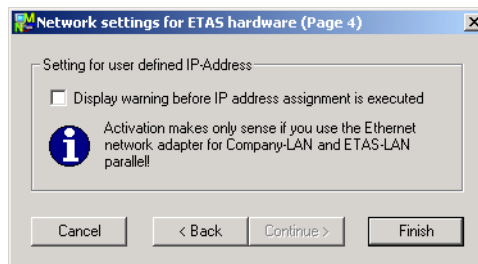
To select the network adapter:

- In the "Available Network Adapters" field, select the network adapters you want to use for the company network and the ETAS network.
You can select only network adapters of those types that are supported by the ETAS network.
- Click the **Continue** button.
The "Network settings for ETAS hardware (Page 2)" dialog window opens.

To define the address pool:



- Click the entry you want to modify in the "Start Address", "End Address" or "Subnet Mask" field.
 - Edit the value directly (text input).
- Or*
- Click the **Default** button.
The Network Manager automatically enters the address range and the setting for the subnet mask. You may accept these settings or overwrite them.
 - Click the **Continue** button.
The "Network settings for ETAS hardware (Page 4)" dialog window opens.



To set a user-defined IP address:

- Activate the **Display warning before IP assignment is executed** option if you want to specify that a warning be displayed before an IP address is assigned to the ETAS hardware.

Note

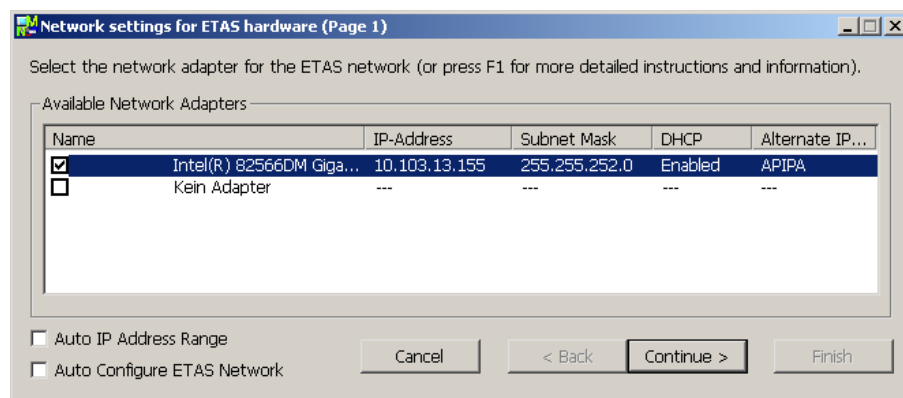
Enabling this warning is useful only if you want to run the PC both in the company network or on an ETAS measurement module in the ETAS network using this network adapter.

- Click the **Finish** button.
The configuration is finished and the dialog box is closed. The settings are saved.
- Restart the ETAS software to make the changes effective.
Restarting is necessary only if the ETAS software did not automatically invoke the configurator during a hardware search or initialization.

7.5.2 Adapter in DHCP Environment

To start the Network Manager:

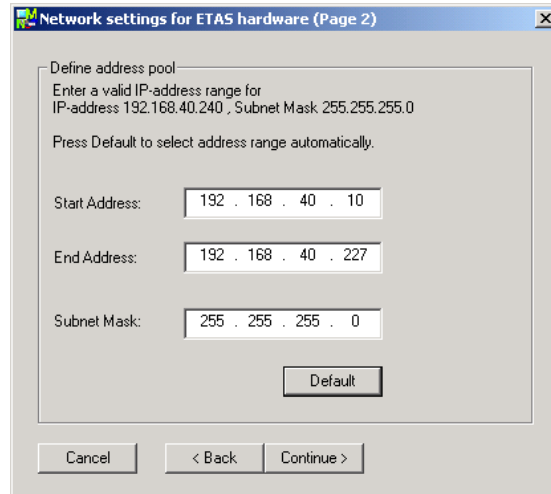
- In the Windows Start menu, go into the program folder of your ETAS software (below **Start** → **Programs** → **ETAS**), and select **ETAS Network Settings**.
The "Network settings for ETAS hardware (Page 1)" dialog window opens.



To select the network adapter:

- In the "Available Network Adapters" field, select the network adapters you want to use for the ETAS network.
You can select only those network adapters whose addressing type the ETAS network supports.

- Click the **Continue** button.
The "Network settings for ETAS hardware (Page 2)" dialog window opens.



To define the address pool:

- Click the entry you want to modify in the "Start Address," "End Address" or "Subnet Mask" field.
- Edit the value using the keyboard (text entry).

Or

- Click the **Default** button.
The Network Manager automatically enters the address range and the setting for the subnet mask. You may accept these settings or overwrite them.
- If you address the network adapter via DHCP using an APIPA or alternative IP address, click the **Finish** button.

The configuration is finished and the dialog window is closed. The settings are saved.

- Restart the ETAS software to make the changes effective.
Restarting is necessary only if the ETAS software did not automatically invoke the configurator during a hardware search or initialization.

7.6 Troubleshooting Ethernet Hardware Access

In certain cases you might have problems accessing ETAS hardware via the Ethernet interface. For instance, if APIPA, a mechanism for IP addressing, has not been enabled on your system, you cannot select a network adapter in the Network Manager. You can find descriptions of potential problems and their solutions in "Troubleshooting General Problems" on page 109.

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 Problems and Solutions

8.1.1 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 XP and Vista 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:

- Open the Registry Editor:
 - Windows 7:
Click **Start** and then click **Run**. Enter `regedit` and click **OK**.
 - Windows Vista:
Click **Start**, enter `regedit` in the entry field, and press <ENTER>.The registry editor is displayed.
- Open the folder `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\`
- Select **Edit** → **Find** to search for the key `IPAuto-configurationEnabled`.

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.

- Set the value of this key 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.
- Close the registry editor.
- Restart your workstation in order to make your changes take effect.

8.1.2 Search for Ethernet Hardware fails

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

If you are using ETAS hardware with ETAS MC 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 under 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

For a detailed description on problems caused by personal firewalls and possible solutions see "Personal Firewall blocks Communication" on page 112.

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 page110)
- 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 (Win Vista):

- To deactivate the network card first select in the Windows start menu the following item:
 - Windows Vista:
Control Panel → Network and Internet → Network and Sharing Center → Manage Network Connections
- Right-click on the used network adapter and select **Disable** in the context menu.
- In order to reactivate the network adapter right-click on it again and select **Enable**.

To deactivate and reactivate the network card (Win 7):

- To deactivate the network card, select **Control Panel → Device Manager** from the Windows start menu.
- In the Device Manager, open the tree structure of the entry **Network Adapters**.
- Click on the used connection to open its "*<connection name> Status*" dialog window.
- Right-click on the used network adapter and select **Disable** in the context menu.
- In order to reactivate the network adapter right-click on it again and select **Enable**.

Cause: Laptop energy management deactivates the network card

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

To switch off power monitoring on the laptop

- From the Windows Start Menu, select
 - Windows Vista:
Control Panel → System and Maintenance → Device Manager.

- Windows 7:
Control Panel → Device Manager.
 - In the Device Manager open the tree structure of the entry **Network Adapter**.
 - Right-click on the used network adapter and select **Properties** in the context menu.
 - Select the **Power Management** tab and deactivate the **Allow the computer to turn off this device to save power** option.
 - Select the **Advanced** tab. If the property **Autosense** is included, deactivate it also.
 - 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`:

- Open the Registry Editor.
- Select under `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\lanmanserver\parameters` the Registry Key `autodisconnect` and change its value to `0xffffffff`.

8.1.3 Personal Firewall blocks Communication

Cause: Permissions given through the firewall block ETAS hardware

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 experiment in ASCET 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:

- 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 used. For more precise information on the port numbers that can be used please refer to your hardware documentation.

The Windows operating systems come with a built-in personal firewall. In addition, it is very common to have personal firewall software from third party vendors, such as Symantec, McAfee or Blacklce installed. The proceedings in configuring the ports might differ for each personal firewall software used. Therefore please refer to the user documentation of your personal firewall software for further details.

As an example for a firewall configuration, you will find below a description on how to configure the widely used Windows XP firewall if the hardware access is prohibited under Windows XP with Service Pack 2.

Solution for Windows XP Firewall, Users with Administrator Privileges

If you have administrator privileges on your PC, the following dialog window opens if the firewall blocks an ETAS product.



To unblock a product:

- In the "Windows Security Alert" dialog window, click on **Unblock**.

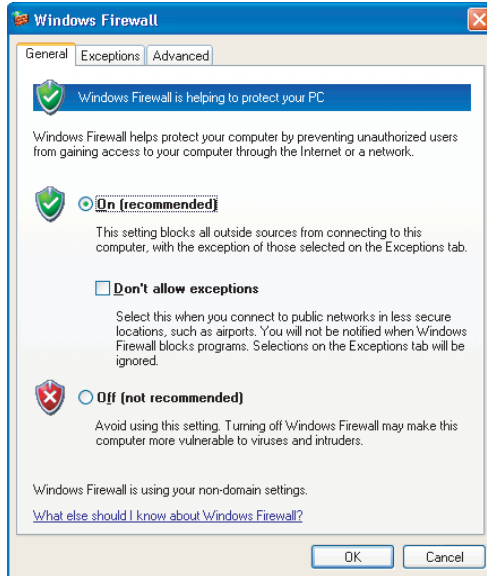
The firewall no longer blocks the ETAS product in question (in the example: ASCET). This decision survives a restart of the program, or even the PC.

Instead of waiting for the "Windows Security Alert" dialog window, you can unblock ETAS products in advance.

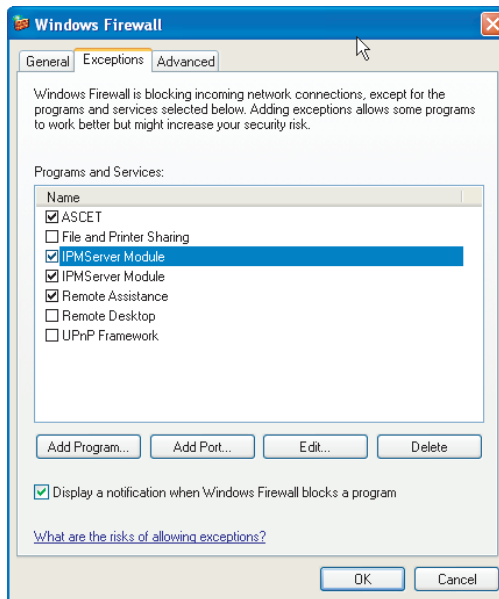
Unblocking ETAS products in the firewall control:

- From the Windows Start Menu, select **Settings** → **Control Panel**.

- In the control panel, double-click the **Windows Firewall** icon to open the "Windows Firewall" dialog window.



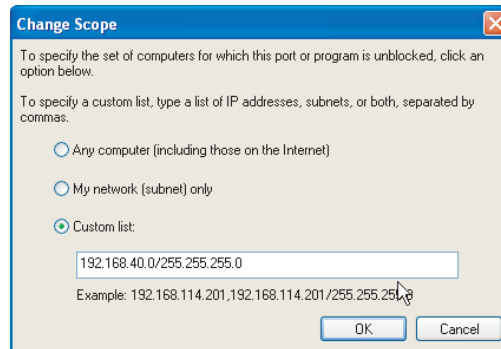
- In the "Windows Firewall" dialog window, open the "Exceptions" tab.



This tab lists the exceptions not blocked by the firewall. Use **Add Program** or **Edit** to add new programs, or edit existing ones.

- Make sure that the ETAS products and services you want to use are properly configured exceptions.

- Open the "Change Setup" window.



- To ensure proper ETAS hardware access, make sure that at least the IP addresses 192 . 168 . 40 . xxx are unblocked.
- Close the "Change Setup" window with **OK**.
- Close the "Windows Firewall" dialog window with **OK**.

The firewall no longer blocks the ETAS product in question. This decision survives a restart of the PC.

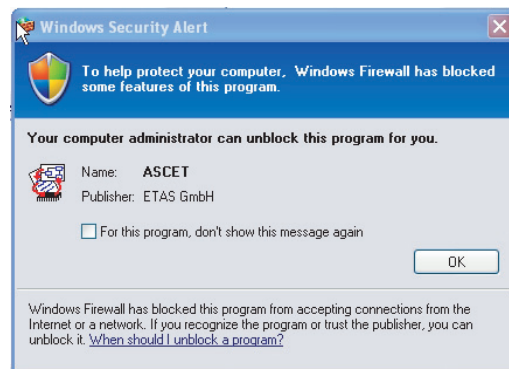
Solution for Windows XP Firewall, Users without Administrator Privileges

This section addresses users with restricted privileges, e.g., no system changes, write restrictions, local login.

Working with an ETAS software product requires "Write" and "Modify" privileges within the `ETAS`, `ETASData`, and ETAS temporary directories. Otherwise, an error message opens if the product is started, and a database is opened. In that case, no correct operation of the ETAS product is possible because the database file and some `*.ini` files are modified during operation.

The ETAS software has to be installed by an administrator anyway. It is recommended that the administrator assures that the ETAS program/processes are added to the list of the Windows XP firewall exceptions, and selected in that list, after the installation. If this is omitted, the following will happen:

- The "Window Security Alert" window opens when one of the actions conflicting with a restrictive firewall configuration is executed.



To unblock a program (no Admin privileges):

- In the "Windows Security Alert" dialog window, activate the option **For this program, don't show this message again**.
- Click **OK** to close the window.

An administrator has to select the respective ETAS software in the "Exceptions" tab of the "Windows Firewall" dialog window to avoid further problems regarding hardware access with that ETAS product.

9 **ETAS Contact Addresses**

ETAS HQ

ETAS GmbH

Borsigstraße 14
70469 Stuttgart
Germany

Phone: +49 711 89661-0
Fax: +49 711 89661-106
WWW: www.etas.com

ETAS Subsidiaries and 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:

ETAS subsidiaries WWW: www.etas.com/en/contact.php
ETAS technical support WWW: www.etas.com/en/hotlines.php

Index

A

ActivateTask 80
 API Functions (ERCOSEK)
 see also *Service Routines (ERCOSEK)*
 API Functions (LEDs)
 see also *Service Routines (LEDs)*
 API Functions (misc)
 see also *Service Routines (misc)*
 API Functions (NVRAM)
 see also *Service Routines (NVRAM)*
 API Functions (Watchdog)
 see also *Service Routines (Watchdog)*
 Application Mode 79
 ASCET options
 "Hardware Connection" node 12
 "Hardware" node 11
 asdWriteUserDebug 98
 asdWriteUserError 98

B

Back-Animation 56
 end 61
 end measurement 61
 Monitors 60
 open experiment environment 57
 select hardware 56
 set up experiment 59

start 56
 start measurement 60
 user interface 58

C

Cache locking 34
 restrictions 35
 set (component) 35
 set (included component) 36
 set (method) 35
 set (parameter) 35
 set (process) 35
 set (variable) 35
 set globally 37
 set in OS editor 36
 Compiler
 GNU Cross ~ 19
 precompiled header 18
 QCC 19
 use own ~ 18
 compiler switches 77
 control unit ES1120 25
 Converting old projects 21

D

Declarations 79, 80
 DeclareAppMode 79
 DeclareTask 80
 DisableAllInterrupts 82
 dT 22
 dT (delta t) 83

E

- EnableAllInterrupts 82
- ES1000
 - TCP/IP protocol 26
- ES1120 25
- ES1130 25
 - dT 22
- ES1130 target 18
- ES1135 25
 - cache locking 34
 - dT 22
 - LED API Functions 97
 - LEDs 34
 - NVRAM 38
 - special features 32
 - Watchdog 32
- ES1135 target 18
- ES581
 - settings 32
- ES910
 - configure 26
- ES910 target 18
- ETAS Contact Addresses 117
- ETAS Network 99
 - activate usage 13
 - Configure network adapter 104
 - DHCP 100
 - Network Manager 101
 - Reserved addresses 103
 - Search error 107
 - x 100
- ETAS network
 - Hardware Connection 13
- Ethernet interface 17
- Experimenting with INTECRIO 47–66
 - back-animation 56
 - calling transfer 48
 - creating an ASCET project 47
 - end back-animation 61
 - executing transfer 52
 - INTECRIO experiment 55
 - option "Ignore internally connected messages" 50
 - prepare project 48
 - project transfer 47
 - select INTECRIO Build process 52
 - select system project 52
 - selecting the INTECRIO version 51
 - selecting the workspace 51

- setting the path for files 51
- starting an experiment 55
- window "INTECRIO Project Transfer" 49

- EXPORT Subdirectory 9

G

- GetDeltaT 83
- GetSystemTime 81
- GetSystemTimeHigh 82
- GetSystemTimeLow 81
- GNU Cross Compiler 19

H

- Hardware Connection
 - with ETAS Network Manager 13
 - without ETAS Network Manager 17
- hardware options 11
- hardware selection window 14
 - open manually 14

I

- installation program 9
- INTECRIO experiment 55
- Interrupts
 - Disable 82
 - Enable 82

N

- Network configuration
 - s. ETAS network
- non-volatile RAM
 - see NVRAM
- NVRAM 38
 - API Functions 84
 - Basics 38
 - clear content 41
 - Data Consistency 41
 - defective content 43
 - Hardware Support 39
 - high level consistency 42
 - initialization of NV variables 41
 - low level consistency 42
 - model-controlled consistency 42
 - no consistency 42
 - NVRAM identifier 40
 - Tips 45
 - update of NV variables 41
- NVRAM Cockpit 43
 - work with ~ 43
- nvrAmCheckForAutoUpdate 87

- nvrAmCheckForInitializedVars 89
- nvrAmCheckRunningUpdate 89
- nvrAmClear 90
- nvrAmDisableAutoUpdate 86
- nvrAmEnableAutoUpdate 86
- nvrAmGetConsistencyLevel 86
- nvrAmGetUpdateAgeUs 89
- nvrAmGetUpdateInterval 85
- nvrAmInitModelVars 84
- nvrAmManualUpdateBackground 87
- nvrAmManualUpdateBlocked 88
- nvrAmManualUpdateExit 87
- nvrAmSetConsistencyLevel 85
- nvrAmSetUpdateInterval 84

- P**
- PPC module ES1130 25
- PPC module ES1135 25
- precompiled header 18
- Product liability disclaimer 7
- project transfer 47
- Prototyping target 18, 47

- Q**
- QCC compiler 19

- R**
- RTPRO-PC 29
 - configure 30
 - ES581 settings 32
 - startup 30

- S**
- Safety Instructions
 - technical state 8
- SCOOP-IX 61–66
 - example 63
- Service Routines
 - GetSystemTime 81
 - GetSystemTimeHigh 82
- Service Routines (ERCOSEK)
 - ActivateTask 80
 - DeclareAppMode 79
 - DeclareTask 80
 - DisableAllInterrupts 82
 - EnableAllInterrupts 82
 - GetDeltaT 83
 - GetSystemTime 81
 - GetSystemTimeHigh 82
 - GetSystemTimeLow 81
 - SetNextAppMode 79
- Service Routines (LEDs)
 - userLed(n)Off 97
 - userLed(n)On 97
 - userLed(n)Toggle 97
- Service Routines (misc)
 - asdWriteUserDebug 98
 - asdWriteUserError 98
- Service Routines (NVRAM)
 - nvrAmCheckForAutoUpdate 87
 - nvrAmCheckForInitializedVars 89
 - nvrAmCheckRunningUpdate 89
 - nvrAmClear 90
 - nvrAmDisableAutoUpdate 86
 - nvrAmEnableAutoUpdate 86
 - nvrAmGetConsistencyLevel 86
 - nvrAmGetUpdateAgeUs 89
 - nvrAmGetUpdateInterval 85
 - nvrAmInitModelVars 84
 - nvrAmManualUpdateBackground 87
 - nvrAmManualUpdateBlocked 88
 - nvrAmManualUpdateExit 87
 - nvrAmSetConsistencyLevel 85
 - nvrAmSetUpdateInterval 84
- Service Routines (Watchdog)
 - wdCheckActive 96
 - wdCheckReducedSafetyMode 96
 - wdDisableAutoService 94
 - wdEnableAutoService 94
 - wdIntAck 96
 - wdIntDisable 95
 - wdIntEnable 95
 - wdIntPend 95
 - wdService 94
 - wdSetEvent 93
 - wdSetPeriod 92
 - wdSetReducedSafetyMode 92
 - wdSetSafetyMode 91
- SetNextAppMode 79
- Switch
 - Application Mode 79
- System Time 81

- T**
- target
 - set up interfaces (with ETAS Network Manager) 13
 - set up interfaces (without ETAS Network Manager) 17
- Target directory 11
- target.ini 17

Task

Activation 80

general description 80

TCP/IP protocol 26

U

userLed(n)Off 97

userLed(n)On 97

userLed(n)Toggle 97

W

Watchdog 32

API Functions 90

interrupt control 33

modes 33

period 33

service 33

service register 33

wdCheckActive 96

wdCheckReducedSafetyMode 96

wdDisableAutoService 94

wdEnableAutoService 94

wdIntAck 96

wdIntDisable 95

wdIntEnable 95

wdIntPend 95

wdService 94

wdSetEvent 93

wdSetPeriod 92

wdSetReducedSafetyMode 92

wdSetSafetyMode 91