# ASCET Rapid Prototyping V5.6

User's Guide

## Copyright

# Contents

# 1    Introduction

The execution of real-time software requires experimenting hardware that is capable of real-time processing. The ASCET **R**apid **P**rototyping V5.6 (ASCET-RP V5.6) software package is used to integrate the ES1000 experimental targets (E-Targets) in ASCET V5.2. Together with I/O periphery, powerful development systems can be built on the basis of these experimental targets.

In addition to the license for the compiler toolset used, the ASCET-RP Package also includes extensions to the ASCET development environment, such as for the homogenous integration of compiler and linker calls and the ERCOS$^{EK}$ operating system kernel for experimental targets.

ASCET-RP contains, in addition, the basic functions of the ASCET base software. ASCET-MD is needed in addition for the modeling functions.

## 1.1    Components

The ASCET-RP V5.6 installation includes the following components:

- Integration of the ES1130 and ES1135[1];
- Integration of the I/O boards, including ETK Bypass and CAN Bypass , for the targets ES1130 and ES1135 (RTIO package);
- GNU compiler;
- Documentation and examples.

## 1.2    Installation

The ASCET base software and the ASCET-RP software are supplied on a common CD-ROM. To install ASCET-RP V5.6 on your PC, you have to install the ASCET base software first, and then the ASCET-RP software, and you have to obtain a license file.
Start first the installation program `ASCET.exe.` Continue with the installation program `ASCET-RP.exe` from the CD-ROM.

Details on the ASCET-RP V5.6 installation can be found in the release note.

Details on licensing are given in the ASCET "Getting Started" manual.

*Sample Files*

After the installation of ASCET-RP V5.6, the sample databases exported from ASCET are located in the `EXPORT` directory of your ASCET installation in the `RTIOTutorial.exp` and `INTECRIO_Tutorial.exp` files.

---

[1] The term *ES113x* is used tthroughout this manual for an arbitrary system controller.

## 1.3 Manual Structure

The ASCET-RP V5.6 user's guide consists of three main sections:

- General Section
- Real-Time Input-Output Package (including Bypass Interface)
- Tutorial

The general section is intended for all users of ASCET-RP V5.6. Here, the users find information about the structure, installation and usage of ASCET-RP V5.6.

The subsequent chapters introduce and explain the functionality and operation of the RTIO package of ASCET-RP V5.6.

The tutorial contains lessons about experimenting with INTECRIO, as well as the configutration of several boards.

## 1.4 Conventions

### 1.4.1 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 box related to the action.

### 1.4.2 Typographic Conventions

The following typographic conventions are used throughout this manual:

| | |
|---|---|
| Select **File → Open**. | Menu options are printed in bold characters. |
| Click **OK**. | Button labels are printed in bold characters. |
| Press <ENTER>. | Key commands are printed in small capitals enclosed in angle brackets. |

| | |
|---|---|
| The "Open File" dialog window opens. | The names of program windows, dialog boxes, fields, etc. are enclosed in double quotes. |
| Select the `setup.exe` file. | Text strings in list boxes on the screen, in program code and in path and file names are printed using the `Courier` font. |
| A *distribution* is always a one-dimensional table of sample points. | Emphasized text portions and newly introduced terms are printed in *italic* font face. |
| The OSEK group (see http://www.osekvdx.org/) has developed certain standards. | Links to internet documents are set in <u>blue, underlined</u> font. |

Important notes for the users are presented as follows:

<u>**Note**</u>

*Important note for the user.*

## 2 Configuring Experimental Targets

ASCET Rapid Prototyping V5.6 (ASCET-RP V5.6) contains the compiler and linker tools required for producing executable files for a transputer or PowerPC target, and an extension of the ASCET development environment for the hardware integration. The target itself can be selected in the target options of the project. That way the targets are fully integrated into ASCET.

The configuration of the compiler and the linker, as well as the description of the interface to the actual target hardware is not described in ASCET directly, but either with the help of the ETAS Network Manager or with the help of `*.ini` files. Chapter 2.1 describes the hardware options of ASCET-RP, hardware connection using 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.

*Structure of the PowerPC E-Target Directories*

Installing ASCET-RP V5.6 results in a target directory with two different PowerPC subdirectories being created in the ASCET directory. These subdirectories contain E-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\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 computer nodes, e.g. compiler-specific make files.

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

## 2.1 The Hardware Options

ASCET-RP adds the "Hardware" node to the ASCET option window for easy setting of the interface. In that node, you can specify the following options. It is suggested that you select the item you use most frequently.
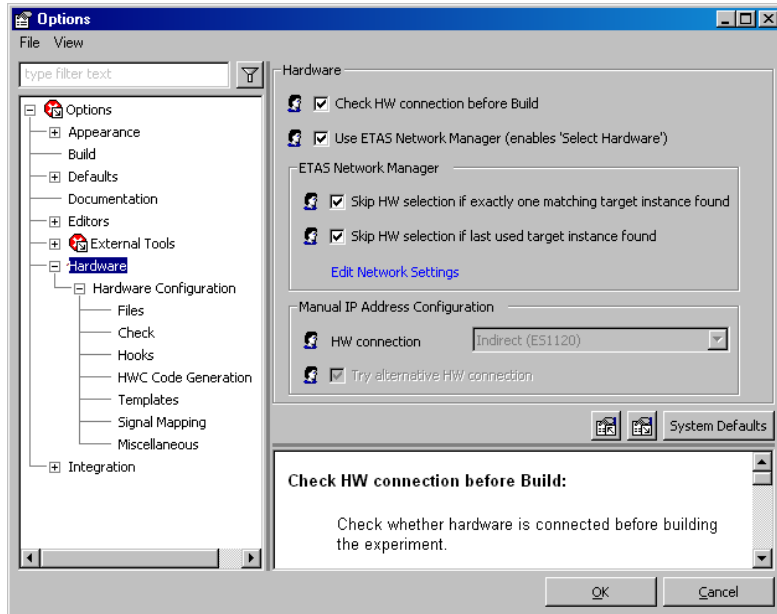


- **Check HW connection before Build**

  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 12) is used (activated, default) or not.
  When the option is activated, the **Select Hardware** button and the **Extras → Select Hardware** menu option in the project editor become available.

---

**Note**

*The following two options and the link are relevant only when you are using the ETAS Network Manager.*

---

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

  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**

  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 12).

---

**Note**

*The following two options are relevant only when you are **not** using the ETAS Network Manager.*

---

- **HW connection**

  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**

  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.

> **Note**
>
> *The following option is always relevant.*

## 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.
- You can select simulation controllers (ES113x) from an ETAS hardware network.

Working with the ETAS Network Manager is described in chapter 12. Here, you find information regarding hardware connection using the EAS Network Manager.

**To activate ETAS Network Manager usage:**

- In the component manager, select **Tools → Options**.

  The "Options" dialog window opens.

- Open the "Hardware" node.

- Activate the **Use ETAS Network Manager (enables 'Select Hardware')** option.

  If *only* this option is activated, the hardware selection window "Experimental Target Hardware Selecvtion" 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:**

> <u>Note</u>
>
> *The* **Select Hardware** *button and the* **Extras → Select Hardware** *menu option are only available when the* **Use ETAS Network Manager (enable 'Select Hardware')** *option is activated.*

- Select **Extras → Select Hardware**

*or*



- click on the **Select Hardware** button.

  The hardware selection window opens.

## 2.2.1    The Hardware Selection Window
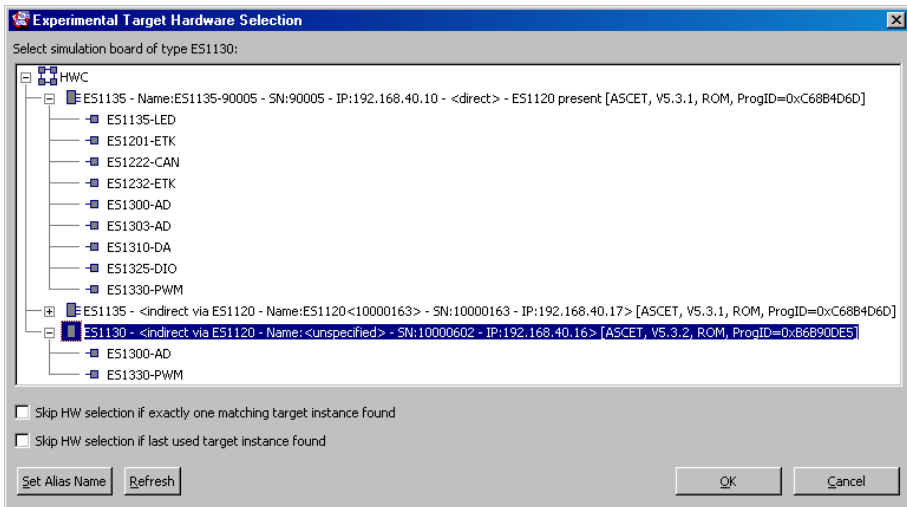


The hardware selection window, named "Experimental Target Hardware Selection", contains the following elements:

- Field "Select simulation board of type *<type>*"[1]

  This field displays, below the main entry HWC (symbol ⊞), all simulation controllers (ES113x, symbol ▤) connected with the PC. The simulation controller label contains, in additon to stating whether the ES113x is connected directly or indirectly to the PC, further information; see page 20. Available boards (symbol ▬) are displayed below the simulation controller.

  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" node 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.

- 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 the 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.

---

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

- `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.

- `<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 the simulation controller is indirectly connected to the PC, i.e. via ES1120, its entry 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.

## 2.3 Interface Setup Without ETAS Network Manager

For special use cases, ASCET-RP offers the possibility to work without the ETAS Network Manager, in accordance with previous ASCET-RP versions. In this case, 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. The options are described in chapter 2.1.

- Open the "Hardware" node.

- 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 before and *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 subirectory 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
```

ETAS provides Ethernet documentation with tips on how to install and configure the network interface of your PC for connecting the ES1000.x system. Once you have installed ASCET manuals, the following Ethernet document is available in the `..\ETAS\ETASManuals\ASCET V5.2\ES1000` directory:

- `ES1000 Ethernet InstallationGuide.pdf` (installation)

## 2.4    Selecting a Compiler

The GNU Cross Compiler (`GNU-C V3.4.4 (Power PC)` in the user interface) is integrated in ASCET-RP for the ES1130 and ES1135 target. As an alternative, you can use the older `GNU-C V2.95.3 (Power PC)` compiler. The Diab Data Compiler can be used for the ES1130 target (not included in delivery).

| Target | Compiler |
|--------|----------|
| ES1130 | GNU Cross Compiler, Diab Data V4.1a |
| ES1135 | GNU Cross Compiler |

**Tab. 2-1**    Target/Compiler Overview

For both versions of the GNU Cross compiler, and for two Diab Data compiler versions, specific make files are provided in the `..\Target\ES113x\trg-make` directory. Version-independent settings are stored in the `settings_<compiler>_common.mk` files. 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 acivated 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 is supplied as a standard part of the ASCET-RP V5.6 package. If you want to use your own compiler, please take the following points into consideration.

1. The path of the `GNU-C V3.4.4 (PowerPC)` code generation option has to be reset to the new compiler in the ASCET dialog `"Options"`.
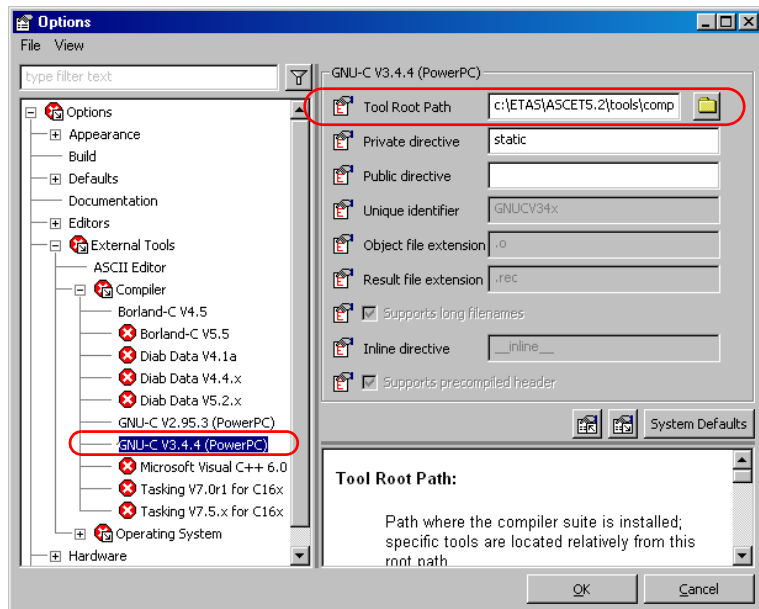
2. In the `comptool.ini` file of the relevant target (ES1130 or ES1135), the `RelativeToolPath` variable has to be changed from `RelativeToolPath=\powerpc-eabi\bin` to (for example) `RelativeToolPath=\win32\bin`.

**Note**

*Paths and system variables should only be set to the version of the compiler to be used in ASCET.*

### 2.4.2 Changing to the GNU Cross Compiler

The ASCET option window, "External Tools\Compiler" node, contains a sub-node with default path for the Diab Data compiler even if the compiler is not installed. Likewise, you can select the Diab Data compiler in the build options of a project even if the compiler is not installed.

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.
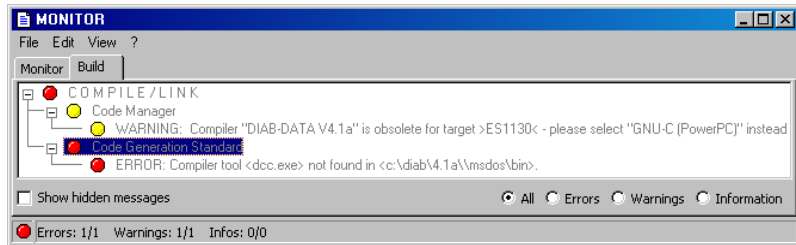


If you are working with an older project that uses the ES1112 target, an error message is displayed when you open the project, and the missing target is replaced by the target PC.
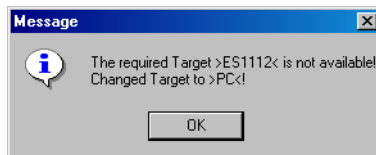


In both cases, you have to select a suitable combination of target and compiler for the project. Proceed as follows.

**To change to the GNU Compiler:**



- In the project editor, click on the **Project Properties** button.

  The "Project Properties" window opens.

- In the "Build" node, select the target `ES1130` or `ES1135` and the `GNU-C V3.4.4 (PowerPC)` compiler.

- Click **OK**.

  Depending on which target the project used, you now have to copy the C-Code. The procedure is described in section "To copy operating system settings and C-Code:" on page 26.

- Finally, select **Component** → **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.
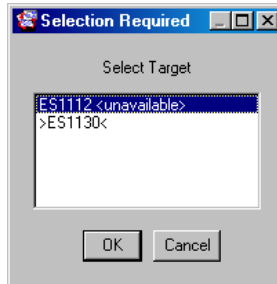


- From the "Selection Required" dialog box, select the original target of the old project.

- Click **OK**.

  The operating system code is copied from the old target to the **ES1130** or **ES1135** target.

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

  The "Selection Required" window opens.
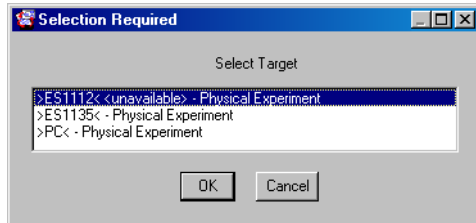


- In the "Selection Required" window, select the original target and experiment of the old project.
- Click **OK**.

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

### 2.4.3 Changing to the Diab Data Compiler

The MS-DOS version of the Diab Data 4.1a Compiler was supplied up to version 4.2 of TIPExp. The Diab Data Compiler can also be used with ASCET-RP V5.6 for the ES1130 target by selecting `Diab Data V4.1a` as compiler. However, it must be ordered and installed separately.

> **Note**
>
> *The ES1135 target **cannot** be used with the Diab Data compiler.*

# 3 Tips on Using ASCET-RP

## 3.1 Preprocessing available Data Bases

ASCET data bases which were made with ASCET versions prior to V4.1.1 must at first be stored at least with an ASCET version V4.1.1, before they can be opened and converted with ASCET V5.1.

**Note**

*Detailed informations to convert very old ASCET projects (with TIPExp V3.x and older, Target PPC) are given e.g. in the TIPExp V4.4 manual.*

## 3.2 Converting Projects for ES1000.1 to ES1000.2/ ES1000.3

ASCET projects which were created for the ES1000.1 E-target have to be converted for the ES1000.2/ ES1000.3 E-target. The following steps have to be executed for the conversion.

**To convert an ASCET project for ES1000.1 to ES1000.2/ ES1000.3**

• Load the ASCET project.

A message is displayed that the ES1112 target is no longer available.
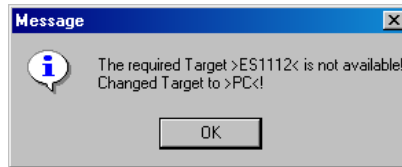


• Click **OK** to confirm the message.

The project opens; the target >PC< is selected instead of the unavailable target.

- Click the **Specify Code Generation Options** button.
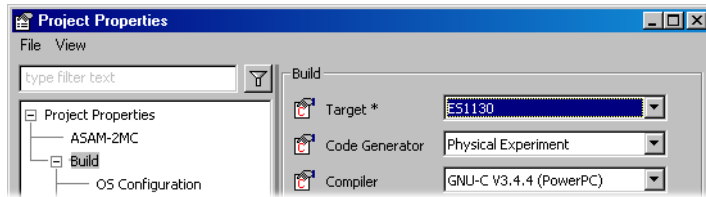


- In the "Settings for" window, "Build" tab, select the following options:
  `Target: >ES1130< or >ES1135<` and
  `Compiler: GNU-C (PowerPC)`.
- Click **OK** to close the "Settings for" window.
- Select the "OS" tab in the project editor.
- In the project editor, select **Operating System → Copy From Target**.
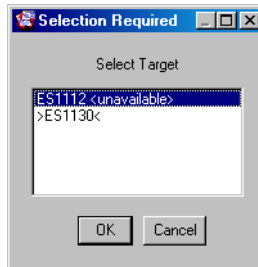


- In the "Selection Required" window, select the original target and click **OK**.

  The operating system code is copied from the `ES1112` target to the `ES113x` target.

- Select **Extras → Copy C-Code From** from the project editor.
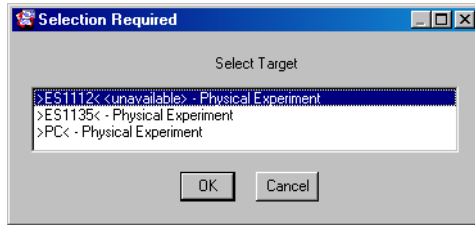


- In the "Selection Required" window, select the original combination of target >`ES1112`< and experiment of the old project.
- Click **OK**.

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

The project can now be edited for the ES1000.2/ ES1000.3 system.

## 3.3    Using dT

The ERCOS$^{EK}$ operating system is implemented for the ES113x target. ERCOS$^{EK}$ enables access to the time dT which has elapsed since the last and second last call of the running task. dT always refers to the task in which the variable is used (see Fig. 3-1).
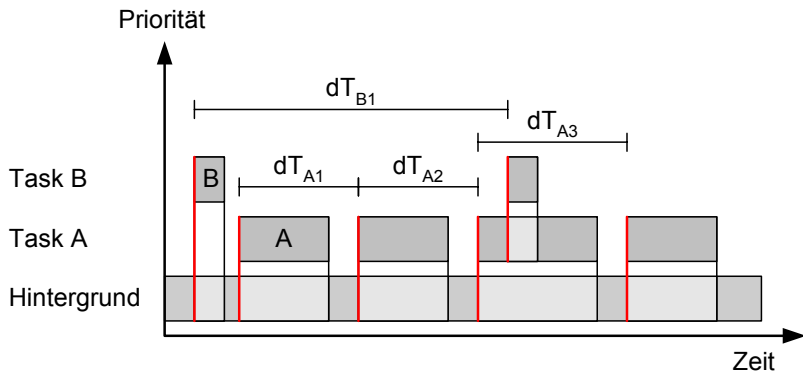


**Fig. 3-1**     dT Scheme

In ERCOS<sup>EK</sup>, `dT` is a global `uint32` variable. It is declared in one of the ERCOS<sup>EK</sup> 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 (`real64`) which contains the time in units of seconds.

If a user does not generate this element in the C-Code editor but still accesses `dT`, no error message appears because `dT` is declared in the ERCOS<sup>EK</sup> files. But as `dT` in ERCOS<sup>EK</sup> and `dT` in ASCET have different units (system ticks or seconds respectively), the calculations are incorrect. The user should therefore ensure that he/she generates the corresponding element with the **dT** button.

# 4        Rapid-Prototyping Experiments

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

## 4.1        Experimenting with ASCET

If you want to run the Rapid-Prototyping experiment in ASCET, you can choose between an online and an offline experiment in the project editor. For more details, please refer to the "Experimenting with Projects" section in the ASCET User's Guide; the ASCET experiment environment is described in detail in the ASCET User's Guide in the section "The Experiment Environment".

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

### 4.1.1        The User Interface

The user interface of the online experiment is very similar to that of the offline experiment. However, the buttons for controlling the experiment and the NVRAM cockpit (ES1135 only), the "Task" combo box and the functions in the **Experiment** menu are different from the offline experiment.

Control buttons (experiment, NVRAM cockpit)        "Task" combo box

*Buttons*

| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
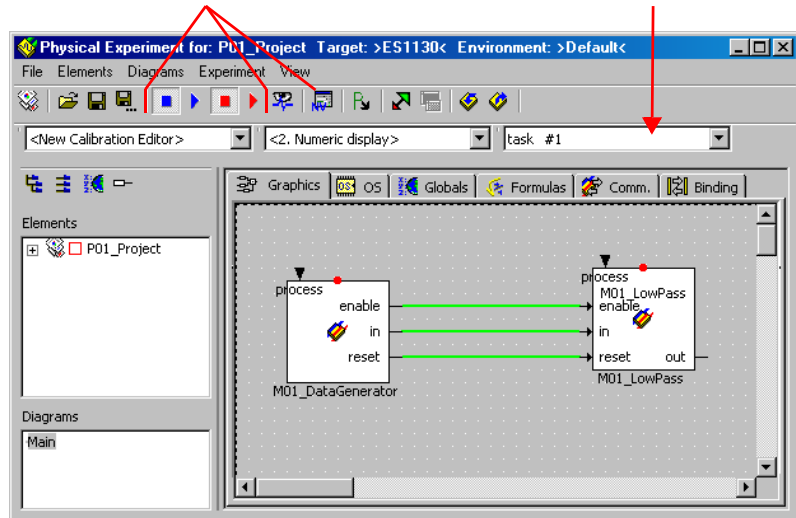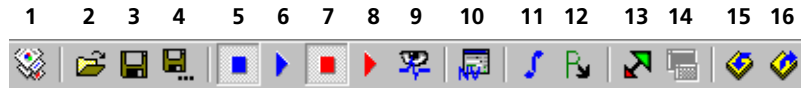
1. Exit to Component (ends the experiment and invokes the project editor)

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 ERCOS (stops the operating system and thus the experiment)

6. Start ERCOS (starts the operating system and thus the experiment)

7. Stop Measurement (stops measurement, i.e. the data display)

8. Start Measurement (starts measurement, i.e. the data display)

9. Open Data Logger

10. Open NVRAM Cockpit (cf. page 76)

    This button *only* exists if you selected the ES1135 as target.

11. Open CT Solver (opens a window in which you can configure the integration method)

    This button *only* exists if you are experimenting with a CT block or a hybrid project.

12. Update Dependent Parameters (updates the values of dependent parameters)

13. Expand / Collapse Window (shows/hides the component display)

14. Always on top (keeps the experiment window on top)

15. Navigate down to child component (shows the selected included component)

16. Navigate up to parent component (shows the parent component)

### *Experiment* Menu

– *Data Logger*

Opens the Data Logger.

– *NVRAM Cockpit*
(Only available when the target ES1135 was selected in the code generation options.)

Opens the NVRAM Cockpit.

– *Stop ERCOS*

Stops the operating system.

– *Start ERCOS*

Starts the operating system.

– *Stop Measurement*

Stops the measurement.

– *Start Measurement*

Starts the measurement.

– *Open Target Debugger*

Opens the debugger window for C code components.

– *Update Calibration Windows*

Updates the content of calibration windows.

– *Close Calibration Windows*

Closes all open calibration windows.

– *Close Measure Windows*

Closes all open measure windows.

The other elements of the user interface correspond to those of the offline experiment; they are described in the "The Experiment Environment" section of the ASCET User's Guide.

### 4.1.2 Running Online Experiments

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

**To start the online experiment:**

- Open the project or component you require.
- If you want to experiment with a component, open the relevant default project.

- In the project properties, "Build" node, select the target `ES1130` or `ES1135` and an appropriate compiler.

  In the "Experiment Target" combo box you can select either `Offline (RP)` or `Online (RP)`.

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

- Select `Online (RP)` from the "Experiment Target" combo box.

  `Offline (RP)` is intended for offline experiments on the target.

- Select **Component → Open Experiment**

*or*

- click the **Open Experiment for selected Experiment Target** button.
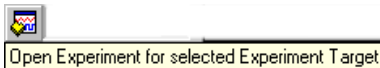
**To select hardware (with ETAS Network Manager):**

> **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 37.*

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.

**What to do in case of an error:**

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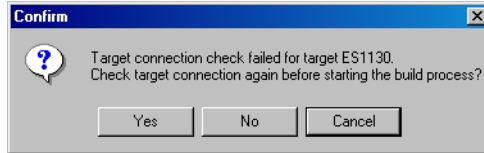


- Click **Yes** to repeat the search for a hardware connection.

*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 experiment environment for the online experiment:**

The default experiment environment for the component is opened immediately after staring the experiment (cf. page 35) when you are working without ETAS Network Manager, or after successful hardware selection (cf. page 36) when you are working with the ETAS Network Manager.

- If several environments have been saved, select the one to be opened.
  For more details, refer to the section "Loading and Saving Environments" in the ASCET User's Guide.

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

  Later on, you can select the acquisition task in the "Task" combo box.

Setting up an online experiment only entails the setting up of the measure and calibration windows. The measure and calibration windows in the online experiment are the same as those in the offline experiment. "The Experiment Environment" section in the ASCET User's Guide explains how to use them.

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

**To start an experiment and measurement:**

- Open the experiment environment for the project you want to experiment with.

- Select **Experiment → Start ERCOS**

*or*



- click the **Start ERCOS** button.

  If you set the relevant options in the ASCET options window, "Experiment" node, variables and parameters are initialized each time this command is called.
  The operating system and the experiment are started. Measure data is not displayed yet.

- Select **Experiment → Start Measurement**

*or*
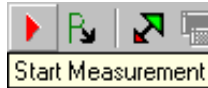
- click the **Start Measurement** button.

**Note**

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

Measurement is started and all values set up in the measure system are displayed in the relevant windows.
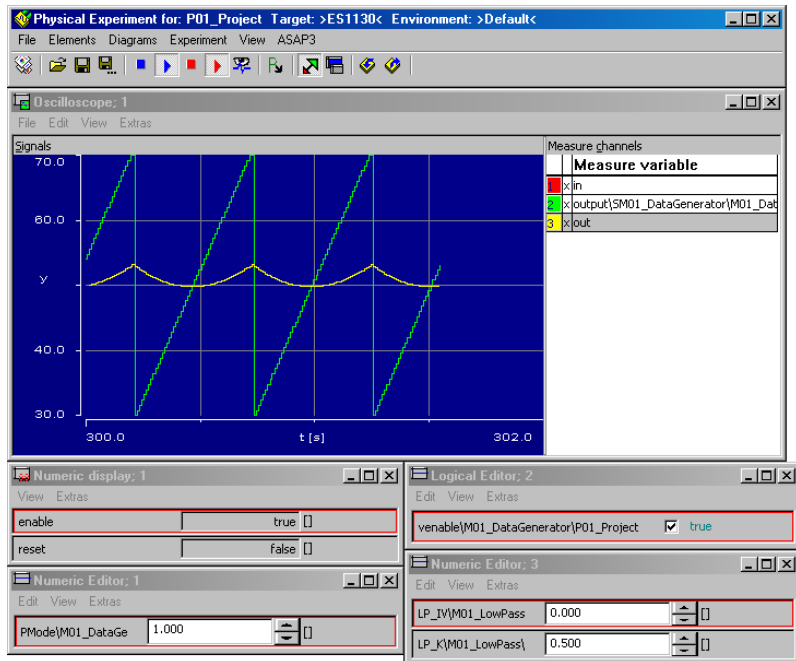


**To stop measurement:**

- 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.

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

**To stop the experiment:**

- Select **Experiment → Stop ERCOS**

  *or*

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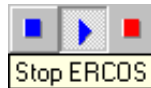
  Any measurement which is currently running is stopped. The operating system, and thus the experiment, is stopped and enters the inactive mode. The inactive mode of the operating system may contain one task with trigger mode `init` that is executed when the operating system is stopped. This can be used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

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

- Click the **Exit to Component** button to exit the experiment environment and to activate the project editor.

- Select **File → Exit** to exit the experiment environment and close the project.

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

As with the offline experiment, components specified in C Code offer additional opportunities for displaying debugger information or error messages during experimenting. You can embed debugger or error messages in your C code. Debugger information is displayed in the Debugger window which

can be opened during experimenting. Error messages are displayed in the ASCET monitor window. The debugger in the online experiment works like the debugger in the offline experiment (see the section "Running Experiments" in the ASCET User's Guide).

4.1.3    Standalone Mode

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

**To load the experiment in the Flash memory:**

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

- Select **Component → Flash Target**.

  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.
  Section "To select hardware (with ETAS Network Manager):" describes the required actions in this window.

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

**To experiment in standalone mode:**

- Select **Component → Reconnect To Experiment**

  *or*

- click on **Reconnect to Experiment of selected Experiment Target** to switch to a running online experiment.
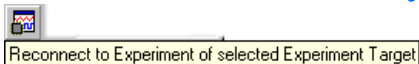
  When you activated the **Use ETAS Network Manager (enables 'Select Hardware')** option in the hardware options (chapter 2.1),

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

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

## 4.2 Experimenting with INCA

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

**To initiate a transfer:**

- Open the project with which you want to experiment.
- From the project properties in the project editor, "Build" node, select the `ES1130` or `ES1135` target and a compiler `GNU-C V3.4.4 (PowerPC)`.



- From the "Experiment Target" combo box, select `INCA`.

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



- Click on **Transfer Project to selected Experiment Target**

  *or*
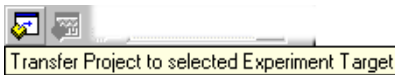
- select **Component** → **Transfer Project**.

The "INCA Project Transfer" dialog opens.
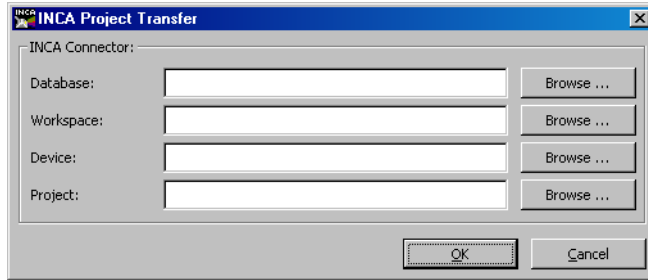


In this window, you define the INCA database, the workspace, and the project within the INCA database you want to use.

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

---

**Note**

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

---

**To set the INCA database path:**

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

*or*

- click on the **Browse** button to search for a database.

  The "Select INCA database path" window opens. The databases are shown by red folders.
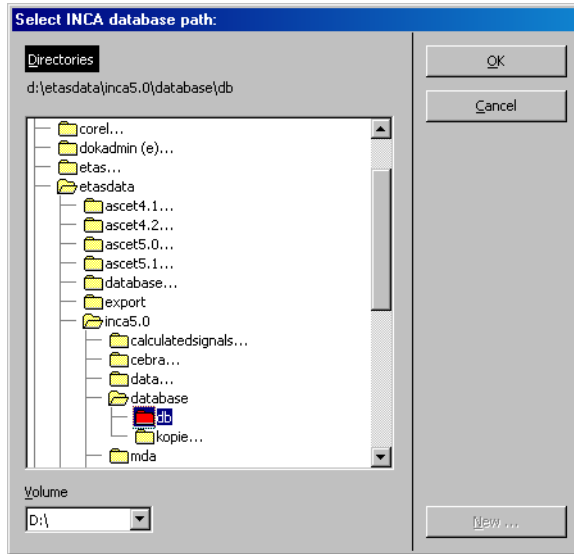


- Select an INCA database path and click **OK**.

---

**Note**

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

---

*Or*

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

**To select an INCA workspace:**

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

*or*

- click on the **Browse** button to search for a workspace.

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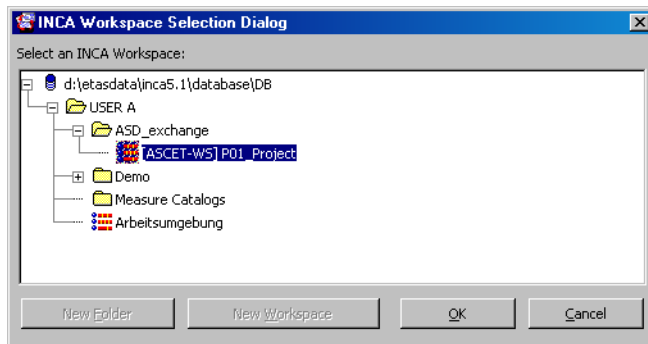


- Select an existing workspace

*or*

- create a new workspace in a new or existing folder. The procedure is described on page 45.
- Once you have selected a workspace, click **OK**.

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

1. Creating a folder

   - In the "INCA Workspace Selection Dialog" window, activate the database name or a folder.

     The **New Folder** button is activated.

   - Click on **New Folder**.

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

  The new folder is created.

2. Creating a workspace

   - Activate a folder.

     The **New Workspace** button is activated.

   - Click on **New Workspace**.

     The "INCA Workspace Template Selection Dialog" window opens.
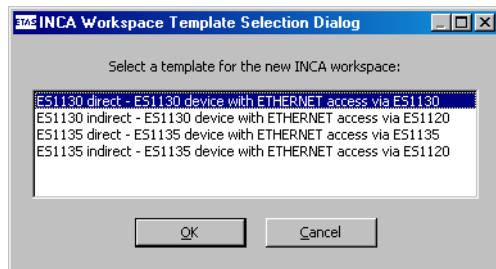


   - Select a template for the workspace and click **OK**.

     The dialog window "Create New INCA Workspace" opens.
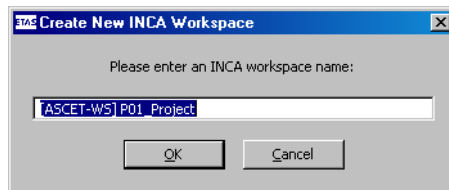


     The name `[ASCET-WS] <ascet project name>` is assigned.

   - Enter a name for the workspace and click on **OK**.

     The new workspace is created in the selected folder and selected.

**To select an INCA device:**

- Enter a device in the "INCA Project Transfer" window in the "INCA Device" field

*or*

- click on the **Browse** button to search for a device.

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



- Select a device and click on **OK**.

**To select an INCA project:**

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

*or*

- click on the **Browse** button to search for a project.

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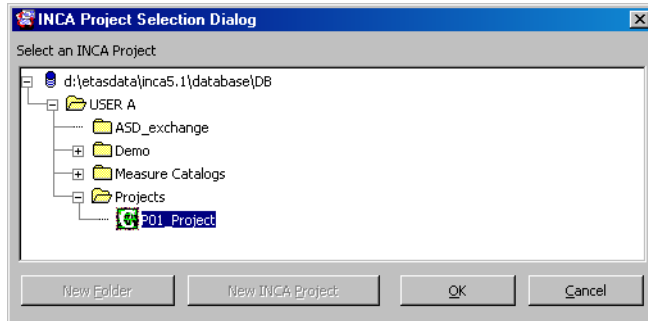


- Select an existing project

*or*

- create a new project in a new or existing folder. The procedure is described on page 48.
- Once you have selected a project, click on **OK**.

**Note**

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

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

1. Creating a folder

   - To create a new folder, proceed as described under "Creating a folder" on page 45.

2. Creating a project

   - Activate a folder.

     The **New INCA Project** button is activated.

- Click on **New INCA Project**.

  The "Create New INCA Project" input window opens. The name [ASCET-Prj] <ascet project name> is assigned.
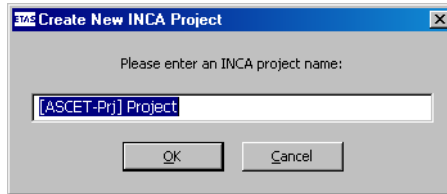
  

- Enter a name for the project and click on **OK**.

  The new project is created in the selected folder and selected.

**To start a transfer:**

- Once you have made all settings in the "INCA Project Transfer" window, click on **OK**.

  The transfer of the ASCET project to INCA is executed. First of all the ASAM-MCD-2MC code is generated and the project is refreshed if necessary (i.e. code generated, compiled and linked).

  If you selected an existing project in the "INCA Project" field, it is overwritten.

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

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

**To use Back-Animation:**

- Start the INCA experiment with your project.

Reconnect to Experiment of selected Experiment Target

- In the ASCET project editor, make sure that `INCA` is selected in the "Experiment Target" combo box.
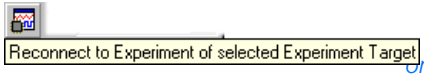
- Klick the **Reconnect to Experiment of Selected Experiment Target** button

  *or*

- select **Component → Reconnect to Experiment**.

  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.
  Section "To select hardware (with ETAS Network Manager):" describes the required actions in this window.

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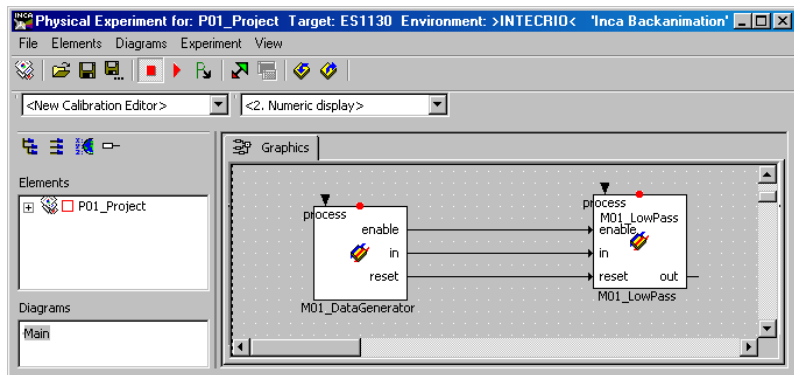


Unlike with the online and offline experiment, this window only contains the "Graphics" tab.

- Create the necessary measure windows (see ASCET User's Guide, "The Measure System" section) and set these up.

- Create the necessary calibration windows (see ASCET User's Guide, "The Calibration System" section) and set these up.

- Select **Experiment → Start Measurement**

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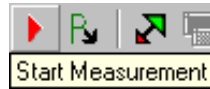
- click the **Start Measurement** button to start measurement.

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

You can load, save and export environments as described in the section "Loading and Saving Environments" in the ASCET User's Guide. When you load an environment which contains unavailable elements (e.g. an oscilloscope), these are ignored.

The monitor function (see the "Monitor" section in the ASCET User's Guide) for monitoring numeric and logical variables is available. You can activate the function for individual or all variables of a component. The setting of the monitor function is saved in the environment.

You can navigate between the components of your project (see the section "Navigating in Block Diagrams" in the ASCET User's Guide).

If your project contains state machines, you can use the animation function for state machines (see the section "Experimenting with State Machines" in the ASCET User's Guide).

You can write data from the experiment into the ASCET model or onto the hard disk; you can also read in data from the hard disk. This is described in the ASCET User's Guide in the section "Manipulating Data".

**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 INCA experiment, however, continues running.

## 4.3 Experimenting with INTECRIO

If you have installed both ASCET-RP and INTECRIO, you can also experiment with your Rapid-Prototyping project in INTECRIO. There is a function for this purpose in the project editor which allows convenient transfer of the experiment.

This chapter contains general instructions for experimenting with INTECRIO. A specific sample task can be found in section 11.1 "Tutorial – Experimenting with 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 55.

- 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` in the build options of the project. `INTECRIO` is preselected in the "Experiment Target" combo box with this target. INTECRIO code generated with this target can be used with each experimental target supported by INTECRIO.

  You can also select the targets `ES1130` or `ES1135`, however, a warning is issued in the ASCET monitor window when you start back animation with these targets.

  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 Activation" field (alarm/software tasks)
- – **Autostart** option (alarm/software tasks)

If you use **Operating System → Copy From Target** to copy the operating system settings to the `Prototyping` target, the deactivated settings are *not* copied.

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

**To call transfer:**

- • Open the project you want to experiment with.
- • Make sure that the `Prototyping` target is selected in the "Build" tab of the project properties.

  The entry 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

*or*
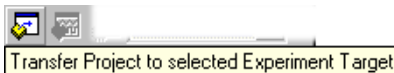
- • select **Component → Transfer Project**

  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.

**INTECRIO Project Transfer**

**Storage location:**

Path: `d:\etasdata\ascet5.2\database\rtio\cgen\P01_Project`  [ Browse ... ]

**INTECRIO Connector:**

Version: `INTECRIO V2.1`  [ Change ... ]

Workspace: [ ]  [ Browse ... ]

System: [ ]  [ Browse ... ]

☐ Trigger INTECRIO Build

☐ Ignore internally connected Messages

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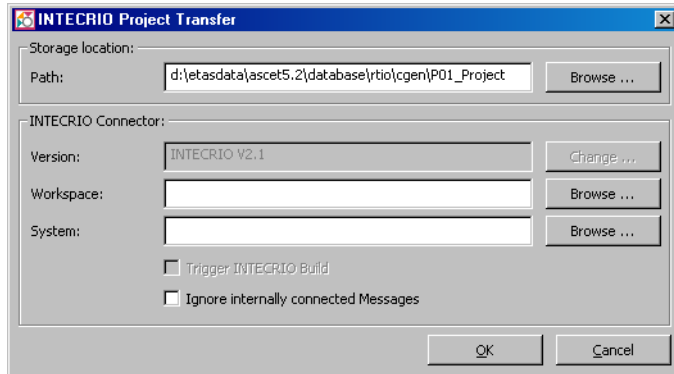
The second step is to enter the path under which the generated files are stored in the "INTECRIO Project Transfer" window. You then have four choices:

• If you only want to generate the *code* required for INTECRIO, all fields except "Paths" remain 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 also have to select the INTECRIO version and enter an INTECRIO workspace. The "Systems" field remains empty.

By default, the version of INTECRIO *last* installed is entered in the "Version" field. If only one INTECRIO version is installed, this is entered; the **Change** button is then 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 the workspace and the 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 useable 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 to convert messages that are read and written in the ASCET model to appear as signal sources, 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.
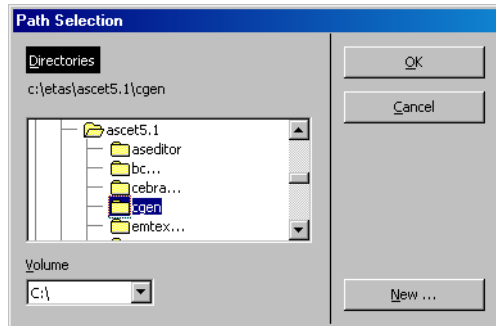


- If necessary, select a volume in the "Volume" combo box.
- Select an existing directory from the "Directories" list

*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 , click the **Change** button next to the "Version" field.

  A selection window opens. It contains all INTECRIO versions installed on your PC.

- Select an INTECRIO version.

- Click **OK**.

  The version is displayed in the "Versions" field. 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.
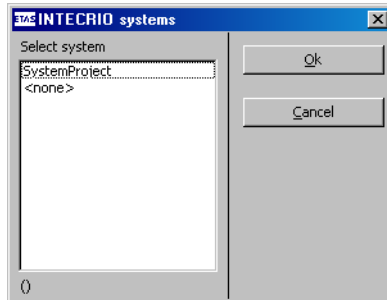


- Select the system project into which you want to add the ASCET project.
- Click **OK**.

  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.

- 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 window opens.
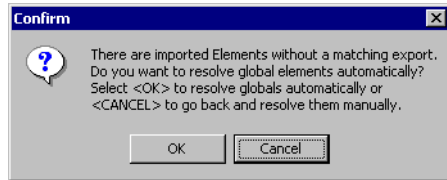
    

    - Click **OK** to automatically resolve the messages.

      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 window opens.
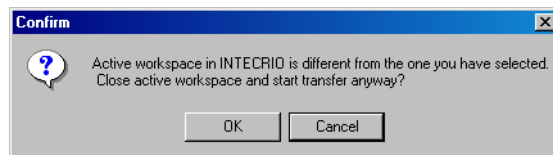
    

    - Click **OK** to close the open workspace and continue the transfer of the ASCET project to INTECRIO.

    *Or*

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

**Rapid-Prototyping Experiments**     **59**

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 (version V1.1) also contains some OS information but none from the deactivated options (see page 52). However, this information is currently not used by INTECRIO.

- `<project name>.a2l`

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

    An 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, a possible 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.

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 INTECRIO User's Guide.

In addition, 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 use Back-Animation:**
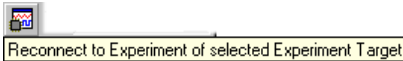


- Start the INTECRIO experiment with your project.

- 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 **Component → Reconnect to Experiment**.

  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.
  Section "To select hardware (with ETAS Network Manager):" describes the required actions in this window.

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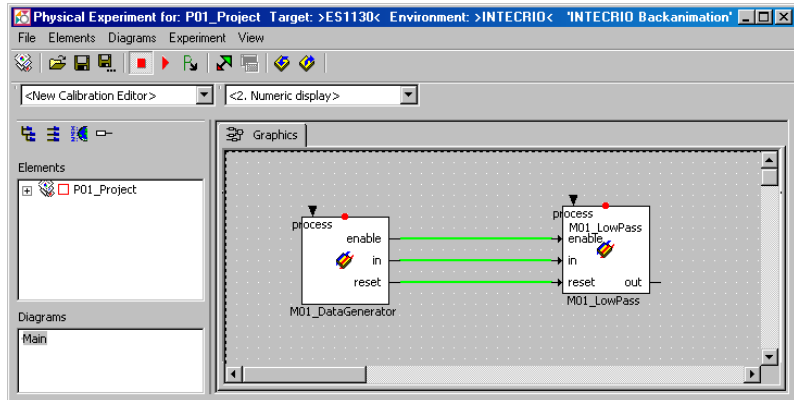


Unlike the online and offline experiment, this window only contains the "Graphics" tab.

• Create the necessary measure windows (see ASCET User's Guide, "The Measure System" section) and set these up.

• Create the necessary calibration windows (see ASCET User's Guide, "The Calibration System" section) and set these up.

• Select **Experiment → Start Measurement**

*or*
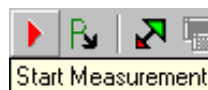
• click the **Start Measurement** button to start measurement.

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

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, the NVRAM cockpit (cf. "NVRAM Cockpit" on page 76) is available.

You can load, save and export environments as described in the section "Loading and Saving Environments" in the ASCET User's Guide. When you load an environment which contains unavailable elements (e.g. an oscilloscope), these elements are ignored.

The monitor function (see the "Monitor" section in the ASCET User's Guide) for monitoring numeric and logical variables is available. You can activate the function for individual or all variables of a component. The setting of the monitor function is saved in the environment.

You can navigate between the components of your project (see the section "Navigating in Block Diagrams" in the ASCET User's Guide).

If your project contains state machines, you can use the animation function for state machines (see the section "Experimenting with State Machines" in the ASCET User's Guide).

You can write data from the experiment into the ASCET model or onto the hard disk; you can also read in data from the hard disk. This is described in the ASCET User's Guide in the section "Manipulating Data".

**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.

# 5 RealTime Input/Output Package

## 5.1 Introduction

The **R**eal**T**ime **I**nput/**O**utput Package (RTIO Package) simplifies the interaction between ASCET and an ETAS experimental system hardware. The ETAS experimental system hardware is a development and experimental platform on a VMEbus basis which can be used universally. The input and output of analog and digital signals is realized with VMEbus interface boards in the ES system.

The RTIO Package means much simpler integration of additional I/O hardware in an ASCET project. A user-friendly user interface enables the configuration of several hardware components, even interrupt-controlled ones.

The imbedding is made by automatic generation of C code for the target system.

## 5.2 Architecture of the RTIO Package

ASCET-RP allows easy prototype development for your ASCET model in a real-time environment on the ES1000 hardware.

For that purpose, the model has to be embedded in your ES1000 environment consisting of your simulation node plus several I/O boards. On the logical level, the ASCET model has to be connected to the I/O channels.
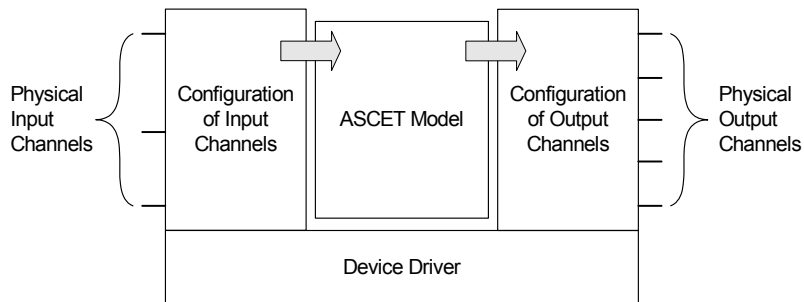


This connection is performed in two steps. The HWC editor (chapter 7) is used to describe and configure the I/O boards, i.e. to configure the physical input and output channels. Each of these channels is then mapped onto one ASCET message, which serves as an input or output to the ASCET model.

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

To generate the prototype, the HWC editor also allows the generation of C code (chapter 8) for the target system, according to the selected experiment type (physical experiment, quantized experiment or imple-mentation experiment). This C code is stored in a C module called HWC (cf. chapter 5.2.1), which belongs to the project and is stored in the ASCET database.
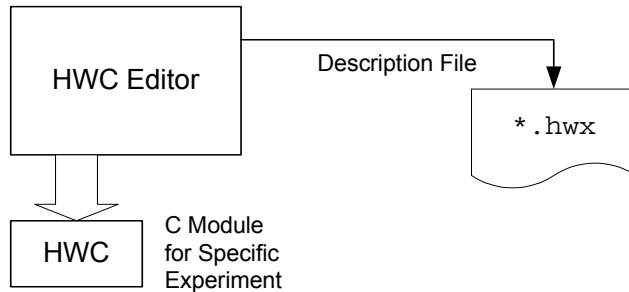


When the code for the entire ASCET model is generated and compiled, the hardware configuration is compiled, too, and linked to the required I/O drivers, to achive an executable prototype.

## 5.2.1    The Hardware Configuration Module

The special characteristic of any ASCET project that uses the RTIO Package, is the Hardware Configuration Module (HWC module). This is a normal C-Code module used as a container for storing the C-Code necessary for the HW link. The RTIO framework identifies the C-Code module by the instance name "HWC".

The HWC module communicates exclusively with the rest of the model using messages. This guarantees a clear separation between HW control and the rest of the model.

The following diagram shows the main connections between the modules in an RTIO project.
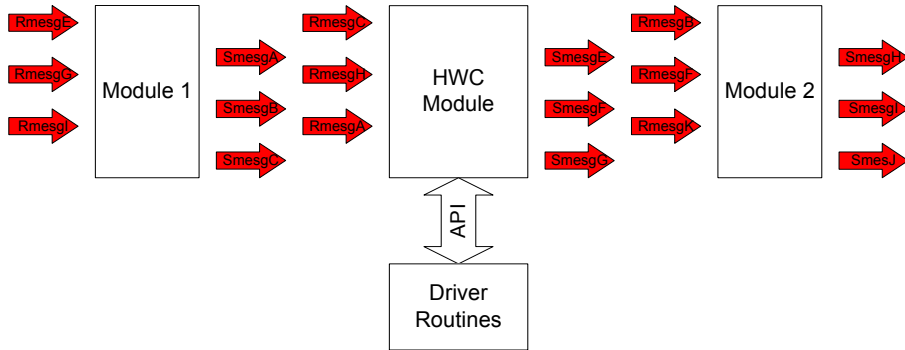


The code generation for the HWC module is coordinated exactly to specific, defined API[1] interfaces to the system-specific and board-specific driver routines. The driver routines are made available to the system as libraries. Apart from these API function calls, the definition of the driver data also takes place in accordance with a standard description structure which is driver-independent.

This results in a modular system that makes it relatively simple to integrate new boards in the RTIO Package.

### 5.2.2 Hardware Configuration Editor

The Hardware Configuration Editor (HWC Editor) is the heart of the RTIO framework.

In Edit mode, the HWC Editor is started from a project and is used to specify the required hardware configuration. A subsequent generation sequence generates all the necessary elements as well as suitable C-Code so that the hardware can be addressed.

---

[1.] API = Application Programming Interface

# 6 Preparatory Measures

This section describes the general prerequisites necessary for the installation and operation of the RTIO Package.

## 6.1 Hardware – ES1000.x Experimental System

Prerequisite for the operation of the RTIO Package is an ETAS experimental system available in various series. The following variants are usually used, although mixing is permissible:

- ES1000.2
- ES1000.3

The RTIO framework supports all ETAS system controller boards currently used. The following diagram shows the standard configurations - special configurations are of course possible.
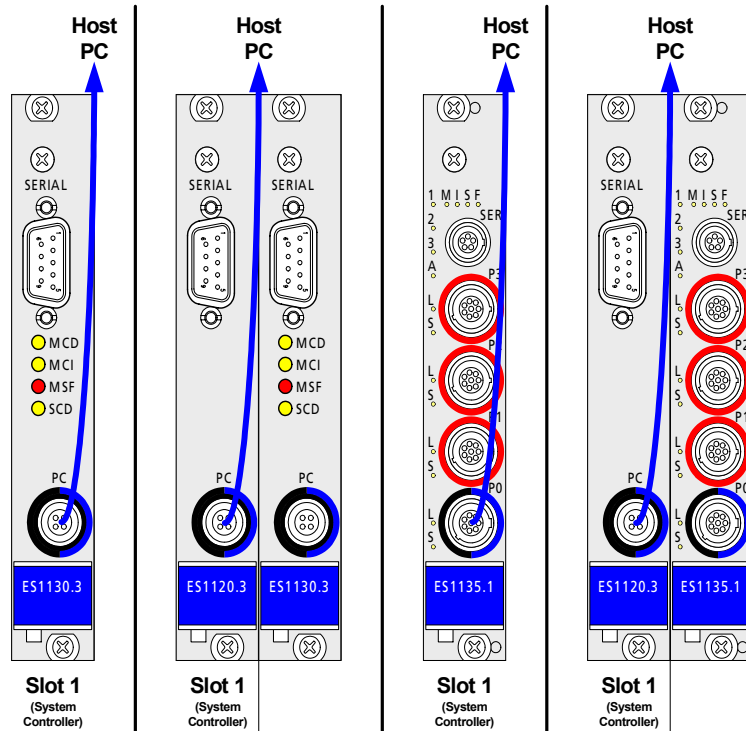


**Fig. 6-1**    Permissible and Supported System Controller Configurations

*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.
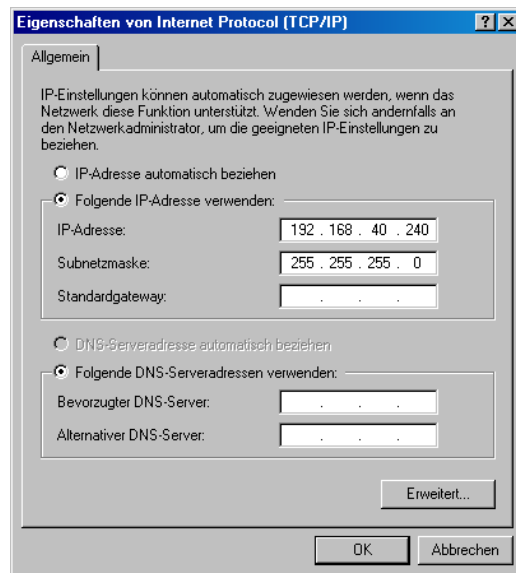


(← Windows® 2000)

- 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.

## 6.2 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.

### 6.2.1 Non-Volatile RAM (NVRAM)

*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.

> ⚠️ **Attention**
>
> *Projects using the NVRAM 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).*

*Hardware Support*

64 kByte of non-volatile RAM (NVRAM) are available in the address space of the ES1135 main processor (IBM750GX). 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 13.3) 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.

The ES1135 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 subclass 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.

*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 ERCOS).

**Stopping the simulation :** When the simulation is stopped (Stop ERCOS), 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 inac-tive).

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 mes-sage 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. How-ever, the API function

```
uint32 nvramClear(void)
```

allows to reset the NVRAM from within the model (see chapter 13.3 "API Functions (NVRAM)").

*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) trig-gered 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 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).

*NVRAM Cockpit*

The NVRAM API (see chapter 13.3) 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 **Experiment → NVRAM Cockpit**

*or*



• click on the **Open NVRAM Cockpit** button.

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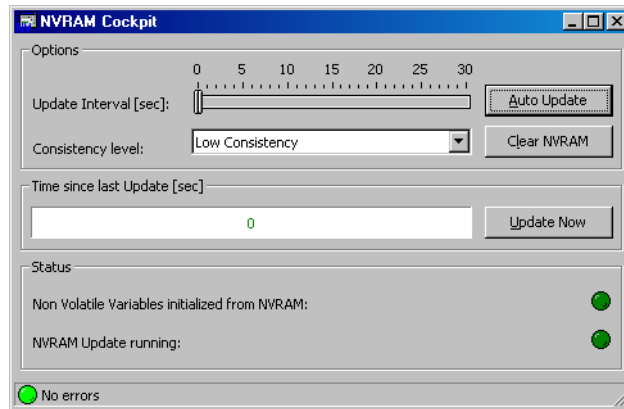
- Use the control elements according to your needs.
- Close the NVRAM cockpit with **Close**.

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).
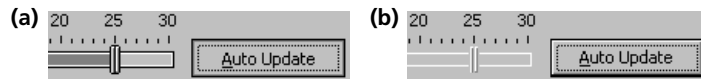


**Note**

*The automatic NVRAM content update works only while the experiment is running. Once you have stopped the experiment with* **Experiment → Stop ERCOS** *or with the* **Stop ERCOS** *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 74.

- **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).

  **(a)** Non Volatile Variables initialized from NVRAM:                    ⬤

  **(b)** Non Volatile Variables initialized from NVRAM:                    ⬤

- **NVRAM Update running**

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

  NVRAM Update running:                                                    ⬤

*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.
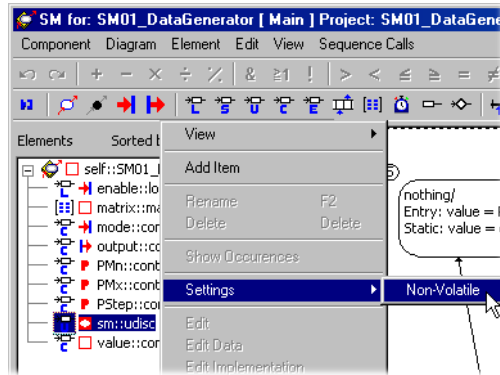


- **Actual size of multi-dimensional elements**

  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 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.

6.2.2    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 vent 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 function only have effects on the interrupt propagation.

The watchdog interrupt is mapped to a HW Task inside ASCET. Inside the ASCET OS editor, the task type `Interrupt` and the ISR Source `Int2` must be selected.
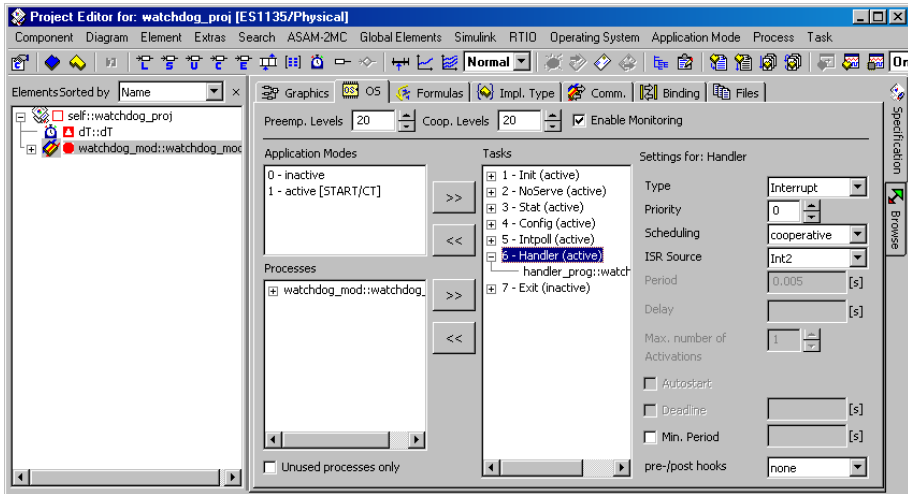


The watchdog handler is running below the ERCOS[EK] level but above other HW interrupts (e.g. from VME Bus), thus the watchdog interrupt is handled even if another HW interrupt is currently handled. Interrupt acknowledgement is done inside the ES1135 firmware, it should therefore not be done inside the 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 µs, 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 13.4 "API Functions (Watchdog)" on page 422.

### 6.2.3 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).
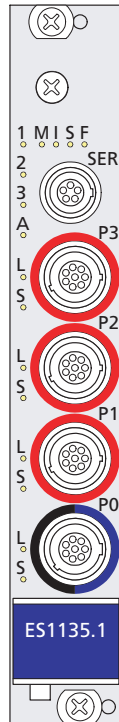


**Fig. 6-2**    ES1135 – Front Panel

The system LEDs are described in the hardware manual.

The programmable LEDs can be accessed either via RTIO (device ES1135-LED, see chapter 10.2) or via the program interface (see chapter 13.5 "API Functions (ES1135 LEDs)")

### 6.2.4 Cache-Locking

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

As a remedy, ASCET-RP V5.6 provides the possibility to mark highly tme-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 assiociative cache, 4-ways), is used for that purpose.

You can mark individual variables or parameters, as well as entire methods or processes (cf. page 83). Components can be marked, too; this mark is adopted by the elements of the component, if applicable (cf. page 84, 84, 85, 87).

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 87). 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, an error message is issued during experiment.

  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:

| | |
|---|---|
| `Automatic` | Variables, parameters, methods and processes adopt the setting of the parent component. This is the default setting. For components, `Automatic` is the same as `Off`. |
| `On` | Cache locking is switched on. |
| `Off` | 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 "Elements" list of a project or component editor, select an included component.
- Select **Element → 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.
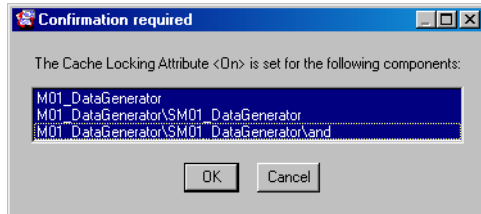


- Select the components to which you want to assign *<setting>*.

- 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 83.*

- 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.
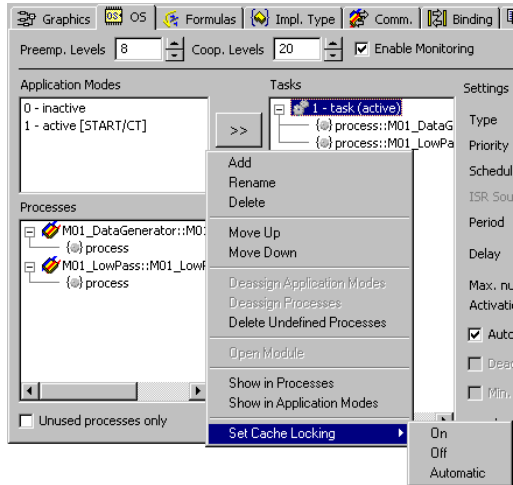


A confirmation window opens that lists the parent components of the processes in the selected tasks, as well as the included components.
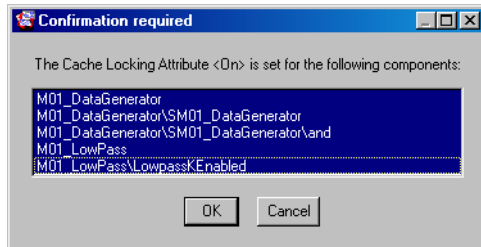


- Select the components to which you want to assign *<setting>*.

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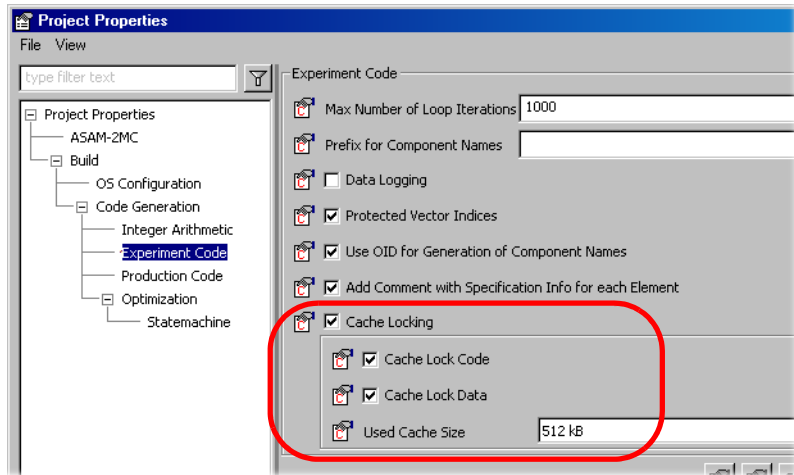
- Click **OK** to close the window and accept your settings.

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

## 6.3     System Software

The RTIO Package described in this manual is either available for or a constituent part of the following software products:

- ASCET Version 5.1.1 and higher with an installed add-on ASCET-RP Version V5.6.0 and higher.

### 6.3.1     System Root Path

With ASCET-RP, a new target directory `ASCET5.1\target` is installed in the subdirectory of your ASCET installation. The system root path must be set to the target subdirectory.

**To determine the system root path:**

- The system root path is set in the "Options" dialog window of ASCET.
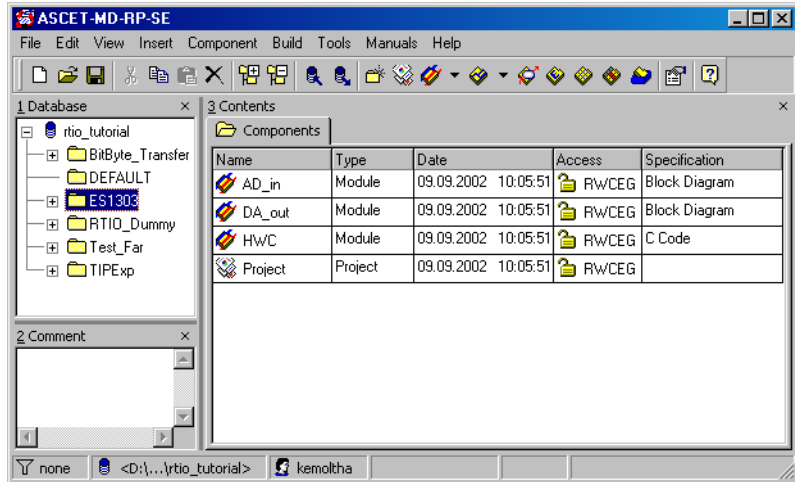
- In the ASCET Component Manager, select
  **Tools → Options**.

  The "Options" dialog window is displayed.
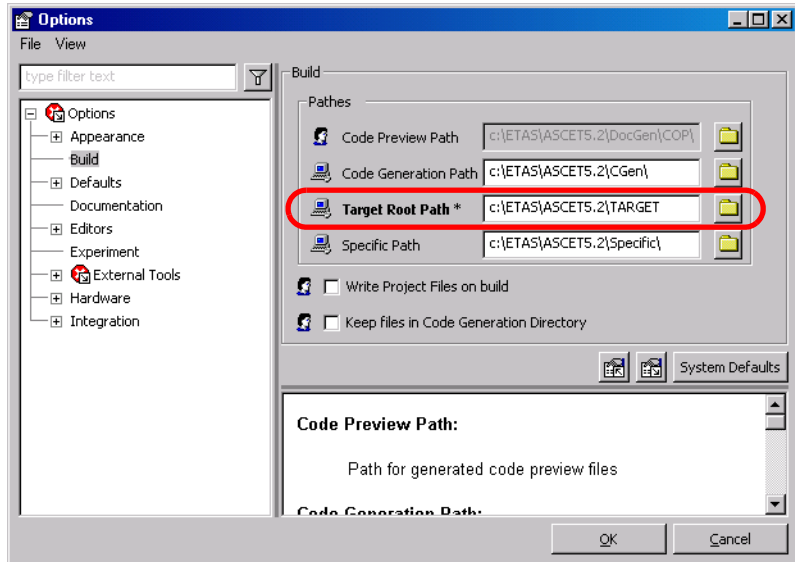


- \TARGET must be selected as "System Root
  Path" in the "Options" dialog window.

### 6.3.2 C-Code Module

Every ASCET project, for which an RTIO hardware link is to take place, requires
a separate C-Code module that has to be generated in the database.

**Note**

*Unlike previous RTIO versions, there are no limitations in terms of the module name ("HWC") in the database, i.e. all names are permissible.*

### 6.3.3 Project

There are a few points that have to be taken into consideration, however, for
an RTIO hardware link to be executed:

- A generated C-Code module has to be added to the project which can
  be used exclusively for RTIO code generation (see previous section). The
  module must contain the instance name "HWC".

- The required target type has to be selected in accordance with the type of system controller used (ES1130 or ES1135).
- The task list in the "OS" tab of the project editor should contain the following tasks so that the RTIO link can be executed as easily as possible:

| Task Name | Task Type | Application Mode | Task |
|-----------|-----------|------------------|------|
| Init | Init | active | Required by the RTIO framework to initialize the hardware drivers |
| Config | Software | active | Required by some hardware drivers for reconfiguration |
| Exit | Init | inactive | Required for releasing driver resources |

Some hardware components also need additional tasks—this is looked at in more detail in the description of the relevant components.

- All messages the RTIO HWC module uses to communicate with the other components in the project (other modules, global messages), have to be generated by the counterpart as "EXPORTED". This is generally the case for send or send-receive messages but you must ensure that this is executed explicitly for receive messages.

Unlike previous RTIO versions, messages generated within hierarchic modules can now also be accessed.

# 7 HWC Editor

The HWC editor is the heart of the RTIO framework and is used to define and describe hardware configurations.

## 7.1 Opening the HWC Editor

First of all, the Project Editor must be opened on the required ASCET project. Providing the RTIO Package has been installed correctly, the main menu bar contains the **RTIO** menu. The HWC editor can now be opened using the menu functions **Open Editor** or **Open Editor with**.
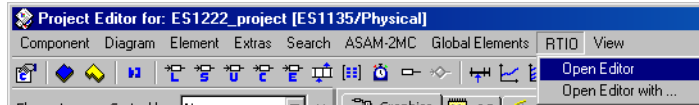


The first time the HWC editor is opened within an ASCET session it takes a little longer as a few system components have to be loaded. (To speed up system start and save resources, some system extensions are only loaded when they are actually needed.)

If the HWC editor was opened with ackage has been installed correctly, the main menu bar contains the **RTIO → Open Editor**, a search is executed in the "Files" tab (File Container) of the ASCET project for an HWC.HWX (or HWC.HWC) file.

If there is such a file, it is automatically loaded in the editor. The HWC.HWX (or HWC.HWC) file is always the one with which the last code generation of the HWC module was executed.
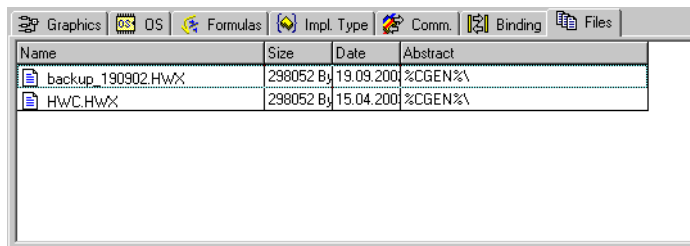


The menu function **Open Editor with** allows the user to select a specific hardware configuration file to be loaded instead of loading HWC.HWX (or HWC.HWC).

In the "HWC Editor Options" window, "Files" tab, you can select the format (*.hwx or *.hwc) you want to use as default. Use **Extras → Options** to open that window.

As an alternative, you can open a particular *.hwc or *.hwx file from the "Files" tab of the project editor. To do so, highlight the file and select the **Project Files → Edit** menu option or the **Edit** context menu.

## 7.2    Controls

This section describes the HWC editor controls.
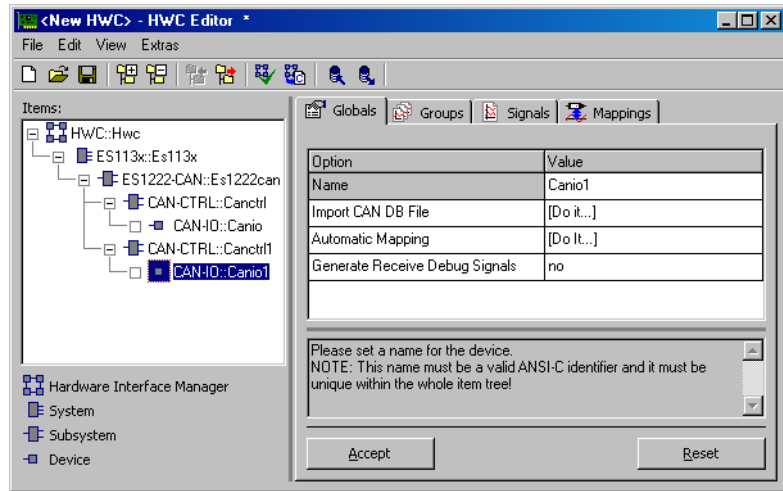


The user interface consists of a toolbar, an "Items" list and a set of tabs which are used for most modifications and entries.

A text field below the tabs contains short information about the selected row. The **Accept** or **Reset** buttons are used to adopt or reject changes, respectively.

### 7.2.1    Toolbar

This section contains a description of the functions that can be activated using the buttons in the toolbar.

> **Note**
>
> *For a more precise description of the individual functions, please refer to the section "Main Menu" on page 98.*

**To create a new hardware configuration:**

- Click on the **New** button to create a new hardware configuration

*or*

- select **File → New.**

**To open a hardware configuration:**

- Click on the **Open** button to open a hardware configuration

*or*

- Select **File → Open.**

**To save a hardware configuration:**

- Click on the **Save** button to save a hardware configuration,

*or*

- select **File → Save**.

- Select **File → Save As** to save the hardware configuration with an arbitrary name.

**To expand all items:**

- Click on the **Expand all** button to completely expand the "Items" list,

*or*

- select **View → Expand all**.

**To collapse all items:**

- Click on the **Collapse all** button to collapse the "Items" list as far as possible,

*or*

- select **View → Collapse all**.

**To search for hardware:**

- To search for connected hardware, click on the **Search for Boards** button,

*or*

- select **Extras → Search for Boards**.

**To add a new item:**

- Click on the **Add Item** button to add a new item to the list of items

*or*

- select **Edit → Add Item**.

**To delete an item:**

- To delete an item, select it in the list and click on the **Delete Item** button

*or*

- select **Edit → Delete Item**.

**To check the hardware configuration:**

- Click on the **Check Hardware Configuration** button to check a hardware configuration

*or*

- select **Extras → Check Hardware Configuration.**

**To start code generation:**

- Click on the **Generate Code For Current Experiment** button to start code generation

*or*

- select **Extras → Generate Code → For Current Experiment.**

**To import a hardware configuration:**

- Click on the **Import** button to import a hardware configuration

*or*

- select **File → Import.**

**To export a hardware configuration:**

- Click on the **Export** button to export a hardware configuration

*or*

- select **File → Export.**

## 7.2.2 "Items" List

The tree-like "Items" list is used to define and display the hardware structure.

The term *item* generally refers to a hardware configuration element. This can either be an entire board or a unit for a specific function.

> **Note**
>
> *Hardware is specified completely if every end node of the hardware tree is of the type "Device".*

The table to the right of the "Items" list always shows the settings for the selected item.

## 7.2.3 Configuration Tabs

Depending on which item is selected in the items list, different tabs are displayed which can be used to make individual settings for the items.

If the selected item is a "Device", 4 tabs are displayed; all other item types only have one tab, "Globals".

The task division of the tabs can be explained as follows:

- **Globals**

  This tab contains all the global settings that refer to the item (e.g. VME-bus address of the I/O board).

- **Groups**

  This tab contains all the settings that refer to a signal group (e.g. gain factor for a multi-channel A/D converter or the identifier of a CAN message).

  > **Note**
  >
  > *All signals in a signal group have the same transfer direction (send or receive) and are acquired at the same time or interval. This is how it is easy to recognize a signal group.*

- **Signals**

  This tab is used for all signal-specific settings (e.g. conversion formula for a signal).

- **Mappings**

  This tab is used to define the allocation between a signal and an ASCET message. The RTIO data flow between the HWC module and the rest of the model can also be controlled and manipulated here.

For more detailed information on tab settings, please refer to "Configuration Tabs" on page 121.

In addition to this, each tab also has the two buttons, **Accept** and **Reset**, which should be used as follows.

**Accept:**

As soon as a tab is opened for the first time, or reopened, a copy is made of the data set in it: this is then displayed in the data table. Any modification of the data made by editing the relevant cell(s) only changes the copy of the original data.

Once editing has been completed, the user has to explicitly write the changed data back to the original data by activating the "Accept" button or he/she is prompted to do so before being allowed to close the tab:
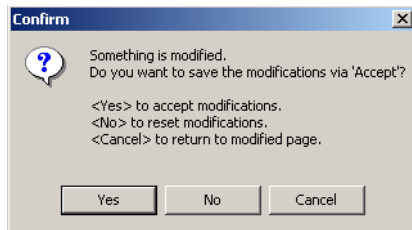


This mechanism is necessary to guarantee a consistent data set at all times as there are sometimes multiple dependencies to be taken into consideration within the "Items" structure.

**Reset:**

This button resets all modifications made after the last clicking of **Accept**.

### 7.2.4 Main Menu

This section describes the menu items in the individual menus.

- **File → New**

  Generates a new hardware configuration. If changes have already been made to an existing hardware configuration but not saved, you are asked the following question:

  

  If you confirm by pressing **Yes**, the configuration in the Project File Container may be saved. Pressing **No** deletes the configuration just created and generates a new hardware configuration. **Cancel** terminates the process.

- **File → Open**

  This menu function opens the "Open Hardware Configuration File" window. Use the "File Type" combo box to select the format (`*.hwx` or `*.hwc`); the "Select from Project Files" list displays all the files of the selected type contained in the Project File Container. Here, you select the file that is loaded in the HWC editor with **OK**.
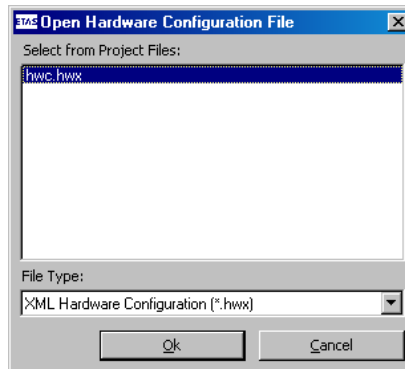
  

  **Note**

  *The* `HWC.HWX` *(or* `HWC.HWC`*) file has a special significance: this file is always generated implicitly or overwritten when code generation is started in the HWC editor.*

- **File → Save**

  Saves the hardware configuration under the same name in the File Container in the ASCET project.

- **File → Save As**

  Saves the hardware configuration under a new name in the File Container in the ASCET project. For that purpose, select the format (`*.hwx` or `*.hwc`) in the "Save Configuration File as" window, and enter the new file name in the "Filename" field.
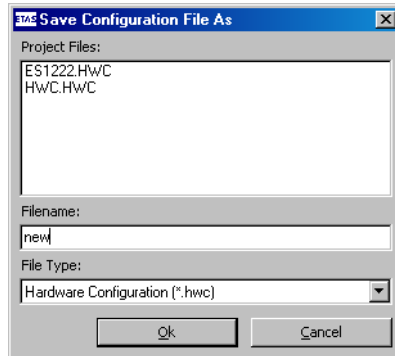
  

  The "Project Files" list contains the existing files of the selected type. To overwrite one of them, click on the name. The file name is written into the "Filename" field. When you klick **OK**, the new hardware configuration is saved under the existing name.

- **File → Save Item as Template**

  This function saves the settings of the item just selected as a template for all items of this type (see **Use Item Templates** in the menu Extras → Options on page 111).

  > **Note**
  >
  > *By default, the hardware configuration is stored in the XML format*
  > `*.itx` *by default.*
  > *The* `*.itp` *format is still valid and can be selected as alternative format. However, it is recommended to use the XML format because the* `*.itp` *format will be discontinued in future ASCET-RPversions.*

It opens the "Save Item As Template" dialog window which can be used to copy the item to the required directory as a template. To do so, you select the format (`*.itx` or `*.itp`) from the "File Type" combo box and enter the required path.
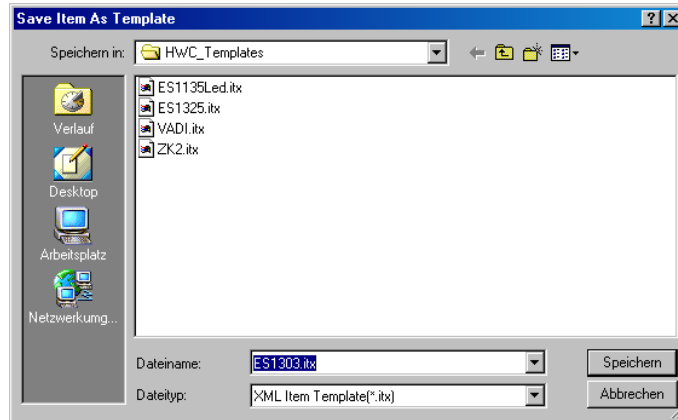
- **File → Import**

    A hardware configuration can be read by the file system using the Import interface. To do so, select the file type (`*.hwx` or `*.hwc`) in the "Import Hardware Configuration" window, select a path and, finally,, the file you want to import.

Existing hardware configurations can be read provided they were cre-
ated with ASCET TIP Exp V4.0.0 or later.
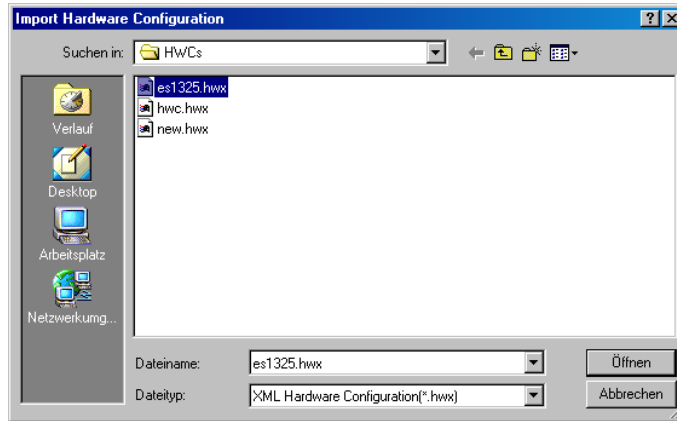


- **File → Export**

    A hardware configuration can be exported to the file system by speci-
    fying a new file name and a file format. Three file formats are available.
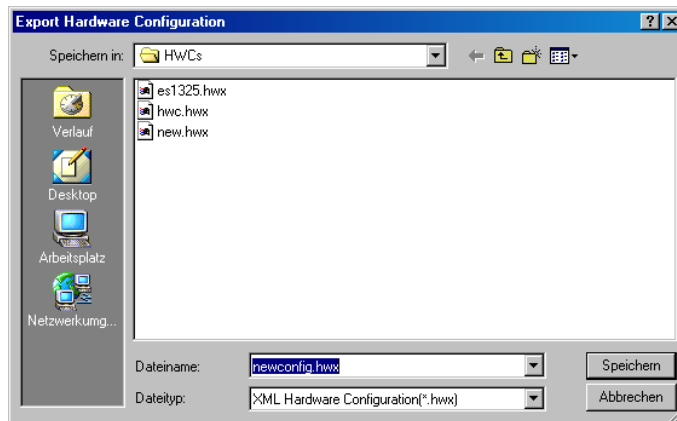
The `*.hwx` and `*.hwc` formats are used for exchanging and archiving of hardware configurations while the `*.csv` (comma separated values) format serves documentation purposes and can be displayed in spreadsheet applications.

> **Note**
>
> *By default, the hardware configuration is stored in the XML format `*.hwx` by default.*
> *The `*.hwc` format is still valid and can be selected as alternative format. However, it is recommended to use the XML format because the `*.hwc` format will be discontinued in future ASCET-RP versions.*

- **File → Exit**

  This menu item closes the HWC editor. If the system detects that a modified hardware configuration has not been saved, it is now possible to save it before the Editor is closed.

*"Edit" Menu*

---

- **Edit → Copy → Item(s)** (<CTRL> + <C>)

  This menu item creates a copy of the selected item with its subtree (if it has one) and transfers it to an internal buffer of the HWC editor.

  The buffer is valid as long as the HWC editor is open.

- **Edit → Copy → Item(s) For Export**

  This menu item works like **Copy → Item(s)**, but the copy is written to an external file (`*.hsx` or `*.hws`). This function is useful for swapping subconfigurations.

> **Note**
>
> *By default, the subtree is stored in the XML format `*.hsx` by default. The `*.hws` format is still valid and can be selected as alternative format. However, it is recommended to use the XML format because the `*.hws` format will be discontinued in future ASCET-RP versions.*

- **Edit → Copy → Item Data**

  This menu item copies the data of an item to an internal buffer (this function does **not** apply to entire subtrees). This function can be used to swap settings between two items.

- **Edit → Paste → To Selected Item** (<CTRL> + <V>)

  This menu item adds an item copied using **Copy → Item(s)** (perhaps with a subtree) below the item currently selected. The following must be true for the function to work correctly:

  1. The item types must correspond to each other; the only exception to this is the possibility to swap item subtrees between system controller items.
  2. It has to be possible to add the item or the subtree below the selected item (the same is true here as with **Add Hardware Item**).

- **Edit → Paste → To Selected Item From Import**

  Works exactly the same as **Paste → To Selected Item**, but the copied part is read from the required file. The same restrictions apply as with the previous menu item.

- **Edit → Paste → Item Data**

  Overwrites the data of the previous item with the data written to the buffer with **Copy → Item Data**.

  Unlike the previous **Paste** menu items, it is also possible to swap data between similar or compatible items (a list of compatible items can be found later). This function is possible if the type of item selected is the same or compatible to the type from which the data was copied. In addition, the structure has to be virtually identical; i.e. the number of rows in the tables in the "Groups", "Signals" and "Mappings" tabs has to correspond.

- **Edit → Add Item** (<INS>)

  This menu item allows you to add a new hardware item to the hardware configuration.

  A selection list is displayed which displays all available items on the hierarchy level immediately below.
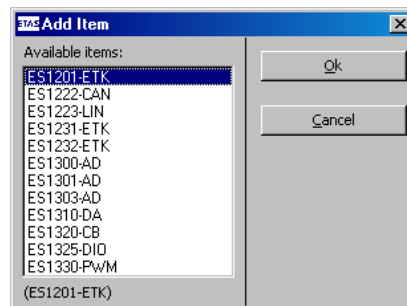
The selected item is then added to the hardware configuration accordingly.

> **Note**
>
> *Depending on the configuration, the process of adding an item may be aborted and an error message issued even if the item has been selected successfully. This happens, for example, if the available resources (ports, slots) are already being used or if a specific adding sequence is obligatory.*

- **Edit → Delete Item** (<DEL>)

  This menu item allows you to remove the marked item from the hardware configuration.

  It is not possible to remove the top item in the hierarchy of the "Hardware Interface Manager" type.

- **Edit → Move Item Up** (<CTRL> + <U>)

  This menu item is used to change the item sequence within a hierarchy level in the hardware configuration. The menu item moves the selected item one element up.

- **Edit → Move Item Down** (<CTRL> + <D>)

  This menu item is used to change the item sequence within a hierarchy level in the hardware configuration. The menu item moves the selected item one element down.

- **Edit → Add Row Before**

  This menu function is only available for items of type "Device" if their number of signal groups or signals can be changed dynamically.

> **Note**
>
> *The system frequently limits the table size. In these cases, the menu functions for adding and deleting table rows are disabled.*

The following must be true for the menu function to be available and selectable (not locked):

1. Item of type "Device" is selected which has "dynamic" features (e.g. CAN-IO device)
2. The "Groups" or "Signals" tab is selected
3. An entry from the "No" column is selected (1st column in the table)

The menu function adds a line *before* the selected cell or the selected area.

**Example:**
A line is to be added in the "Signals" tab in front of the selected cell
area (no. 3-4); i.e. line 3 (signal "Signal 6") is newly created.
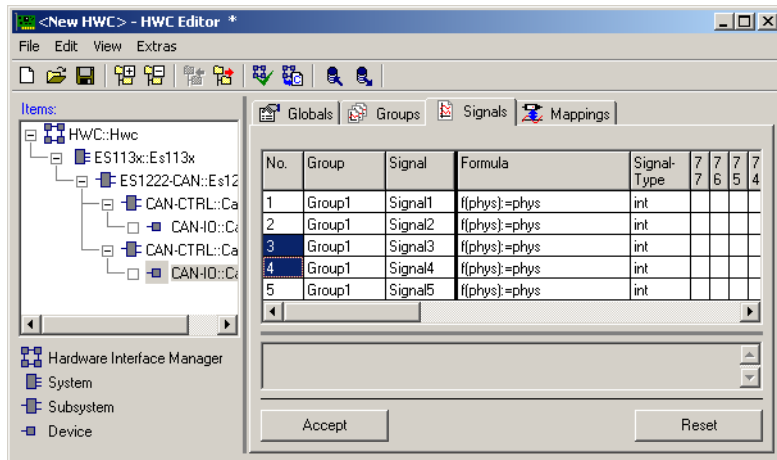


**Fig. 7-1**    HWC editor with a marked area in front of which a line is to be inserted
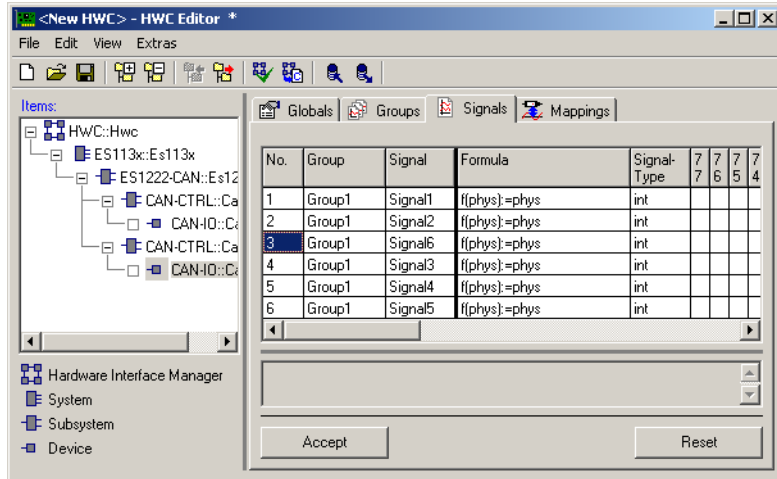
**Fig. 7-2**    HWC editor after a new line is inserted ("Signal 6" signal high-lighted)

- **Edit → Add Row After**

  Works like **Add Row Before**, but a line is added **after** the selected cell or the selected area.

- **Edit → Add Multiple Rows Before**

  Works like **Add Row Before**, but several rows can be added at a time. The number of new lines to be inserted can be entered in a dialog box:
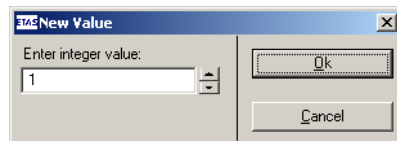


- **Edit → Add Multiple Rows After**

  Works as described under **Add Multiple Rows Before**, but rows are added here *after* the selected cell or the selected area.

- **Edit → Delete Row(s)**

  Deletes the row or the area which is marked in the "No" column.

  **Note**

  *One signal group or one signal must remain in the table for system reasons. If you try to delete the last row, an error message is displayed.*

"*View*" *Menu*

- **View → Expand all**

  This menu function opens the tree structure in the "Items" field as far as possible.

- **View → Collapse all**

  This menu function closes the tree structure in the "Items" field as far as possible.

- **View → Show All**

  This menu function shows all options or columns that can be displayed.

  The table of a tab can be structured so that some of the possible options or columns are not shown by default. This has the advantage that options not often used or rarely needed can be masked out and the table is much clearer.

  **Note**

  *If the table is confirmed with the **Accept** button and the hardware configuration is saved, the reloaded table is displayed exactly as it was saved, i.e. newly shown options/columns remain displayed.*

- **View → Hide All hidable**

  Is the opposite of the previous menu function, i.e. all options/columns that can be hidden are hidden.

- **View → Hide Selected**

  Hides the currently selected option or column if it can be hidden.

"*Extras*" *Menu*

- **Extras → Map Item Signals**

  Opens the "Automatic Mapping" window; see "Automatic Mapping" on page 182.

- **Extras → Check Item**

  This function makes explicit checking of all settings within the selected item on the one hand and checking against the current ASCET project on the other. This checking also takes place implicitly before every RTIO code generation.

  RTIO code generation is only possible when this check is completed successfully. Warnings are permissible, but should be taken into consideration in each individual case.

- **Extras → Check Hardware Configuration**

  This menu item makes explicit checking of all settings possible in terms of consistency within the hardware configuration on the one hand and checking against the current ASCET project on the other. This checking also takes place implicitly before every RTIO code generation.

  RTIO code generation is only possible when this check is completed successfully. Warnings are permissible, but should be taken into consideration in each individual case.

- **Extras → Generate Code → For Current Experiment**

  This menu item starts the RTIO code generation for the experiment currently selected in the ASCET project editor. The following steps are executed:

  1. The content of the entire HWC module is deleted (all processes generated by the HWC module are removed from the task list).

  2. All necessary elements are created in the HWC module (messages, parameters, variables...).

  3. The C-Code for the header and for the required processes of the HWC module is generated.

  4. All processes of the HWC module are sorted into the relevant tasks.

- **Extras → Generate Code → For Phys. and Quant. Experiment**

  This function is exactly the same as the previous function with the difference that the code for the physical and quantization experiment is generated in the HWC module.

- **Extras → Generate Code → For Phys., Quant. and Impl. Experiment**

  This function is exactly the same as the previous one but code is generated here for the physical, quantization and implementation experiment.

- **Extras → Item Code Generation On/Off**

  This function allows to switch on/off the code generation for a selected item (and the structure below). By default, code generation is switched on. When you select this function for the first time, code generation is switched off, and the selected item (and the structure below) is crossed out.
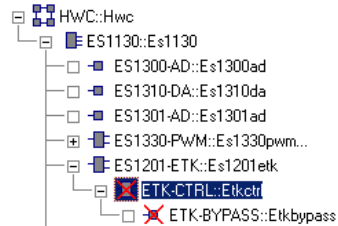
  

  When you generate code now, the crossed-out item(s) are ignored.

  When you select **Extras → Item Code Generation On/Off** once more, code generation is switched on again, and the symbol is no longer crossed out.

- **Extras → Clear HWC Module**

  Deletes the entire content (elements and processes) of the HWC module.

- **Extras → Options**

  Opens the options window in the "Hardware Configuration" node where the settings of the HWC editor can be modified.
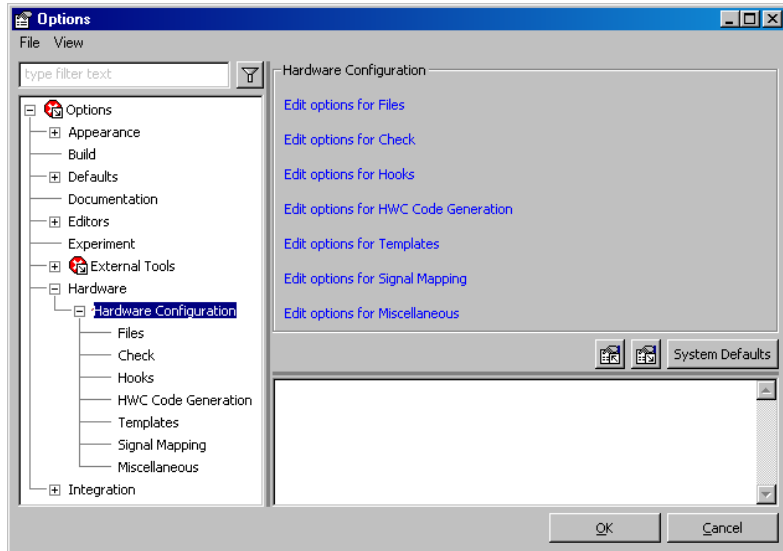


The nodes and their options are described below.

**"Files" Node**

*Default storage format: XML (\*.hwx and \*.itx) / Legacy (\*.hwc and \*.itp)*

This option determines the default format for hardware configuration files and item templates.

*Validate XML files*

This option determines whether the XML code in `*.hwx` and `*.itx` files ist checked for syntactical correctness. When the option is activated, the reading process is aborted when a syntax error is detected.

*Verify XML Checksum*

This option determines whether the checksum is verified upon reading `*.hwx` and `*.itx` files. This checksum verification checks whether the file in question was generated by ASCET-RP and not changed afterwards. If the option is activated, a warning is issued when the checksum verification fails.

*Export ASAM-2MC for Bypass Items*

If the option is activated, and the user performs an HWX export that includes one or more bypass items, each ASAM-MCD-2MC file assigned to these bypass items is exported to the same directory as the `*.hwx` file.

If the option is deactivated, or the user performs an HWC (Legacy) export, only the `*.hwx` or `*.hwc` file is exported.

**"Check" Node**

*After loading*

Activates or deactivates implicit checking which is executed when a new hardware configuration is loaded.

*Before saving*

Activates or deactivates the checking of the hardware configuration executed automatically before a configuration is saved.

*After "Accept"*

Activates or deactivates automatic checking that is initiated when the **Accept** button is pressed. For performance reasons, only the data consistency within the tab currently selected is checked.

**"Hooks" Node**

*Post Check Hook*

Activates or deactivates the usage of an additional user-defined check after the check via **Check Hardware Configuration**.

---

**Note**

*The following three options are only available if **Post Check Hook** is activated.*

---

*Post Check Command*

Contains path and filename of the user-defined check command.

*Post Check HW Configuration File*

Determines where the hardware configuration is saved prior to calling the command.

*Post Check Log File*

Determines where the file containing warnings and errors from the user-defined check is stored after the execution. Messages in the log file conform to the following format description:

```
<log message entry> --> <log message type> ': ↵
    (' <location> ') >>> ' <log message>

<log message type> --> {ERROR|WARNING}

<location> --> <hwc item name> '::'
    <hwc item type> '"' <tab name> [' \ '   ↵
    <column or row name> [' \ ' <row number>]]

<hwc item name> --> HWC item name as in    ↵
    hardware configuration

<hwc item type> --> HWC item type as in    ↵
    hardware configuration

<tab name> --> {'Globals' | 'Groups' |     ↵
    'Signals' | 'Mappings'}

<row or column name>--> Name of the row (for ↵
    'Globals' tab) or column (other tabs)   ↵
    to which the message belongs

<row number> --> Row number for tabs except  ↵
    'Globals'
```

The existence of the log file is be checked after command execution. A non-existing log file is interpreted as error.

*Post Generation Hook*

Activates/deactivates the usage of user-defined code generation.

**Note**

*The following three options are only available if* **Post Generation Hook** *is activated.*

*Post Generation Command*

Contains path and filename of the user-defined code generation command.

*Post Generation HW Configuration File*

*Determines where the hardware configuration is saved prior to calling the command.*

*Post Generation Log File*

Determines where the file containing warnings and errors from the user-defined code generation is stored after the execution. Messages in the log file conform to the same format description as for "Post Check Log File".

### "HWC Code Generation" Node

*Automatic Repair*

Activates or deactivates the possibility of automatically recovering a few errors discovered in the hardware configuration. This possibility is, however, not supported very frequently at the moment.

*Store HW Configuration in generated C-Code*

If activated, the hardware configuration, in an XML format, is added as a comment to the C code of the HWC module.

**Note**

*A hardware configuration can be of significant size.*

*Log Level: silent / brief / verbode*

Specifies the scope of the protocol which is generated in the HWC code generation.

### "Templates" Node

*Use Item Templates*

Activates or deactivates the use of item templates.
Item templates are used to overwrite the default values specified by the system with individual settings. When the Template option is activated, the default values specified by the system are overwritten by the values that were saved in the template directory (using the **Edit → Save Item As Template** menu) each time items are added (using the **Edit → Add Item** menu).

*Template Path*

This allows you to select the directory that is used to store the item templates.

**"Signal Mapping" Node**

*Message Creation Target: Project / Module*

This option determines whether ASCET messages generated during automatic mapping are created in the project itself (Project, default) or in one of the included modules.

*Module Instance Name*

This option is only effective when "Message Creation Target" is set to Module.

Enter the name of the module for the generated messages in the input field. You can use any module included directly or indirectly (in another module) in the project. For modules in modules, the module names are separated by a dor. If, e.g., module A contains module B, you enter A.B if you want to create the messages in module B.

If the selected module is not included in the project, the following error message appears:

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

*Create Signal Type specific Messages: no / yes*

This option determines whether the signal type is considered when ASCET messages are generated. If the option is set to yes, the model type of a message is determined from the signal type or signal according to the following table.

| Device | Signal type (CAN/LIN/ETK) or Signal | Model type of message |
|---|---|---|
| CAN-IO, CAN-Bypass, LIN-IO, ETK-BYPASS, ETK-BYPASS-ADV | real | cont |
| | uint[a] | udisc |
| | int[a], (s)int[b] | sdisc |
| | Bool, uint (only one Bit set to 1) | log |
| ES1300-AD, ES1303, ES1310, ES1330-PWM | (all signals) | cont |
| ES1135-LED, ES1320-CB/DIO, ES1325-LED | (all signals) | log |

| Device | Signal type (CAN/LIN/ETK) or Signal | Model type of message |
|--------|--------------------------------------|------------------------|
| ES1325-INPUT, ES1325-OUTPUT | Error State, State | log |
| | Active Time, Additive Active Time, Duty Cycle, Frequency, Inverse Duty Cycle, Inactive Time, Period Duration, Ratio Active Time/Inactive Time, Ratio Inactive Time/Active Time | cont |
| | Counter Value, No. of Periods | udisc |

a: If the formula of a bypass signal does not correspond to the identity, an ASCET message of model type cont is generated.
b: only available for CAN-IO and LIN-IO

If the option is deactivated, messages of model type cont are generated in all cases.

*Create Signal Direction specific Messages: no / yes*

This option determines whether the signal direction is considered when ASCET messages are generated. If the option is set to yes, the signal direction determines the message tybe according to the following table:

| Transfer direction | Message generated in module/project |
|--------------------|--------------------------------------|
| send | send message |
| receive | receive message |

If the option is deactivated, send&receive messages are generated.

**"Miscellaneous" Node**

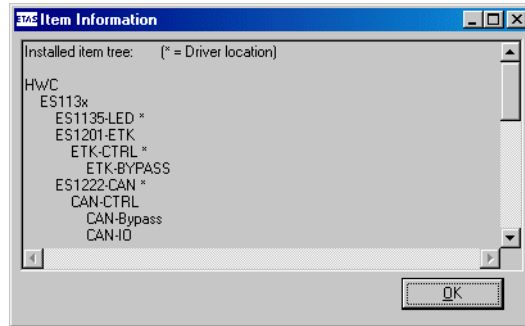*Data digits after decimal point: 1...6*

*Data digits before decimal point: 1...9*

These two options define the format in which the values in the "Data" column in the "Mappings" tabs of the devices are displayed.

Representation: <Data digits before...>.<Data digits after...>

- **Extras → Installed Items**

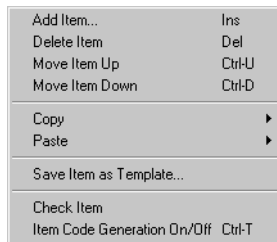  This menu function lists all items currently supported by the system and their hierarchic dependencies.



- **Extras → Search for Boards**

  Starts the automatic hardware detection, see chapter 7.3.

### 7.2.5 Context Menu ("Items" List)

In the "Items" list, you can use the following context menu.



The context menu functions are also accessible via the main menu in the menu bar. They are described in previous chapters.

### 7.3 Automatic Hardware Detection

In ASCET-RP V5.6, the HWC editor offers the automatic hardware detection, an overview of the boards connected to the system, and their IO functionality. You can display the overview, as well as insert boards from the overview into the hardware configuration.

**To start automatic hardware detection:**

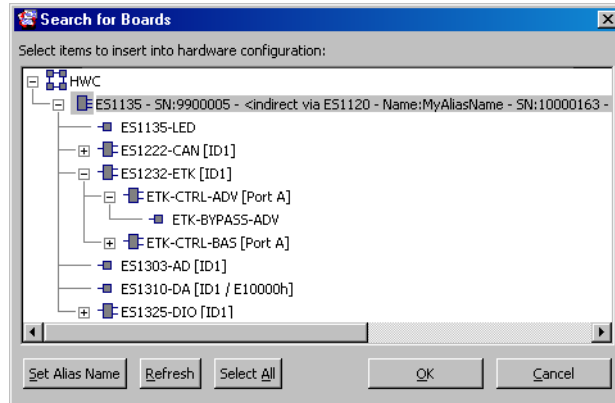- In the "Items" field, select an entry.

- Click on the **Search for Boards** button

  *or*

- select **Extras → Search for Boards**.

  The detected hardware is displayed in the "Search for Boards" window.

The "Search for Boards" window is similar to the hardware selection window described in chapter 2.2.1.



- Field "Select items to insert into hardware configuration" (display field)

  This field displays, below the main entry HWC (symbol ), all simulation controllers (ES113x, symbol  ) connected with the PC. The simulation controller label contains, in additon to stating whether the ES113x is connected directly or indirectly to the PC, further information; see page 20.

  Available boards (symbol ), with all available subsystems (symbol ) and devices (symbol  ), are displayed below the simulation controller.

- Button **Set Alias Name**

  You can use this button to assign an arbitrary name to the ES113x or ES1120.

- Button **Refresh**

  Selects the entire content of the display field.

- Button **Select All**

  This button updates the display 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 insert the selection into the "Items" list of the HWC editor, or click **Cancel** to close the "Search for Boards" window without inserting the selection.

**To insert hardware from the overview into the configuration:**

- In the "Search for Boards" window, selectone or more items.

  When you select an item, the parent items are inserted, too, if they do not exist. Child items are not inserted automatically, they must be selected explicitly.

  When inserting, e.g., a device, the respective board is inserted as well. If, on the other hand, the board is inserted, respective devices are not inserted.

*Or*

- Click **Select All** to select all detected hard-ware.

  For boards with several, mutually exclusive, subsystems or devices (e.g., ES1222 with CAN-IO and CAN-Bypass), a valid default configuration, i.e. one of the items, is selected. The other items remain unconsidered.
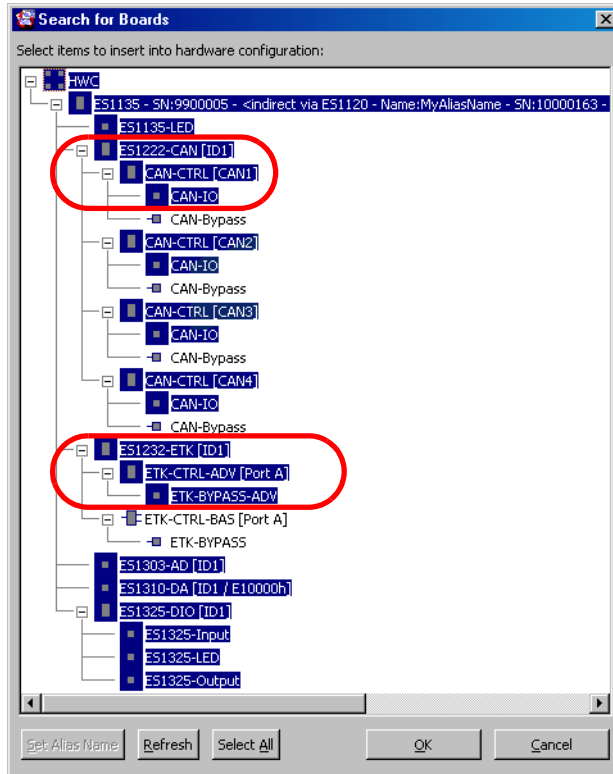


In detail, the default configurations look as follows:

| ES1222-CAN | ES1232-ETK |
|---|---|
| CAN-CTRL | ETK-CTRL-ADV |
| CAN-IO | ETK-BYPASS-ADV |

- Click **OK** to insert the selected items.

  Items not yet present in the "Items" list are inserted, existing items are skipped. A message window shows the result.
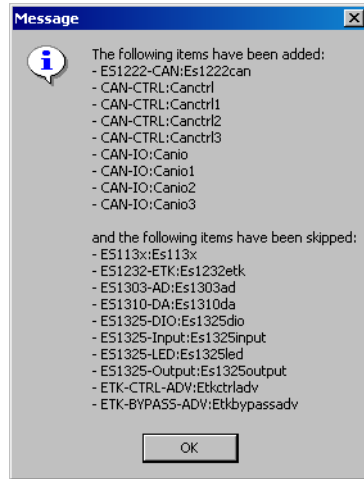


## 7.4 Configuration Tabs

All item settings can be made in the Configuration tabs. The "Globals" tab is available for all item types; the "Groups", "Signals" and "Mappings" tabs are only available for the "Device" items.

### 7.4.1 General Tips

*Editing the Option / Cell*

A cell can be selected and edited by clicking the mouse in the tab table. You can also navigate in the table using the arrow keys.

Some columns are also capable of being selected several times, i.e. several cells can be selected within the column at one time (e.g. the "Formula" column).

Multiple selection possibilities of contiguous fields of a column:

- Click the left-hand mouse button and drag the mouse at the same time over the desired fields of the column

  *or*

- click the left-hand mouse button in the first and the last desired field of the column by pressing the <SHIFT> key.

  By pressing <F2>, or with a further mouse click while pressing the <Shift> key (works *only* with combo boxes and input fields), the edit mode is reached.

- A multiple selection with the keyboard is also possible by keeping the <SHIFT> key held down.

  With pressing <F2>, the edit mode is reached.

Multiple selection possibilities of discontiguous fields of a column:

- A multiple selection can take place by clicking the left-hand mouse button and keeping the <CTRL> key held down. The last field to be selected must be marked with one double-click .

  The edit mode is reached.

- A multiple selection is also possible by clicking the left-hand mouse button and by keeping the <CTRL> key held down. All desired fields must be marked.

  With pressing <F2>, the edit mode is reached.

<ENTER> confirms the input, while <ESC> cancels the edit mode.

*Editing*

Not all options can be modified by the user; some options are permanently locked and are only used to display certain status values. Other options can be locked or ready to accept entries for some of the time which might depend on other option settings in this or another item in the items hierarchy.

*Modified Status*

The HWC editor registers whether a table value has already been modified or whether it is currently being modified. The following message is displayed if another item or tab is selected without the change first being made valid using the **Accept** button:
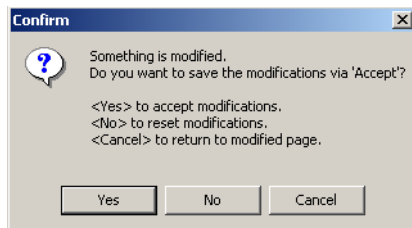
Save the changes with **OK**, or cancel the switch to another item or tab with **Cancel**.

*Scrolling the Table*

The table may be too big to be displayed in a tab. If this is the case, "scrollbars" are displayed.
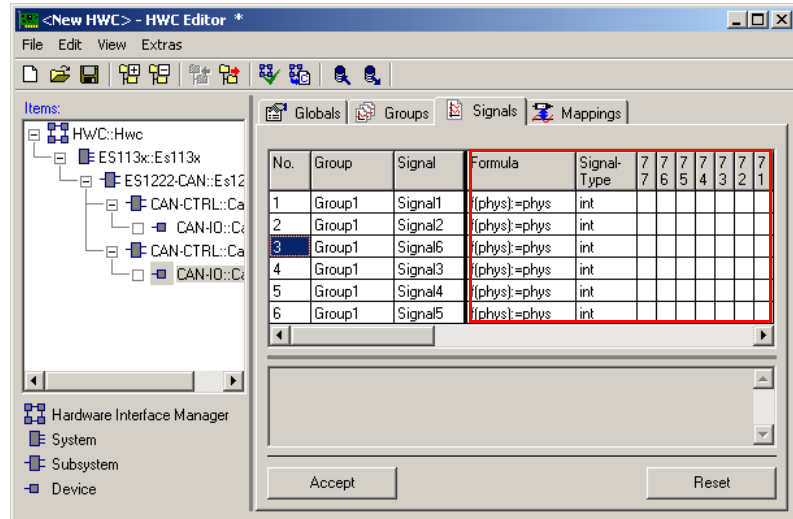


**Fig. 7-3**     Scrollable Area of a Configuration Tab

The scroll bars do **not** move the entire table, but just the variable part (the marked area in Fig. 7-3); the rest remains unchanged.

> **Note**
>
> *There might be so many columns displayed in the left-hand (static) part of the table that the right-hand, scrollable part no longer fits into the tab which makes the user think that "scrolling" does not work. This problem can either be solved by enlarging the HWC editor or by hiding static columns that are not required.*

*Changing the Width of the Columns*

The width of the columns can be changed as required by clicking the left-hand mouse button on the edge of the column, and keeping it held while you adjust the width. The newly specified column widths are then saved with the hardware configuration providing they are confirmed using the **Accept** button.

After a reload, they are shown as required.

*Help Text*

The text field at the bottom of the tab allows you to display a short help text on the option currently selected.

### 7.4.2 Default Options in the "Globals" Tab

This section describes the default options in the "Globals" tab. Further item-specific options can be found in the respective sections in chapter 10 "HWC Items".

*Name*

This is where an individual name can be specified for the item. The name must be a valid, ANSI-C compatible name and be unique in the entire hardware tree.

This option is available in every "Globals" tab.

*Init Task*

This is where you select the task in which the hardware driver is to be initialized. This is usually a "real" init task. In exceptional cases, it can also be a "software" task, but this would be quite unusual and results in a warning being displayed (see the section "Preparatory Measures" on page 69).

This option is available if the item is connected with a low-level driver.

**Note**

*When tasks are being processed, you must always ensure that this task is executed before another task, connected to this driver, (Start, Config...), can be executed. You must also ensure that the "Init" call only occurs after there has been an "Exit" call. If not, there is a runtime error in the experiment which results in the driver being "locked" (Driver Lock)!*

A consistent task sequence must be adhered to:

1. Init Task
2. Start Task
3. Stop Task
4. Exit Task

*Exit Task*

This is where the task is selected in which the hardware driver is to be deinitialized. This is usually a "real" exit task (= inactive init task), could, however, also be a "software" task but this would be quite unusual and certainly result in a warning being displayed (see the section "Preparatory Measures" on page 69).

An "exit task" is available if the item is connected with a low-level driver.

**Note**

*Please stick to the task order (see "Init Task" on page 124)!*

*Config Task*

The choice of a config task to change definite configuration parameters at runtime is contained only for reasons of the compatibility with other ETAS products (LabCar). For the use of ASCET-RP is this position not relevant.

*IRQ Handler Task*

The task selected here must always be a "software" task, it must be generated as a "software" task in the task list in the ASCET project editor. At least 2 (max. 50) must be entered in the "Max. No. of Activations" field.

An "IRQ Handler Task" can be required by drivers which work in Interrupt mode.

**Note**

*This task must be available **exclusively** for the item or the relevant hardware driver and must not be used by any other item or user process. Two items of the same type (e.g. two ES1222s) are **not** allowed to share the same task!*

*Device Manager Task*

This task is required by some drivers to ensure a certain basic supply of the driver independent of the actual data exchanged (e.g. error handling, bus monitoring).

Usually a timer task has to be specified here. The cycle time can vary considerably and is specified in more detail in the tab's help text or in the relevant item documentation.

*Version*

Displays the version number of the item which can be used to check the version with the low-level hardware driver.

"Version" is hidden by default.

*Format*

Display of the format of the item which can be used for the compatibility check with the low-level hardware driver.

"Format" is hidden by default.

### 7.4.3 Default Options in the "Groups" Tab

This section describes the default options in the "Groups" tab. Further item-specific options can be found in the respective sections in chapter 10 "HWC Items".

*No*

Is responsible for the numbering of the lines so that, for example, error messages can be assigned.

*Device*

This is where the corresponding item name is displayed.

"Device" is hidden by default.

*Group*

This is where the designation of the signal group is specified. This name can usually be edited, but has to be a valid ANSI-C name.

> **Note**
>
> *All signal groups within one device have to have different names!*

*Direction*

Specifies the transfer direction of the relevant signal group. If the setting is locked, the transfer direction is specified by the hardware and cannot be modified (e.g. an A/D board is always "receive", a D/A board is always "send"). With some devices (e.g. CAN-IO), you can choose the transfer direction.

*Task*

This specifies the task in which the data transfer is to take place. It is also possible to specify several tasks - data transfer then takes place in each of the tasks. "Timer" tasks or "software" tasks are generally permissible. With some devices, no specification can be made here; data transfer then takes place in a different way, usually interrupt-controlled.

**Note**

*If more than one task is selected for the same signal group, consistency problems may occure during execution time because the accompanying RTIO process is then assigned to more than one task.*
*The RTIO processes generated by the HWC editor however are non-reentrant. Therefore, you must avoid to construct OS configurations that may cause coincidences at the execution of the RTIO process.*
*Most simply, this can be achieved by using exclusively cooperative tasks when assigning more than one task to the same signal group. If preemptive tasks shall be used nevertheless, it lies in the responsibility of the user to ensure a correct OS configuration.*

### 7.4.4    Default Options in the "Signals" Tab

This section describes the default options in the "Signals" tab. Further item-specific options are given in the respective sections in chapter 10 "HWC Items".

*No*

Is responsible for the numbering of the lines so that, for example, error messages can be assigned.

*Device*

This is where the corresponding item name is displayed.

"Device" is hidden by default.

*Group*

This is where the corresponding signal group name is displayed. With some devices, such as "CAN-IO", the allocation of the signal to a signal group can be defined freely.

"Group" is hidden by default.

*Direction*

This is where the transfer direction of the corresponding signal group (and hence the signal) is displayed.

"Direction" is hidden by default.

*Task*

This is where the assigned task is displayed in which the relevant signal group is transferred.

"Task" is hidden by default.

*Signal*

This is where a signal name can be specified. With some devices, the specification of the signals is also fixed and cannot be modified by the user. If an input is possible, the name must correspond to ANSI-C guidelines.

**Note**

*All names of signals which are allocated to a signal group must differ!*

*Formula*

If necessary, a signal value can be assigned a linear conversion in this column. A 1:1 conversion is selected by default (formula `f(phys):= phys`). With some devices, this formula is fixed and the column locked. This is the case, for example, with the ETK-BYPASS device with which the signal is generated from ASAM-MCD-2MC information and the formula is therefore fixed.

This formula is used during signal processing. Signal processing includes three layers: transformation, quantization, and mapping. A signal group received from the RTIO driver is directly sent to signal processing.

In the *transformation layer*, the signal group is split into the individual I/O signals. With the help of the formula specified in the "Signals" tab, the I/O signals are converted into the physical representation of the corresponding ASCET messages.

For quantisation or implementation experiments, the *quantization layer* adjusts these physical signals to the quantization of the ASCET messages. Here, the formula specified in the implementation of the respective ASCET message is used.

In the *mapping layer*, the quantized signals are copied onto the corresponding ASCET messages.

## 7.4.5 Default Options in the "Mappings" Tab

This section describes the standard options in the "Mappings" tab. Further item-specific options can be found in the respective sections in chapter 10 "HWC Items".

*No*

Is responsible for the numbering of the lines so that, for example, error messages can be assigned.

*Device*

This is where the corresponding item name is displayed.

"Device" is hidden by default.

*Group*

This is where the corresponding signal group name is displayed.

"Group" is hidden by default.

*Direction*

This is where the transfer direction of the corresponding signal group (and hence the signal) is displayed.

"Direction" is hidden by default.

*Task*

This is where the assigned task is displayed in which the relevant signal group is transferred.

"Task" is hidden by default.

*Signal*

This is where the signal name is displayed.

*ASCET Message*

If necessary, an allocation to an ASCET message can be made here ("mapping"). All EXPORTED messages in the project which correspond to the current signal are displayed to the user in a selection dialog window. These are all exported receive messages for a "receive" signal and all exported send messages for a "send" signal.

In the "Message selection" dialog window, the required message can be searched for using a character string whereby the character string is shown at the bottom of the dialog window.
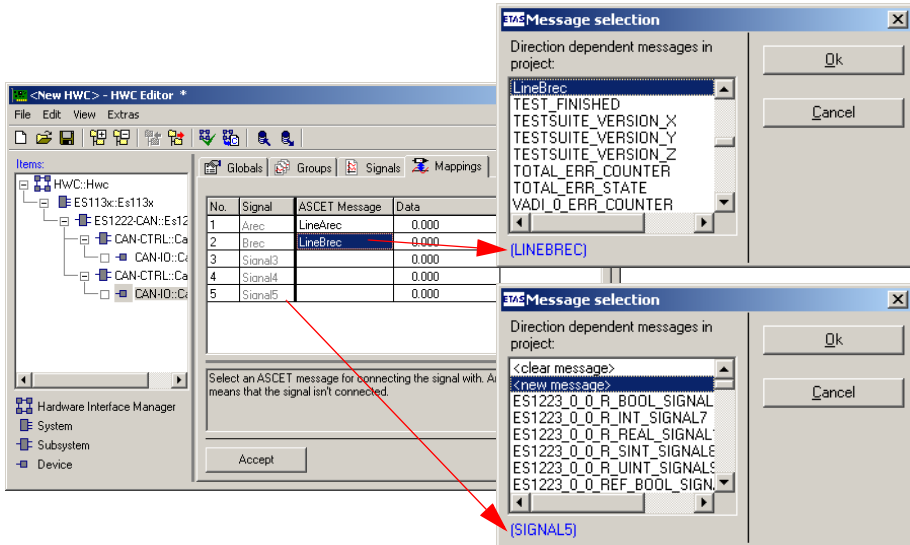


**Fig. 7-4**    "Message selection" Dialog Window

The following takes place when the selection window is opened (see Fig. 7-4):

- • if there is a message in the "ASCET Message" column, the message is used as a search key
- • if there is no message in the "ASCET Message" column, the relevant signal name is used as a search key

A message entered in the "ASCET Message" column can be deleted in the "Message selection" dialog window using `<clear message>`.

A new message can be generated by selecting `<new message>`. The following dialog window opens:
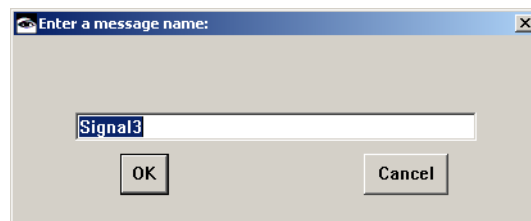
The new message name, by which the message is to be entered in the project as a global "send-receive message", can be entered in the dialog window.

*Data*

Here, the default values for send messages are entered. The RTIO code generation sets the ASCET parameters to the specified values.

*Explanation*

This column contains information about the signals. Not all items have this column.

# 8 Code Generation

RTIO code generation cannot be started until the implicit check of the hardware configuration has been completed without any errors being returned (warnings are permissible but should certainly be taken seriously).

Code generation first deletes the entire content of the HWC module (elements, code and processes) before the contents are newly regenerated.

> **Note**
>
> *The user should never modify the generated code manually as correct functioning cannot be guaranteed after manual changes are made. It could even result in the hardware being damaged in extreme cases!*

## 8.1 HWC Module

This section gives a brief overview of the generated code. This will give an experienced user a feel for the working of the RTIO link and alleviate error search.

This section is not of interest to an inexperienced user.

### 8.1.1 Elements

The following diagram is intended to give an overview of the data flow between the project and the HWC module:



x = <ItemName>,[<GroupName>],[<SignalName>]
y = any usual ASCET Message name
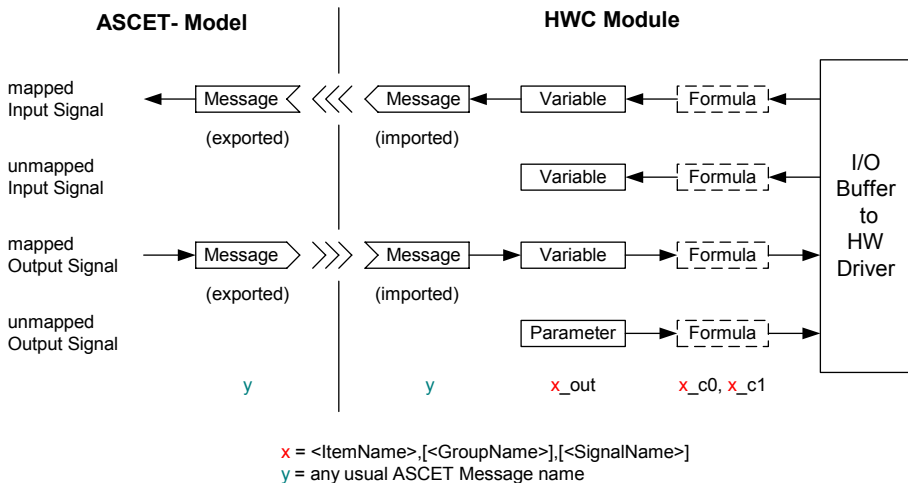
*Messages*

For every entry in the "ASCET Message" column ("Mappings" tab) in the HWC Editor, a message element with the same name is created which is always "imported" (shown by "y").

*I/O Interface*

Every signal that communicates with the hardware driver is routed within the HWC module via an I/O interface (shown by x_out). Depending on whether it is a "send" or "receive" signal or whether the signal is "mapped" to a message, either a variable or a parameter with the name x_out is generated.

*Formula*

If a linear conversion formula has been selected for the signal, there are relevant parameters for the coefficients c0 and c1 .

*Configuration Parameters*

An additional calibration parameter is generated for every configuration value. The name of the parameter is composed of the "Item" and/or "Group" prefix and a parameter designation which, however, is individual for each item.

*External Code*

The code for an interrupt vector table and for the necessary interrupt vectors is generated in the external code of the HWC module.

*Header Code*

The data buffers for data exchange with the low-level drivers and the data structures for initializing the drivers are defined in the header code.

*Processes*

The processes generated by the RTIO framework are divided into processes for driver control and processes for data exchange.

The following processes exist for driver control:

- <item_name>_InitCode_<task_name>_HWCF
- <item_name>_StartCode_<task_name>_HWCF
- <item_name>_ConfigCode_<task_name>_HWCF
- <item_name>_DeviceManagerCode_<task_name>_HWCF
- <item_name>_AcknowledgeCode_<task_name>_HWCF
- <item_name>_AnalyzeCode_<task_name>_HWCF
- <item_name>_StopCode_<task_name>_HWCL
- <item_name>_ExitCode_<task_name>_HWCL

A proprietary process is generated for data exchange with the hardware driver for every signal group defined.

The following processes exist for data exchange:

- `<item_name>,<signal_group_name>_<task_name>_HWCF` ("receive" process)

- `<item_name>,<signal_group_name>_<task_name>_HWCL` ("send" process)

`HWCF` means that the process is to be inserted into the task specified with <task_name> **before** the existing processes; `HWCL` means that the process is to be inserted **after** the existing processes.

## 8.2 Process Order

The RTIO framework ensures consistent hardware communication thanks to the automatic sorting of the processes into the task list.

> **Note**
>
> *This allocation is executed each time RTIO code is generated in accordance with the scheme defined. The user should never change the allocation of the "HWC" processes. This can lead to unforeseeable results which in turn may lead to the hardware being damaged!*

If all processes generated by the RTIO framework were to be allocated to a task, the following process sequence would result:

- Init
- Analyze
- Acknowledge
- DeviceManager
- Start
- Config
- first "receive" signal group
- ...
- last "receive" signal group
- USER PROCESSES
- first "send" signal group
- ...
- last "send" signal group

- *Stop
- *Exit

The processes are processed for the "items" from top to bottom in accordance with the order which is defined by the hierarchical structure, i.e. the system controller ES113x first has to be initialized before a subordinate hardware driver can be initialized.

The opposite order applies for the processes marked with a "*" ("Stop" and "Exit"); i.e. a hardware driver first has to be ended before the system controller is ended.

# 9    The ETK Bypass (ES1200/ES1201/ES1231/ES1232)

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

## 9.1    ETK Bypass: Definition

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

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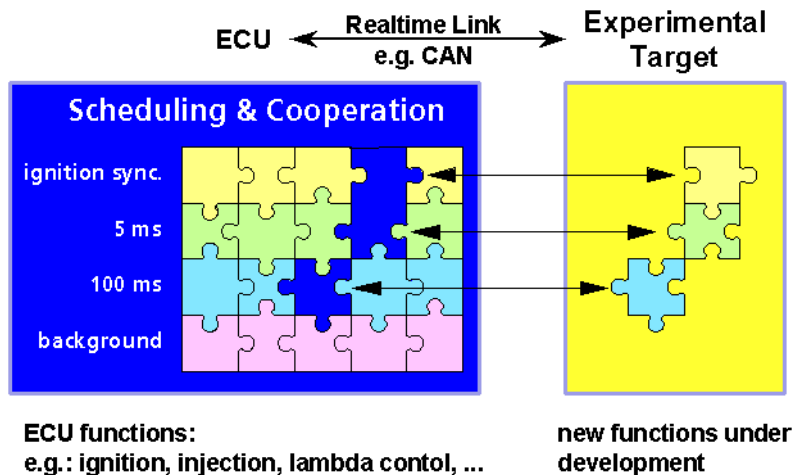


The connection to the ECU is implemented via an ETK (emulator test probe). A dual ported RAM is used for the communication between ECU and ETK.

## 9.2 Hardware Configuration of an ETK Bypass

The figure below shows a sample configuration for an ETK bypass application with the ES1000.2 system:
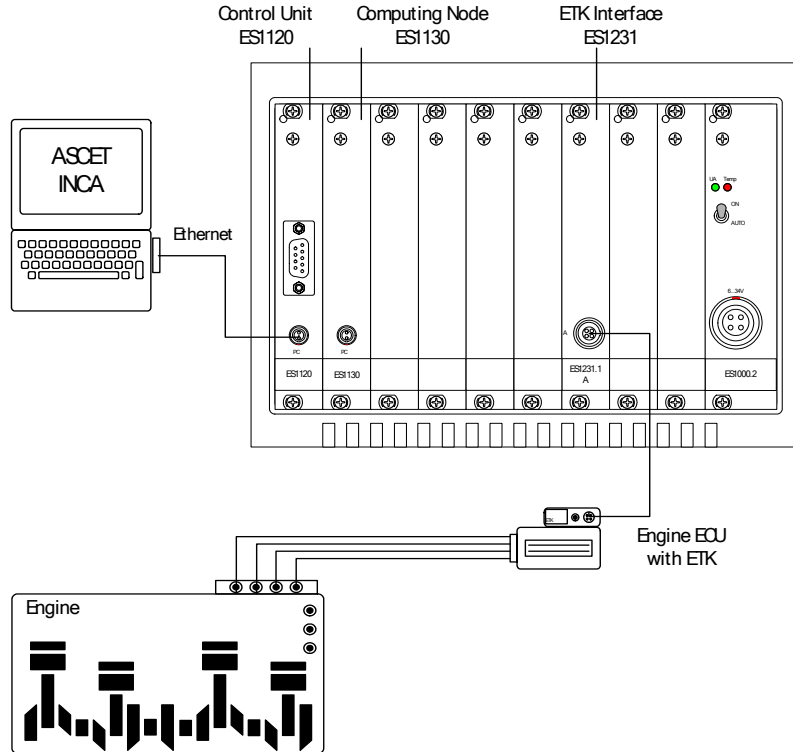


In an ETK bypass application, the outsourced functions are specified in ASCET; code is then generated from the specification that can be executed on the PPC module of the ETAS experimental system. The generated code is downloaded by the host PC to the ETAS experimental system. The ETAS experimental system is connected to the ETK of the control unit via the ETK interface ES1200, ES1201, ES1231, or ES1232.

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

## 9.3    ASCET Project for the ETK Bypass

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

- The HWC module (cf. chapter 5.2.1) must be added to the ASCET RTIO project. The name of the instance must be `HWC`.

- The task list in the OS Editor must include the tasks `Init` (Type: Init / Application Mode: active) and `Exit` (Type: Init / Application Mode: inactive). The task names "Init" and "Exit" are standard names used by the RTIO package. If other task names are used, the task assignment for each new component has to be specified manually in the HWC Editor

- The task list in the OS Editor requires two special, individual tasks. These tasks are used by the RTIO package to assign the signal groups which provide the bypass data received from the driver or the bypass data to be sent to the driver. In any case, these tasks must have the type *software*. The task assignment must be specified manually in the HWC Editor.

- All messages, including Receive messages to be available for the RTIO communication, must be declared as "Exported".

- Global messages are available for the RTIO communication.

The following illustration shows the finished configuration in the "OS" tab:
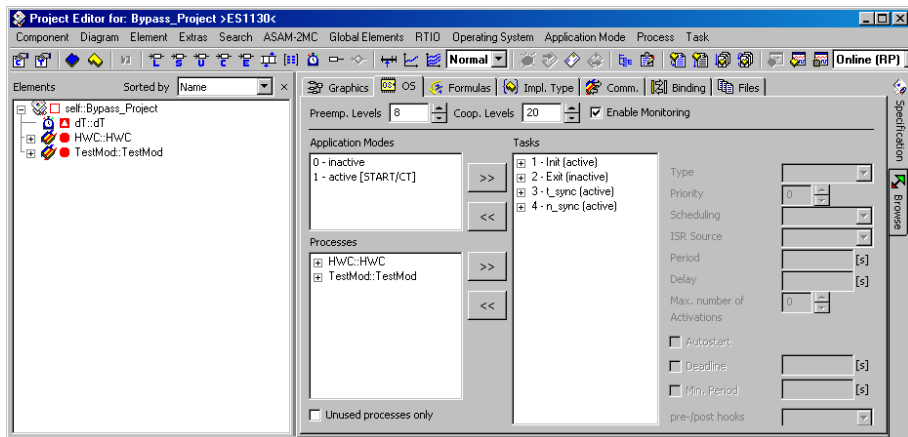
The following table shows the tasks settings:

| Task Name | Applica-tion Mode | Trigger Mode | Prio. | Group | Max. No of Act. | Period |
|-----------|-------------------|--------------|-------|-------|-----------------|--------|
| Init | active | Init | - | - | - | - |
| Exit | inactive | Init | - | - | - | - |
| t_sync | active | Soft-ware | 16 | preemptive | 1 | - |
| n_sync | active | Soft-ware | 17 | preemptive | 1 | - |

## 9.4 How the ETK Bypass Works

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

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

The bypass input data is the data that the ES1000 reads from the control unit. This data is used to calculate the outsourced functions. The calculation results are returned to the control unit program as bypass output data.

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

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

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



— · — · — ▶    Bypass Output Data
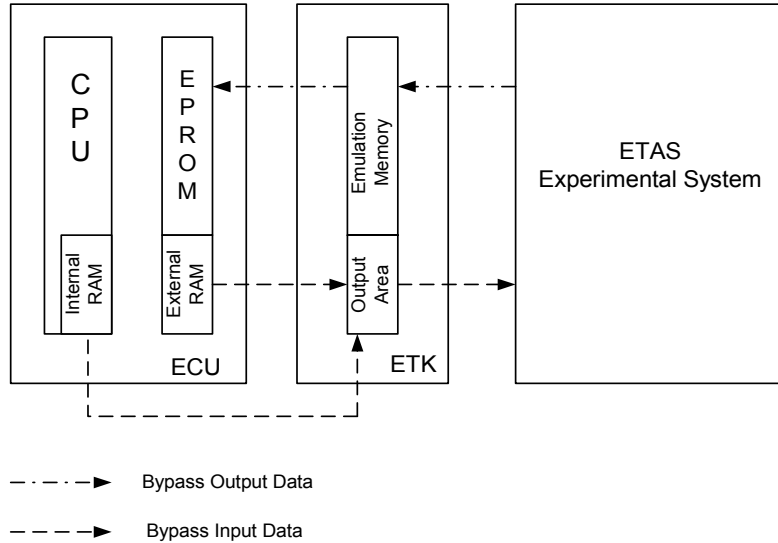
— — — — ▶    Bypass Input Data

The data flows cyclically within the bypass project, i.e. the bypass input data is first read and then processed by the ES1000 system. Then the bypass output data calculated in the ES1000 is written back to the control unit where it is processed further. Reading the bypass input data is synchronized by a time- or angle-synchronous grid in the control unit. Writing the data back to the control unit is not synchronized.

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

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

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

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

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

```
/begin DISTAB_CFG

0xC             /* type of display table:      */
                /* 0xC =DISTAB12, 0xD =DISTAB13 */

0x1             /* Data type of display table: */
                /* 1=byte 2=word (ECU Data Mode) */
                /* additional code table for   */
                /* distab13 depending on bus   */
                /* width/bus access (see distab13 */
                /* spec. for more information)  */

MSB_LAST        /* Byte Order: MSB_FIRST/MSB_LAST */

0x383000        /* Trigger Segment Address      */

0x0             /* Trigger Configuration        */

TRG_MOD 0xB7    /* Dyn. length for TRG_MOD      */
                /* (special code)               */

/end DISTAB_CFG
```

**Note**

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

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

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

With ES1201 and ES1231, the control unit has two bypass channels of which one (channel A, higher priority) runs in the angle-synchronous grid and the other (channel B, lower priority) in the time-synchronous grid. Each of the two bypass channels is defined by various pieces of address information and the

size of data buffers. With ES1232, up to 32 channels exist (16 bypass channels and 16 measurement channels) whose names and priorities are determined automatically (see chapter 10.8.7 on page 221).

*Bypass Communication (AML V1.1)*

The bypass communication is explained here using bypass channel A of the ES1231. The process is the same for both channels. Tab. 9-1 contains the names and description of the parameters used in the ASAM-MCD-2MC file (`*.a2l`, AML V1.1) to define both channels.

> **Note**
>
> *The parameter names are not part of the ASAM-MCD-2MC standard; they are not necessarily included in the `*.a2l` file. They are added manually to the examples of this manual to provide clarity.*

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

**Tab. 9-1**    Bypass communication parameters (AML V1.1)

The parameters for channel A are provided in a `QP_BLOB` in the `IF_DATA ETK` section of the ASAM-MCD-2MC file.

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

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

Fig. 9-1 schematically shows the process of the bypass cycle with the DISTAB data exchange method using the example of bypass channel A. The numbers represent the individual steps of the bypass cycle:
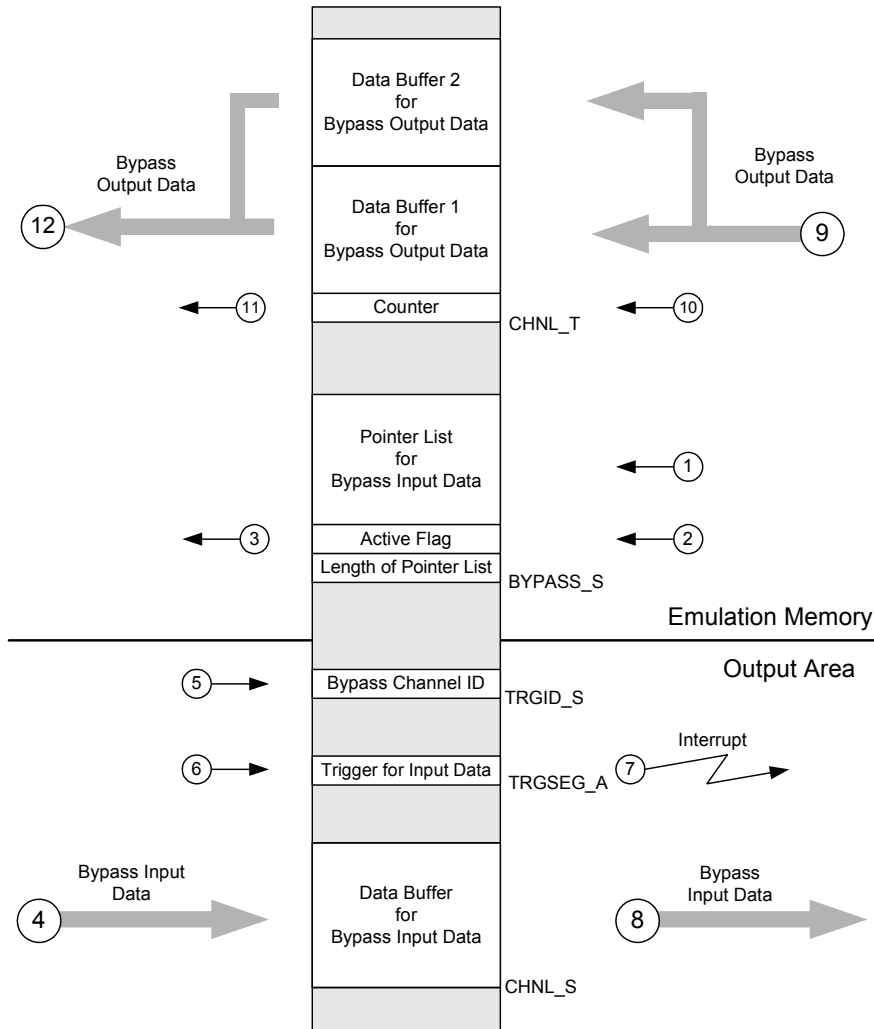


**Fig. 9-1**    Schematic process of a bypass cycle (DISTAB method)

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

| DISTAB12 | DISTAB13 |
| --- | --- |
| For each byte to be sent to the simulation computer, there is a pointer pointing to the address of the byte in the control unit memory. With the DISTAB 12 method, data is always transmitted in bytes; therefore, two pointers are required to send a word. After filling the pointer list, its length is written into the first byte of the list. | Bytes 4 to 7 of the pointer list contain the number of 8-byte, 4-byte, 2-byte, and 1-byte signals. The following bytes contain the signal addresses, beginning with the first 8-byte signal, and ending with the last 1-bytre signal. Each address covers 4 bytes; the byte order of the addresses is determined by the `Byte Order` parameter (cf. page 142). |

The bypass offsets (cf. step 11, page 147) are transferred in this step, too.

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

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

3. The control unit continuously checks the active-flag byte of the pointer list. Once the last bit is set to 1, the bypass is activated and the data transmission begins.

4. The bypass input data is written by the control unit into the appropriate data buffer, `CHNL_S`.

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

5. The control unit writes the ID of the current bypass channel (i.e., 4 for bypass channel A) to the address `TRGID_S`. This is necessary because only two addresses are available for the channel IDs, while there are a total of five channels. Two channels are used as bypass channels; the other three are available for calibration purposes. The ES1232 has up to 32 channels, 16 bypass channels and 16 measurement channels.

6. The trigger address `TRGSEG_A` is loaded by the control unit with a random value.

7. Writing to the trigger address triggers an interrupt in the simulation computer.

8. By reading the channel ID from the address `TRGID_S`, the simulation computer determines which channel was initialized and then reads the corresponding data buffer. From the information in the ASAM-MCD-2MC file, such as the conversion formula, the simulation computer is able to convert the raw data from the control unit into physical model variables that can then be processed by the ASCET model.

9. The simulation computer computes the functions outsourced to the bypass model on the basis of the bypass input data. The results are then written back to the appropriate data buffer as bypass output data. For each bypass channel there are two data buffers that are alternately written to. The simulation computer first internally (i.e. in the simulation computer) increments the counter that is later written to the `CHNL_T` address. The counter is incremented when the interrupt in step 7 occurs. Depending on the contents of the counter, one of the data buffers is used for writing. If the count is odd, the first data buffer is used; if it is even, the second buffer is used.

10. After writing to the data buffer, the internal count is written to the `CHNL_T` address.

11. The value of the counter is read by the control unit. The control unit decides on the basis of this value which data buffer is used for reading. If the value is odd, the first data buffer is read from; if it is even, the second buffer is read from. This ensures that the data consistency of the bypass output data is maintained.

The layout of these two buffers depends on the selected bypass output signal. Thus, the ECU must know where in the buffer the signal value actually is located. For that purpose, the signal position relative to the buffer start address—the *bypass offset*—is determined. For safety reasons, no pointer list exists for the bypass signals; instead, the ECU software contains a special bypass offset parameter for each signal. The

bypass offsets are filled into the bypass offset parameters in step 1 (cf. page 146). These parameters can be directly accessed because they are located in an ECU application data area allocated in the ETK RAM.

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

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

*Other AML versions*

A `QP_BLOB` of version AML V1.1.1 does not differ from the above AML V1.1 example (see page 143).

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

```
/begin IF_DATA ETK

   /begin SOURCE   "BYPASS A4"

    0

    0

    /begin

     QP_BLOB

     0x0100     /* version 1.0              */
     15         /* hardware trigger         */
     INDIRECT   /* indirect/direct triggering */
     5          /* raster number/priority   */
                /* (32..1)                  */
     BYPASS     /* raster type              */
                /*(BYPASS/MEASUREMENT)      */
     0x38302E   /* trigger id/flag address for */
                /* indirect triggering      */
     100        /* Max. length of display table */
                /* in bytes                 */
```

```
            0x81025E    /* address of the display table */
                        /* in the memory                */
            0x3801E0    /* address, where the ECU        */
                        /* writes the display values    */
            100         /* Max. size of bypass receive  */
                        /* table                         */
            0x0         /* StartAddress of the Adress    */
                        /* table for bypass output       */
            0x8103F2    /* Output address of the         */
                        /* bypass table                  */
            2           /* worst case raster timing      */
        /end QP_BLOB

    /end SOURCE

    ...

  /end IF_DATA
```

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

**Note**

*AML V1.1.x* *describes ETK data for 3+2 measurement rasters, it is valid for ES1232 [ETK-CTRL-BAS], ES1231, and ES1200/1.* **AML V1.2 and higher** *describes ETK data for multirasters, it is valid for ES1232 [ETK-CTRL-ADV].*

## 9.6    Initially Required Information and Data

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

*Control Unit Program*

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

*ASAM-MCD-2MC File*

In addition to other tasks, the ASAM-MCD-2MC file describes the data structures of the control unit. The ETK bypass package uses the ASAM-MCD-2MC file to access the input and output variables of the bypass project. Therefore, it is crucial to have the proper ASAM-MCD-2MC file matching the current soft-

ware version of the control unit. The supported AML version depends on the hardware. The descriptions of the various boards include the necessary information.

*Data Format of the Control Unit Processor*

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

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

*Base Addresses*

The start addresses of the transfer channels and the trigger addresses must be specified in the ETK bypass package, or the start addresses are specified in the ASAM-MCD-2MC file and can be read by the HWC Editor. The significance of the base addresses is explained in chapter 9.5 "Data Exchange Between Control Unit and ETAS Experimental System" on page 141.

*Parameters for Activating the Bypass*

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

*Bypass Output Variables*

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

*Bit Masks for the Transformation in the NEAR Region*

The DISTAB 12 method supports only 16-bit memory addresses for transfer between the control unit and the bypass computer. For control units using memory addresses larger than 16 bits, these have to be transformed to 16 bits by means of a data page pointer. The `longAdrANDMask` and `longAdrOR-Mask` bit masks are used for this purpose. The bit masks need to be specified in the HWC Editor.

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

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

# 10 HWC Items

All HWC items available with ASCET are described in this chapter.

## 10.1 Implemented Items

Select **Options → Show Installed Items** in the HWC editor to display the implemented items of the RTIO Package.
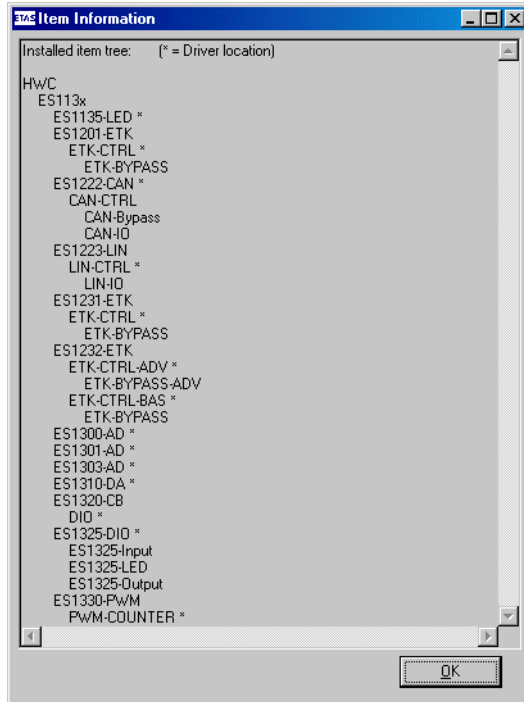


**Fig. 10-1** "Item Information" Dialog Box (* = Position of the HW driver)

The following sections show the HWC editor tabs for each item. Only the item-specific options are described. Options that apply to all items are described in the chapter "Configuration Tabs" on page 121. The "Mappings" tab is not shown as there are no item-specific options in this tab.

## 10.2 ES1135-LED

This chapter describes the settings for the LEDs on the ES1135 front panel.

This section describes the global options of the LED Device.
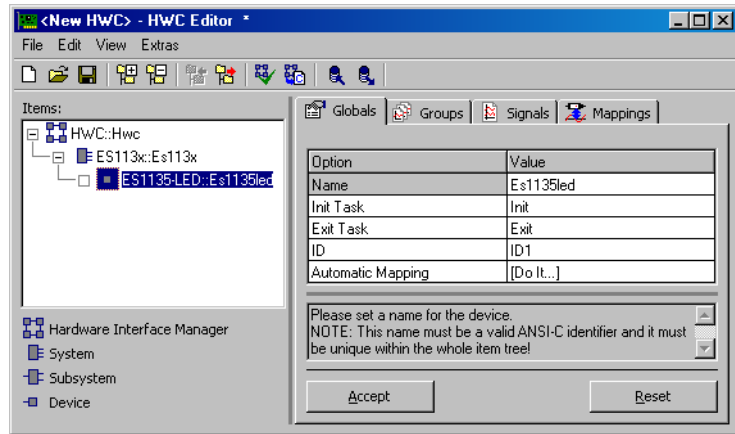


**Fig. 10-2**    The "Globals" Tab of the ES1135-LED Device

*Automatic Mapping*

This option makes automatic assignment between signals and (ASCET) messages possible. The assignment takes place as described in section "Automatic Mapping" on page 182.

### 10.2.2 Groups (ES1135-LED Device)

This section describes the signal-group-specific options of the ES1135-LED Device.
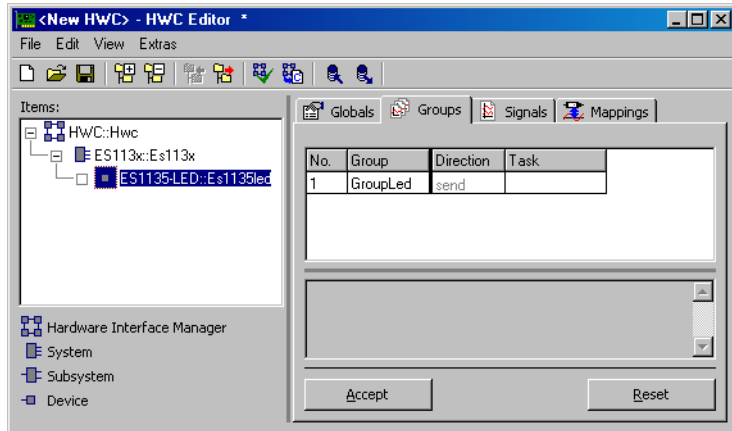


**Fig. 10-3**     The "Groups" Tab of the ES1135-LED Device

There are no item-specific columns defined for the ES1135-LED Device in the "Groups" tab. The possible item settings are described in section 7.4.3 on page 126.

### 10.2.3 Signals (ES1135-LED Device)

This section describes the signal-specific options of the ES1135-LED Device.
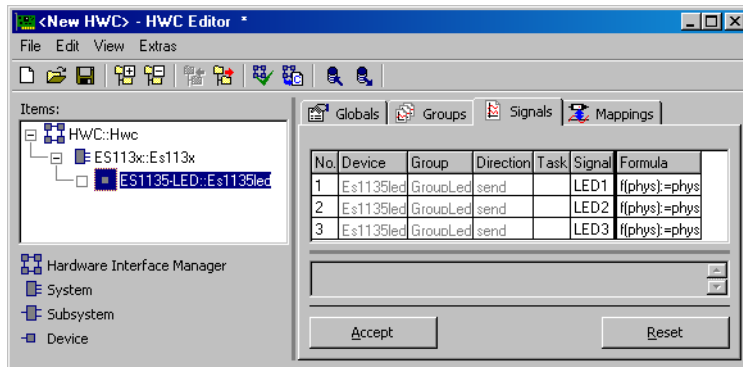


**Fig. 10-4**     The "Signals" Tab of the ES1135-LED Device

There are no item-specific columns defined for the ES1135-LED Device in the "Signals" tab. The possible item settings are described in section 7.4.4 on page 127.

## 10.2.4 Mappings (ES1325-LED Device)

There are no item-specific columns defined for the ES1135-LED Device in the "Mappings" tab. The possible item settings are described in section 7.4.5 on page 129.

## 10.3 ES1201-ETK

The ETK interface boards ES1200 and ES1201 enable the experimental system to be connected to ECUs with an ETK. Using an interference-immune serial interface (8 MBit/s), the contents of the ECU memory can be transferred to the ETAS experimental system or alternatively be modified by this system.

The ES1200 provides one ETK interface while the ES1201 has two ETK interfaces. Otherwise, the two boards are identical in every respect.

### Note

*The ES1201 board requires special system services to be able to communicate with an ETK. These services can only be provided by an ES1111 (VCU) or an ES1120 (VCU2) system controller board. (For more details, please see the section "Hardware – ES1000.x Experimental System" on page 69).*

## 10.3.1 Globals (ES1201-ETK Subsystem)
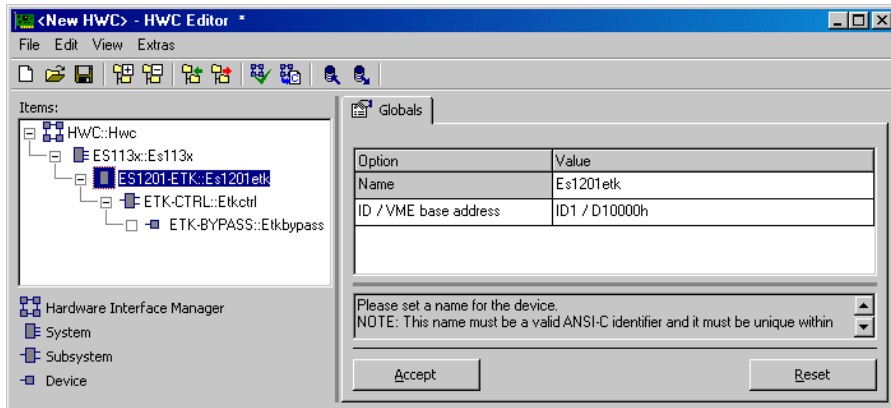


**Fig. 10-5**     The "Globals" Tab of the ES1201-ETK Subsystem

*ID / VME base address*

This line is responsible for the setting of the VME base address. This setting has to correspond to the switch settings of the ES1201. Two different VME base addresses can be selected for the ES1201; this means that up to two ES1201s can be operated in an ETAS experimental system.

Up to two ETK-CTRL subsystems can be assigned to the ES1201-ETK subsystem. These ETK-CTRL subsystems correspond to the two ETK interface controllers on the board.

The ES1201-ETK subsystem is used to integrate the ES1200. As this board has only one ETK interface, only one ETK-CTRL subsystem can be assigned (with ETK Port A).

10.3.2    Globals (ETK-CTRL Subsystem)

In the "Globals" tab of an ETK-CTRL subsystem, a physical ETK controller or an ETK port is assigned to the ETK-CTRL subsystem.
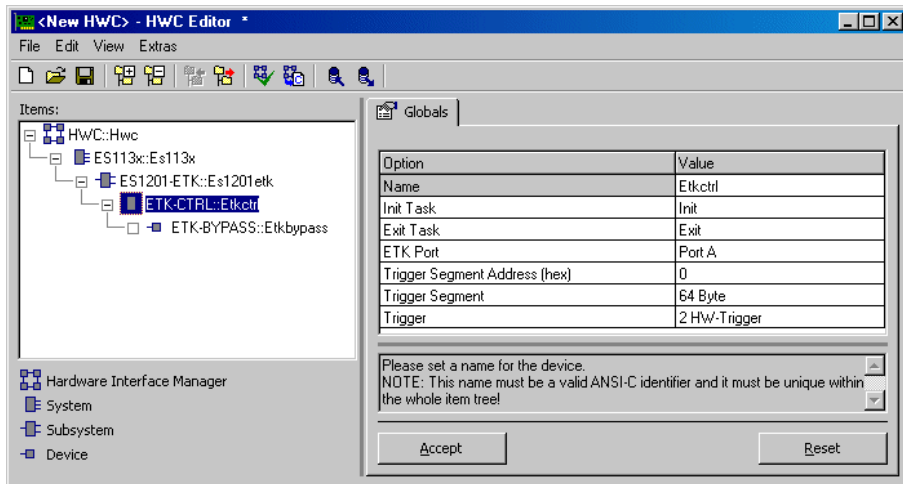


**Fig. 10-6**    The "Globals" Tab of the ETK-CTRL Subsystem

*ETK Port*

One of the two independent ETK channels (port A, port B) and hence one of the two ETK controllers of the ES1201 is assigned as an ETK interface. Only port A can be selected for the ES1200.

*Trigger Segment Address (hex)*

Here, the trigger segment address of the DISTAB procedure has to be specified.

If an ETK bypass item with a relevant ASAM-MCD-2MC project is assigned to the ETK-CTRL subsystem, this setting cannot be edited and is made automatically providing the relevant variable in the ASAM-MCD-2MC project was correctly defined.

The trigger segment address determines the location of the hardware trigger addresses which are used to control data transmission via the bypass channels. The location of the hardware trigger addresses in relation to the trigger segment address is constant with the DISTAB data exchange procedure.

The start address of a 64-byte trigger segment (with 8-bit or 16-bit wide bus access) must be at an even address which can be divided by 64: the start address of a 128-byte trigger segment (with 32-bit wide bus access) correspondingly has to be at an address that can be divided by 128.

The format of the trigger segment is defined in the DISTAB 12 and DISTAB 13 interface description.

*Trigger Segment*

This is where the trigger segment size has to be defined.

If an ETK bypass item with a relevant ASAM-MCD-2MC project is assigned to the ETK-CTRL subsystem, this setting cannot be edited and is made automatically providing the relevant variable in the ASAM-MCD-2MC project was correctly defined.

The address space of the trigger segment must not be used by the ECU. With 16-bit-wide trigger addresses, the trigger segment is 64 bytes wide and with 32-bit-wide trigger addresses, the trigger segment is 128 bytes wide.

*Trigger*

This is where the number of hardware trigger addresses used for the bypass (2 or 16) is set.

If an ETK bypass item with a relevant ASAM-MCD-2MC project is assigned to the ETK-CTRL subsystem, this setting cannot be edited and is made automatically providing the relevant variable in the ASAM-MCD-2MC project was correctly defined.

### 10.3.3    Globals (ETK-BYPASS Device)

This ETK-BYPASS device is used to define all variables necessary for bypass operation with a bypass-capable ECU equipped with an ETK and the relevant software.
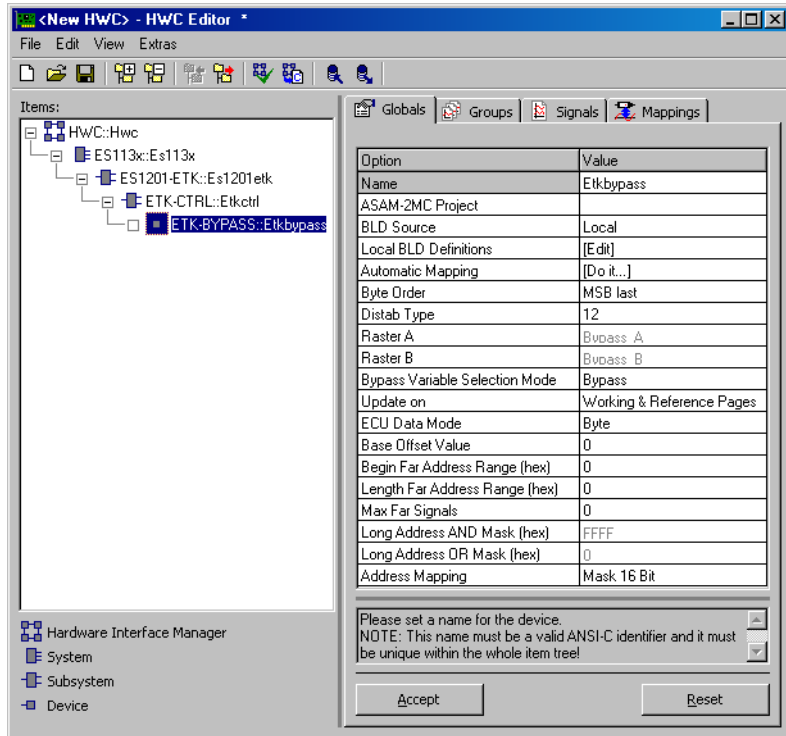


**Fig. 10-7**    The "Globals" Tab of the ETK-BYPASS Device

*ASAM-2MC Project*

The ETK bypass requires an ASAM-MCD-2MC project (AML V1.1) which can be generated in the database by reading an ASAM-MCD-2MC description file.

> **Note**
>
> *ASAM-MCD-2MC projects using AML V1.2 or higher* **cannot** *be read. The following error message appears when such a project is assigned to the device:*
> ```
> Error: Incompatible version 0x<version> of QP_BLOB
> found in SOURCE '<sName>'. Expected version = 0x1.
> Selected ASAM-2MC Project is not suitable for the basic
> ETK-Controller (ES1232/ETK-CTRL-BAS, ES1231/ETK-CTRL,
> ES1201/ETK-CTRL) ! Please use an advanced ETK-Control-
> ler (ES1232/ETK-CTRL-ADV)!
> ```

The ASAM-MCD-2MC file contains, under the `IF_DATA ETK` label (in older versions also `IF_DATA ASAP1B_ETK`), the bypass description. For each bypass channel, a `QP_BLOB` exists which contains the channel settings; an example is given in section "Bypass Communication (AML V1.1)" on page 143. The `TP_BLOB` contains general settings.

```
/begin TP_BLOB

   0x1000100    /* TP_BLOB version        */
   2            /* Project Base Address   */
   0x0          /* RESET_CFG (only PPC family CPU)*/
   /begin DISTAB_CFG 0xC 0x1 MSB_LAST 0x383000 0x0
    TRG_MOD 0xB7
   /end DISTAB_CFG
   CODE_CHK   /* check whether program and data  */
              /* are matching                    */
      0x0  /* program ID address in data range   */
      0x0  /* program ID length in data range    */
      0x0  /* program ID address in external RAM */
      0x0  /* program ID length in external RAM  */
   ETK_CFG 0xF 0xF0 0xFF 0x3 0xFD 0xEE 0xFF 0x1
              /* ETK configuration        */
```

```
      RESERVED 0x810000     /* start address        */
               0x8103F9     /* length               */
               EXTERN       /* memory attribute     */
               0x-1 0x-1 0x-1 0x-1 0x-1
                            /* mirror offsets 1 - 5  */

   /end TP_BLOB
```

Necessary information not given in the ASAM-MCD-2MC file can be inserted manually in the configuration tabs of the HWC editor.

Click in the "Value" column next to the "ASAM-2MC Project" option. A dialog window opens from which the required ASAM-MCD-2MC project can be selected.
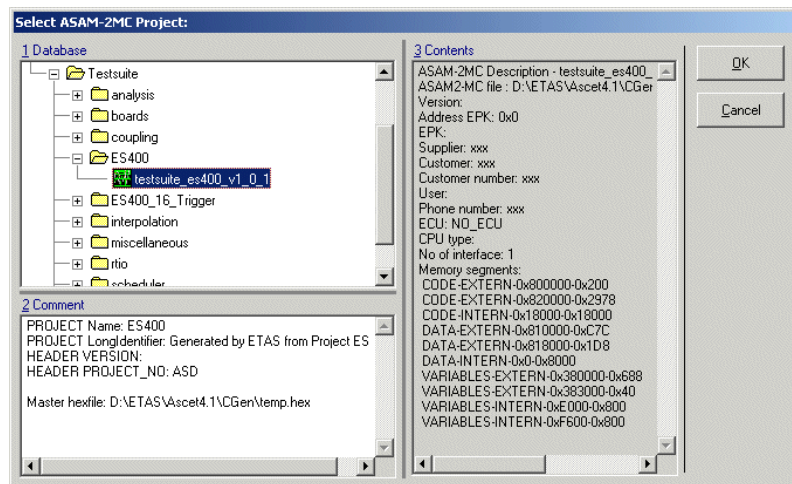


**Fig. 10-8**    "Select ASAM-2MC Project" dialog window

*BLD Source*

This option is used to determine which source is responsible for the definition of the dependency between the required ASAM-MCD-2MC variables for the bypass output values (BLD = Bypass Label Dependence).

The following two possibilities are usually at your disposal:

ASAM-2MC File    This means the definition of the dependencies between the bypass output values within the ASAM-MCD-2MC file. This, however, is not supported by many ASAM-MCD-2MC files.

Local            Local means the local definition is used from the ETK-BYPASS device (see following section "Local BLD Definitions")

The data transmission of the bypass output values from ASCET to the ECU takes place using an individual data buffer for each grid (angle-synchronous / time-synchronous).

Some ECU programs support flexible allocation of bypass variables to this data buffer. This flexible allocation is certainly necessary when more bypass variables are made available by the ECU software than can be transferred in the available data buffer.

Depending on the ECU program, up to 3 additional characteristics are necessary for every bypass output value (defined as a measurement) to define the bypass variable:

| | |
|---|---|
| Byte offset or vector parameter: | Defines the byte offset of the bypass output value in relation to the start of the data buffer |
| Bit offset: | Is only available for bit values and specifies the bit position within a byte |
| Source: | A few ECU programs support a free allocation of a bypass variable to a bypass grid or source (angle-synchronous / time-synchronous). The parameter contains the relevant value for this (0 = angle-synchronous / 1 = time-synchronous) |

The following figure shows an example:
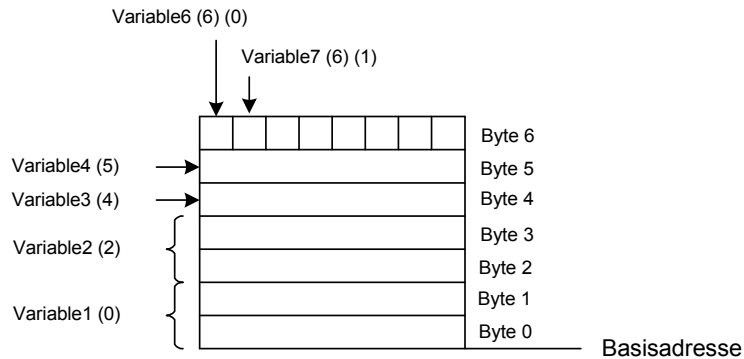


In this case, `Variable6` is at bit 0 of byte 6 in the data buffer for the bypass output values; the offset values are thus byte offset: 6 / bit offset: 0.

`Variable7`, also a bit value, is at bit 1 in byte 6: the values are accordingly byte offset: 6 / bit offset: 1.

`Variable3` is a byte and has byte offset 4 as it is at byte 4 of the memory area; `Variable4` has byte offset 5.

`Variable1` is at the start of the data buffer and has byte offset 0. `Variable2` has byte offset 2 as `Variable1` is a word occupying two bytes.

Click on the entry [Edit] in the "Value" column to open a small text editor. There, the local definition of the dependencies between the bypass labels is defined (BLD Bypass Label Dependencies).

The text is built in lines whereby each line describes exactly one bypass output signal. The definition of the label dependencies of a bypass output signal has the following syntax:

`<Grid>, <Signal>, [<Byte Offset>], [<Bit Offset>], [<Source>]`

| | |
|---|---|
| `Grid:` | "A" (angle-synchronous), "B" (time-synchronous) or "AB" (angle- and time-synchronous) |
| `Signal:` | Name of the ASAM-MCD-2MC measurement of the bypass output value |
| `Byte Offset:` | Name of the optional ASAM-MCD-2MC characteristic which is normally always available and is used to define the byte offset. |
| `Bit Offset:` | Name of the optional ASAM-MCD-2MC characteristic which is used to define the bit position with bit values |
| `Source:` | Name of the optional ASAM-MCD-2MC characteristic. If more than one raster is available (AB), the currently selected grid is entered here. |
| `//` | Is used to introduce a comment until the end of the line |

The offsets for the parameters defined here can be configured automatically with ASCET.

Example of a local BLD definition:

```
1                   // current version of the BLD

sgetas2             // (optional) name of the dependent
                    // ASAP2 file
                    // (is ignored by BLD reader)

A, t1b_8, EBOTT1_8  // definition of a bypass signal
                    // available for angle sync sample
                    // grid

B, B_t1b, EBOTBT1, BOBBT1
                    // definition of a bypass bit
                    // signal available for time
                    // sync grid
```

```
AB, B_Test, B_Test_Vector,, B_Test_Channel
                    // definition of a bypass signal
                    // available for both sample grids
                    // with definition for the actual
                    // sample grid (source) parameter
A, log_uint8_0_A, log_uint8_0_offset_A.Model_Byp_A,
   log_uint8_0_bitOffset_A.Model_Byp_A
                    // definition of a bypass signal
                    // available for angle sync sample
                    // grid
```

This last example corresponds to the following section MEASUREMENT of the ASAM-MCD-2MC file (AML V1.1):

```
/begin MEASUREMENT log_uint8_0_A
   ""
   UBYTE
   ident
   1
   100
   0.0
   1.0
   READ_ONLY
   BIT_MASK 0x8
   ECU_ADDRESS 0xFD00
/end MEASUREMENT
```

*Automatic Mapping*

For more information please refer to the description "Automatic Mapping" in section 10.4.3 on page 179.

*Byte Order*

This line displays the word data storage format (MSB first / MSB last) of the ECU processor. This information is read from the relevant ASAM-MCD-2MC project (see page 142).

Overview of the common "byte order" terms:

| | | |
|---|---|---|
| MSB first | big endian | Motorola |
| MSB last | little endian | Intel |

*Distab Type*

This line shows which DISTAB procedure is used for exchanging data with the ECU. This information is read from the relevant ASAM-MCD-2MC project (see page 142).

DISTAB 12 supports signals which are up to two bytes long

DISTAB 13 supports signals which are 1, 2, 4 and 8 bytes long

*Grid A / B*

These lines display the designations read for the two grids A and B from the ASAM-MCD-2MC project (angle-synchronous and time-synchronous).

*Bypass Variable Selection Mode*

In this line, you can specify whether all ECU variables which were defined as "measurements" in the ASAM-MCD description are displayed ("all") in the selection list for bypass output values ("send" signal groups in the "Groups" tab) or just those variables which were identified as bypass output values in the BLD definition ("bypass").

For the bypass input values ("receive" signal groups in the "Groups" tab), which are not affected by this setting, *all* measurement variables defined in the ASAM-MCD-2MC project are available regardless of the selection in this line.

*Update on*

This line allows you to select whether data transfer between the simulation processor and the ECU should take place during bypass communication on the working page, reference page or working and reference page of the ETK.

*ECU Data Mode*

This line shows which access mode (byte access / word access) the ECU processor uses to access the data memory.

This setting is taken from the ASAM-MCD-2MC project (see page 142).

*Base Offset Value*

This line allows you to move the byte offset of every bypass output value 0, 2 or 8 bytes up.

The default value for the byte offset with DISTAB 12 is 0. With DISTAB 13, the default value for the byte offset is 8. Settings other than those listed here are special solutions and must be coordinated with the ECU programmer.

*Begin Far Address Range (hex)*

This line is used to specify the start address of the FAR address range. This specification is usually only necessary for the DISTAB 12 procedure.

Variables from the FAR address range of the ECU cannot be read or written by ECUs with C16x microcontrollers that use the DISTAB 12 procedure. All values in the Far address range are read by individual ETK accesses which slows down data transfer.

*Length Far Address Range (hex)*

This line is used to specify the length of the FAR address range. This specification is usually only necessary for the DISTAB 12 procedure.

*Max Far Signals*

This line is used to specify the maximum number of variables that can be read from the FAR address range. This specification is usually only necessary for the DISTAB 12 procedure.

Reading large numbers of variables from the FAR address range has a negative effect on the runtime of the ETK bypass. As few values as possible should be read from this range.

*Long Address AND / OR Mask (hex)*

These two lines show the masks selected in the "Address Mapping" line.

*Address Mapping*

This line can only be edited when the DISTAB 12 procedure was selected.

It is used to define bit masks which is then in turn used to execute an address transformation. ECUs with a C16x microcontroller, that use memory addresses larger than 16 bits, can be moved with these bit masks to addresses within the 16-bit address range of the DISTAB 12.

Formula:

```
<ECU_address> :=
        <Memory_address> & <AND Mask> | <OR Mask>
```

The masks are shown in the "Long Address AND / OR Mask (hex)" lines. The following table contains the possible selections for "Address Mapping" and the correspondins masks.

| Address Mapping | Long Address | |
| --- | --- | --- |
| | AND Mask (hex) | OR Mask (hex |
| Mask 16 Bit | 0xFFFF | 0x0000 |
| C167x DPP0 | 0x3FFF | 0x0000 |
| C167x DPP1 | 0x3FFF | 0x4000 |
| C167x DPP2 | 0x3FFF | 0x8000 |
| C167x DPP3 | 0x3FFF | 0xC000 |

## 10.3.4  Groups (ETK-BYPASS Device)

The number of signal groups (4) is fixed for the ETK-BYPASS device. The first two signal groups concern the first grid, usually the angle-synchronous one. The last two signal groups concern the second bypass grid which is usually the time-synchronous grid.

Each grid consists of two signal groups; a send and a receive signal group.

The data of the receive signal group is transferred using the standard DISTAB procedure and the data of the send signal group is exchanged between the ECU and RTIO using the bypass table procedure.
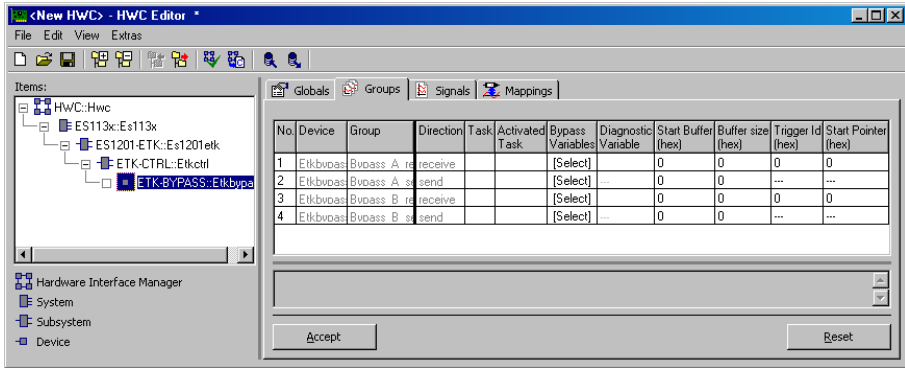


**Fig. 10-9**    The "Groups" Tab of the ETK-BYPASS Device

**Note**

*After changing the group name, the signal name, or the signal direction, an ASCET message mapped previously may not be mapped automatically any longer and then has to be mapped again manually.*

*Task*

The following task allocation should be used for a correct bypass function:
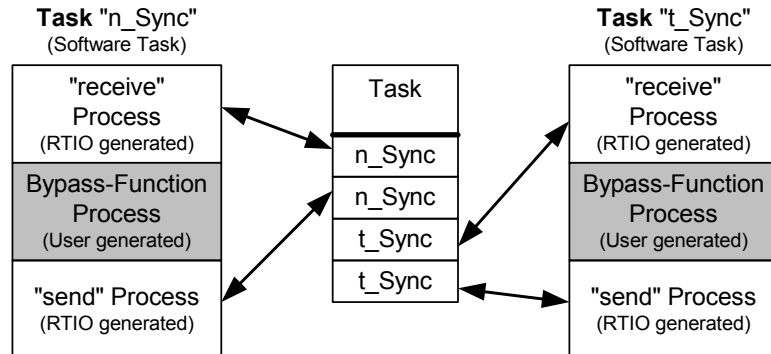


**Fig. 10-10**    The ETK Bypass Standard Task Allocation

*Activated Task*

If software tasks are entered in this column, they are activated when sending or receiving data by an implicit "activate Task" call.

*Bypass Variables*

See section "Bypass Variables" on page 195 in chapter 10.5.4.

*Diagnostic Variable*

See section "Diagnostic Variable" on page 196 in chapter 10.5.4.

*Start Buffer (hex)*

The base addresses of the relevant data buffers for the bypass input data and output data must be entered in this column.

Normally, these address specifications are contained in the ASAM-MCD-2MC project, providing it is an ASAM-MCD-2MC project which also correctly supports bypass operation. In this case, the relevant values from the ASAM-MCD-2MC project are entered and entry is then locked.

The relevant ASAM-MCD-2MC parameters for this are (Tab. 9-1, "Parameter" column):

`CHNL_S, CHNL_T, CHNL_X, CHNL_Y`

*Buffer size (hex)*

The values of the relevant data buffers for the bypass input data and output data must be entered in this column.

Normally, these specifications are contained in the ASAM-MCD-2MC project, providing it is an ASAM-MCD-2MC project which also correctly supports bypass operation. In this case, the relevant values from the ASAM-MCD-2MC project are entered and entry is then locked.

The relevant ASAM-MCD-2MC parameters for this are (Tab. 9-1, "Parameter" column):

`BPMAX_S, BPMAX_T, BPMAX_X, BPMAX_Y`

*Trigger Id (hex)*

The addresses for the trigger identifiers for the DISTAB procedure used must be entered in this column for the "receive" signal groups.

Normally, these specifications are contained in the ASAM-MCD-2MC project, providing it is an ASAM-MCD-2MC project which also correctly supports bypass operation. In this case, the relevant values from the ASAM-MCD-2MC project are entered and entry is then locked.

The relevant ASAM-MCD-2MC parameters for this are (Tab. 9-1, "Parameter" column):

`TRGID_S, ---, TRGID_X, ---`

*Start Pointer (hex)*

The start addresses of the pointer lists for the DISTAB procedure used must be entered in this column for the "receive" signal groups.

Normally, these specifications are contained in the ASAM-MCD-2MC project, providing it is an ASAM-MCD-2MC project which also correctly supports bypass operation. In this case, the relevant values from the ASAM-MCD-2MC project are entered and entry is then locked.

The relevant ASAM-MCD-2MC parameters for this are (Tab. 9-1, "Parameter" column):

`BYPASS_S, ---, BYPASS_X, ---`

10.3.5    Signals (ETK-BYPASS Device)
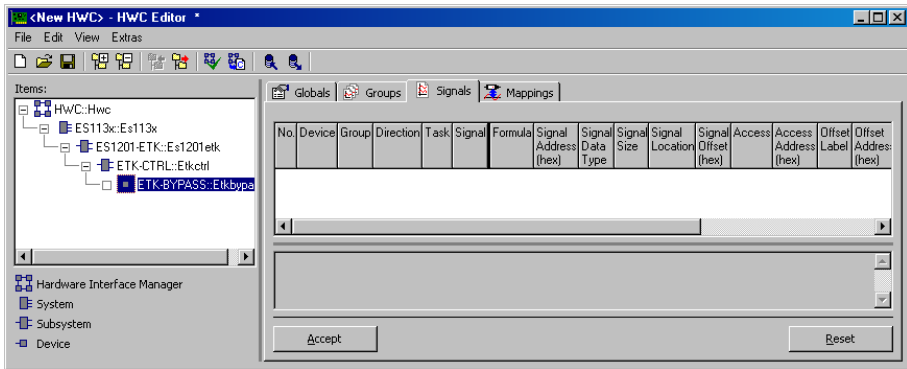


**Fig. 10-11**    The "Signals" Tab of the ETK-BYPASS Device

> **Note**
>
> *None of the columns of the "Signals" tab can be edited by the user. They are solely intended for the display of status values for the bypass variables.*

The bypass variables are sorted into the relevant signal group according to their byte size. The valid order for "send" signals is 8-4-2-1 bytes.

With "receive" signals, a distinction has to be made as to whether they are internal variables of the ECU controller which can only be read indirectly using the DISTAB procedure or whether the variables are in the external memory range of the ECU controller which can also be read directly by the ETK. This means the valid order here is 8-4-2-1 bytes directly and then 8-4-2-1 indirectly.

### Signal Address (hex)

This column displays the ASAM-MCD-2MC measurement memory addresses of the bypass variables.

### Signal Data Type

This column displays the data types of the bypass variables.

### Signal Size

This column displays the data size of the bypass variables in bytes.

### Signal Location

This column is only relevant for "receive" signals and displays the memory location (internal / external) of the bypass variables.

- Internal:
  "Internal" means that the bypass variable is in the internal RAM of the ECU controller and can thus only be read via the DISTAB mechanism.

- External
  "External" means that the value is in the external RAM of the ECU controller and can **possibly** be read directly by the ETK.

### Signal Offset (hex)

This column displays the (0-based) index of the value within the data buffer.

### Access

This column is only relevant for "receive" signals and displays the access method used to access the measured values:

"Distab"    The value is read using the DISTAB procedure

"Direct"    The value is read using the direct ETK access

### Access Address (hex)

This column shows the target addresses of the bypass variables which may have been recalculated.

The ASAM-MCD-2MC project may contain "memory segments" for the ETK access which contain a conversion for certain addresses to access the target address.

In addition, the address masks "AND Mask" and "OR Mask" from the "Globals" tab may be taken into consideration for the DISTAB 12 procedure.

Formula:

```
<target_address> :=
          (<signal_address> & <AND Mask> | <OR Mask>)
```

(possibly with "memory segment" conversion)

*Offset Label*

This column is only relevant for "send" signals and displays the name of the vector parameter providing a BLD reference has been defined for the bypass variable (for more information, please refer to the section "Local BLD Definitions" on page 162).

*Offset Address (hex)*

This column is only relevant for "send" signals and displays the address of the vector parameter providing a BLD reference has been defined for the bypass variable (for more information, please refer to the section "Local BLD Definitions" on page 162).

*Offset Value (hex)*

This column is only relevant for "send" signals and displays the value to which the parameter is adjusted when bypass is started to activate the relevant bypass variable. This value is only available if a BLD reference has been defined for the bypass variable (for more information, please refer to the section "Local BLD Definitions" on page 162).

Formula for the calculation of the value:

```
Offset Value =
      Signal Offset + Base Offset Value
```

(The Base Offset Value is taken from the "Globals" tab.)

*Bit Label*

This column is only relevant for "send" signals and displays the name of the bit offset parameter providing a BLD reference has been defined for the bypass variable and it is a bit value (for more information, please refer to the section "Local BLD Definitions" on page 162).

*Bit Address (hex)*

This column is only relevant for "send" signals and displays the address of the bit offset parameter providing a BLD reference has been defined for the bypass variable and it is a bit value (for more information, please refer to the section "Local BLD Definitions" on page 162).

*Bit Value*

This column is only relevant for "send" signals and displays the value to which the bit offset parameter is adjusted when bypass is started to specify the relevant bit position within a data byte (0-based). This value is only available providing it is a bit value and a BLD reference has been defined for the bypass variable (for more information, please refer to the section "Local BLD Definitions" on page 162).

*Src Label*

This column is only relevant for "send" signals. It displays the name of the "Source" parameter providing a BLD reference has been defined for the bypass variable (for more information, please refer to the section "Local BLD Definitions" on page 162).

*Src Address (hex)*

This column is only relevant for "send" signals. It displays the address of the "Source" parameter (Src Label) providing a BLD reference has been defined for the bypass variable (for more information, please refer to the section "Local BLD Definitions" on page 162).

*Src Value*

This column is only relevant for "send" signals. It displays the value to which the parameter (Src Label) is adjusted at the start of bypass to tell the ECU which grid the relevant bypass variable shall be updated in. If the variable is to be displayed in the angle-synchronous grid (A), `Src Value` is 0, if the variable is to be displayed in the time-synchronous grid (B), `Src Value` is 1.

This value is only available if a BLD reference has been defined for the bypass variable (for more information, please refer to the section "Local BLD Definitions" on page 162).

10.3.6    Mappings (ETK-BYPASS Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

## 10.4 ES1222-CAN (CAN-IO)

The ES1222 board is used as a CAN interface in VMEbus systems. The board makes four CAN channels available which have separate CAN controllers of type Intel 82527.

The board also makes it possible to trigger interrupts on the simulation processor with received CAN messages so that a permanent polling for messages can be done without. Each CAN controller can process up to 255 send and 255 receive messages. The system controller is considerably relieved as a proprietary processor is used.

> **Note**
>
> *Because the FIFO memory contains only 82 messages per channel it isn't possible, to send the complete 255 messages at the same time. The messages have to submit from several tasks at various times to the FIFO memory so that the FIFO memory isn't overfilled.*

This section describes the RTIO integration of the ES1222 board (previously referred to as the VSIC board). In the HWC editor, the ES1222 board is integrated by selecting the "ES1222-CAN" item.

In principle, all distant terminals can connected over this board which communicate about CAN with tightly defined messages.

### 10.4.1 Globals (ES1222-CAN Subsystem)
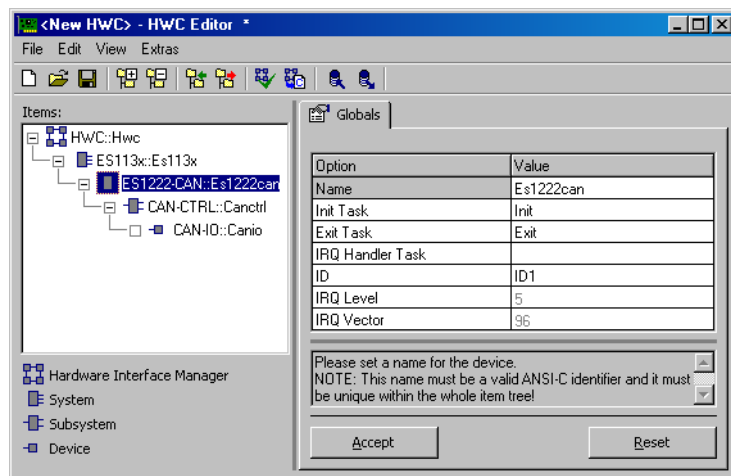


**Fig. 10-12**   The "Globals" Tab of the ES1222-CAN Subsystem

Up to four CAN-CTRL subsystems can be assigned to the ES1222-CAN subsystem. These "CAN-CTRL" items correspond to the four Intel 82527 CAN controllers on the board.

*IRQ Handler Task*

An additional "IRQ Handler Task" is required to support interrupts of the ES1222 board. This task must be generated as a "software" task in the task list in the ASCET project editor, and at least 2 (max. 50) must be entered in the "Max. No. of Activations" field.

*ID*

The board number of the board to be addressed has to be entered in the "ID" field.

**Note**

*The ES1222 board is an "Auto-ID" board which is automatically assigned an ID number and a free address space by the Hardware Manager when the system is switched on. Boards of the same type (e.g.two ES1222s) are numbered from left to right, i.e. the left-hand board is assigned the number one, the next one number two etc. There are 4 IDs available; this means that up to 4 boards can be operated in an experimental system.*

## 10.4.2    Globals (CAN-CTRL Subsystem)

A physical CAN controller or CAN connector is assigned to the CAN-CTRL subsystem in the "Globals" tab of a CAN-CTRL subsystem.
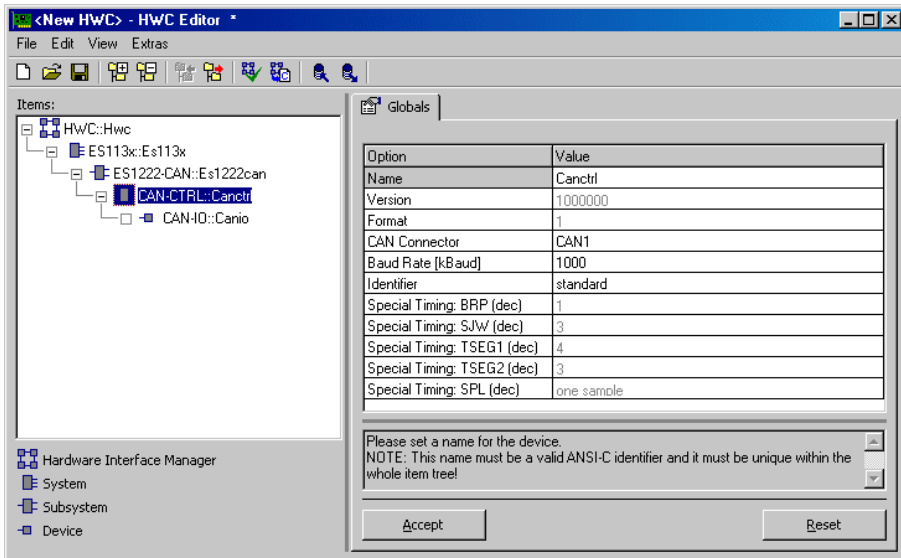


**Fig. 10-13**    The "Globals" Tab of the CAN-CTRL Subsystem

*CAN Connector*

This is where the required CAN controller or CAN connector (port A, port B, port C or port D) is selected.

*Baud Rate [kBaud]*

This is where you can select the required baud rate. The 8 standard baud rates are available (1000, 500, 250, 125, 100, 50, 20,10 kBaud). It is also possible to select the `<Special Timing>` setting which activates the low-level control of the CAN controller in terms of bit timing and baud rate. The five options described below are necessary for specifying the "Special Timing" settings.

*Identifier*

With CAN messages, you can choose between *standard frames* with 11 bit identifiers or *extended frames* with 29 bit identifiers. The length of the identifier field (*standard / extended*) can be specified in this line.

> **Note**
>
> *A mixture of* standard frames *and* extended frames *is **not** supported for CAN-CTRL devices.*

When the *standard* identifier has been selected, only 11 bit values are allowed in the "Groups" tab of the CAN-IO device (cf. "Identifier dec/hex" on page 185). When you enter larger values, the *most significant bits* (MSB) are truncated. A warning is **not** given.

When the *extended* identifier has been selected, only 29 bit values are allowed. When you enter larger values, the *most significant bits* (MSB) are truncated. A warning is **not** given.

*Special Timing: BRP (dec)*

This option is hidden by default. It can only be edited if the "Baud Rate" option is set to `<Special Timing>`.

This parameter is used to set the "Baud Rate Prescaler" which determines the baud rate from the input clock of the CAN controller. The value range of this setting is 0 - 63.

For more information on this setting, refer to the Intel 82527 CAN Controller data sheet.

*Special Timing: SJW (dec)*

This option is hidden by default. It can only be edited if the "Baud Rate" option is set to `<Special Timing>`.

This parameter is used to set the "Synchronization Jump Width". The value range of this setting is 0 - 3.

For more information on this setting, refer to the Intel 82527 CAN Controller data sheet.

*Special Timing: TSEG1 (dec)*

This option is hidden by default. It can only be edited if the "Baud Rate" option is set to `<Special Timing>`.

This parameter is used to set "Time Segment 1" and determines the time segment before the sampling time. The value range of this setting is 2 - 15.

For more information on this setting, refer to the Intel 82527 CAN Controller data sheet.

*Special Timing: TSEG2 (dec)*

This option is hidden by default. It can only be edited if the "Baud Rate" option is set to `<Special Timing>`.

This parameter is used to set "Time Segment 2" and determines the time segment after the sampling time. The value range of this setting is 1 - 7.

For more information on this setting, refer to the Intel 82527 CAN Controller data sheet.

*Special Timing: SPL (dec)*

This option is hidden by default. It can only be edited if the "Baud Rate" option is set to "`<Special Timing>`".

This parameter is used to set the "Sampling Mode" and determines how often the signal is sampled to determine the logical state.

For more information on this setting, refer to the Intel 82527 CAN Controller data sheet.

The formula for calculating the CAN bus frequency (from Intel 82527 CAN Controller data sheet) is:

```
CAN bus frequency =
        10 MHz / [(BRP + 1) x (3 + TSEG1 + TSEG2)]
```

### 10.4.3 Globals (CAN-IO Device)

This CAN-IO device can be used for the simple simulation of a CAN bus partic-ipant which can send and receive CAN messages.
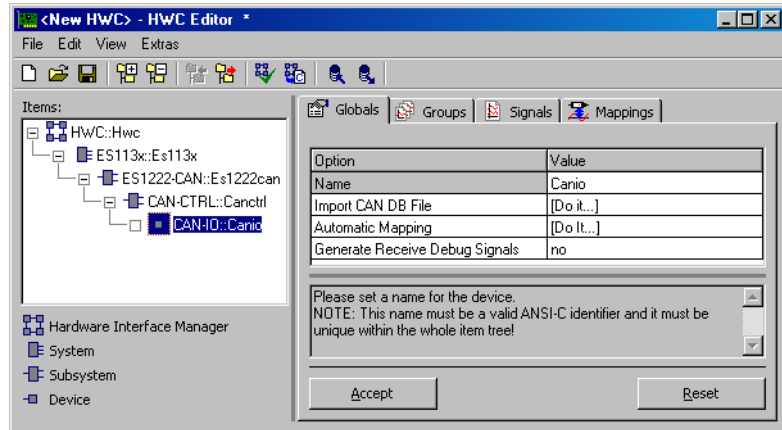


**Fig. 10-14** The "Globals" Tab of the CAN-IO Device

*Import CAN DB File*

This option is used to read a CAN database file which was created with the CANdb data management program made by the company Vector Informatik. CAN messages and signals can be generated automatically if necessary using this file.

A dialog box opens after you press [Do it...]:
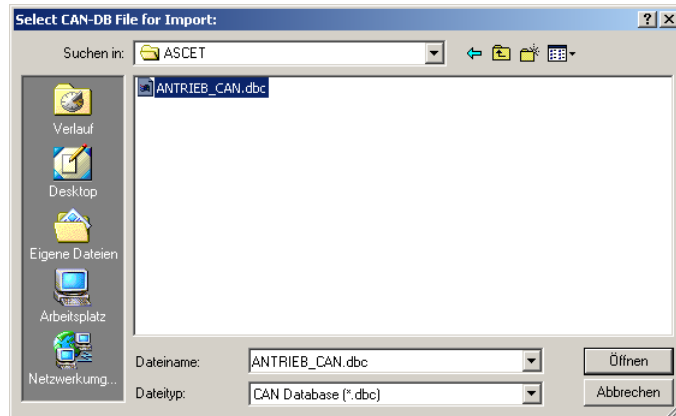
The CAN DB file to be imported can be selected in this dialog box.

After clicking the **Open** button, the following dialog box is displayed and can be used to specify what happens next:
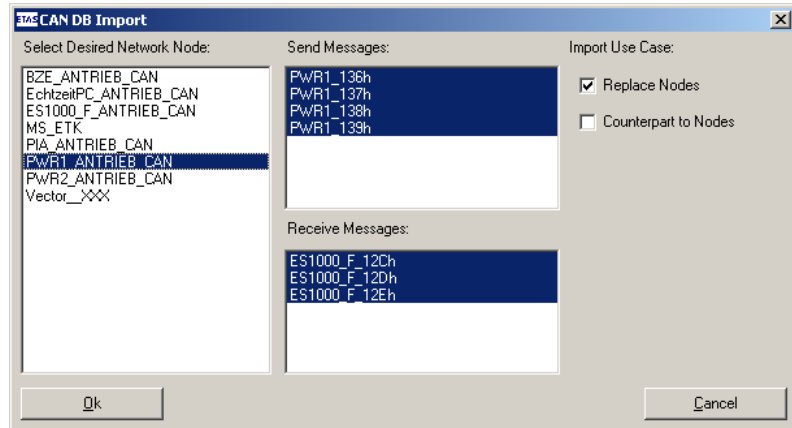


**Fig. 10-15**    CAN DB Import Dialog

A CAN DB file normally describes several nodes of a CAN network. All existing nodes are listed in the "Select Desired Network Nodes" list. The two lists to the right of it ("Send Messages" / "Receive Messages") list all CAN messages defined for the node currently selected. The messages selected are used for import. "Import Use Case" can also be selected. The **Replace Nodes** option means that the CAN-IO device assumes the role of the network node; i.e. a send message of the node is also used as a send message of the CAN-IO device etc..

The **Counterpart to Nodes** option means that the CAN-IO device is the counterpart to the network node; i.e. a send message of the node results in a receive message etc.

After you confirm with **OK**, the data to be imported (CAN messages and signals) with the signal groups and signals is checked. The result of this check is then shown in the dialog box displayed below:
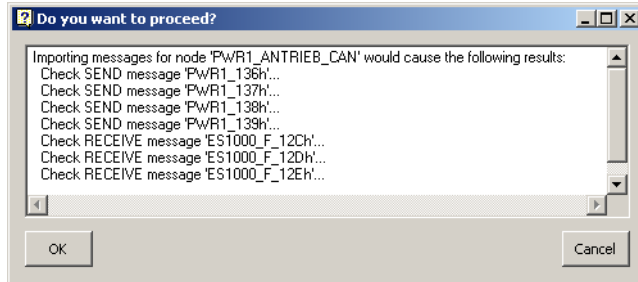
**Fig. 10-16** "CAN DB Check" Dialog Box

So far, the existing CAN-IO device has not been changed. The actual import procedure does not start until the **OK** button is pressed.

The import procedure inserts the imported messages in the signal groups and ensures that relevant signals are defined.

Once import is completed, the detailed protocol of the import procedure is shown in the "Monitor" dialog box:
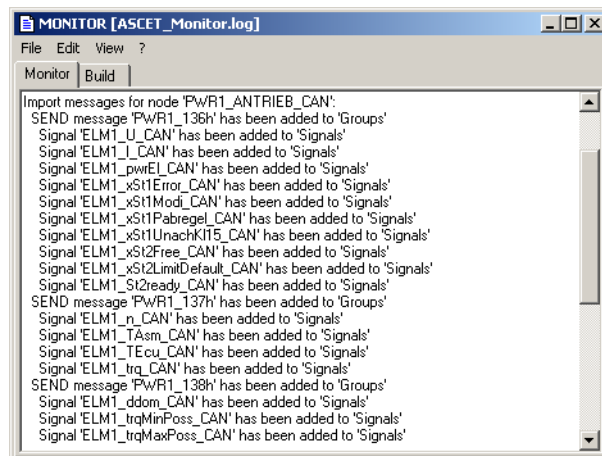


**Fig. 10-17** "Monitor" Window with Import Protocol

**Attention during import:**

When a CAN DB file is imported, only 29 Bit (identifier *extended*), or 11 Bit (identifier *standard*; cf. "Identifier" on page 177), are inserted into the identifier field automatically.

When you selected the *standard* identifier in the CAN Controller item, but the CAN DB file contains signals with 29 Bit identifiers (ID > $2^{31}$), two things happen:

- 11 bits (bits [28...18]) are inserted into the identifier field. The remaining bits (MSB) are rejected.
- Warnings are displayed in the monitor window.

Since only one identifier can be selected, CAN DB files containing both *standard* and *extended* identifiers cause problems.

*Automatic Mapping*

This option makes automatic assignment between signals and (ASCET) messages possible.

Click on [Do it...] or select **Extras → Map Item Signals** to open the "Automatic Mapping" window.
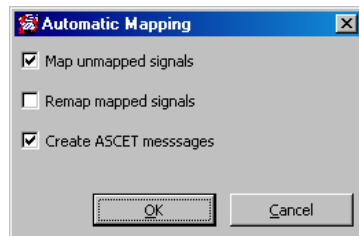


**Fig. 10-18** "Automatic Mapping" dialog window with mapping status

The next part of the procedure can now be selected from the window. The following commands are possible:

- **Map unmapped signals**:
  All signals are mapped that have not been mapped so far (= empty "ASCET Message" field in the "Mappings" tab), i.e. an ASCET message with the same name is searched for for each of these signals. If none is found, the relevant signal is not mapped if **Create ASCET messages** is deactivated.

- **Remap mapped signals**:
  All signals are mapped that were already mapped, i.e. an ASCET message with an identical name is searched for for every mapped signal. If none is found, the relevant signal is no longer mapped (= empty "ASCET Message" field in the "Mappings" tab).

- **Create ASCET messages**:
  When this option is activated, a global message is generated for each missing ASCET message. This means that all selected signals are mapped after this action.

Use the "Signal Mapping" node in the options window (see page 115) to determine whether the messages generated with **Create ASCET Messages** are generated in the project or in one of its included modules. Use the same tab to determine whether transfer direction or signal type shall be considered when creating

*Generate Receive Debug Signals*

If this option is activated (= yes), two additional signals are generated for every "receive" signal group.

- <GroupName>_Diag_dT
- <GroupName>_Diag_Rec

The *...dT* signal specifies the difference in seconds to the previous message received.

> **Note**
>
> *This signal can be used for a normal message receipt (IRQ = no) to monitor the receipt. If, for example, the CANbus is interrupted, the value increases permanently in the grid of the receive task. This value is not protected against overflow (which occurs at approx. 300s) for performance reasons!*

In contrast, no real receipt monitoring can be executed when a message is received with an interrupt (IRQ = yes) as the calculation of the value does not take place until the interrupt task. As the calculation does not take place if no message is received, the old value is frozen. This means that it is impossible to tell when an interrupt is received, whether the receipt takes place in a regular grid or whether it is interrupted.

The *...Rec* signal is described as true every time a message is received. If this signal is mapped to a send-receive message and then reset to false by the application every time it is read, the application can easily determine whether a message has been received between the cycles.

The CAN messages are specified in the form of signal groups in this CAN-IO tab.
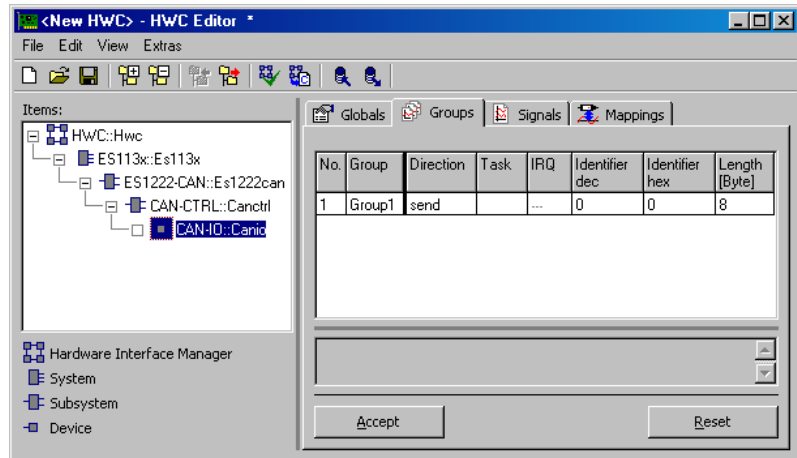


### Note

*New signal groups or CAN messages can be generated via the context menu in the "Groups" tab (for more details please refer to the section "View" Menu on page 108).*

### Note

*After changing the group name, the signal name, or the signal direction, an ASCET message mapped previously may not be mapped automatically any longer and then has to be mapped again manually.*

*Direction*

This is where you can determine the direction of the CAN message ("send" = send message, "receive" = receive message).

*Task*

This is where the task is specified in which the message is to be sent or received. If a receive message is to be received in interrupt mode, this setting is reset and locked.

*IRQ*

This option specifies whether the relevant receive message should be received in interrupt mode or not. For "normal" receipt, the CAN message has to be polled within a task as many times as it can be sent by the counterpart. CAN messages which are not sent in any fixed grid or even only occur sporadically can be problematic in this operating mode. Interrupt reception is ideal for this as it triggers message processing exactly when the message was received.

*Identifier dec/hex*

This is where the message identifier is entered. The value can vary in size here depending on the setting chosen for the identifier type in the superior CAN controller item (standard or extended):

Standard identifier:      11-bit      2 047 dec           7 FF  hex

Extended identifier:      29-bit      536 870 911 dec   F FF FF FF  hex

Each signal group or CAN message has to have a unique identifier.

*Length [Byte]*

Specifies how many useful data bytes the relevant message can transfer (1..8).

*Activated Task\**

For "receive" CAN messages[1] (signal groups), a software task may be entered here that is always activated when the relevant signal group has been received, after the receiving task has been executed. The task entered here can, e.g., be used for post-processing like cleaning up.

This column is hidden by default.

*Prescaler\**

This allows you to set when a VMEbus interrupt is triggered to transfer data with a message received in interrupt mode. The default setting "1" means that for every message received, data transfer also takes place. If this value, for example, is increased to "2", a VMEbus interrupt is only triggered every second message received which then transfers the data of the message last received.

---

[1] It is possible to specify such a task for "send" groups, too, but there is no recommended usage for such a utilization. (As the sending process is handled asynchronously, the CAN message may or may not have been sent when that task is activated.)

This column is not displayed by default.

### 10.4.5  Signals (CAN-IO Device)

The CAN I/O signals are specified in this CAN-IO tab.
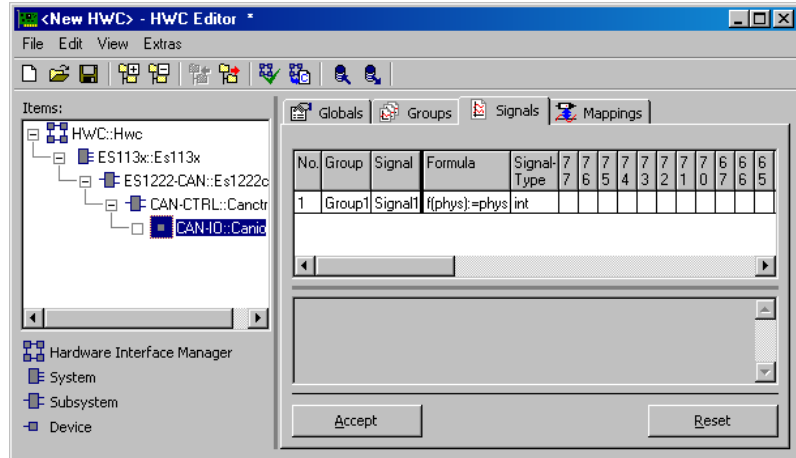
*Group*

This is where a signal is allocated to the required signal group.

*Signal Type*

This is where the signal type is determined in which the signal is transferred via the CAN bus.

The following settings are possible:

| Signal Type | Data Type |
|---|---|
| int | Characterizes a signed signal in the default complement to two data format (max. 32-bit) |
| (s)int | Characterizes a signed signal in which the sign is transferred as the most significant bit and then the absolute value of the signal is transferred. If the sign bit is set, this is a negative number (max. 32-bit) |
| uint | Characterizes an unsigned signal (max. 32-bit) |
| bool | Characterizes a Boolean signal. Only one bit can be marked in the bit matrix. |
| real | Characterizes a floating-point value in "standard IEEE float (4 byte)" format. Accordingly, only 32 bits can be marked in the bit matrix. |

The following table provides an overview of the IEEE floating-point formats:

| Format | Sign | Exponent | Fraction |
|---|---|---|---|
| Float | 1 bit | 8 bits | 24 bits |
| Double (not supported) | 1 bit | 11 bits | 52 bits |

*7654321.. (Bit matrix)*

A CAN message can transfer up to 8 data bytes. A bit matrix can specify which bits each signal requires or occupies (a signal = a row).

The columns are structured as follows:

| 7 | 7 | ... | 0 | 0 | Byte number |
|---|---|---|---|---|---|
| 7 | 6 | ... | 1 | 0 | Bit number |

**Significance of the bit fields:**

| Empty field | The relevant signal does not use the bit |
|---|---|
| Occupied field | The signal requires this bit at the position |
| "X" field | The relevant bit is not available for data transfer because the signal has fewer useful data bytes (see "Length" in the "Groups" tab). |

**Operating the bit fields:**

The required bit cell can be selected using the arrow keys on the keyboard.

Use the number keys to assign the bit with the correct value (different values are important for "block building", see below).

Clicking with the mouse also "toggles" the relevant cell between "empty" and "1". If you press the <Alt> key and click the mouse simultaneously, the value is incremented from "1" to "9" (important for "block building").

You can select several bits simultaneously as follows:

| 7 7 | ... | 4 0 | 3 7 | 3 6 | 3 5 | 3 4 | 3 3 | 3 2 | 3 1 | 3 0 | 2 7 | 2 6 | 2 5 | 2 4 | 2 3 | 2 2 | 2 1 | 2 0 | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 0 7 | 0 6 | 0 5 | 0 4 | 0 3 | 0 2 | 0 1 | 0 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Klicken um erstes Bit auszuwählen

Klicken um zweites Bit auszuwählen

<Shift> Taste drücken und nochmals klicken um zweites Bit auszuwählen

Der Block wurde gefüllt
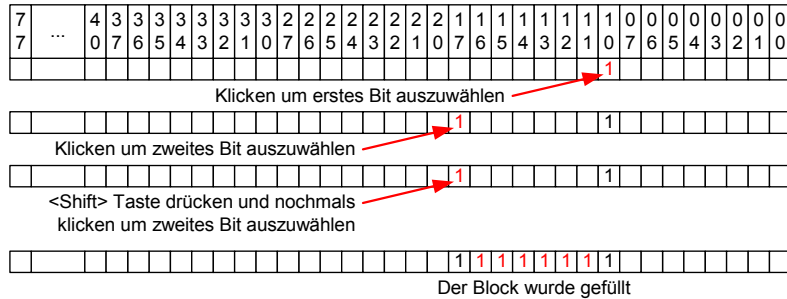
Data blocks can be created using different numbers ("1111   2222...") with which virtually every signal transferred can be described. The numbers used to form the data blocks have the significance that the block with the highest number ("2222") specifies the block which contains the most significant bits during transfer. The block with the smallest number ("1111") contains the least significant bits during transfer. With the numbers available (1...9), you can write a signal with up to 9 bit blocks. As the representation of the bits corresponds exactly to the form in which an Intel signal is transferred, block building is necessary for the representation of signals in Motorola format as soon as the signal is longer than 8 bits.

Examples of the definition of different signals:

| 7 | ... | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

16 Bit Signal im Intel Format

| | | | | | | | | | | | | | | | | | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Andere Beschreibung des selben Signals

| | | | | | | | | | | | 2 | 2 | 2 | 2 | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

12 Bit Signal (Intel Format) mit Lücke

| | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

16 Bit Signal im Motorola Format

## 10.4.6 Mappings (CAN-IO Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

## 10.5 ES1222-CAN Bypass (CAN Bypass Protocol CBP)

CAN bypass applications are less complex but also less efficient than ETK bypass applications. Only the ECU software has to be adapted in a CAN bypass (customization); modifications to the hardware are not necessary.

A CAN bypass is of course only possible with ECUs which have a CAN interface. Two fundamental methods exist to create a Bypass with CAN:

- CAN Bypass over messages and assigned signals, which are firmly predefined in the ECU software. This isn't made possible about the CAN Bypass Device but about the CAN-IO Device (see chapter 10.4 on page 174).

- CAN Bypass over an agreed protocol, in which a selection of the sizes to be transferred can be made. Please pay attention to the licensing agreement in the following chapter.

## 10.5.1 License legal note for the CAN Bypass protocol (CBP)

The CAN Bypass protocol (CBP) is a development of the Robert Bosch GmbH Stuttgart and is common applied by the Robert Bosch GmbH and the ETAS GmbH. All rights reserved by the Robert Bosch GmbH.

The use of the CAN Bypass protocol in the interaction of ASCET-RP and ECUs of the Robert Bosch GmbH, which integrates this protocol, is possible without further licensing agreement.

You receive information about suitable ECUs with CBP support from the ECU manufacturer.

An interface description of the CBP in coherence with ASCET-RP is available at the ETAS GmbH on enquiry.

Note

**Note**

*The surrender of the interface description is only carried out, if the corresponding license agreement is accepted. There is not an entitlement to this license due to an ASCET-RP licensing agreement!*

### 10.5.2 Hardware Configuration of a CAN Bypass

The following figure shows a sample configuration of a CAN bypass application with the ES1000.2/ ES1000.3:
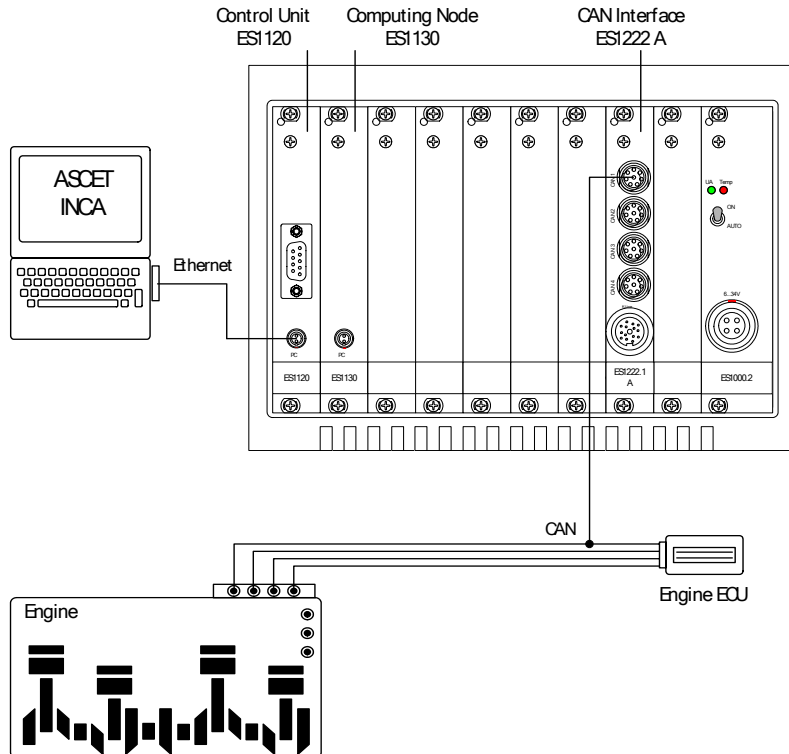
As with all rapid-prototyping applications, the host PC is connected to the ETAS experimental system via the host link interface. The functions developed with ASCET run on the PPC module, ES113x. The ECU is connected to the ETAS experimental system via the ES1222 CAN interface board.

Communication with the ECU takes place using the CAN bypass protocol (CBP). (© Copyright Robert Bosch GmbH).

10.5.3    Globals (CAN-Bypass Device)
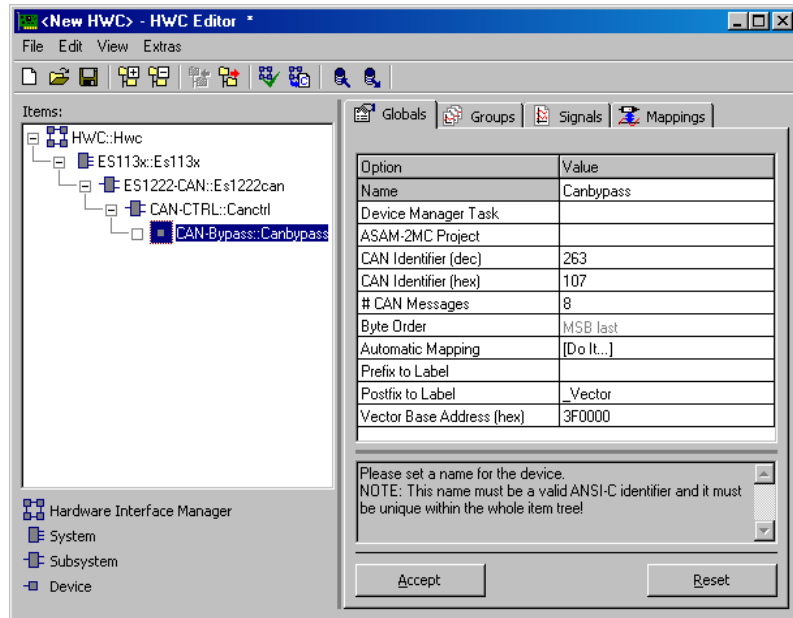


**Fig. 10-19**    The "Globals" Tab of the CAN Bypass Device

*Device Manager Task*

A timer task has to be assigned here which is available exclusively to this bypass. The period duration of the timer task has to be set between 0.05 seconds and 0.8 seconds.

*ASAM-2MC Project*

An ASAM-MCD-2MC project suitable for the ECU and which has already been read into the database has to be selected here.

*CAN Identifier (dec)*

This is where the CAN identifier of the CAN message command is specified in decimal notation. The CAN message command is the message with the highest identifier of the CAN message block (the default value for CBP is 263). The length of the identifier (11 bit / 29 bit) depends on the "Identifier" setting in the "Globals" tab of the CAN-CTRL subsystem.

The format of the CAN message block and the command message is defined in the CBP interface description.

*CAN Identifier (hex)*

This is where the CAN identifier of the CAN message command can be specified in hexadecimal notation (the default value for CBP is 107H).

*# CAN Messages*

This is where the number of available CAN messages for the CAN message block has to be set (4 .. 16). The CAN message block normally has 8 CAN messages with the CAN bypass protocol.

**Note**

*The higher the number of CAN messages, the more bypass variables can be transferred. As the ECU software also has to provide the resources, however, this entry cannot be set freely but must be coordinated with the ECU software version.*

*Byte Order*

This line displays the word data storage format (MSB first / MSB last) of the ECU processor. This information is read from the allocated ASAM-MCD-2MC project.

Overview of the common "byte order" terms:

| | | |
|---|---|---|
| MSB first | big endian | Motorola |
| MSB last | little endian | Intel |

*Automatic Mapping*

For more information please refer to the description "Automatic Mapping" in section chapter 10.4.3, section "Automatic Mapping" on page 182.

This is where the name extensions can be specified to be able to identify the relevant bypass parameters from the bypass output value (ASAM-MCD-2MC measurement). This name reference is used to detect the available bypass output values ("send" signals to the ECU) and to determine all the necessary information for a bypass signal.

The following figure clarifies the situation:
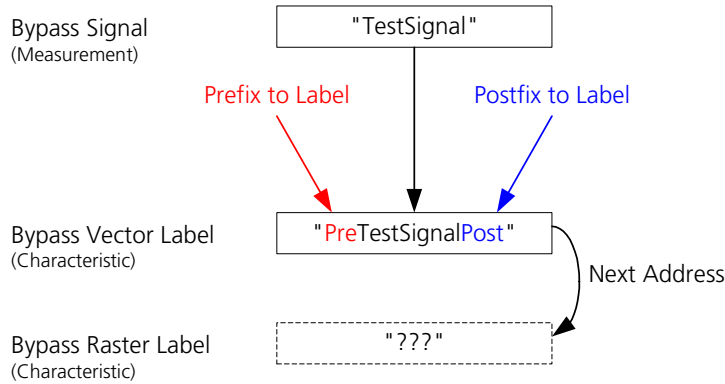


**Fig. 10-20**   Relation between the Bypass Labels

**Bypass Label Task:**

| | |
|---|---|
| Bypass Vector Label | Specifies the vector address of the bypass variable. A 0 means that the value is not transferred. |
| Bypass Raster Label | Specifies whether the bypass variable is transferred in the angle-synchronous grid (value = 0) or in the time-synchronous grid (value = 1). |

*Vector Base Address (hex)*

This is where a base address can be specified which is used to calculate the target address of the bypass output values ("send" signals to the ECU) in the ECU. This is necessary if the target address is outside the address space which can be addressed via the 16 bits available in the CAN bypass protocol.

The target address is calculated as follows:

```
<bypass address> :=

<vector address> + <ecu calibration offset> - <base
address>
```

`<ecu calibration offset>` is determined via the ASAM-MCD-2MC project.

### 10.5.4 Groups (CAN-Bypass Device)
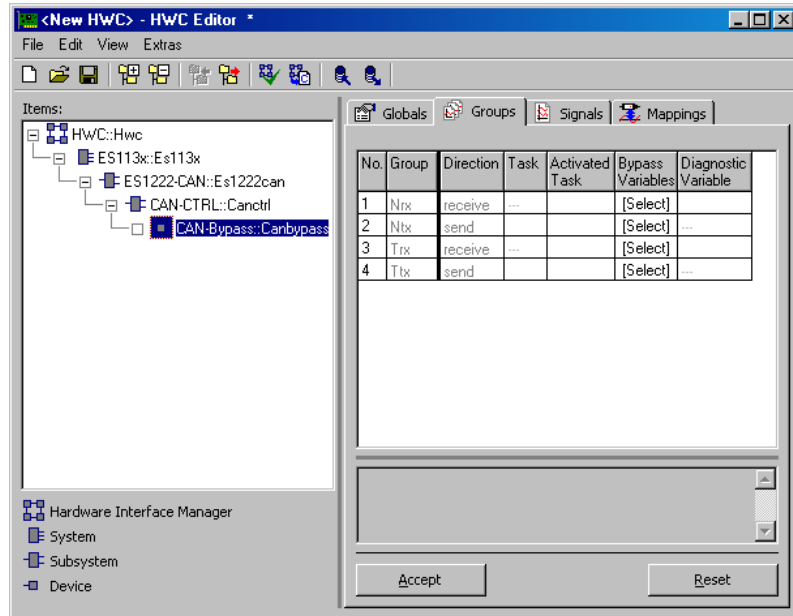


**Fig. 10-21** The "Groups" Tab of the CAN-Bypass Device

---

**Note**

*After changing the group name, the signal name, or the signal direction, an ASCET message mapped previously may not be mapped automatically any longer and then has to be mapped again manually.*

---

*Group*

The CAN-Bypass device supports up to two grids - one in send and one in receive direction. The signal groups have the following significance:

Nrx   Speed-synchronous data from the ECU to the CAN-Bypass device

Ntx   Speed-synchronous data from the CAN-Bypass device to the ECU

Trx   Time-synchronous data from the ECU to the CAN-Bypass device

Ttx   Time-synchronous data from the CAN-Bypass device to the ECU

This is the column in which software tasks have to be specified which are to be activated when speed-synchronous or time-synchronous data is received. The calculation of the bypass functions and the returning of the results to the ECU also take place in these tasks (cf. the corresponding entries in the "Task" column).

*Bypass Variables*

By selecting a cell from this column, the Select Variables dialog box is opened from which the bypass variables (measurements) for the relevant signal group can be selected.
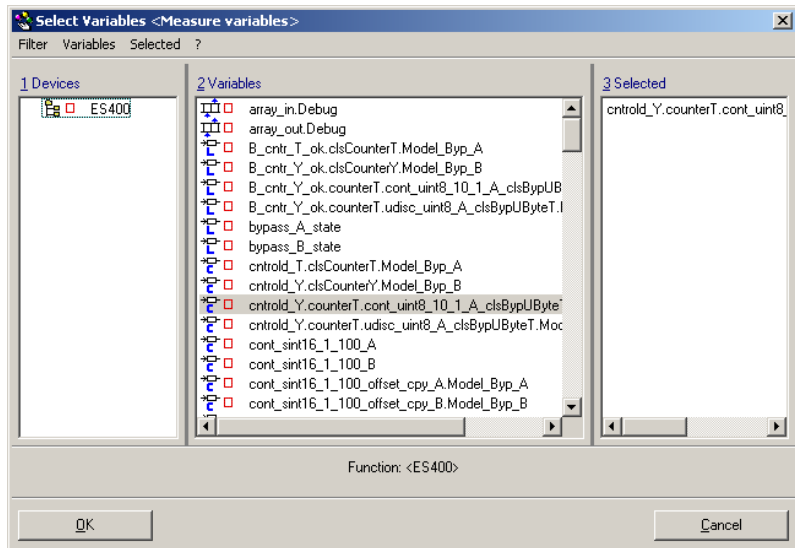


**Fig. 10-22**   The "Select Variables" Dialog Box

---

**Note**

*Non-linear formulas are not supported. If a variable with non-linear ormula is selected nonetheless, a warning of the following kind is displayed:*

<RTIO Toolbox WARNING>
Error on processing signal '<signal_name>' formula: Can't handle '<formula_name>' for bypass; formula is converted to 'FormulaId'!

All measured values defined in the ASAM-MCD-2MC project are basically available to the "receive" signal groups. Only those measured values are available for the "send" signal groups which are supported by the ECU software as bypass intervention variables and are clearly bypass intervention variables from their name reference (see also "Prefix / Postfix to Label" in the "Globals" tab).

<div style="border-left: 4px solid;">

**Note**

*The number of measured values is limited and depends on the data type of the measured values (byte, word...) as well as the number of available bypass CAN messages (entry "# Bypass Messages" in the "Globals" tab).*

</div>

*Diagnostic Variable*

If necessary, an additional value of the ECU can be selected here for diagnostic purposes for every "receive" signal group (e.g. to monitor bypass operation).

Selection takes place as described under "Bypass Variables".

### 10.5.5 Signals (CAN-Bypass Device)
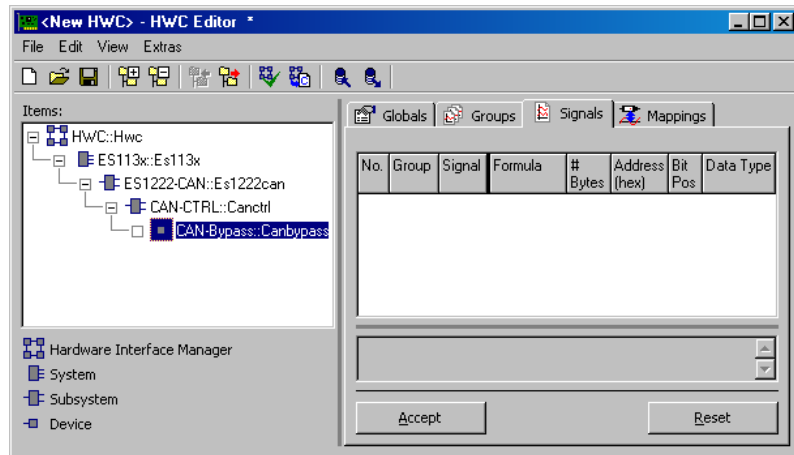


**Fig. 10-23** The "Signals" Tab of the CAN-Bypass Device

<div style="border-left: 4px solid;">

**Note**

*None of the columns of the "Signals" tab can be edited by the user. They are intended purely for display purposes of status values for the bypass variables.*

</div>

The bypass variables are sorted according to their byte size within the relevant signal group (8-4-2-1 bytes).

*# Bytes*

The number of bytes per bypass variable is displayed in this column.

*Address (hex)*

The addresses of the bypass variables are displayed in this column.

Receive values have a 32-bit address space; send values have a 16-bit address space.

*Bit Pos*

This column is unused as the standard ECU software at the moment does not support bypass of bit values.

*Data Type*

The data type of the bypass variables is displayed in this column.

*Byte Offset*

Display of the offset of the bypass variable within the relevant signal group.

## 10.5.6 Mappings (CAN-Bypass Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

## 10.6 ES1223-LIN

The ES1223 board is used as an LIN interface in VMEbus systems. The board provides four LIN channels. It also makes it possible to trigger VMEbus interrupts with received messages so that permanent polling after messages is not necessary. Each LIN controller can process up to 64 messages. The use of an independent processor means the system controller is considerably relieved.

This section describes the RTIO link of the ES1223 board. The ES1223 board is integrated in the HWC editor by selecting the "ES1223-LIN" item.

### 10.6.1 Globals (ES1223-LIN Subsystem)
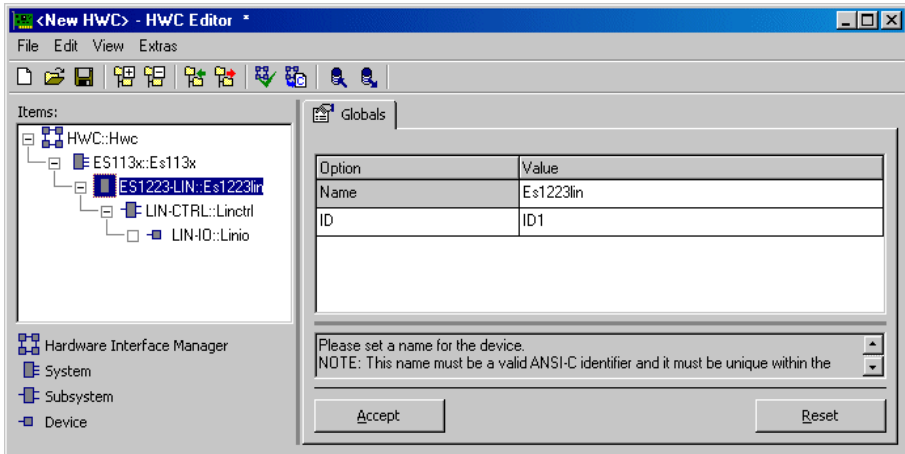


**Fig. 10-24**   The "Globals" Tab of the ES1223-LIN Subsystem

*ID*

The board number of the board to be addressed has to be entered in the "ID" selection field.

> **Note**
>
> *The ES1223.1 board is what is referred to as an "Auto-ID" board which is automatically assigned an ID number and a free address space by the hardware manager when the system is switched on. Boards of the same type (e.g. two ES1223.1s) are numbered from left to right, i.e. the left-hand board is assigned number one, the next board number two. There are 2 IDs available; this means that up to 2 ES1223.1 boards can be operated in an experimental system.*

### 10.6.2 Globals (LIN-CTRL Subsystem)

An "Init Task", an "Exit Task" and an "IRQ Handler Task" are assigned to the LIN-CTRL subsystem in the "Globals" tab.

Up to four LIN-IO devices can be assigned to the LIN-CTRL subsystem corresponding to the four LIN interfaces on the board.
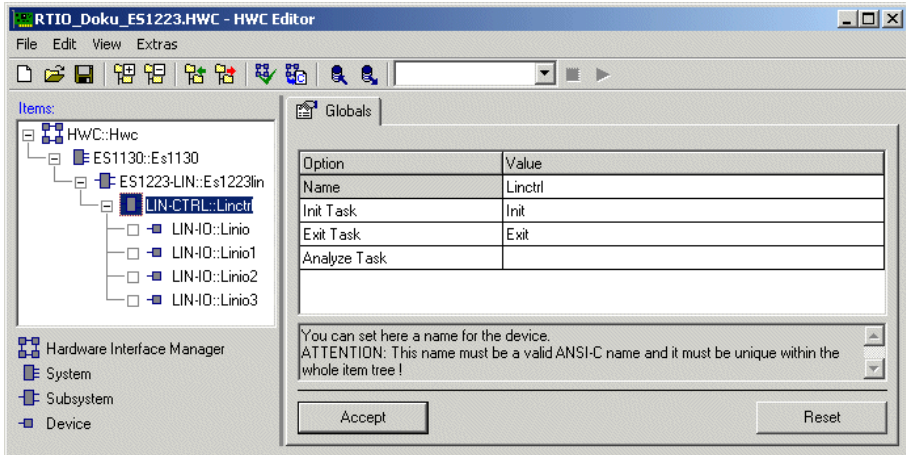


**Fig. 10-25**   The "Globals" Tab of the LIN-CTRL Subsystem

*IRQ Handler Task*

An additional "IRQ Handler Task" is required to support interrupts of the ES1223 board. This task must be created as a "software" task in the task list in the ASCET project editor whereby "Max. No. of Activations" has to be at least 2 (maximum of 50).

### 10.6.3 Globals (LIN-IO Device)
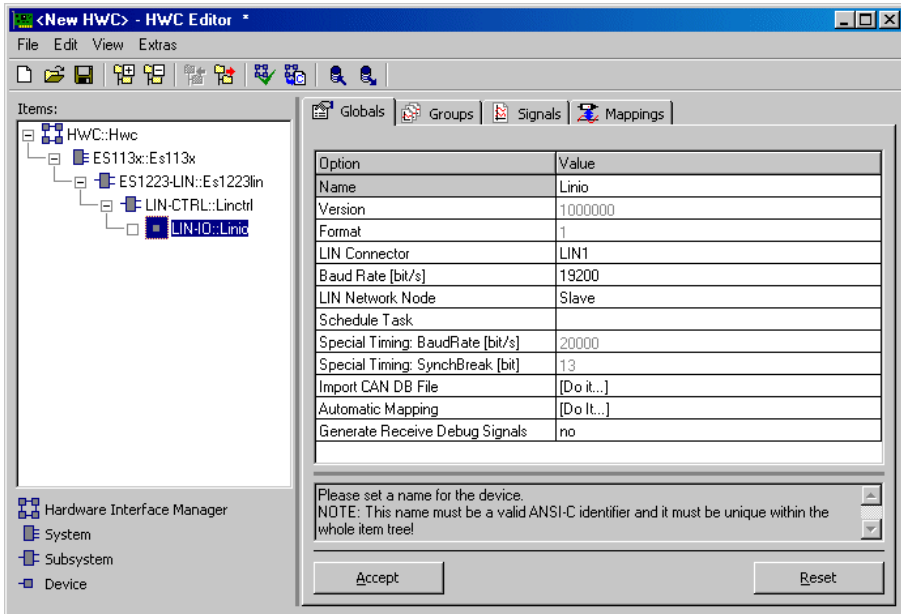


**Fig. 10-26** The "Globals" Tab of the LIN-IO Device

*LIN Connector*

This is where the desired LIN connector is selected (LIN1 to LIN4). Each LIN-IO device has to be assigned to its own connector.

*Baud Rate [bit/s]*

This is where the required baud rate can be selected. The standard baud rates (19200, 9600, 2400 bit/s) are available. It is also possible to select the <Special Timing> setting which activates low-level addressing of the LIN controller in terms of the bit-timing and baud rate.

These settings can be specified in the "Special Timing" options described on page 201.

*LIN Network Node*

This option determines whether the selected LIN-IO device works as a master or slave control unit. Please note that only one master unit is possible in the LIN network.

*Schedule Task*

This option is only available if the "LIN Network Node" option is set to `Master` for the selected LIN-IO device. It is used to define a task for a master control unit which is always executed when the send queue is empty. This is useful for the continuous sending of messages.

*Special Timing: BaudRate [bit/s]\**

This option is masked out by default. It can only be edited if the "Baud Rate" option is set to `<Special Timing>`.

Unlike with the standard transfer rates (see "Baud Rate [bit/s]" on page 200), any baud rate between 10 and 20,000 bit/s can be entered here.

*Special Timing: SynchBreak [bit]\**

This option is masked out by default. It can only be edited if the "Baud Rate" option is set to `<Special Timing>`.

The synchbreak length includes the actual synchbreak (low phase) as well as the synch delimiter (1 bit). Unlike the standard (at least 14 bits), any synch length between 8 bits and 15 bits (low time) can be entered here.

*Import CAN DB File*

This option is used to import a CAN/LIN database file which was created using the CANdb data administration program, developed by the company Vector Informatik. This file can be used to generate LIN messages and signals automatically when required.

Please consult the section "Import CAN DB File" on page 179 for details on how to execute the import.

*Automatic Mapping*

This option makes the automatic mapping of signals and (ASCET) messages possible.

For more details, consult the section "Automatic Mapping" on page 182.

*Generate Receive Debug Signals*

If this option is activated (= `yes`), two additional signals are generated for every "receive" signal group.

- `<GroupName>_Diag_dT`
- `<GroupName>_Diag_Rec`

For more details, consult the section "Generate Receive Debug Signals" on page 183.

## 10.6.4 Groups (LIN-IO Device)

This LIN-IO tab specifies LIN messages in the form of signal groups.
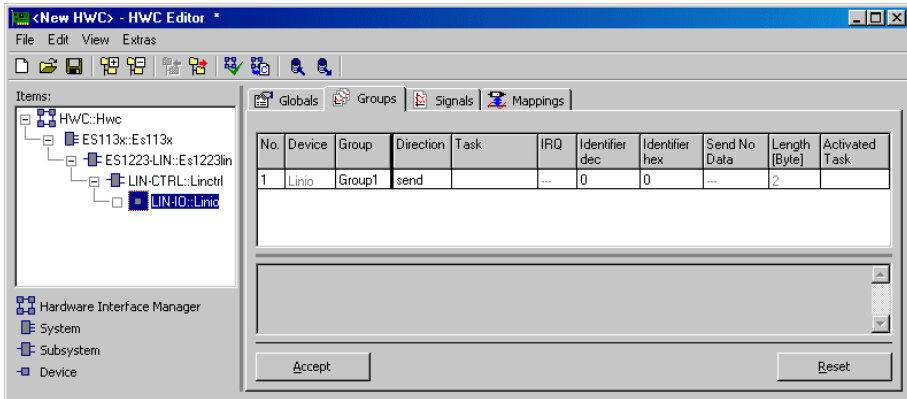


**Fig. 10-27**   The "Groups" Tab of the LIN-IO Device

New signal groups or LIN messages can be generated using the shortcut menu in the "Groups" tab (see the section " "View" Menu" on page 108).

### Note

*When the signal group name, signal name or signal direction is changed, any ASCET message which may have been assigned may not be assigned automatically any more and may then have to be reassigned manually.*

Please consult section 10.4.4 "Groups (CAN-IO Device)" on page 184 for a description of the options. The LIN-IO device has the following special features:

- the "Length [Byte]" option is masked out and write-protected by default for the LIN-IO device as the length of the LIN message is coded in the identifier
- the "Prescaler" option is not available
- the "Send No Data" option is LIN-specific and is described here

*Send No Data*

This option is only available if the "LIN Network Node" optionin the "Globals" tab is set to Master, and send is selected in the "Direction" column in the "Groups" tab.

If "Send No Data" is set to Yes, only the header of the message is transferred, not the data. The length of the message is thus Length [Byte] = 0.

> **Note**
>
> *To receive data from a slave node with a master node, two signal groups* **with the same identifier** *have to be created:*
>
> *- one signal group with the direction "send" and the option "send no data",*
> *- one signal group with the direction "receive".*
>
> *The master node sends the header of the message with the first signal group; with the second, it receives the data from the slave node.*

### 10.6.5    Signals (LIN-IO Device)

The LIN signals are specified in this LIN-IO tab.
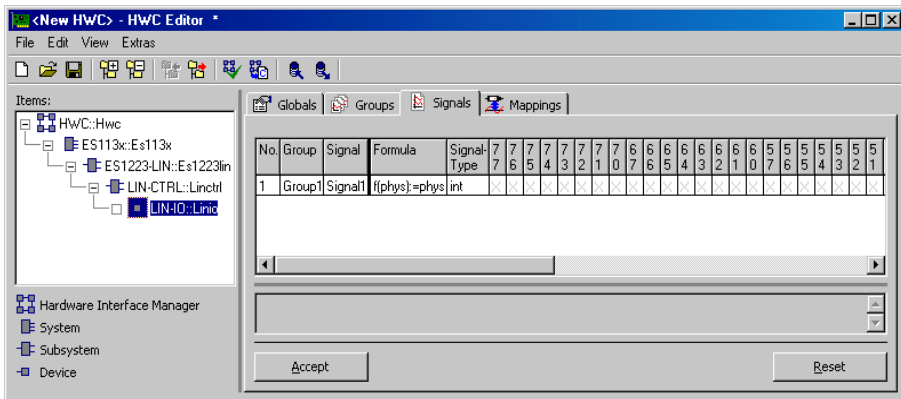


**Fig. 10-28**    The "Signals" Tab of the LIN-IO Device

New signals can be generated via the shortcut menu in the "Signals" tab (see section ""View" Menu" on page 108).

Please refer to section 10.4.5 "Signals (CAN-IO Device)" on page 186 for a description of the columns of the table.

> **Note**
>
> *The length of a message is automatically coded in the identifier ("Groups" tab). The top 3 bits of the identifier specify the length - 2, 4, and 8 bytes are all possible. This means that all higher fields of the bit matrix are deactivated.*

### 10.6.6 Mappings (LIN-IO Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

### 10.6.7 Runtime Behavior

The LIN bus is managed in interrupt mode by the ES1223 processor. Due to the interrupt management, send and receive commands can be delayed by up to 250 µs.

## 10.7 ES1231.1-ETK

The ETK interface board ES1231 is the successor to the ES1200 board and is also used to link an ECU with an ETK to an experimental system.

The board has extended functionality in comparison to its predecessor, such as block transfer mode and a higher transfer rate which enables a much higher data rate with corresponding support. This board is also equipped with the memory-saving "Auto-ID" technology.

> **Note**
>
> *The ES1231 board requires special system services to communicate with an ETK which can only be provided by an ES1120 (VCU2) system controller board. (See the section "Hardware – ES1000.x Experimental System" on page 69.)*

### 10.7.1 Globals (ES1231-ETK Subsystem)
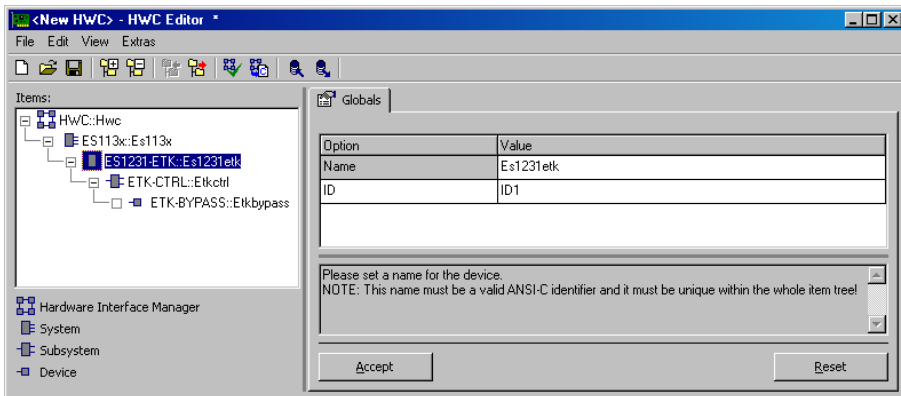


**Fig. 10-29**   The "Globals" Tab of the ES1231-ETK Item

*ID*

The board number of the board to be addressed has to be entered in the "ID" field.

**Note**

*The ES1231 board is an "Auto-ID" board which is automatically assigned an ID number and a free address space by the Hardware Manager when the system is switched on. Boards of the same type (e.g.two ES1231s) are numbered from left to right, i.e. the left-hand board is assigned the number one, the next one number two etc. There are 4 IDs available; this means that up to 4 boards can be operated in an experimental system.*

As the ES1231 only has one ETK controller, only one "ETK-CTRL" item can be inserted (only port A is supported).

10.7.2    Globals (ETK-CTRL Subsystem)

See chapter 10.3.2 "Globals (ETK-CTRL Subsystem)" on page 157.

10.7.3    Globals (ETK-BYPASS Device)

See chapter 10.3.3 "Globals (ETK-BYPASS Device)" on page 159.

10.7.4    Groups (ETK-BYPASS Device)

See chapter 10.3.4 "Groups (ETK-BYPASS Device)" on page 167.

### 10.7.5    Signals (ETK-BYPASS Device)
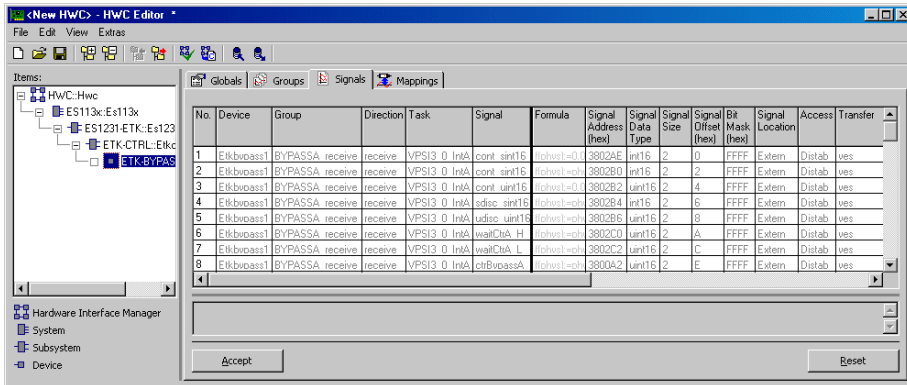


**Fig. 10-30**    The "Signals" Tab of the ETK-BYPASS Device

> **Note**
>
> *None of the columns of the "Signals" tab can be edited by the user. They are solely intended for the display of status values for the bypass variables.*

The "Signals" tab corresponds largely to the description given in chapter 10.3.5 "Signals (ETK-BYPASS Device)" on page 170. Only two additional columns cab be found, which are described here.

*Bit Mask (hex)*

ES1231 always reads a byte (8 bit) or a word (16 bit) from the ETK controller. ASAM-MCD-2MC and the HWC editor, on the other hand, allow the usage of bit signals. The "Bit Mask (hex)" column contains the mask which is used to determine the signal from the value read by the controller (bit-wise AND with the mask).

In principle, appropriate masks (`0x1, 0x2, 0x4, 0x8, 0x10, 0x20, 0x40,0x80`) can be used to create eight signals from one byte value. With FAR addressing, two byte values are created from one word value (masks `0x00FF` and `0xFF00`).

The column is hidden by default.

*Transfer*

When two or more signals are created from one byte value, the system notices that the value was previously read. Transfer time can be saved by using a local copy for further accesses to the value. To use this feature, insert `Yes` in the "Transfer" column at the first access to a given address, and `No` at all further accesses to the same address.

The column is hidden by default.

### 10.7.6 Mappings (ETK-BYPASS Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

## 10.8 ES1232 -ETK

The ES1232 ETK interface board is the successor to the ES1231 board and is also used to link an ECU with an ETK to an experimental system.

The board features extended functions in comparison to its predecessor, such as indirect addressing, transfer rates of 8 Mbit/s and 100 Mbit/s, up to 32 measurement rasters, importing of new ASAM-MCD-2MC files (AML V1.2.0; ETK-CTRL-ADV subsystem, cf. chapter 10.8.5), and support of serial ETKs.

### 10.8.1 Globals (ES1232-ETK Subsystem)
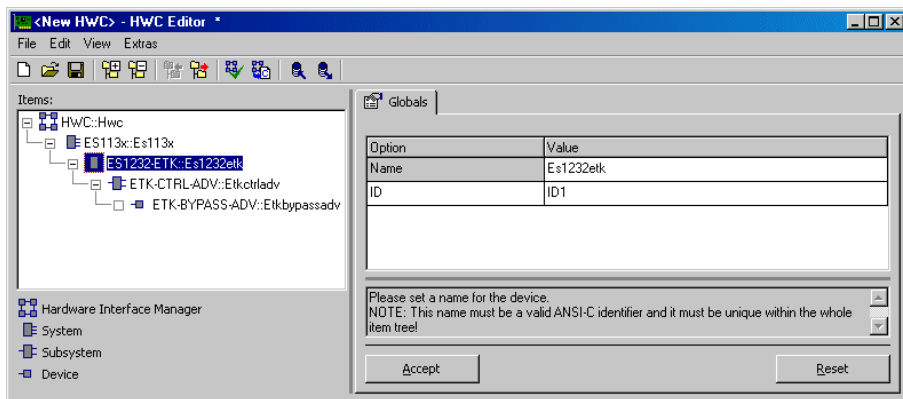


**Fig. 10-31**   The "Globals" Tab of the ES1232-ETK Item

Enter the board number of the addressed board in the "ID" field. The board number also indicates the board positon in the VME system; ID1 means the first ES1232 from the left, ID2 means the second, etc.

---

**Note**

*The ES1232 board is an "Auto-ID" board which is automatically assigned an ID number and a free address space by the Hardware Manager when the system is switched on. Boards of the same type (e.g.two ES1232s) are numbered from left to right, i.e. the left-hand board is assigned the number one, the next one number two etc. There are 4 IDs available; this means that up to 4 boards can be operated in an experimental system.*

Since the ES1232 only has one ETK controller, only one "ETK-CTRL" item can be inserted (only port A is supported).

### 10.8.2 ETK-CTRL-BAS Subsystem
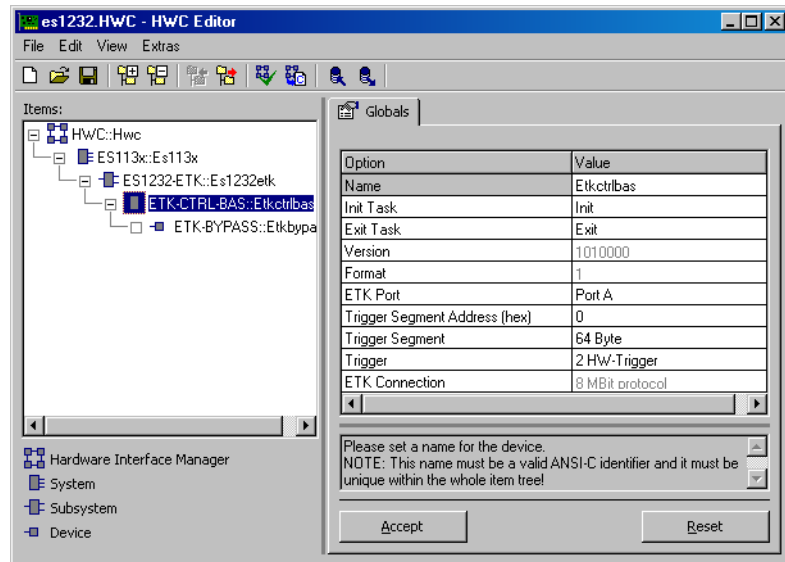


**Fig. 10-32** The "Globals" Tab of the ETK-CTRL-BAS subsystem

The settings for ETK-CTRL-BAS are identical to those for the ETK-CTRL item of the ES1231-ETK device (chapter 10.7 "ES1231.1-ETK" on page 204), with the exception of the "ETK Connection" option.

*ETK Connection*

This line shows the transfer rate (8 Mbit/s or 100 Mbit/s). The field cannot be edited; the value is read from the ASAM-MCD-2MC file (cf. page 216).

### 10.8.3 ETK-BYPASS Device

The settings for the ETK-BYPASS device associated with the ETK-CTRL-BAS subsystem correspond in structure and functionality to the ETK-BYPASS item of the ES1231-ETK board (chapter 10.7 "ES1231.1-ETK" on page 204).

### 10.8.4 100 Mbit/s for Existing Projects (ETK-CTRL-BAS Subsystem)

It is possible to use existing projects (AML V1.1.0) with ES1232 and the ETK-CTRL-BAS subsystem.

The transmission rate is specified in the `TP_BLOB` of the ASAM-MCD-2MC file (`*.a2l`). For that purpose, AML V1.1.1 and higher versions contain the `INTERFACE_SPEED` parameter.

If you want to continue using the old project, without the high transfer rate of 100 Mbit/s, yo need not change the ASAM-MCD-2MC file. Proceed as follows.

**Using an existing project with 8 MBit/s:**

- In the HWC editor, create the required item tree (see chapter 7.2 and chapter 11.2.3).
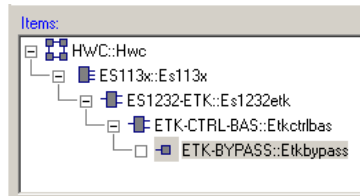


- Use **Edit → Copy → Item(s)** and **Edit → Paste → to selected Item** to copy the existing ETK-BYPASS device of the ES1231 with all settings to the ES1232, where appropriate.

- Use **Edit → Copy → Item Data** and **Edit → Paste → Data to selected Item** to copy the ETK-BYPASS data of the ES1231 to the ES1232.

**Note**

*Only the **ETK-BYPASS device, or its data,** can be copied, and only **between** the **ES1231 and ES1232 with ETK-CTRL-BAS** subsystem. If you try to copy other items or data (see figure below), the procedure is cancelled, and an error message is displayed.*
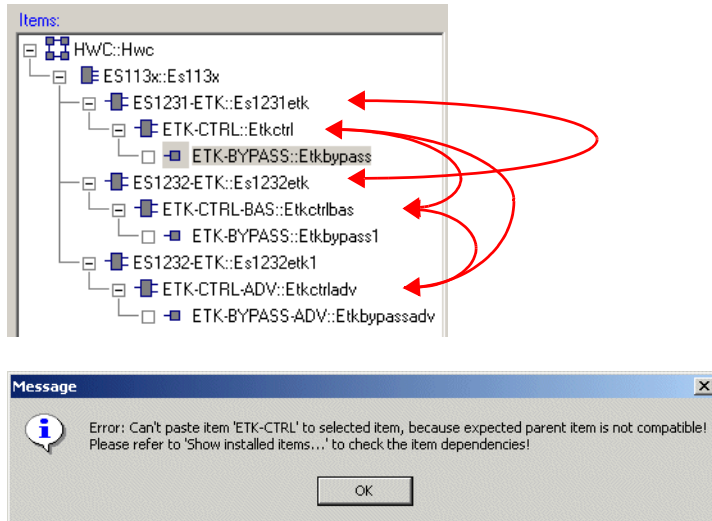




- Click the "ASAM-2MC Project" field on the "Globals" tab of the ETK-BYPASS device (ES1232 / ETK-CTRL-BAS) to select the ASAM-MCD-2MC file (cf. "ASAM-2MC Project" on page 160).

The following warning is shown in the ASCET monitor window:
`Parameter 'Interface Speed' is`

The "ETK Connection" line on the "Globals"
tab of the ETK-CTRL-BAS item shows the
transfer rate of 8 Mbit/s.

If you want to use the old project with the high transfer rate of 100 Mbit/s, you
have to change the ASAM-MCD-2MC file and adjust the settings in the HWC
editor. Proceed as follows.

**Using an existing project with 100 Mbit/s:**

- In the HWC editor, create the required item
  tree as described on page 209.
- Open the ASAM-MCD-2MC file in a text edi-
  tor.

  In AML V1.1.0, the `TP_BLOB` is defined as fol-
  lows:

```
/begin TP_BLOB 0x1000001 0x0 0x0

  /begin DISTAB_CFG  0xC 0x1 MSB_LAST 0x383000 0x0

    TRG_MOD 0xB7

  /end DISTAB_CFG

  CODE_CHK

  0x0

  0x0

  0x0

  0x0

  ETK_CFG 0xF 0xF0 0xFF 0x3 0xFD 0xEE 0xFF 0x1

  RESERVED 0x810000 0x8103F9 EXTERN 0x-1 0x-1 0x-1
          0x-1 0x-1

/end TP_BLOB
```

For illustration purposes, the first line is
repeated with additional comments.

```
/begin TP_BLOB
    0x1000001    /* TP_BLOP version        */
    0x0          /* Project Base Address   */
    0x0          /* RstCfg parameter (MPC) */
```

- To use the 100 Mbit/s, update the `TP_BLOB` version and replace the `Project Base Address` parameter with the `INTERFACE_SPEED` parameter.

**Note**

*Make sure that the `TP_BLOB` you change is placed in an* **`IF_DATA ETK`** *section. Only in this form, the switch to 100 MBit/s is valid.*

The modified line must read as follows (corresponds to AML V1.1.1/1.2):

```
/begin TP_BLOB
   0x1000100  /* TP_BLOP version */
   2          /* InterfaceSpeed 1=8MBit, 2=100MBit */
   0x0        /* RstCfg parameter (MPC) */
```

- Save the file.
- In the Component Manager, select **Project →
Read ASAM-2MC** to import the changed file into the database.
- In the HWC editor, select the "Globals" tab for the ETK-BYPASS device (ES1232 / ETK-CTRL-BAS).
- Click the "ASAP2 Project" field to select the newly imported ASAM-MCD-2MC project (cf. "ASAM-2MC Project" on page 160).

The "ETK Connection" line on the "Globals" tab of the ETK-CTRL-BAS item shows the transfer rate of 100 Mbit/s.

## 10.8.5    Globals (ETK-CTRL-ADV Subsystem)

On the "Globals" tab of an ETK-CTRL-ADV subsystem, a physical ETK controller or an ETK port is assigned to the ETK-CTRL subsystem.
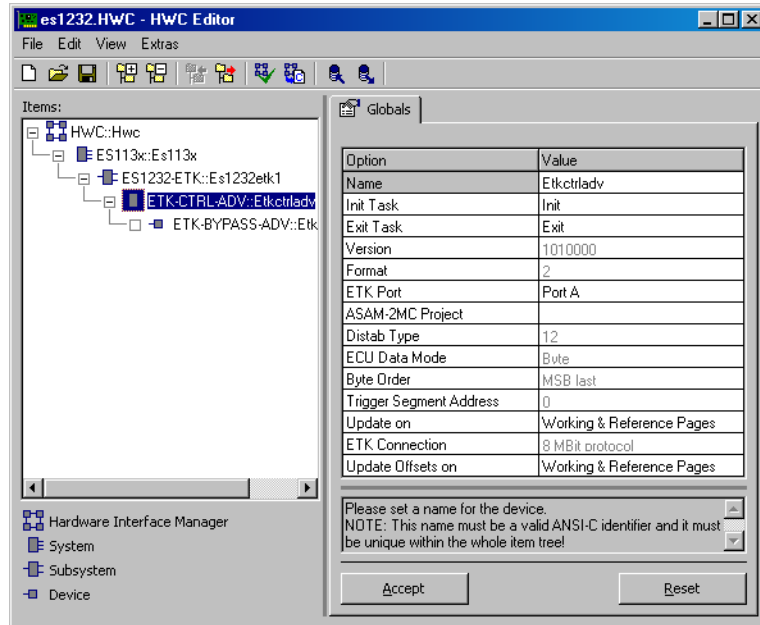


**Fig. 10-33**   The "Globals" Tab of the ETK-CTRL-ADV Subsystem

*ETK Port*

With the ES1201 (cf. chapter 10.3.2), this line is used to select one of two independent ETK channels (Port A, Port B), and thus one of two ETK controllers of the board. With the ES1232, you can only select Port A.

*ASAM-2MC Project*

The ETK bypass requires an ASAM-MCD-2MC project (AML V1.2) which can be generated in the database by reading an ASAM-MCD-2MC description file. The selection window for these projects is shown in Fig. 10-8 on page 161.

> **Note**
>
> *ASAM-MCD-2MC projects using AML V1.1.x **cannot** be read. The following error message appears when such a project is assigned to the device:*
> ```
> Error: Incompatible version 0x<version> of QP_BLOB
> found in SOURCE '<sName>'. Expected version = 0x1.
> Selected ASAM-2MC Project is not suitable for the
> advanced ETK Controller ETK-CTRL-ADV ! Please select
> another one!
> ```

In the following cases, an ASAM-MCD-2MC project *cannot* be read:

- The ASAM-MCD-2MC project uses *AML V1.1.x*. The following error message appears:

  ```
  Error: Incompatible version 0x<version> of
  QP_BLOB found in SOURCE '<sName>'. Expected ver
  sion = 0x1. Selected ASAM-2MC Project is not
  suitable for the advanced ETK Controller ETK-
  CTRL-ADV ! Please select another one!
  ```

- The ASAM-MCD-2MC project uses *DISTAB 16*. If possible, the DISTAB type is converted to 13 during read-in. The following error message appears:

  ```
  WARNING(): (ETK-CTRL-ADV::Etkctrladv "Globals \
  Distab Type ") >> Distab Type (16) in ASAM-2MC
  description not supported. Setting Distab Type
  to 13.
  ```

The `IF_DATA ETK` section of the ASAM-MCD-2MC file contains the bypass description. Older versions sometimes used the section name `IF_DATA ASAP1B_ETK`; this is not supported for ES1232 with ETK-CTRL-ADV.

An ASAM-MCD-2MC project with an `IF_DATA ASAP1B_BYPASS` section of version 1.1 can be read if the `Base Offset Value` in the said section has a value of 0, 2, or 8. In all other cases, the project cannot be read.

```
/begin IF_DATA ASAP1B_Bypass

   /begin TP_BLOB

      0x100 /* Bypass version */

      0x101 /* Blob version */
```

```
      BYPASS_CONFIGURATION

    0  /* BASE_OFFSET_VALUE                    */
        /* value: (0/2) valid for Distab 12    */
        /* value: (0/8) valid for Distab 13    */
        /* Allows to define the Bypass Settings */
        /* used in the bypass tool:            */
      FAR_ADDRESS
        0x384000        /* FAR_ADDRESS_BEGIN   */
                        /* address value       */
        0x004000        /* FAR_ADDRESS_LENGTH  */
                        /* size value          */
      ADDRESS_MAPPING
        Mask16Bit    /* [Mask16Bit, DPP0, DPP1, ↵
                                   DPP2, DPP3]  */
        /* Distab12 only! Ignored for Distab 13 */
  /end TP_BLOB

/end IF_DATA
```

- The content of Base Offset Value is written to the "Base Offset Value" field (ETK-BYPASS-ADV device, "Globals" tab, cf. page 220). Values 0 and 2 are allowed for DISTAB12, values 0 and 8 are allowed for DISTAB13. For foridden values, an error message is issued during a hardware configuration check.

  If the `IF_DATA ASAP1B_BYPASS` section is older than version 1.1, the "Base Offset value" field can be edited manually.

- if the section contains a configuration of the FAR address range (`FAR_ADDRESS_BEGIN` and `FAR_ADDRESS_LENGTH`), the values are written to the "Begin Far Address Range (hex)" and "Length Far Address Range (hex)" fields (ETK-BYPASS-ADV device, "Globals" tab).

  If the section contains no such configuration, the fields can be edited manually.

- if the section contains an "Address Mapping" configuration, and if DISTAB12 is used, the value is written to the "Address Mapping" field (ETK-BYPASS-ADV device, "Globals" tab).

  If the section contains no such configuration, the fields can be edited manually.

  If DISTAB13 is used, the configuration is igored.

A `QP_BLOB` with the channel settings exists for each bypass channel; an example is given in section "Other AML versions" on page 148. The `TP_BLOB` contains general settings.

```
/begin TP_BLOB

  0x1000100    /* TP_BLOP version          */
  2            /* InterfaceSpeed 1=8MBit,  */
               /*                2=100MBit */
  0x0          /* RstCfg parameter (MPC)   */
  /begin DISTAB_CFG  0xC 0x1 MSB_LAST 0x383000 0x0
   TRG_MOD 0xB7
  /end DISTAB_CFG
  CODE_CHK 0x0 0x0 0x0 0x0
  ETK_CFG 0xF 0xF0 0xFF 0x3 0xFD 0xEE 0xFF 0x1
  RESERVED 0x810000 0x8103F9 EXTERN 0x-1 0x-1 0x-1
     0x-1 0x-1

/end TP_BLOB
```

> **Note**
>
> *The parameter in the* `DISTAB_CFG` *section are described on page 142, the parameters in the* `CODE_CHK` *and* `RESERVED` *sections are described on page 160.*

Here - unlike ES1201, ES1231 and ES1232/ETK-CTRL-BAS -, all necessary settings *must* be provided in the ASAM-MCD-2MC file. No facilities are provided to enter them manually.

This also applies to local BLD definitions, which could be entered via the ASAM-MCD-2MC file or via a text editor ("Local BLD Definitions" on page 162) in older versions. With ES1232/ETK-CTRL-ADV, providing the required information in a `MEASUREMENT` section with `KP_BLOB` in the ASAM-MCD-2MC file is mandatory.

```
/begin MEASUREMENT log_uint8_0_A
```

```
          ""
          UBYTE
          ident
          1
          100
          0.0
          1.0
          READ_ONLY
          BIT_MASK 0x8
          ECU_ADDRESS 0xFD00
          /begin IF_DATA ASAP1B_Bypass
           /begin KP_BLOB
             BUFFER_OFFSET
                "log_uint8_0_offset_A.Model_Byp_A"
             /* no SOURCE_ID */
             BIT_OFFSET
                "log_uint8_0_bitOffset_A.Model_Byp_A"
             POSSIBLE_SOURCES 5
           /end KP_BLOB
          /end IF_DATA
      /end MEASUREMENT
```

This MEASUREMENT section defines the same variable as the section on page 164.

*Distab Type*

This line shows which DISTAB procedure is used for exchanging data with the ECU. This information is read from the relevant ASAM-MCD-2MC project (see page 142).

- DISTAB 12 supports signals that are up to two bytes long
- DISTAB 13 supports signals that are 1, 2, 4 and 8 bytes long

*ECU Data Mode*

This line shows which access mode (byte access / word access) the ECU processor uses to access the data memory. It is relevant only if DISTAB12 is selected.

The line cannot be edited; the setting is taken from the ASAM-MCD-2MC project (cf. page 142).

*Byte Order*

This line displays the word data storage format (MSB first / MSB last) of the ECU processor. This information is read from the relevant ASAM-MCD-2MC project (cf. page 142).

*Trigger Segment Address*

This line specifies the trigger segment address of the DISTAB procedure. The field cannot be edited; the value is read from the ASAM-MCD-2MC project (cf. page 142), provided the corresponding parameter was properly defined.

The trigger segment address determines the location of the hardware trigger addresses which are used to control data transfer via the bypass channels. The location of the hardware trigger addresses in relation to the trigger segment address is constant with the DISTAB data exchange procedure.

The start address of a 64-byte trigger segment (with 8-bit or 16-bit wide bus access) must be an even-numbered address that is divisible by 64; correspondingly, the start address of a 128-byte trigger segment (with 32-bit wide bus access) has to be an address that is divisible by 128.

The format of the trigger segment is defined in the DISTAB 12 and DISTAB 13 interface description.

*Update On*

This line allows you to select (for parallel ETKs) whether, during bypass communication, the display table transfer (see "Bypass Communication (AML V1.1)" on page 143) between the simulation processor and the ECU shall take place on the `Working Page`, `Reference Page` or `Working & Reference Pages` of the ETK.

For serial ETKs with all display tables located in the RAM area of the memory, `RAM page` is selected automatically. In this case, the line cannot be edited.

*ETK Connection*

This line shows the transter rate (8 Mbit/s or 100 Mbit/s). The field cannot be edited; the value is read from the ASAM--MCD-2MC project (cf. page 216).

*Update Offsets On*

This line allows you to select (for parallel and serial ETKs) whether, during bypass communication, the bypass offset transfer (see page 147) between the simulation processor and the ECU shall take place on the `Working Page`, `Reference Page`, `Working & Reference Pages` or `RAM page` (serial ETKs only) of the ETK.

For serial ETKs, `RAM page` is selected by default.

10.8.6    Globals (ETK-BYPASS-ADV Subsystem)

This ETK-BYPASS device is used to define all variables necessary for bypass operation with the ES1232/ETK-CTRL-ADV.
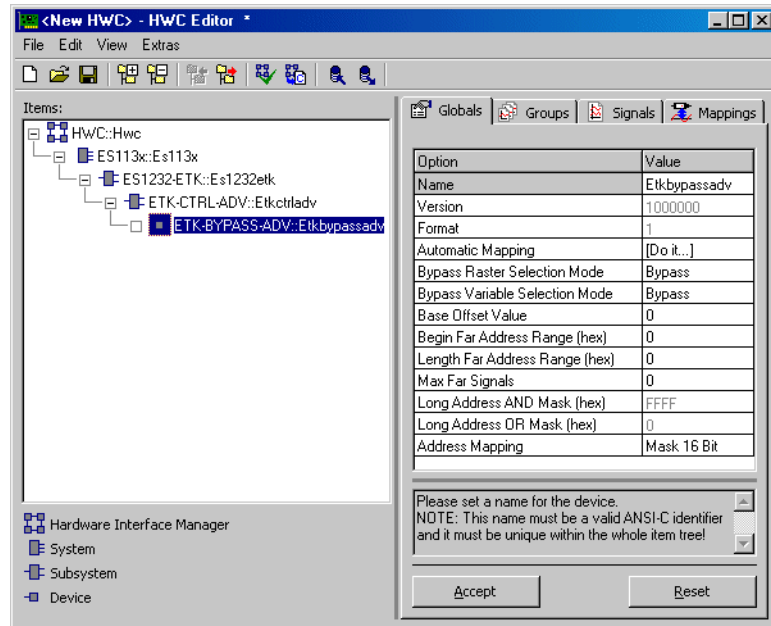


**Fig. 10-34**    The "Globals" Tab of the ETK-BYPASS-ADV Device

*Automatic Mapping*

For more information, refer to the description "Automatic Mapping" on page 182 in chapter 10.4.3.

*Bypass Raster Selection Mode*

This line determines whether bypass and measurement rasters (`Bypass& Measurement`) or only bypass rasters (`Bypass`; default) are shown on the "Groups" tab.

if you select `Bypass&Measurement`, you can measure the signals of the measurement rasters in an ASCET experiment even if only INCA hooks have been defined.

if you change this setting, any signals you selected for bypass rasters will be kept. Signals selected for measurement rasters, however, will be removed when you change the setting from `Bypass&Measurement` to `Bypass`.

Bypass and measurement rasters are treated alike, with one exception: the RTIO driver initializes bypass rasters with master access, whereas measurement rasters have to be initialized with slave access. Since INCA always uses master access for measurement rasters, errors may occur in two cases.

1. An INCA measurement is already running when the ASCET bypass is started. The RTIO driver cannot initialize the measurement rasters due to missing access rights.

2. An INCA measurement is started during an ASCET bypass experiment. Due to higher access rights, INCA "steals" the measurement rasters from ASCET.

In both cases, error messages are displayed.

*Bypass Variable Selection Mode*

Refer to the description "Bypass Variable Selection Mode" on page 165 in chapter 10.3.3.

*Base Offset Value*

This line allows you to move the byte offset of every bypass output value up by 0, 2, 4 or 8 bytes.

The default value for the byte offset with DISTAB12 is 0. With DISTAB13, the default value for the byte offset is 8.

*Begin Far Address Range (hex)*

Refer to the description "Begin Far Address Range (hex)" on page 166 in chapter 10.3.3.

*Length Far Address Range (hex)*

This line is used to specify the length of the FAR address range. This specification is usually only necessary for the DISTAB 12 procedure.

*Max Far Signals*

Refer to the description "Max Far Signals" on page 166 in chapter 10.3.3.

*Long Address AND / OR Mask (hex)*

Refer to the description "Long Address AND / OR Mask (hex)" on page 166 in chapter 10.3.3.

*Address Mapping*

Refer to the description "Address Mapping" on page 166 in chapter 10.3.3.

### 10.8.7 Groups (ETK-BYPASS-ADV Subsystem)

The number of signal groups in the ETK-BYPASS-ADV device is defined in the ASAM-MCD-2MC file. Depending on the setting for the "Bypass Raster Selection Mode" option on the "Globals" tab (chapter 10.8.6), the signal groups for bypass and measurement rasters are created or deleted automatically.

> **Note**
>
> *When the "Bypass Raster Selection Mode" is switched from* `Bypass&Measurement` *to* `Bypass`, *any signals you selected for bypass rasters will be kept. Signals selected for measurement rasters, however, will be removed*

Each bypass raster consists of two signal groups, one send-group and one receive-group. Each measurement raster consists of one receive signal group.

The data of the receive signal group is transferred using the standard DISTAB procedure, and the data of the send signal group is exchanged between the ECU and RTIO using the bypass table procedure.
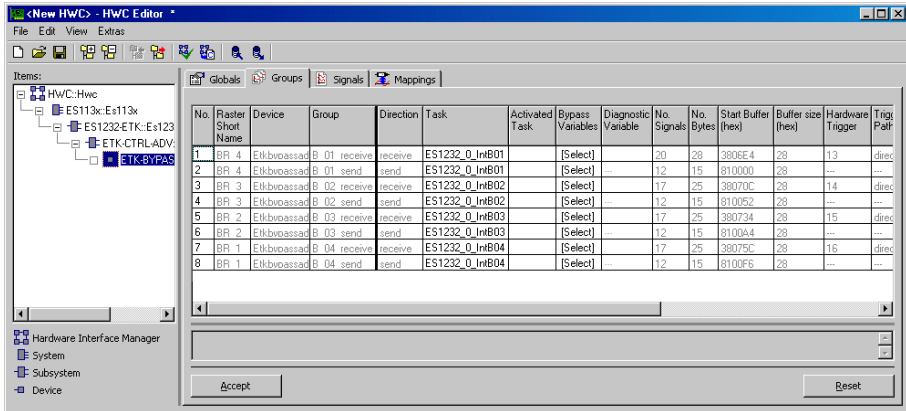


**Fig. 10-35**  The "Groups" Tab of the ETK-BYPASS-ADV Device

*Raster Short Name*

This column displays type (MT for measurement rasters or BT for bypass rasters) and number (priority) of the raster, e.g., MT_12 or BT_31. By default, the signal groups are sorted by descending number (and thus, priority).

The column cannot be edited; the values are read from the ASAM-MCD-2MC project.

*Task*

Refer to the description "Task" on page 168 in chapter 10.3.4.

*Activated Task*

Refer to the description "Activated Task" on page 169 in chapter 10.3.4.

*Bypass Variables*

Refer to the description "Bypass Variables" on page 195 in chapter 10.5.4.

*Diagnostic Variable*

Refer to the description "Diagnostic Variable" on page 196 in chapter 10.5.4.

*No. Signals*

The number of signals selected for the chosen signal group is displayed. The value is updated automatically when you change the signal selection.

*No. Bytes*

The number of bytes required by the signals selected for the chosen signal group is displayed. The value is updated automatically if you change the signal selection.

*Start Buffer (hex)*

The base addresses of the relevant data buffers for the bypass input data and output data are displayed in this column.

The address specifications are read from the ASAM-MCD-2MC project; you cannot edit this column.

*Buffer Size (hex)*

The sizes of the relevant data buffers for the bypass input data and output data are displayed in this column.

The specifications are read from the ASAM-MCD-2MC project; you cannot edit this column.

*Hardware Trigger*

The numbers of the hardware triggers assignd to the receive signal groups are displayed in this column.

The specifications are read from the ASAM-MCD-2MC project; you cannot edit this column.

*Trigger Path*

This column shows the way signal transfer via trigger is performed for receive signal groups:

- `direct` - The raster is directly assigned to a hardware trigger. Once the ECU has written the trigger address, data acquisition can take place directly (without reading a trigger flag).

- `indirect` - Several rasters are assigned to the same hardware trigger. To determine the raster that activated the trigger, a raster-specific "Trigger Flag Address" has to be read.

Direct transfer is possible whenever a sufficient number of hardware triggers are available. This is usually not the case for serial ETKs which must use indirect transfer.

For send signal groups, this column is of no importance; "`---`" is displayed in the respective fields.

*Trigger Flag Address*

This column is used only when the `indirect` "Trigger Path" has been selected. In that case, the raster-specific flag required for correct data acquisition is displayed. The values are read from the ASAM-MCD-2MC project.

In connection with the 8 Mbit/s mode, the trigger flag corresponds to the trigger identifier in other ETK bypass devices (see section "Trigger Id (hex)" on page 169).

If the `direct` "Trigger Path" is selected, "`---`" is displayed.

You cannot edit this column.

*Start Pointer (hex)*

The start addresses of the pointer lists for the DISTAB procedure used is displayed in this column for the "receive" signal groups.

The specifications are read from the ASAM-MCD-2MC project; you cannot edit this column.

### 10.8.8    Signals (ETK-BYPASS-ADV Device)
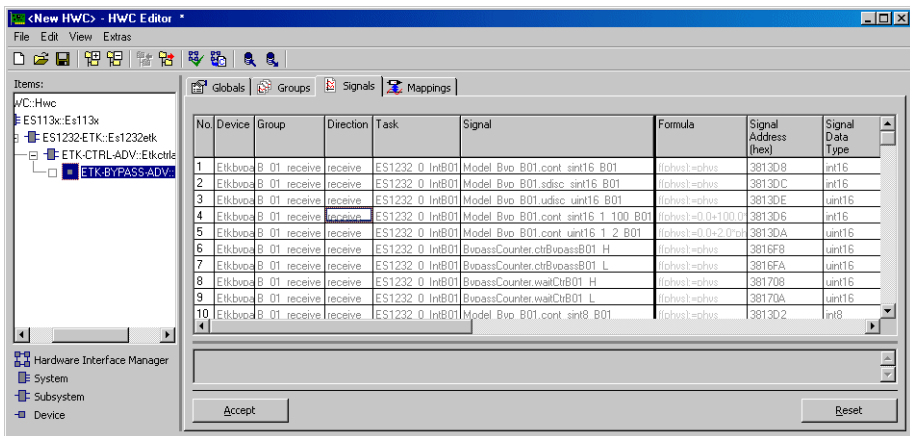


**Fig. 10-36**    The "Signals" Tab of the ETK-BYPASS-ADV Device

Please refer to chapter 10.3.5 on page 170 and chapter 10.7.5 on page 206.

### 10.8.9    Mappings (ETK-BYPASS-ADV Device)

The possible settings are described in section 7.4.5 on page 129.

## 10.9    ES1300-AD

The analog input board ES1300 enables the acquisition of analog signals both differentially (max. 8 channels) and single-ended (max. 16 channels).

### 10.9.1    Globals (ES1300-AD Device)

This section describes the global options of the ES1300-AD device.
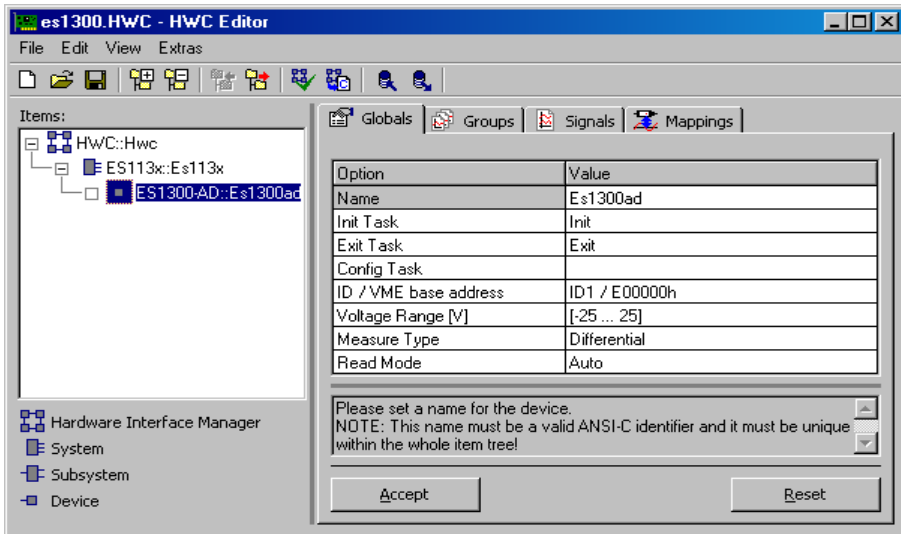


**Fig. 10-37**    The "Globals" Tab of the ES1300-AD Device

*Init Task*

The task for initializing the ES1300 is assigned in this line (Type: Init / Application Mode: active).

*Exit Task*

The task to be executed with the ES1300 when the experiment stops is assigned in this line (Type: Init / Application Mode: inactive).

*ID / VME base address*

This line is responsible for the setting of the VME base address. This setting has to correspond to the jumper settings of the ES1300. Four different VME base addresses can be selected for the ES1300 (ID1/E00000h, ID2/E00100h, ID3/E00200h, ID4/E00300h); this means that up to four ES1300s can be operated in one ES1000 system. The ES1300 occupies an address space of 256 words from the base address.

*Voltage Range [V]*

This is where the input voltage range is specified for each channel of the ES1300. This setting must correspond to the jumper settings of the ES1300. The following voltage ranges can be selected: -25V to 25V, -50V to 50V and 0V to 50V.

*Measure Type*

This is where the measure type is set differentially (max. 8 channels) or single-ended (max. 16 channels). This setting must correspond to the jumper settings of the ES1300.

*Read Mode*

The read mode is specified in this line. With the ES1300, you can select whether the measured A/D values are transferred to ASCET by the hardware driver before or after a new A/D sampling. The following settings are possible:

- auto:
  The driver decides whether to transfer the values from the last grid or values from the current grid based on the time elapsed between two ASCET accesses.

- wait for new values:
  The driver waits for values from the current grid before transfer.

- get old values:
  The driver immediately transfers the values from the last grid.

## 10.9.2    Groups (ES1300-AD Device)

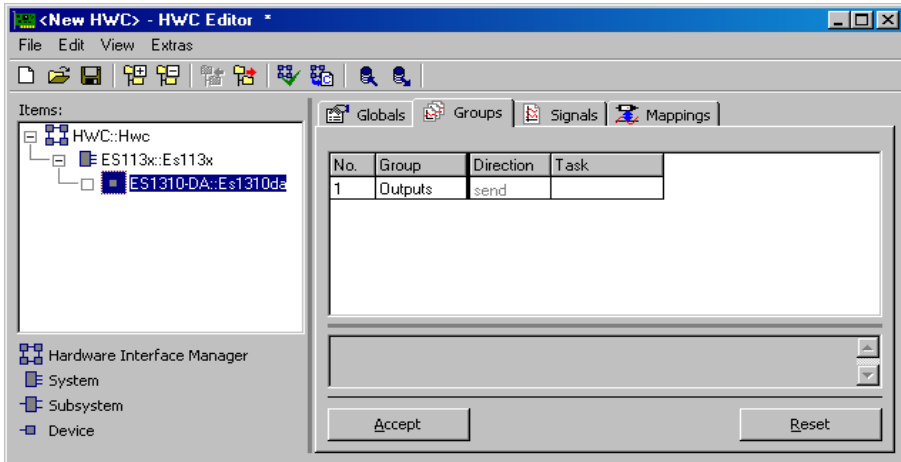This section describes the signal-group-specific options of the ES1300-AD device.
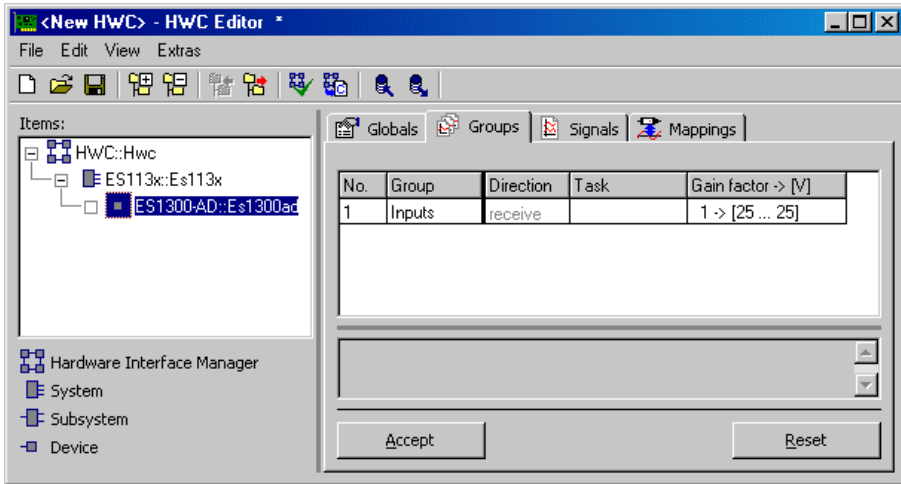


**Fig. 10-38**    The "Groups" Tab of the ES1300-AD Device

*Gain factor -> [V]*

This column is used for setting the programmable gain factor of the ES1300 for the analog input signals. The following gain factors can be selected: 1, 5, 10, 50, 100 or 500. These gain factors cause a higher resolution of the measurement range; thus, only a limited part of the input voltage range is available as effective measurement range.

The resulting effective measurement range is shown in the "Gain factor" column, too. It depends on the input voltage range ("Globals" tab, see page 226) and the gain factor.

### 10.9.3 Signals (ES1300-AD Device)

This section describes the signal-specific options of the ES1300-AD device.
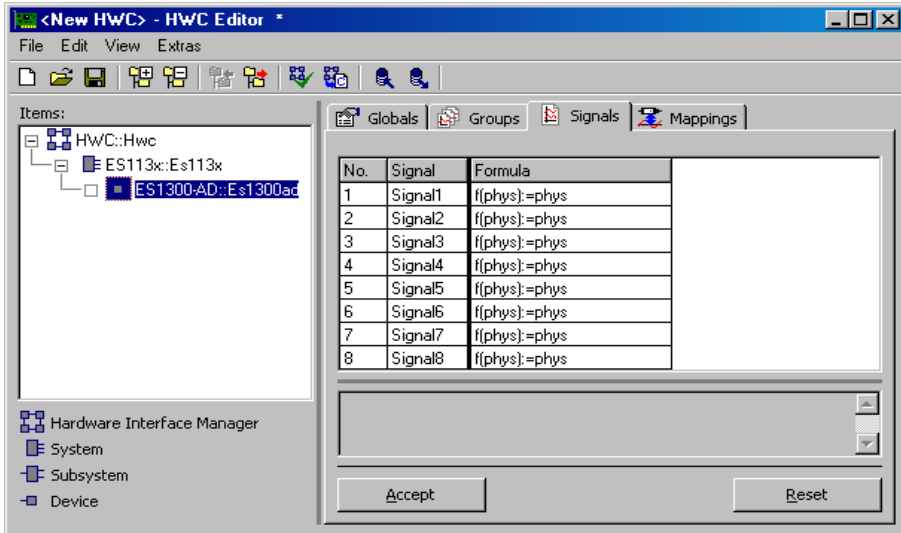


**Fig. 10-39**   The "Signals" Tab of the ES1300-AD Device

*No.*

The maximum number of analog input signals that can be used (8 / 16) is determined for the ES1300 with the `Measure Type` setting (`differen-tial / single-ended`) in the "Globals" tab.

### 10.9.4 Mappings (ES1300-AD Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

### 10.10 ES1301-AD

The analog input board ES1301 enables the acquisition of up to seven analog signals.

### 10.10.1 Globals (ES1301-AD Device)

This section describes the global options of the ES1301-AD device.
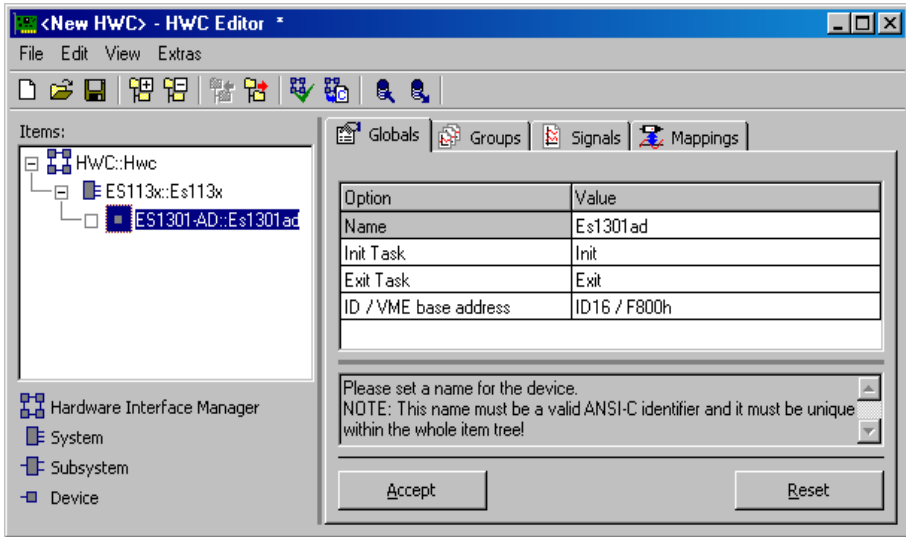


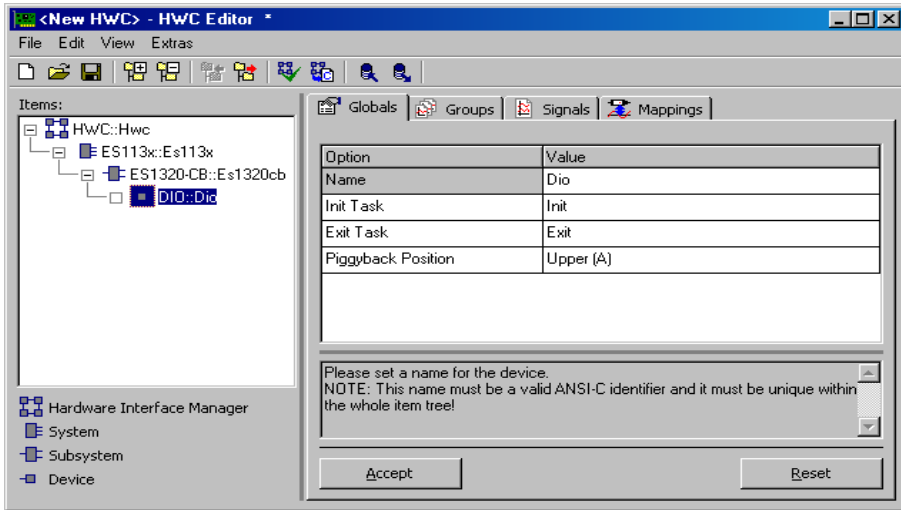**Fig. 10-40**   The "Globals" Tab of the ES1301-AD Device

*Init Task*

The task for initializing the ES1301 is assigned in this line (Type: Init / Application Mode: active).

*Exit Task*

The task to be executed with the ES1301 when the experiment stops is assigned in this line (Type: Init / Application Mode: inactive).

*ID / VME base address*

This line is responsible for the setting of the VME base address. This setting has to correspond to the settings of the ES1301 coded by solder straps. Five different VME base addresses can be selected for the ES1301(`ID16/F800h`, `ID15/F000h`, `ID14/E800h`, `ID12/D800h`, `ID8/B800h`); this means that up to five ES1301s can be operated in one ES1000 system. The ES1301 occupies an address space of 2048 words from the base address.

## 10.10.2  Groups (ES1301-AD Device)

This section describes the signal-group-specific options of the ES1301 device.
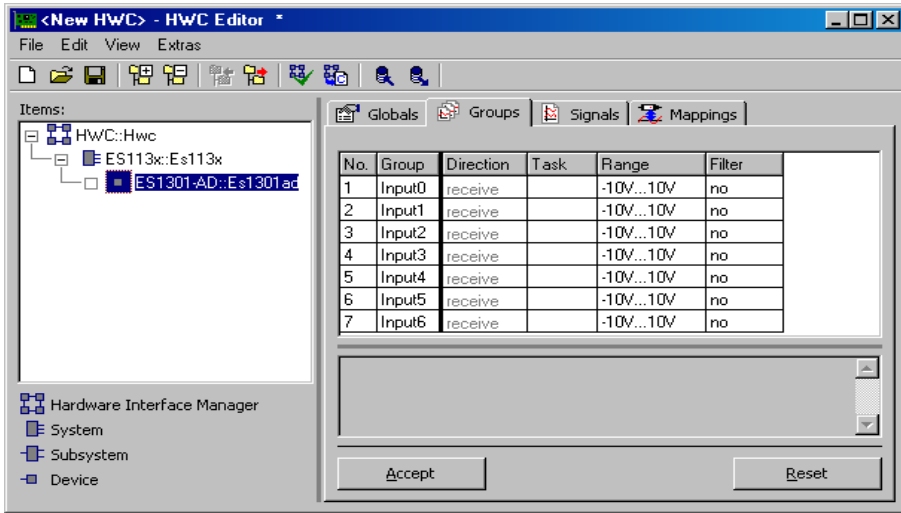


**Fig. 10-41**   The "Groups" Tab of the ES1301-AD Device

*Range*

This is where the input voltage range is specified for each channel of the ES1301. The following input voltage ranges can be selected: -10V to 10V, -5V to 5V, 0V to 10V and 0V to 5V. The configuration for selecting the input voltage range takes place by software; no jumper settings are necessary.

*Filter*

This is where a low-pass filter of $2^{nd}$ order can be set for each channel. The following low-pass filters can be selected: 100 Hz, 200 Hz, 500 Hz, 1 KHz, 2 KHz, 5 KHz and 10 KHz.

### 10.10.3 Signals (ES1301-AD Device)

This section describes the signal-specific options of the ES1301-AD device.
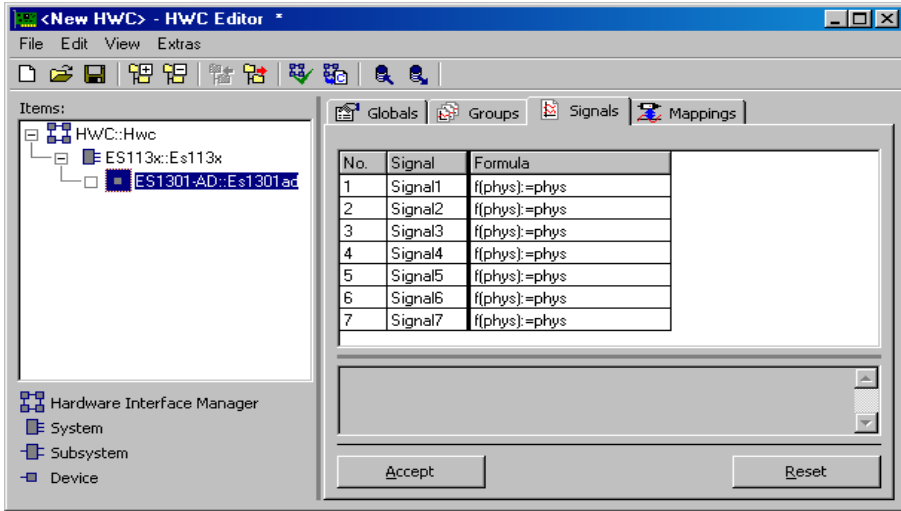


**Fig. 10-42** The "Signals" Tab of the ES1301-AD Device

There are no item-specific columns defined for the ES1301-AD device in the "Signals" tab.

### 10.10.4 Mappings (ES1301-DA Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

## 10.11 ES1303-AD

The analog input board ES1303 is designed to acquire analog signals (max. 16 channels). It was developed for high sampling rates (max. 100 kHz, allows the acquisition of signals of up to 50 kHz) and high resolution. Up to four ES1303 boards can be used simultaneously.

Signal acquisition can be controlled in terms of time (start/stop) due to the two additional trigger inputs. This allows both continuous measuring and the triggered acquisition of individual measured values (single-shot; see ES1303 user's guide) possible.

Each of the 16 analog channels can be used by ASCET-RP or INCA; mixed operation is possible. However, if an ES1303 configuration is running with INCA, ASCET-RP cannot overwrite this configuration. An error message is displayed. The two digital channels can only be used by one of the two programs.

Globals (ES1303-AD Device)

This section describes the global options of the ES1303-AD device.
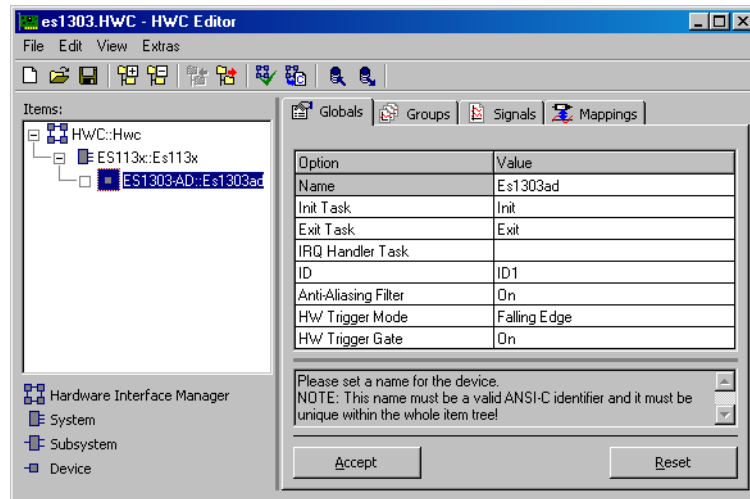


**Fig. 10-43**    The "Globals" Tab of the ES1303-AD Device

*Init Task*

The task for initializing the ES1303 is assigned in this line (Type: Init / Application Mode: active).

*Exit Task*

This line assigns the task which is executed by the ES1303 if the experiment stops (Type: Init / Application Mode: inactive).

*IRQ Handler Task*

The IRQ handler task (Type: Software / Application Mode: active) is required if you want to analyze signal channels in interrupt mode.

This task must be generated as a "software" task in the task list in the ASCET project editor, and at least 2 (max. 50) must be entered in the "Max. No. of Activations" field.

> **Note**
>
> *This task must be available **exclusively** for the element or the respective hardware driver. It must not be used by any other element or user process. Not even two elements of the same type (e.g., two ES1303), may share the same task!*

This line is disabled when the trigger mode Off is selected (see "HW Trigger Mode").

*ID*

The board number of the board to be addressed has to be entered in the "ID" field.

> **Note**
>
> *The ES1303 board is an "Auto-ID" board which is automatically assigned an ID number and a free address space by the Hardware Manager when the system is switched on. Boards of the same type (e.g.two ES1303s) are numbered from left to right, i.e. the left-hand board is assigned the number one, the next one number two etc. There are 4 IDs available; this means that up to 4 boards can be operated in an experimental system.*

*Anti-Aliasing Filter*

This line activates (On) or deactivates (Off) the anti-aliasing filter. The filter is required to limit the bandwidth of the input signal, thus ensuring that the sampling theorem (sampling rate at least twice the bandwidth of the input signal) is not violated.

The filter can only be switched on or off for all channels together. The hardware, on the other hand, allows different settings for four groups of four channels (channels 1–4, channels 5–8, channels 9–12, channels 13–16). Problems can arise when both a and INCA access some channels with different filter settings. If, for example, ASCET accesses channels 1 and 2 without filter, and INCA accesses channels 4 and 5 with filter, a conflict occurs because the channels of the first group (1-4) have to be accessed with the same filter setting. An error message is displayed. The conflict is resolved if INCA accesses channels 5 and 6 instead.

The anti-aliasing filter is optimized for a 100 kHz sampling rate. When the filter is not used, a lowpass filter and a cutoff frequency of 225 kHz are active.

*HW Trigger Mode*

This line is used to set the trigger mode, which determines the start of triggered measurements. The following modes are available:

- Off – the trigger signals are not used.
- RISING EDGE – the measurement is started with the rising edge of the first trigger signal (Mode III in the ES1303 user's guide).
- FALLING EDGE – the measurement is started with the falling edge of the first trigger signal (Mode III in the ES1303 user's guide).

When the "IRQ" column ("Groups" tab) is set to No for a signal group (cf. "IRQ" on page 236), the trigger mode selection is irrelevant.

*HW Trigger Gate*

This line is used to determine whether the second trigger signal is used (ON) or not used (OFF) to generate the trigger.

When the trigger gate is set to ON, the edge of the first trigger signal selected as "HW Trigger Mode" starts the measurement only if, at the same time, the second trigger signal is in high state. If the trigger gate is set to OFF, each edge of the first trigger signal selected as "HW Trigger Mode" starts the measurement. The following table summarizes the conditions for the start of a measurement.

| HW Trigger Mode | HW Trigger Gate | Trigger Signal 1 | Trigger Signal 2 |
|---|---|---|---|
| RISING EDGE | ON | rising edge | high |
| RISING EDGE | OFF | rising edge | any |
| FALLING EDGE | ON | falling edge | high |
| FALLING EDGE | OFF | falling edge | any |

If the "IRQ" column ("Groups" tab) is set to No for a signal group (cf. "IRQ" on page 236), the HW Trigger Gate selection is irrelevant.

## 10.11.2    Groups (ES1303-AD Device)

This section describes the signal-group-specific options of the ES1303-AD device.
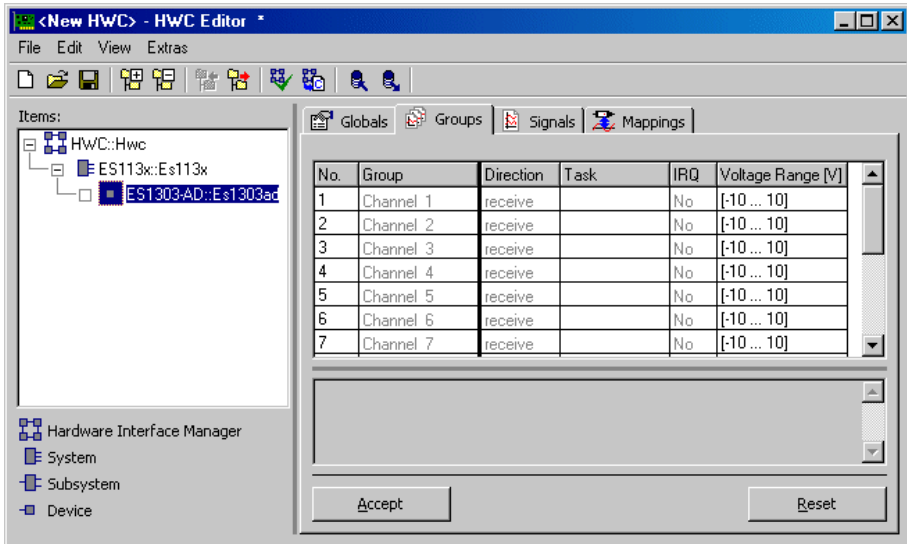


**Fig. 10-44**    The "Groups" Tab of the ES1303-AD Device

The 16 signal groups are predefined. One signal with predefined name belongs to each signal group.

> **Note**
>
> *A future variant of the ES1303 will have only 8 channels. When more than 8 channels were configured in the HWC editor, but a board with only 8 is used, an error message is displayed when the model is started. In that case, the ES1303 is ignored.*

*Task*

This is where the task is specified in which the signal is to be received. You can select several tasks to receive the signal in more than one raster.

If a receive message is to be received in interrupt mode, this setting is reset and locked.

*IRQ*

This option specifies whether the relevant receive signal is to be received in interrupt mode (`Yes`) or not (`No`). It is disabled when the trigger mode `Off` is selected (see "HW Trigger Mode" ).

For "normal" receipt ("IRQ" set to `No`), the sampling or polling frequency in a task must be at least twice the signal frequency, due to the Shannon Theorem (see section 3.1.2 in the ES1303 user's guide). Signals which are not sent in any fixed grid or even only occur sporadically can be problematic in this operating mode. Interrupt reception is ideal for this as it triggers signal processing exactly when the signal was received.

Each signal group can be set up individually. The "HW Trigger Mode"  and "HW Trigger Gate"  settings apply to all signal groups which are received in interrupt mode.

An empty group with "IRQ" set to `No` and and no task selected, generates no process. A warning is displayed in the ASCET monitor window.

*Voltage Range [V]*

The input voltage range for each channel of the ES1303 is specified in this column. The following voltage ranges can be selected: -10 V to 10 V or -60 V to 60 V.

### 10.11.3 Signals (ES1303-AD Device)

This section describes the signal-specific options of the ES1303-AD device.
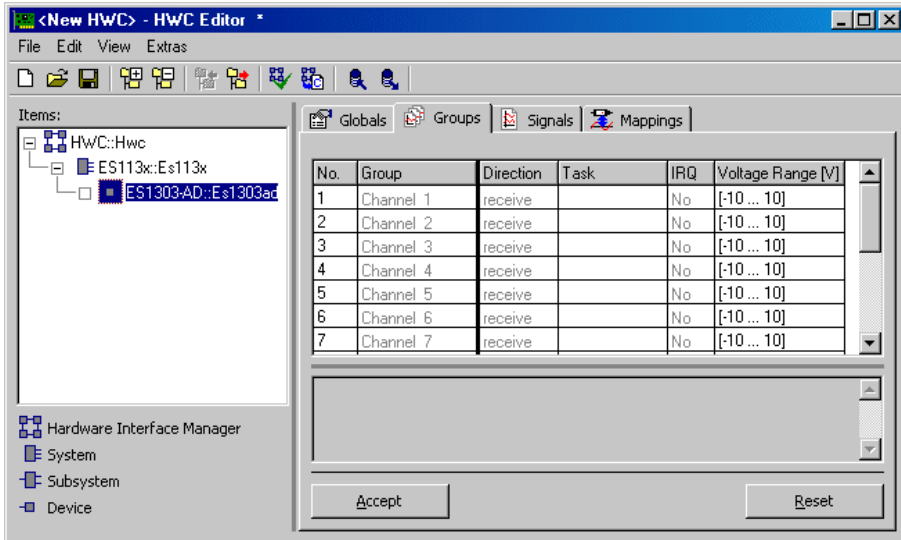


**Fig. 10-45**   The "Signals" Tab of the ES1303-AD Device

Unlike the ES1300 (cf. chapter 10.9.3), you can only select the formula in the "Signals" tab; the signal name is predefined. A description of the various columns is given in chapter 7.4.4.

### 10.11.4 Mappings (ES1303-AD Device)

The possible settings are the same for all items. They are described in chapter 7.4.5 on page 129.

## 10.12 ES1310-DA

The analog output board ES1310 enables the output of eight analog signals.

## 10.12.1    Globals (ES1310-DA Device)

This section describes the global options of the ES1310-DA device.
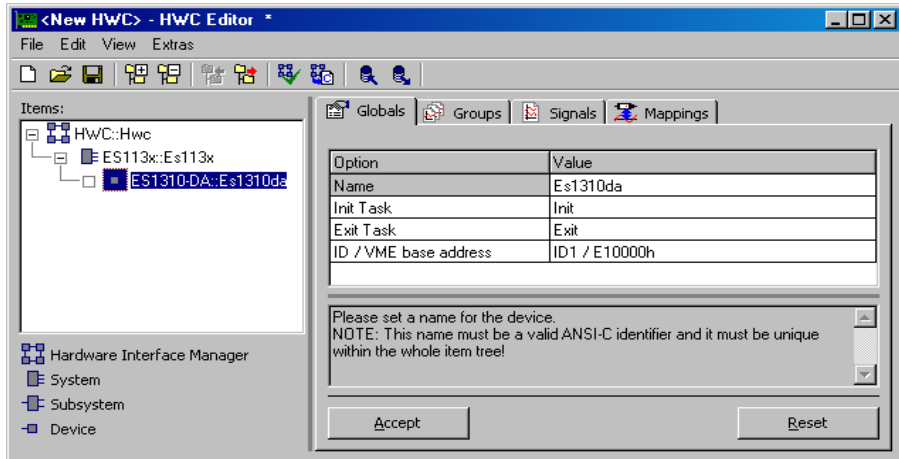


**Fig. 10-46**    The "Globals" Tab of the ES1310-DA Device

*Init Task*

The task for initializing the ES1310 is assigned in this line (Type: Init / Application Mode: active).

*Exit Task*

The task to be executed with the ES1310 when the experiment stops is assigned in this line (Type: Init / Application Mode: inactive).

*ID / VME base address*

This line is responsible for the setting of the VME base address. This setting has to correspond to the jumper settings of the ES1310. Two different VME base addresses can be selected for the ES1310 (`ID1/E10000h`, `ID2/E20000h`); this means that up to two ES1310s can be operated in one ES1000 system. The ES1310 occupies an address space of 1024 words from the base address.

## 10.12.2    Groups (ES1310-DA Device)

This section describes the signal-group-specific options of the ES1310-DA device.



**Fig. 10-47**    The "Groups" Tab of the ES1310-DA Device

There are no item-specific columns defined for the ES1310-DA device in the "Groups" tab.

### 10.12.3 Signals (ES1310-DA Device)

This section describes the signal-specific options of the ES1310-DA device.
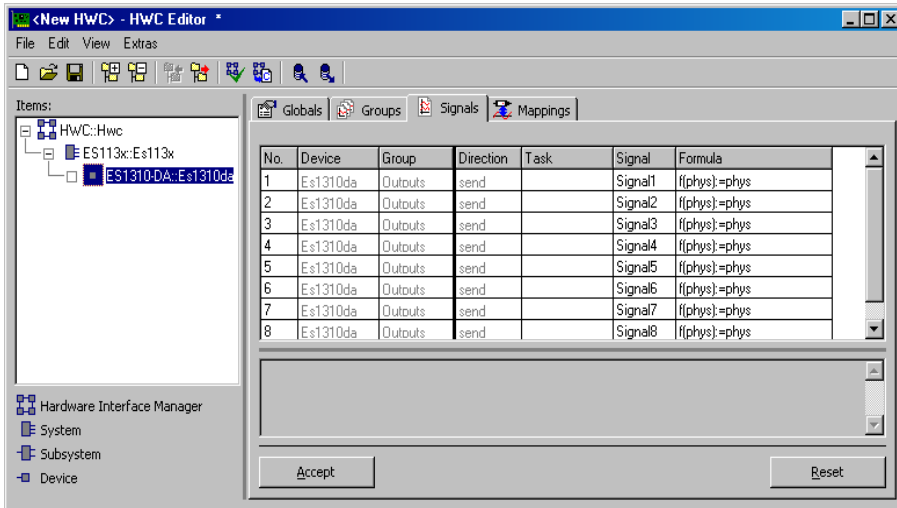


**Fig. 10-48** The "Signals" Tab of the ES1310-DA Device

There are no item-specific columns defined for the ES1310-DA device in the "Signals" tab.

### 10.12.4 Mappings (ES1310-DA Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

### 10.13 ES1320-CB (DIO)

The digital input/output board ES1320 (DIO) makes it possible to measure and output twenty digital signals. The ES1320 hardware consists of a carrier board (CB) with two DIO piggyback modules.
In the HWC Editor items list, the carrier board is added as a subsystem and the DIO piggyback as a device.

### 10.13.1 Globals (ES1320-CB Subsystem)

This section describes the global options of the ES1320-CB subsystem.
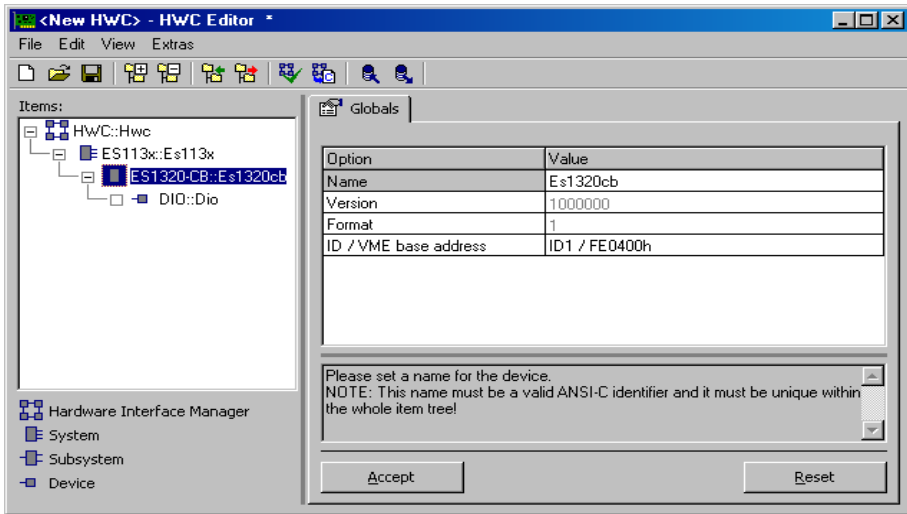


**Fig. 10-49**  The "Globals" Tab of the ES1320-CB Subsystem

*ID / VME base address*

This line is responsible for the setting of the VME base address. This setting has to correspond to the jumper settings of the ES1320. Eight different VME base addresses can be selected for the ES1320 (`ID1/FE0400h`, `ID2/FE0C00h`, `ID3/FE1400h`, `ID4/FE1C00h`, `ID5/FE2400h`, `ID6/FE2C00h`, `ID7/FE3400h`, `ID8/FE3C00h`, `ID9/FE4400h`, `ID10/FE4C00h`, `ID11/FE5400h`, `ID12/FE5C00h`, `ID13/FE6400h`, `ID14/FE6C00h`, `ID15/FE7400h`, `ID16/FE7C00h`); this means that up to eight ES1320s can be operated in one ES1000 system. The ES1320 occupies an address space of 256 words from the base address.

### 10.13.2 Globals (DIO Device)

This section describes the global options of the DIO device.



**Fig. 10-50**   The "Globals" Tab of the DIO Device

*Init Task*

The task for initializing the ES1320 is assigned in this line (Type: Init / Application Mode: active).

*Exit Task*

The task to be executed with the ES1320 when the experiment stops is assigned in this line (Type: Init / Application Mode: inactive).

*Piggyback Position*

This line assigns either the upper (A) or lower (B) DIO piggyback module as a digital interface.

### 10.13.3 Groups (DIO Device)

This section describes the signal-group-specific options of the DIO device.
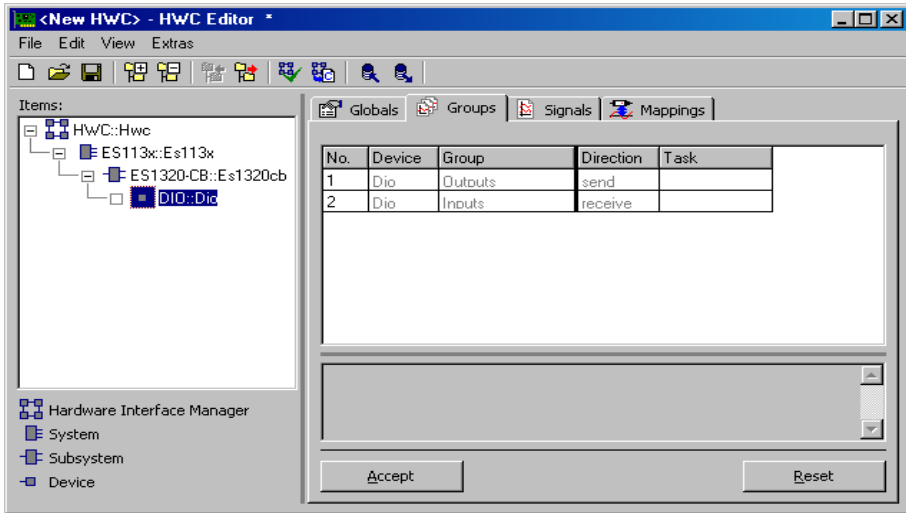


**Fig. 10-51**   The "Groups" Tab of the DIO Device

There are no item-specific columns defined for the DIO device in the "Groups" tab.

### 10.13.4 Signals (DIO Device)

This section describes the signal-specific options of the DIO device.
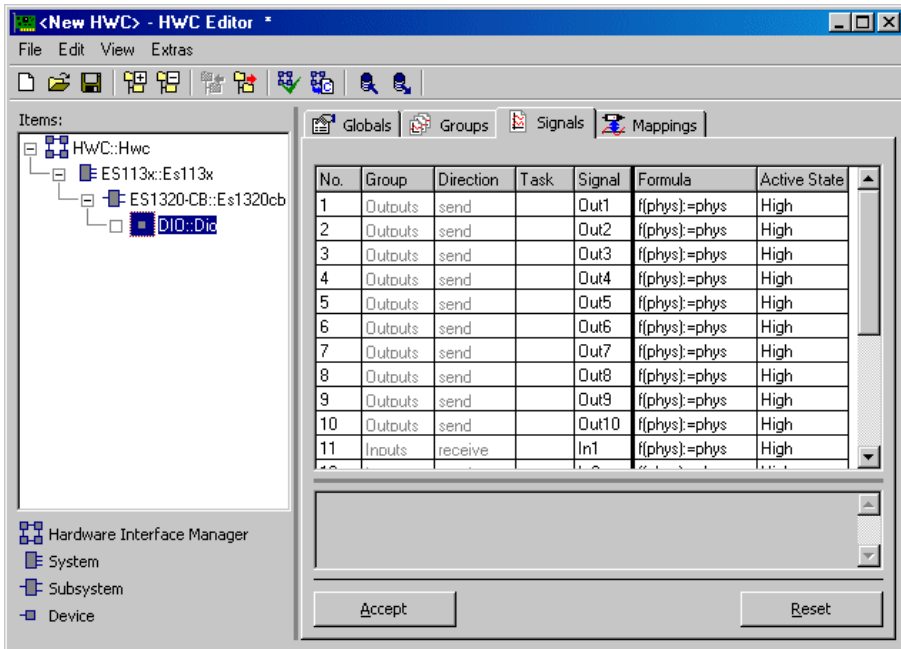


**Fig. 10-52**  The "Signals" Tab of the DIO Device

*Active State*

You can use this column to define whether the relevant digital input or output signal is inverted (`Active Low`) or not (`Active High`) by the driver routine.

### 10.13.5 Mappings (DIO Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

## 10.14 ES1325-DIO

The digital input/output board ES1325 (DIO) can acquire and output digital signals on sixteen channels each. Two trigger inputs make it possible to control signal acquisition and signal output. Up to four ES1325 Boards can be operated simultaneously with an ES1130 or an ES1135. For each ES1325 Board, you can use a maximum of one Input Device, one Output Device and one LED Device.

This section describes the global options of the ES1325-DIO Subsystem.
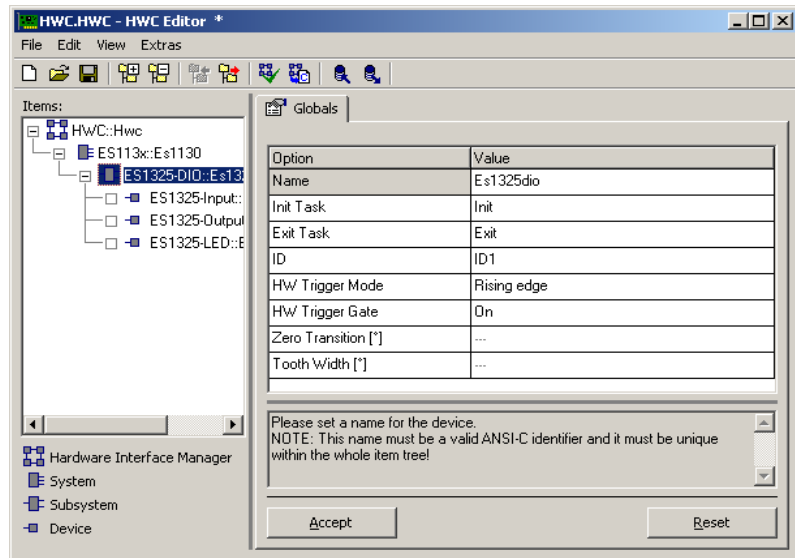


**Fig. 10-53**    The "Globals" Tab of the ES1325-DIO Subsystem

*ID*

This is where the board is assigned an ID for the board number.

---

**Note**

*The ES1325 board is what is referred to as an "Auto-ID" board which automatically receives an ID number and a free address space from the Hardware Manager when the system is powered on. Boards of the same type (e.g. two ES1325s) are numbered from left to right, i.e. the board on the left is assigned the number one, the next one the number two etc.. 4 IDs are available; this means that up to 4 boards can be operated in one experimental system.*

---

*HW Trigger Mode*

This is where the trigger mode is set which determines the start of triggered operations. This option is only available if the "HW Trigger Gate" parameter is set to "**On**" (see "HW Trigger Gate" on page 247).

The following modes are available:

- **Off**

  Signals at the trigger inputs are not evaluated.

- **Rising edge**

  Once it has been evaluated, a rising edge at trigger input 1 starts the acquisition or generation of signals at the digital input and/or digital output channels.
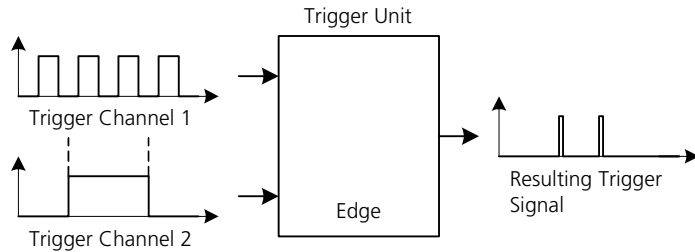


- **Signal state**

  A rising (falling) edge at trigger input 1 determines the start (end) of a period. An operation is run on the digital input channels during this period at trigger input 1.
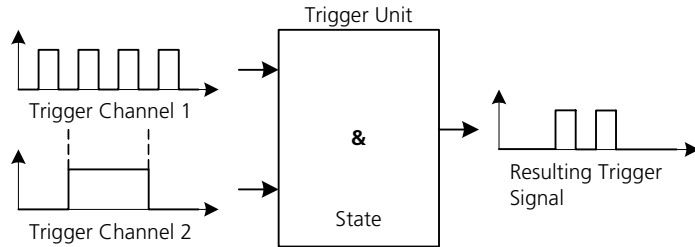


  In "Signal state" trigger mode, an optional request delay can be used (see "Signals" on page 253). This time has to be shorter than the hold time of the resulting trigger signal as data acquisition is otherwise not started.

- **Angle based**

  Trigger input 1 and trigger input 2 both evaluate continuous PWM signals.

  The following applies to crankshaft signals:

  – Trigger input 1 = zero transition,

– Trigger input 2 = tooth width.

A single PWM period at trigger input 2 corresponds to an angle segment. The angle position is determined by counting the PWM periods.
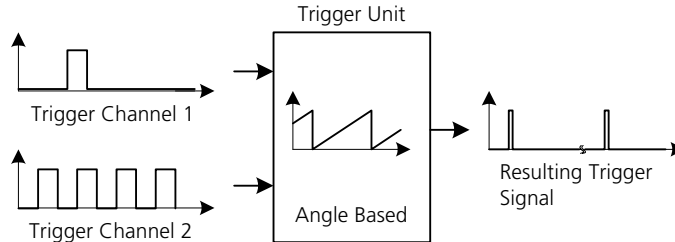


In this operating mode, a counter is checked on the trigger unit with the two trigger inputs.

A rising edge at trigger channel 1 resets the counter to zero and a falling edge at trigger channel 2 increases the counter by 1. This is how, for example, a crankshaft angle can be acquired if trigger channel 2 receives a falling edge with constant angle segments (e.g. 6° tooth width) and trigger channel 1 receives a rising edge with every full revolution (in zero angle transition). To guarantee correct operation, the signal at trigger channel 2 has to show continuous and equidistant angle segments. The signal at trigger channel 1 has to show the zero transition at the rising edge. The falling edge of this signal has to follow after a certain hold time. This is between the falling and the next rising edge of the signal at trigger channel 2 (angle segment signal).

With the trigger modes **Rising edge** and **Signal state**, trigger channel 2 can also be defined as a trigger gate (see "HW Trigger Gate" on page 247).

*HW Trigger Gate*

This is where the mode which determines the evaluation of suitable edges at the trigger inputs is set. The parameter is set if trigger channel 2 is used to release trigger generation (TTL high level corresponds to "trigger enabled").

The following modes are available:

- **Off**

  Every edge at trigger input 1 selected under "HW Trigger Mode" triggers an operation at the digital input and/or output channels.

- **On**

  Edges at trigger input 1 only trigger an operation at the digital input and/or output channels during a high level at trigger input 2.

When the "**Angle based**" trigger mode is selected (see "HW Trigger Mode" on page 245), the "HW Trigger Gate" parameter is automatically set to "**On**". When the "**Off**" trigger mode is selected (see "HW Trigger Mode" on page 245), the "HW Trigger Gate" parameter cannot be edited.

*Zero Transition [*]*

This is where the zero transition at trigger input 1 is defined.

| Resolution | | Tooth width |
|---|---|---|
| Value range | Min. | 2 x tooth width |
| | Max. | 32767 x tooth width |
| Typical value for a crankshaft signal | | 360° or 720° |

This option is only available in "**Angle based**" trigger mode (see "HW Trigger Mode" on page 245). It can be edited within the value range.

*Tooth Width [*]*

This is where the tooth width at trigger input 2 during a PWM period is defined.

| Resolution | | 0.5° |
|---|---|---|
| Value range | Min. | 0.5° |
| | Max. | 720° |
| Typical value for a crankshaft signal | | 6° |

This option is only available in "**Angle based**" trigger mode (see "HW Trigger Mode" on page 245). It can be edited within the value range.

## 10.14.2 Globals (ES1325-Input Device)

This section describes the global options of the Input Device.
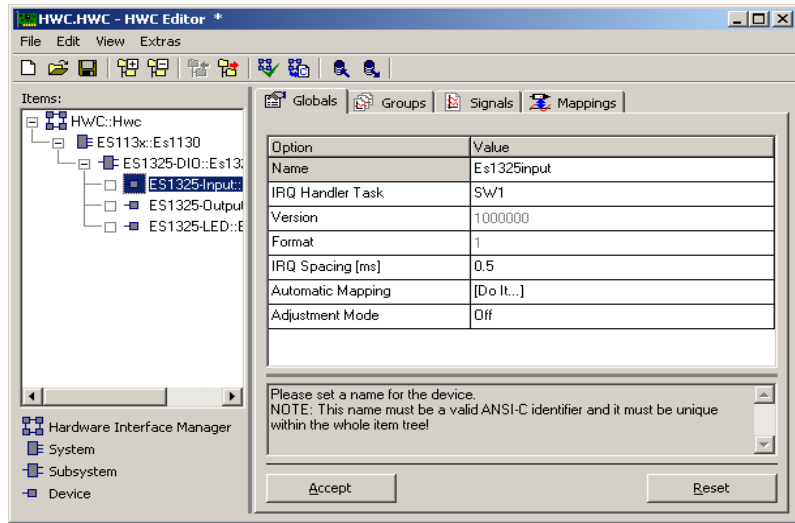


**Fig. 10-54**  The "Globals" Tab of the ES1325-DIO Input Device

*IRQ Handler Task*

The IRQ Handler task (type: Software / Application mode: active) is required if signal channels are to be evaluated in Interrupt mode.

This task must be generated as a "software" task in the task list in the ASCET project editor, and at least 2 (max. 50) must be entered in the "Max. No. of Activations" field.

---
**Note**

*This task has to be of the type "Software" and be **exclusively** available to the item or the relevant hardware driver and cannot be used by any other item or user processes. Two items of the same type (e.g. two ES1325s) can-**not** share the same task!*

---

This option is locked if the `Off` trigger mode is selected (see "HW Trigger Mode" on page 245).

This is where the minimum time between two interrupt requests generated by the ES1325 is defined.

| Resolution | | 0.1 ms |
|---|---|---|
| Value range | Min. | 0 ms |
| | Max. | 25.5 ms |

This option is not available in "`Off`" trigger mode (see "HW Trigger Mode" on page 245). It can be edited within the value range.

*Automatic Mapping*

This option makes the automatic mapping of signals and (ASCET) messages possible.

For more details, consult the section "Automatic Mapping" on page 182.

*Adjustment Mode*

Adjustment Mode makes it possible to influence signal acquisition online if necessary, i.e. during the runtime of the ASCET model.

The following modes are available:

- **`Off`**

  Adjustment Mode is deactivated. A "ChXXMsr" measure group is displayed for every "ChXX" input channel.

- **`On`**

  Adjustment Mode is activated. In addition to the "ChXXMsr" measure group, an online adjustment group "ChXXAdj" is displayed for every "ChXX" input channel. This "ChXXAdj" group also contains configuration parameters that can be adjusted online in the "Signals" column in the "Groups" tab.

Adjustment Mode can only be selected if the HW Trigger Mode is set to "`On`" (see "HW Trigger Mode" on page 245).

This section describes the signal-group-specific options of the Input Device.
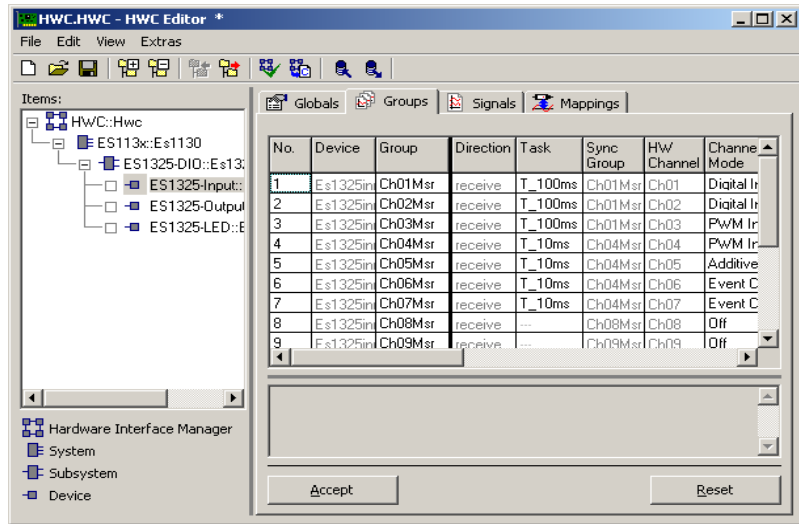


**Fig. 10-55**    The "Groups" Tab of the ES1325-DIO Input Device

*Sync Group*

Signal groups which are transferred synchronously in the same task (or group of tasks) are combined to form one process. This process is given the same name as the group name shown in this column.

If several tasks are selected, only those processes of the signal groups which have an identical entry in the "Task" field can be combined to form one common process.

The signal groups within this process are transferred synchronously.

*Hardware Channel*

Input channels of the ES1325 (Ch01 to Ch16)

*Channel Mode*

This column determines the type of digital input signal to be evaluated for each input channel of the ES1325.1:

- Off

    No evaluation of the digital input signal.

- Digital Input

  Evaluation of the state of the digital input signal at the time of the request.

- PWM Input

  Evaluation of all relevant information of the last complete PWM signal before the request.

- Additive Time

  Addition of the duration of the active phases of a signal during a defined time interval.

- Event Counter

  Addition of relevant signal edges during a defined time interval.

The definition of whether rising, falling or both edges are to be evaluated takes place in the "Significant Edge" parameter.

Data transfer in the channels is only activated either when a task is selected or the "IRQ" parameter is set to "**Yes**" (see "IRQ" on page 252), if in fact it is available.

*Use HW Trigger*

This is where it is determined whether the digital input signal is to be acquired synchronously with an external trigger signal. The following modes are available:

- **Yes**

  Digital signal acquisition is synchronized with an external trigger.

- **No**

  The digital signals are acquired in Polling mode. The digital input signal is not synchronized with an external trigger signal.

The "Trigger Mode" parameter is determined for all hardware channels together in "Globals" for the ES1325-DIO Subsystem.

*IRQ*

This is where it is determined whether an interrupt is generated after signal detection.

- **Yes**

  An IRQ Handler Task (see "IRQ Handler Task" on page 249) is activated. After data acquisition, an interrupt request is generated to transfer data to the ASCET model.

- **No**

    The acquired signals are available for processing in a buffer memory. The data transfer takes place triggered by the ASCET model in Polling mode in the task determined in the "Task".

The "IRQ" option is only active if the "Use HW Trigger" parameter is set to "**Yes**" (see "Use HW Trigger" on page 252).

*Activated Task*

Clicking the "Activated Task" field opens the "Task Selection" dropdown list. The task activated when a signal group is received is selected and assigned there according to the channel.
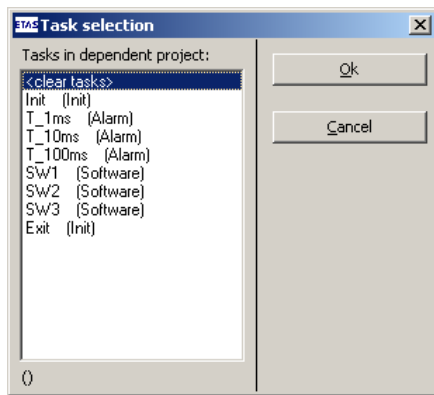


This option is only available if the "IRQ" parameter is set to "**Yes**" (see "IRQ" on page 252).

*Signals*

Clicking the [Select] button in the "Signals" field opens the "Signal Selection" dropdown list.
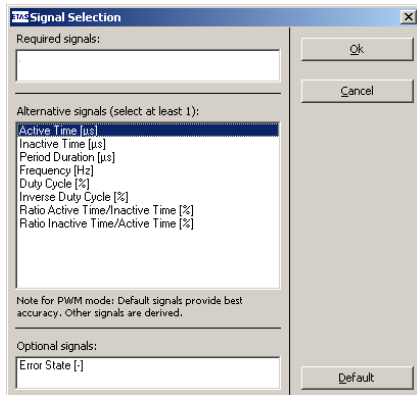
Signal components are assigned to every signal.

The following signals are available for the "ChXXMsr" measure groups:

- State
- Active Time [μs]
- Inactive Time [μs]
- Period Duration [μs]
- Frequency [Hz]
- Duty Cycle [%]
- Ratio Active Time/ Inactive Time [%]
- Ratio Inactive Time/ Active Time [%]
- Additive Active Time
- Counter Value

The following signals are available for the online "ChXXAdj" adjustment groups (see "Adjustment Mode" in the "Globals" tab, page 250):

- Time Delay [μs]
- Angle Delay [μs]
- Angle Interval [μs]

Depending on the parameters "Channel Mode", "Use HW Trigger" and "HW Trigger Mode" some signals are obligatory or default values are selected for the signal group.

*Active State*

The function of the active state is assigned to a level of the input signal. The following modes are available:

- High

    The high level of the input signal is assigned the "active" state for further processing.

- Low

    The low level of the input signal is assigned the "active" state for further processing.

*Significant Edge*

This is where an event (Channel mode "`Event Counter`") or the start of a PWM period (Channel mode "`PWM Input`") is assigned to an edge. The following modes are available:

- Inactive-Active

    The transition of the input signal from inactive to active (see "Active State" on page 255) triggers the event.

- Active-Inactive

    The transition of the input signal from active to inactive (see "Active State" on page 255) triggers the event.

- Both

    Transitions of the input signal from inactive to active and from active to inactive (see "Active State" on page 255) trigger the event.

This setting is not available for the Channel modes "`Digital Input`" and "`Additive Time`" (see "Channel Mode" on page 251).

*Hysteresis*

A type of hysteresis must be assigned to every digital input channel. Available options:

- TTL
- User defined

|  | TTL | User defined |
|---|---|---|
| Lower threshold value | 1.728 V | User-specific |
| Upper threshold value | 2.304 V | User-specific |

When the "**User defined**" option is selected, the lower and upper threshold value of the hysteresis have to be defined in accordance with the definition (see "Low Thresh. [V]" on page 256 and "High Thresh. [V]" on page 256).

*Low Thresh. [V]*

In this cell, the lower threshold value for the input signal is assigned to every digital input channel. The lower threshold value has to be defined at least one step under the upper threshold value.

| Resolution | | 0.144 V |
|---|---|---|
| Value range | Min. | 0 V |
| | Max. | Upper threshold value - 0.144 V |

This option is locked if the Hysteresis mode "**TTL**" is selected (see "Hysteresis" on page 255).

*High Thresh. [V]*

In this cell, the upper threshold value for the input signal is assigned to every digital input channel. The upper threshold value has to be defined at least one step above the lower threshold value.

| Resolution | | 0.144 V |
|---|---|---|
| Value range | Min. | Lower threshold value + 0.144 V |
| | Max. | 36 V |

This option is locked if the Hysteresis mode "**TTL**" is selected (see "Hysteresis" on page 255).

*Timeout [ms]*

In this cell, a period of time is assigned to every digital input channel in which the input signal has to change state at least once. If the signal does not change state within this defined period of time, the relevant error state is generated.

| Resolution | | 2.5 ms |
|---|---|---|
| Value range | Min. | 0 ms (no timeout assigned) |
| | Max. | 163837.5 ms |

### 10.14.4 Signals (ES1325-Input Device)

This section describes the signal-specific options of the Input Device.
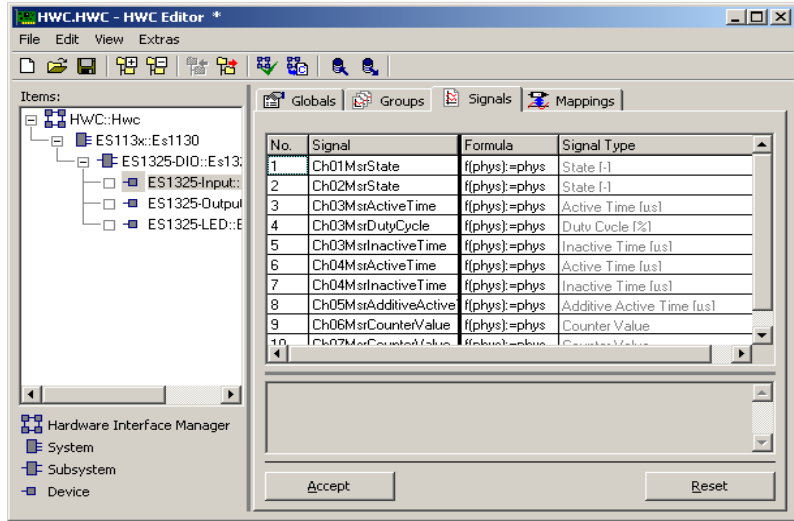


**Fig. 10-56**   The "Signals" Tab of the ES1325-DIO Input Device

*HW Channel*

Display of the input channels of the ES1325.1 (Ch01 to Ch16).

*Signal Type*

This is where the signal type is displayed which is assigned to this signal. This assignment can be modified in the "Signals" parameter of the "Groups" tab.

### 10.14.5 Mappings (ES1325-Input Device)

The possible settings are described in section 7.4.5 on page 129.

## 10.14.6 Globals (ES1325-Output Device)

This section describes the global options of the Output Device.
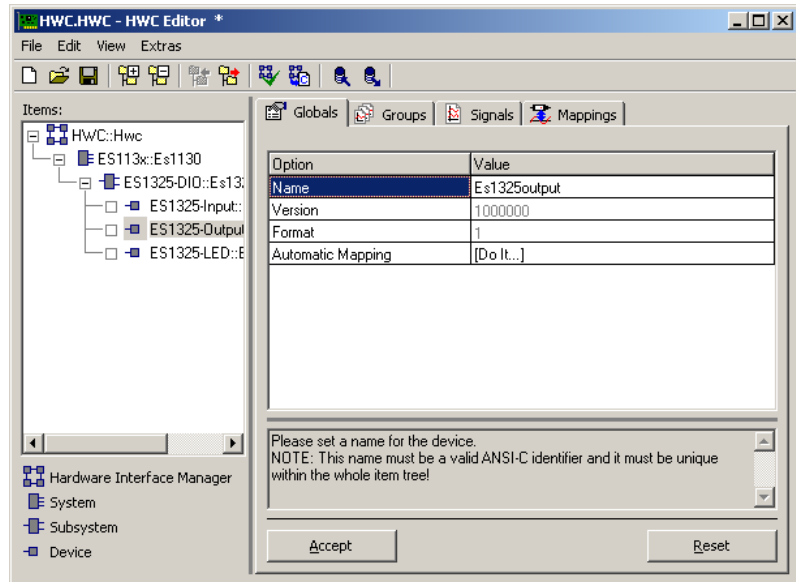


**Fig. 10-57**   The "Globals" Tab of the ES1325-DIO Output Device

*Automatic Mapping*

This option makes the automatic mapping of signals and (ASCET) messages possible.

For more details, consult the section "Automatic Mapping" on page 182.

This section describes the signal-group-specific options of the Output Device.
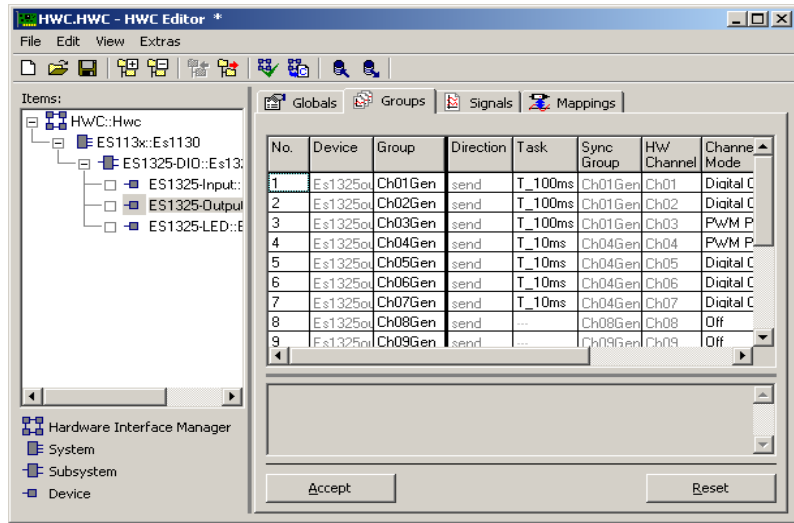


**Fig. 10-58**    The "Groups" Tab of the ES1325-DIO Output Device

*Sync Group*

Signal groups which are transferred synchronously in the same task (or group of tasks) are combined to form one process. This process is given the same name as the group name shown in this column.

If several tasks are selected, only those processes of the signal groups which have an identical entry in the "Task" field can be combined to form one common process.

The signal groups within this process are transferred synchronously.

*HW Channel*

Output channels of the ES1325.1 (Ch01 to Ch16)

*Channel Mode*

This column determines the type of digital output signal to be evaluated for each output channel of the ES1325.1:

- Off

   No evaluation of the digital output signal.

- Digital Output

**259**

The state of an output channel can be set to active or inactive with Request. The request can be triggered by the simulation processor.

- PWM Periodic Output

  PWM signals are generated with no limitation of the output duration.

  Exactly two signal components are required to describe the generated PWM signals (see "Signals" on page 260).

- PWM Interval Output

  PWM functionality is used to generate single pulses or intervals of PWM signals. A further request which is received while the first request is being processed is rejected.

Data transfer in the channels is only activated either when a task is selected or the "IRQ" parameter is set to "**Yes**" (see "IRQ" on page 252), if in fact it is available.

*Use HW Trigger*

This is where it is determined whether the digital output signal is to be output synchronously with an external trigger signal. The following modes are available:

- **Yes**

  Digital signal output is synchronized with an external trigger.

- **No**

  The digital signals are output in Polling mode. The digital output signal is not synchronized with an external trigger signal.

The "Trigger Mode" parameter is determined for all hardware channels together in "Globals" for the ES1325-DIO Subsystem.

*Signals*

Clicking the [Select] button in the "Signals" field opens the "Signal Selection" dropdown list. The signals to be generated can be selected.
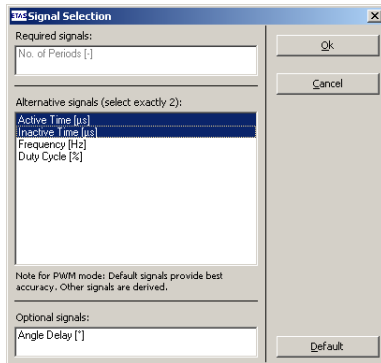
Signal components are assigned to every signal.

- State [-]
- Active Time [μs]
- Inactive Time [μs]
- Frequency [Hz]
- Duty Cycle [%]
- Time Delay [μs]
- No. of Periods
- Angle Delay [°]

Depending on the parameters "Channel Mode", "Use HW Trigger" and "HW Trigger Mode" some signals are obligatory or default values are selected for the signal group.

*Active State*

The function of the active state is assigned to a level of the output signal. The following modes are available:

- High

  The high level of the output signal is assigned the "active" state for further processing.

- Low

  The low level of the output signal is assigned the "active" state for further processing.

### 10.14.8    Signals (ES1325-Output Device)

This section describes the signal-specific options of the Output Device.
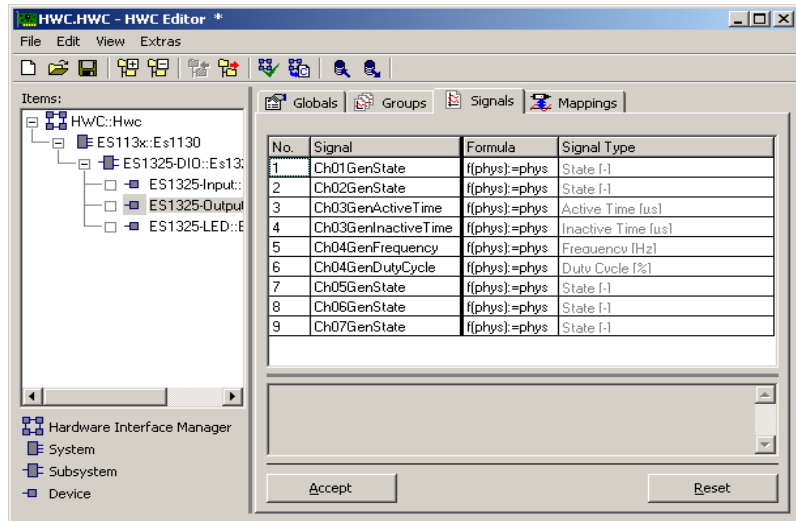


**Fig. 10-59**    The "Signals" Tab of the ES1325-DIO Output Device

*HW Channel*

Output channels of the ES1325.1 (Ch01 to Ch16)

*Signal Type*

This is where the signal type assigned to this signal is displayed. This assignment can be modified in the "Signals" parameter of the "Groups" tab.

### 10.14.9    Mappings (ES1325-Output Device)

The possible settings are described in section 7.4.5 on page 129.

Globals (ES1325-LED Device)

This section describes the global options of the LED Device.
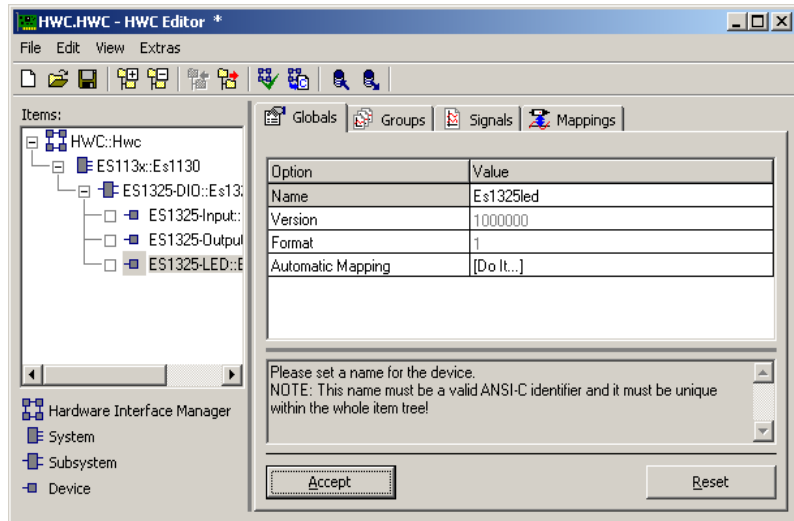


**Fig. 10-60** The "Globals" Tab of the ES1325-DIO LED Device

*Automatic Mapping*

This option makes the automatic mapping of signals and (ASCET) messages possible.

For more details, consult the section "Automatic Mapping" on page 182.

## 10.14.11  Groups (ES1325-LED Device)

This section describes the signal-group-specific options of the LED Device.
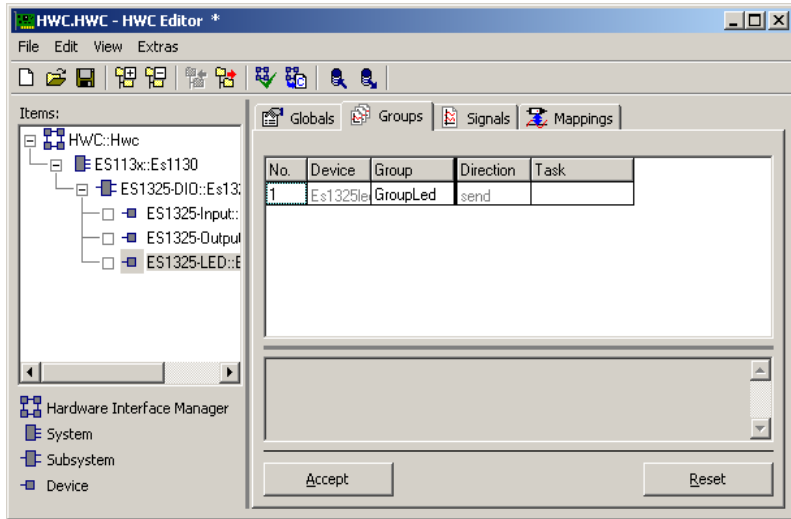


**Fig. 10-61**   The "Groups" Tab of the ES1325-DIO LED Device

There are no item-specific columns defined for the ES1325-LED Device in the "Groups" tab. The possible item settings are described in section 7.4.3 on page 126.

*Task*

The relevant task has to be selected in this field.

## 10.14.12 Signals (ES1325-LED Device)

This section describes the signal-specific options of the LED Device.
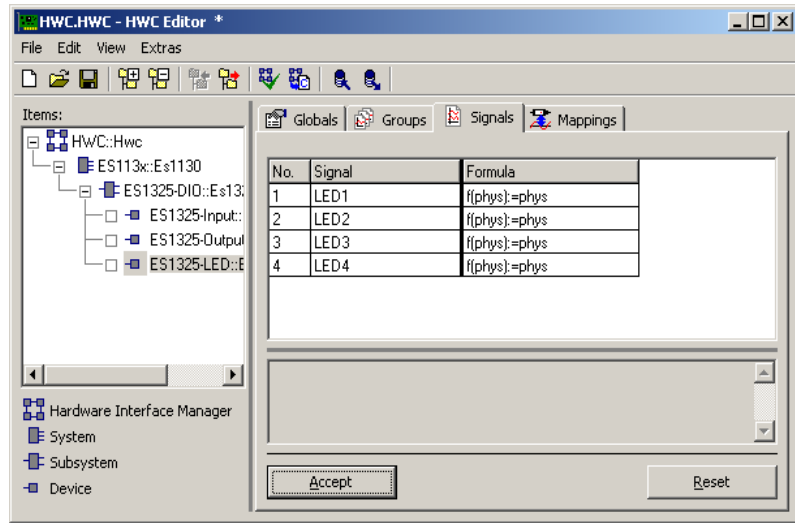


**Fig. 10-62**   The "Signals" Tab of the ES1325-DIO LED Device

There are no item-specific columns defined for the ES1325-LED Device in the "Signals" tab. The possible item settings are described in section 7.4.4 on page 127.

## 10.14.13 Mappings (ES1325-LED Device)

There are no item-specific columns defined for the ES1325-LED Device in the "Mappings" tab. The possible item settings are described in section 7.4.5 on page 129.

## 10.15    ES1330-PWM

The counter board, ES1330, enables the acquisition and output of PWM signals. The ES1330 has six counter components (Am9513A). Each counter component has five 16-bit-wide counters with a count frequency of 4 MHz. Thirty counter inputs and sixteen counter outputs are divided into six internal ports "Port 1 .. Port 6" on the ES1330. Every counter component occupies one of

these ports. Access to the ports takes place via the SUB-D connector of the ES1330. In the HWC Editor items list, the ES1330 counter board is added as a subsystem; the counter component as a device.

> **Note**
>
> *The table in chapter "Connector X1: Digital Inputs and Outputs" of the manual for the ES1330 board (in the* `ETASManuals\ASCET5.1\ES1000` *folder of your ASCET installation) lists the possible combinations of ports and counter inputs or outputs.*

### 10.15.1 Globals (ES1330-PWM Subsystem)

This section describes the global options of the ES1330-PWM subsystem.
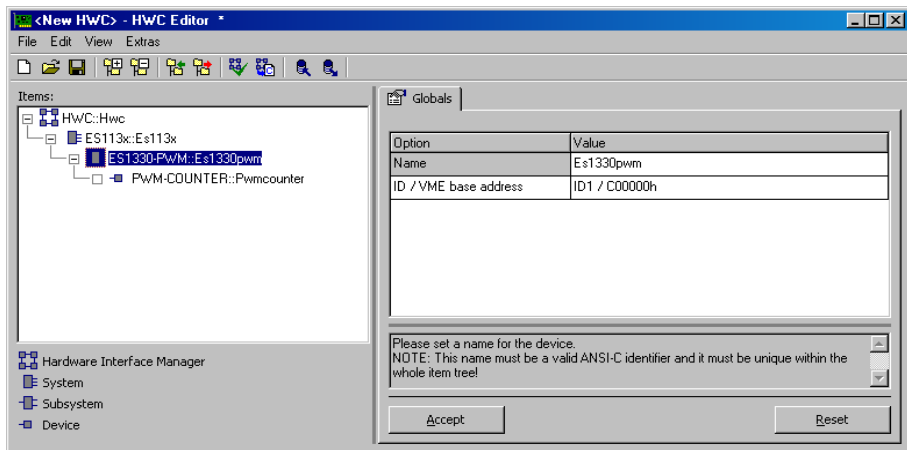


**Fig. 10-63**    The "Globals" Tab of the ES1330-PWM Subsystem

*ID / VME base address*

This line is responsible for the setting of the VME base address. This setting has to correspond to the jumper settings of the ES1330. Four different VME base addresses can be selected for the ES1330 (`ID1/C00000h`, `ID2/C00100h`, `ID3/C00200h`, `ID4/C00300h`); this means that up to four ES1330s can be operated in one ES1000 system. The ES1330 occupies an address space of 256 words from the base address.

## 10.15.2    Globals (PWM-COUNTER Device)

This section describes the global options of the PWM-COUNTER device.
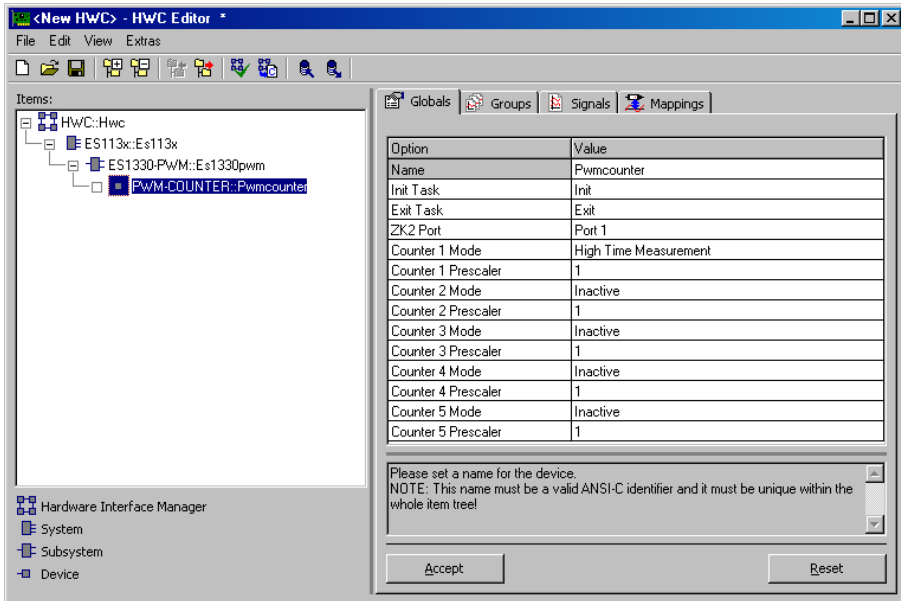


**Fig. 10-64**    The "Globals" Tab of the PWM-COUNTER Device

*Init Task*

The task for initializing the ES1330 is assigned in this line (Type: Init / Application Mode: active).

*Exit Task*

The task to be executed with the ES1330 when the experiment stops is assigned in this line (Type: Init / Application Mode: inactive).

*ZK2 Port*

This is where the port (and hence the counter component) is assigned. Every counter component has its own port.

*Counter 1 Mode .. Counter 5 Mode*

This is where the counter mode is selected for each of the five counters of the counter component. The following list shows the available counter modes.

- Inactive

  The counter is not used.

- PWM-Generator (PWM_Gen)

  The counter is used to generate PWM signals.

- Period-Measurement (P_Meas)

  The counter is used to measure the period of a PWM signal.

- `High Time Measurement`

  In this mode, the duration of a pulse with an active level (here: high-active) is measured.

- `Low Time Measurement`

  In this mode, the duration of a pulse with an active level (here: low-active) is measured.
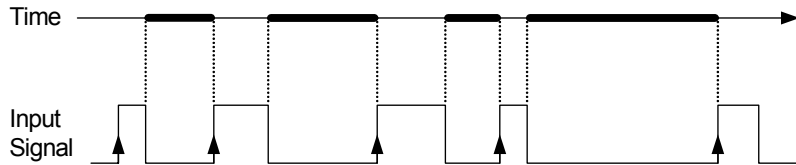


- `Additive High Time Measurement`

  The simulation processor usually acquires measurements from the ES1330 board periodically. With additive measurements, the length of time the signal was active (here: high-active) between two consecutive read-accesses of the measure tab is measured.

- `Additive Low Time Measurement`

  The simulation processor usually acquires measurements from the ES1330 board periodically. With additive measurements, the length of time the signal was active (here: low-active) between two consecutive read-accesses of the measure tab is measured.
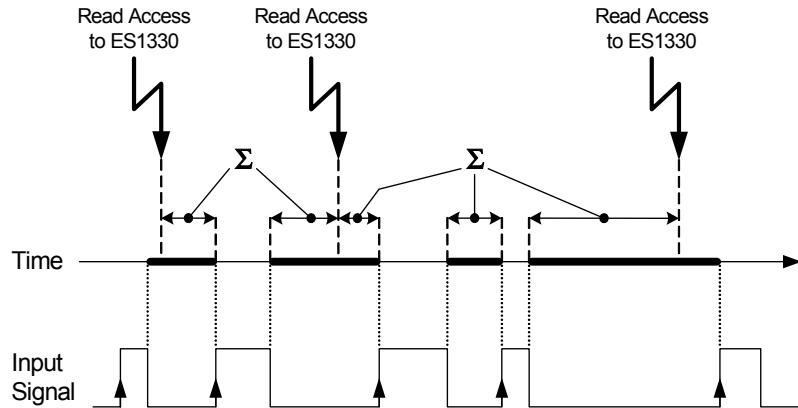


*Counter 1 Prescaler .. Counter 5 Prescaler*

These lines can be used to specify a prescaler for the input frequency of 4 MHz. The scaled input frequency is used as a clock signal for the first four counters. The prescaler makes it possible to generate and evaluate PWM signals with a longer period duration; the resolution of the PWM signals is, however, reduced because of this. The following prescalers can be selected: 1, 10, 100, 1000 and 10000.

### 10.15.3 Groups (PWM-COUNTER Device)

This section describes the signal-group-specific options of the PWM-COUNTER device.
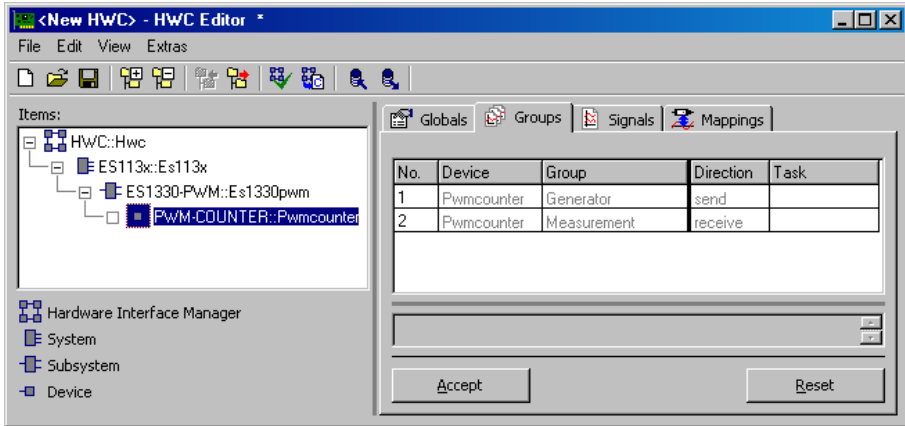


**Fig. 10-65** The "Groups" Tab of the PWM-COUNTER Device

There are no item-specific columns defined for the ES1330-PWM subsystem in the "Groups" tab.

### 10.15.4 Signals (PWM-COUNTER Device)

This section describes the signal-specific options of the PWM-COUNTER device.
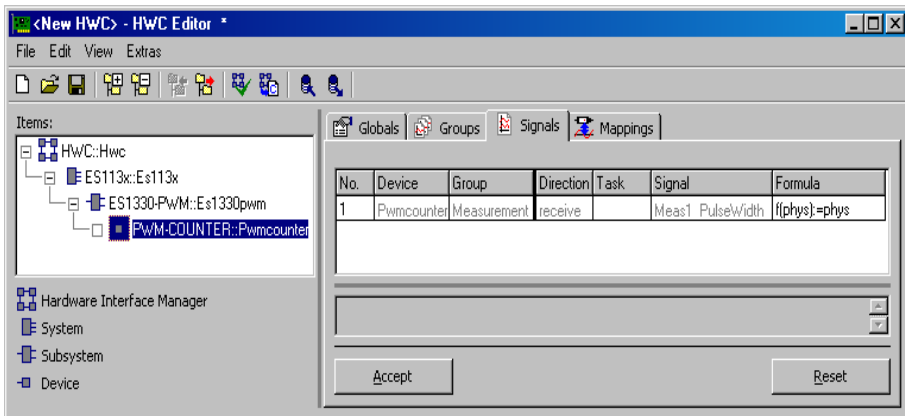


**Fig. 10-66** The "Signals" Tab of the PWM-COUNTER Device

*No.*

The number and significance of the signals is determined by the "ZK2 Port and Counter X Mode" settings from the "Globals" tab.

*Signal*

This is where names (ANSI-C) for the signals are specifically defined. The following naming conventions apply (X = number of the counter):

- GenX_Frequency

  Frequency of the PWM output channel in Hz.

- GenX_DutyCycle

  Duty cycle of the PWM output channel in %.

- MeasX_Period

  Period duration of the PWM evaluation channel in seconds.

- MeasX_DutyCycle

  Result of the pulse width measurement (single pulse or additive).

### 10.15.5    Mappings (PWM-COUNTER Device)

The possible settings are the same for all devices. They are described in section 7.4.5 on page 129.

# 11 Tutorial

The tutorial describes an INTECRIO experiment (chapter 11.1), as well was the setup of an ES1222 (chapter 11.2), ES1303 (chapter 11.3), and ES1325 board (chapter 11.4 and chapter 11.5), using supplied examples.

ASCET-RP V5.6 includes the sample files `INTECRIO_Tutorial.exp` and `RTIOTutorial.exp`. During the installation, the sample file is stored in the subdirectory `ASCET5.2\export` of your ASCET installation.

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

> **Note**
>
> *This tutorial is written under the assumption that 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.*

**To create a new 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.
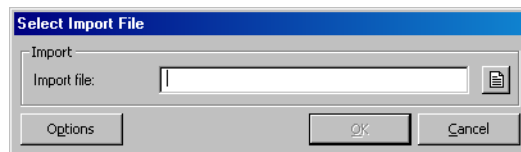


- Use the button to select the `RTIOTutorial.exp` file from the `ASCET5.2\export` subdirectory of your ASCET installation.

  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..
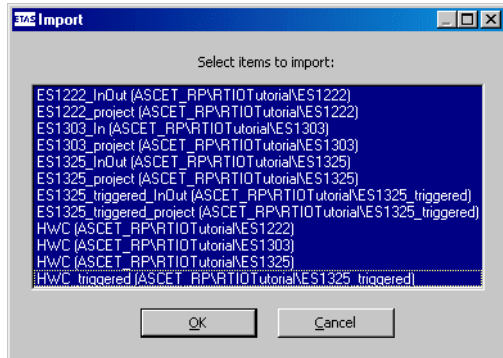


- Click **OK**.

  The files are imported. A list of the imported database items is shown in the "Imported Items" window.
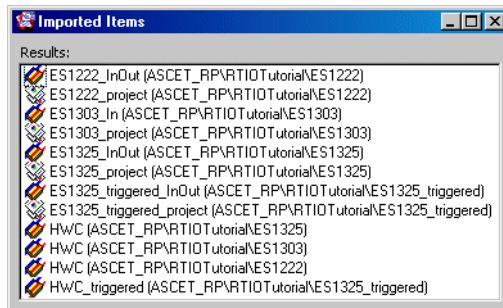


- Close the "Imported Items" window.

- Import the `INTECRIO_Tutorial.exp` file the same way.

  Make sure all components are selected for the import.



```
LowpassKEnabled (ETAS_IconLib\TransferFunction\Lowpass)
LowpassKEnabled (SystemLib_ETAS\Transferfunction\Lowpass)
M01_DataGenerator (ASCET_RP\INT_Tutorial)
M01_LowPass (ASCET_RP\INT_Tutorial)
P01_Project (ASCET_RP\INT_Tutorial)
SM01_DataGenerator (ASCET_RP\INT_Tutorial)
```

**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`.
- Enter the subnet mask `255.255.255.0`.
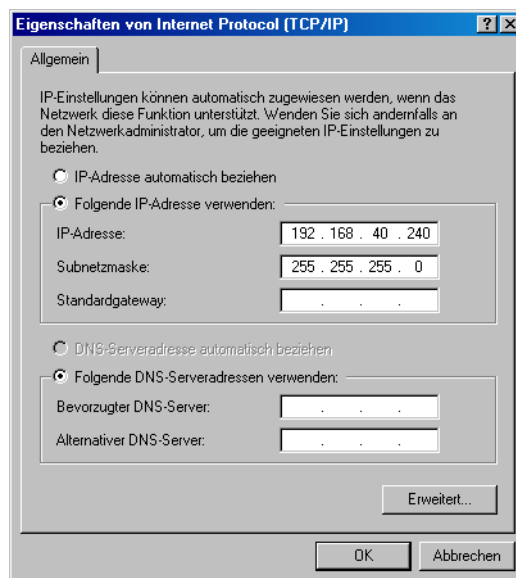


(← Windows® 2000)

## 11.1 Tutorial – Experimenting with INTECRIO

This chapter explains how to transfer ASCET projects to INTECRIO as well as how to use Back-Animation (see page 62) when experimenting with INTEC-RIO. 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 `INTECRIO_Tutorial.exp` 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 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 INTEC-RIO 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.

### 11.1.1 Preparations

First of all, make the necessary preparations.

**To copy the INTECRIO workspace:**

The prepared INTECRIO workspace is located in the `ETAS\ASCET5.2\export` directory of your ASCET installation.

- Copy the `INTECRIO_Tutorial_Workspace` directory to your hard disk, e.g. to

  `ETASData\INTECRIO2.1\INTECRIO_Tutorial_Workspace`.

**To select a target:**

- Open the project `P01_Project` in the project editor.

- Open the "Project Properties" window.
- In the "Build" node, select the target `Proto-typing and` the compiler `GNU-C V3.4.4 (PowerPC)`.

  If no predefined operating system settings for this target are loaded, copy the settings with **Operating System → Copy from Target**.

## 11.1.2    Transferring the Project

The next step is to transfer the project to INTECRIO.

**To transfer the project to INTECRIO:**

- Open the project `P01_Project`.

  `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 available.

- Click the **Transfer Project to selected Experiment Target** button.
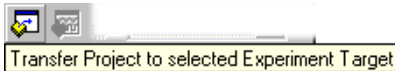
  The "INTECRIO Project Transfer" window opens.
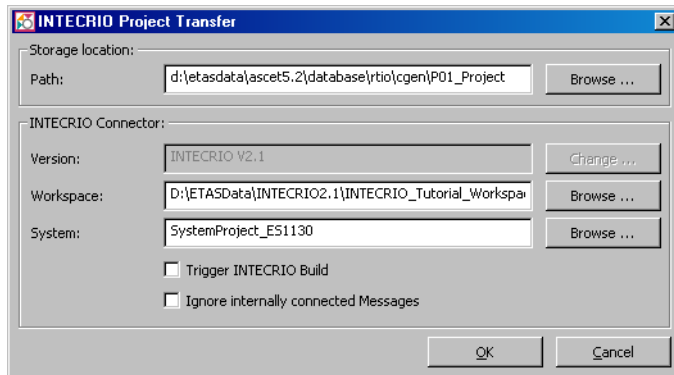
- In the "Path" field, enter a path for the generated files.

- Use the **Change** button next to the "Version" field to select the INTECRIO version you will use.

  If only one INTECRIO version is installed on your computer, the button is deactivated.

- 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.

  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:
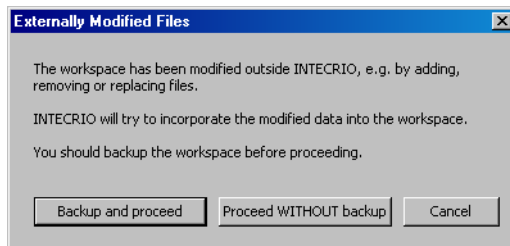


Click **Proceed without Backup** to continue.

## 11.1.3 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 `UserAppMode` 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 INTEC-RIO 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 INTECRIO 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.eex` file from the `EE\Experiments\INTEC-RIO_Tutorial` subdirectory of your INTEC-RIO workspace.

  As you have already started the simulation, values are displayed immediately.

Your INTECRIO experiment environment then looks something like this:
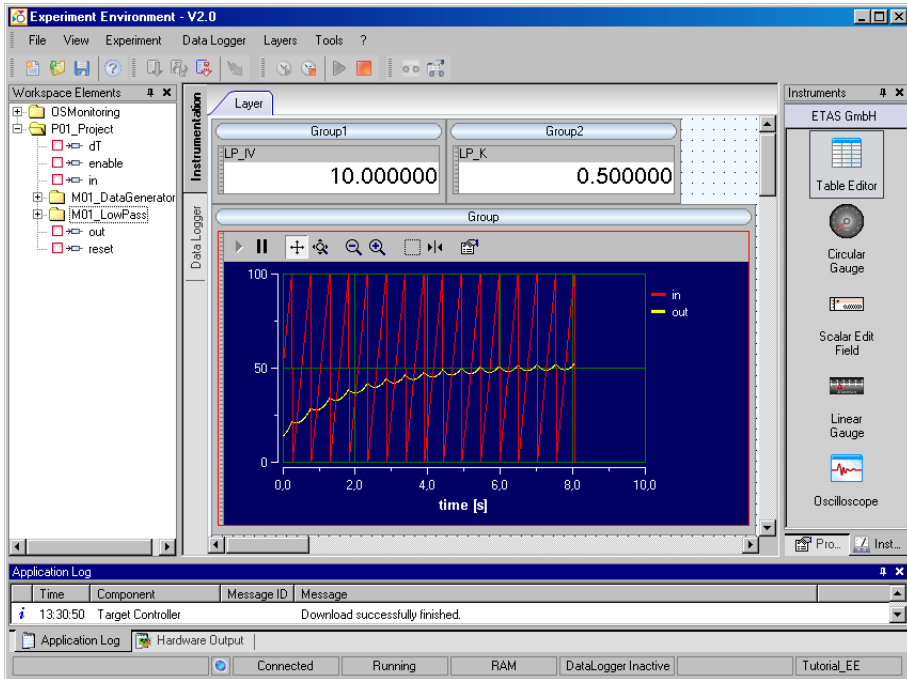


### 11.1.4 Using Back-Animation

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

**To start Back-Animation:**
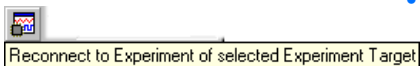


- Click the **Reconnect to Experiment of Selected Experiment Target** button in the ASCET project editor.

  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 measure and calibration windows opens.
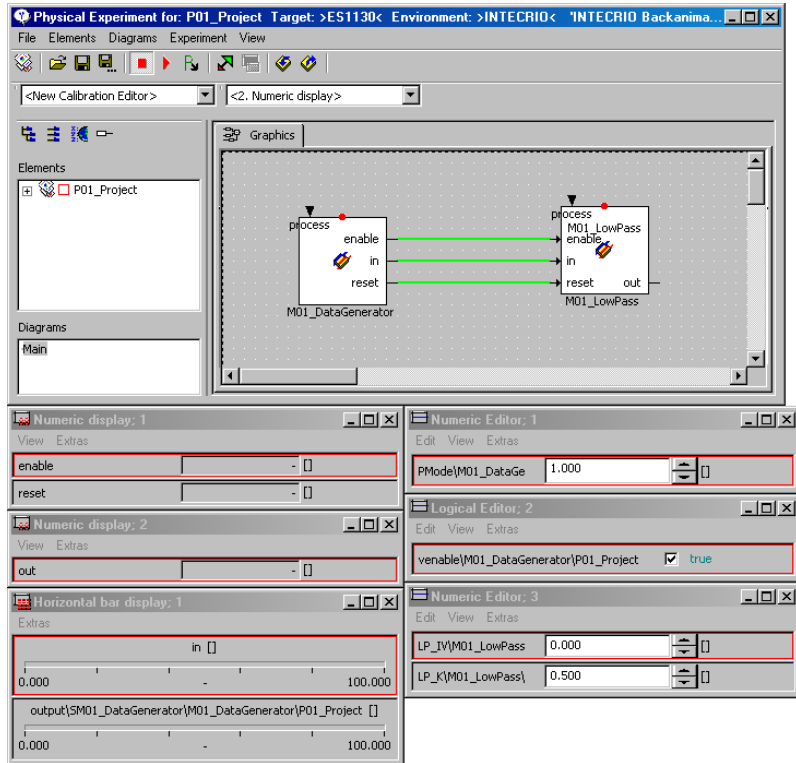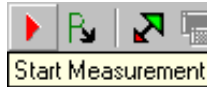
- Click the **Start Measurement** button.

  Measuring is started in the ASCET experiment; values are displayed in the measure windows.
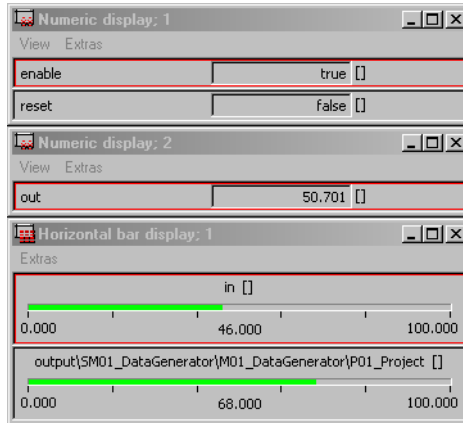


You can now calibrate values either in the ASCET experiment or in the INTEC-RIO experiment. The modified values are transferred to the INTECRIO experiment and displayed and used there.

**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 ("Group1") in the INTECRIO experiment 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 `venable` 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`.
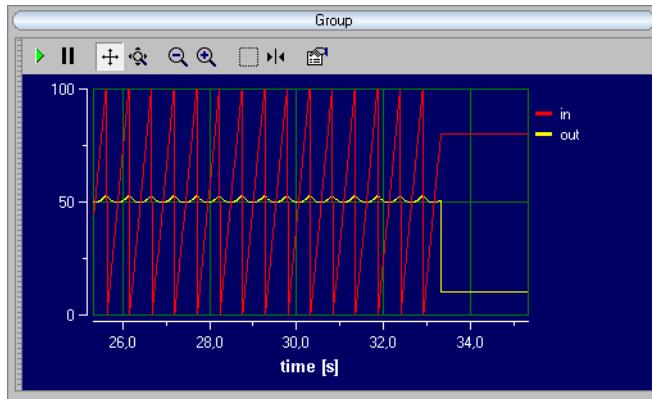


- Set `venable` back to `true` and then `PMode` to 2 to select the other signal generator.

  The display in the INTECRIO oscilloscope changes accordingly.
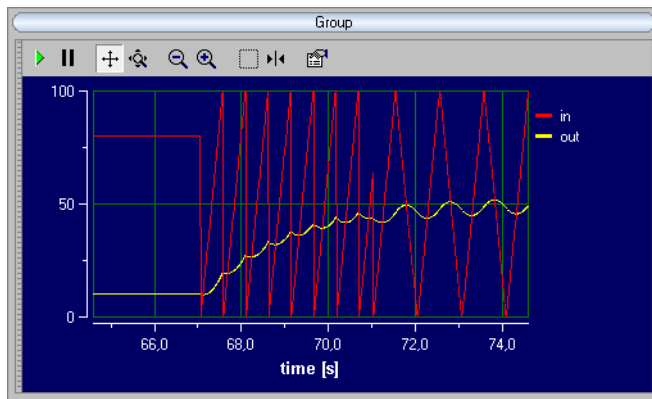
**To view the ASCET components:**

- 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.
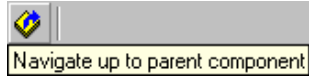
- Select **View → Monitor All**.

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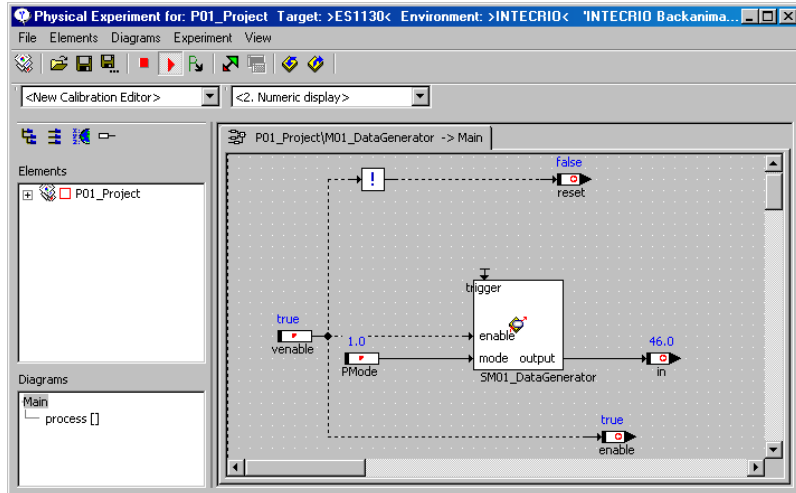


- Navigate through the model specification to the state machine SM01_DataGenerator.

- Right-click one of the states and select **Ani-mate States** from the context menu.

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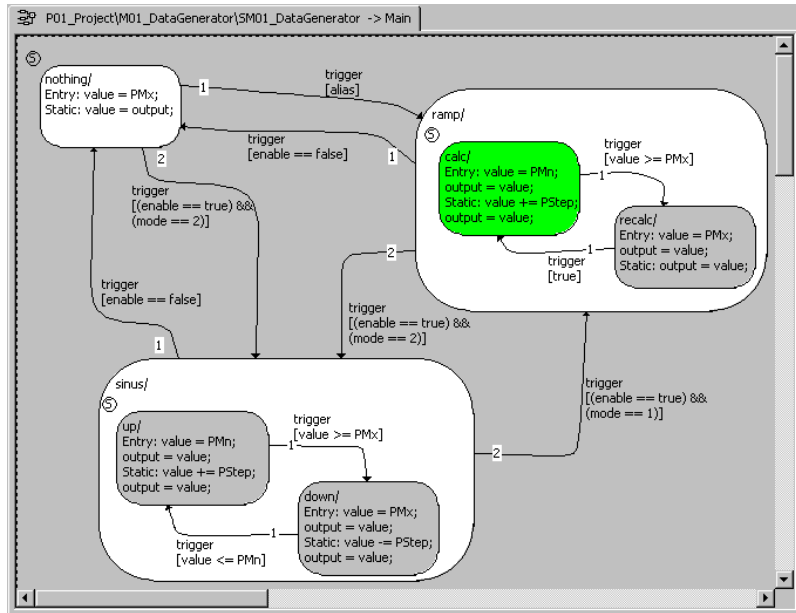


## 11.2 Tutorial – ES1222 (CAN-IO)

The example for ES1222 contains three variables which are sent by CAN channel 1 and received by CAN channel 3. If both channels **are connected together with a terminator** (see Fig. 11-3 on page 311), the experiment can be performed.

The model is present and the preparations required for RTIO integration of the ES1222 (tasks, messages and HWC module) have been made; you have to set up the hardware configuration and create the code.
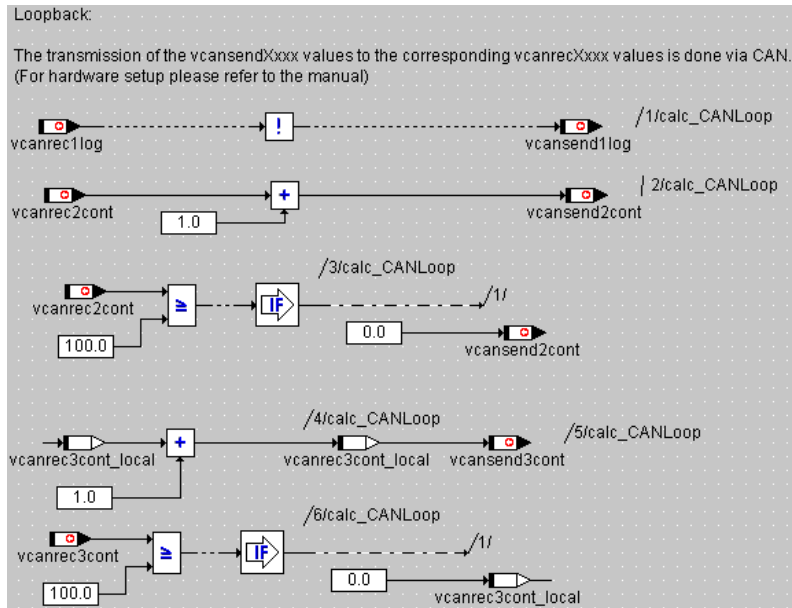
**Fig. 11-1**    ES1222 - `ES1222_InOut` module
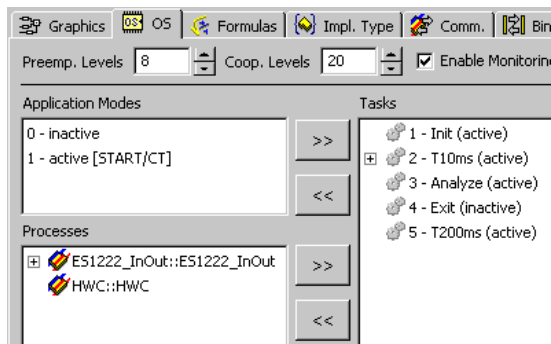


**Fig. 11-2**    ES1222 – Presets in the OS editor

## 11.2.1 The ES1222 Board

The figure below shows the front panel of the ES1222 CAN board.
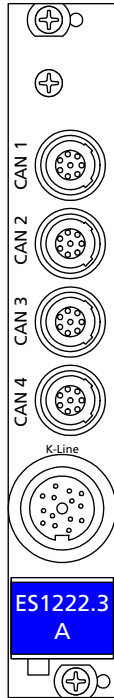


The inputs CAN1 to CAN4 are independent and galvanically isolated CAN interfaces; the K wire input is a serial K and L line interfaces used for connecting VMEbus systems with external devices.

*Jumpers of the ES1222*

For the experiment to work, you must verify the correct configuration of the jumpers.
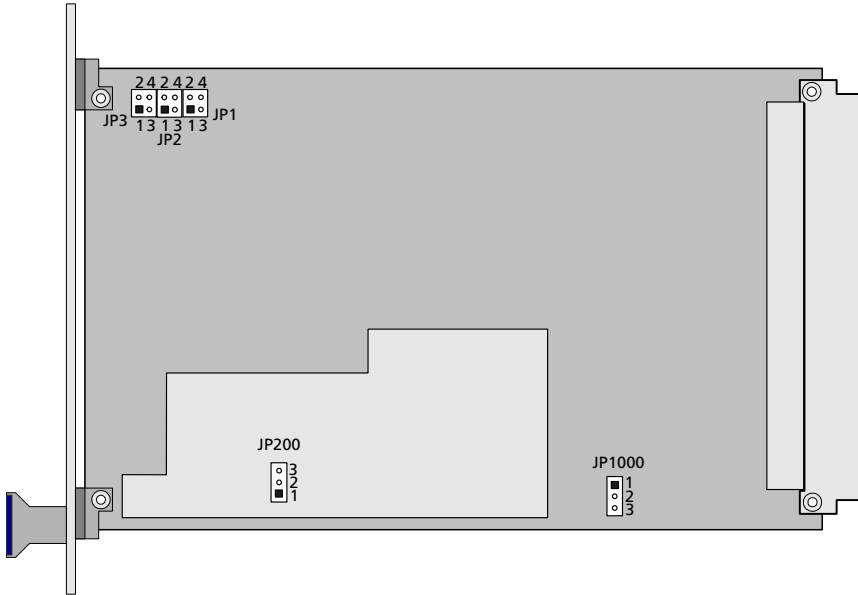


| Jumper | PIN | Meaning |
| --- | --- | --- |
| JP1 | open | CAN1 and CAN2 are independent |
| JP2 | open | |
| JP3 | open | |
| JP1000 | 1-2 closed | |
| JP200 (ES1222.3 only) | any | K Line is not sed in the example |

*Operating Several ES1222 Boards*

Up to four ES1222 boards can be used in one ES1000 system.

## 11.2.2    Sample Project

**How to open the exercise example:**

- In the Component Manager, select the `ASCET_RP\RTIOTutorial\ES1222` folder.
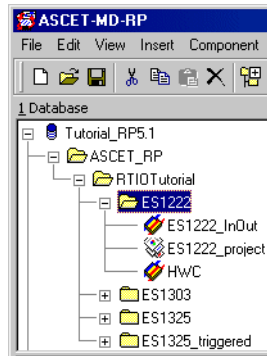


- Select the `ES1222_project` project.
- Open the project.
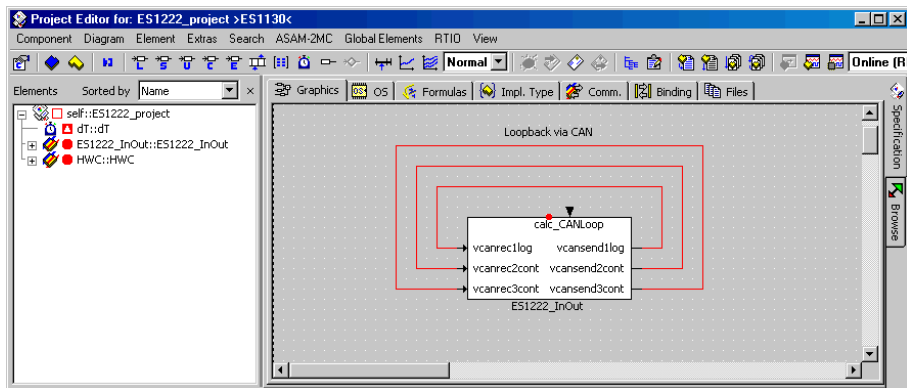




- Click the **Project properties** button. The "ProjectProperties" window opens.

- In the "Build" node, select the options

  ```
  Target:     >ES1130< or >ES1135<,
  Compiler:   GNU-C * (PowerPC).
  ```
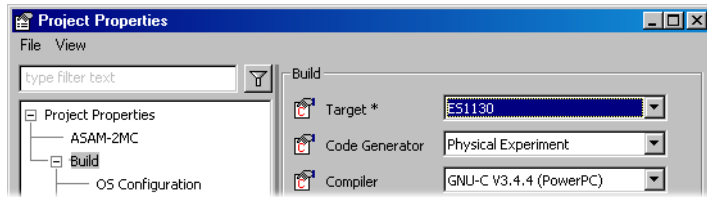
**Note**

*Only messages declared as "Exported" are available for the RTIO communication.*

### 11.2.3    Creating the Hardware Configuration

<u>**Note**</u>

*Normally, you must create the C code module* HWC, *and insert it into the project, before you edit the hardware configuration. In the tutorial, however, this step has been performed for you.*

**How to open the HWC editor:**

- In the project editor, select **RTIO → Open Editor**.

  The HWC editor is opened.
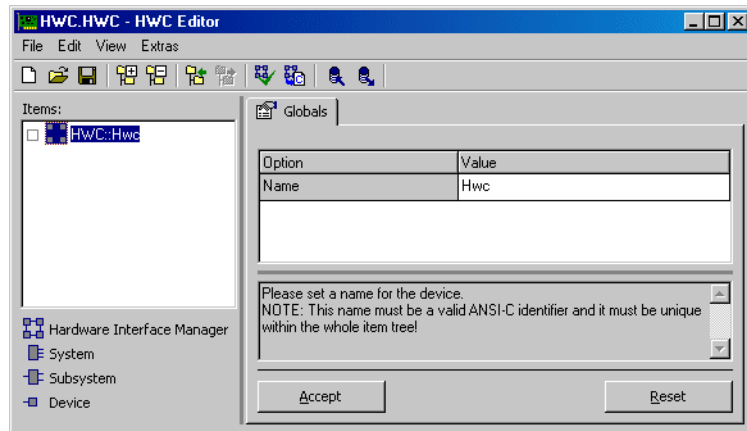


**How to create the hardware configuration (HWC):**

The hardware must be described as a tree structure in the items list. The HWC item always exists and forms the root of the tree.

- In the HWC editor, select **Edit → Add Hardware Item**

  *or*

- click the **Add Item** button.
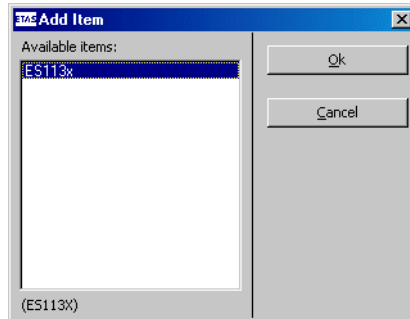
  The "Add Item" window is displayed.



**Add Item** always opens the list of available items of the next hierarchy level.

- Select the `ES113x` entry.

  This entry is used to describe the ES1000.x system with integrated ES1130 or ES1135 PowerPC processing node.

- Click **OK**.

  The `ES113x` item is added to the "Items" list.
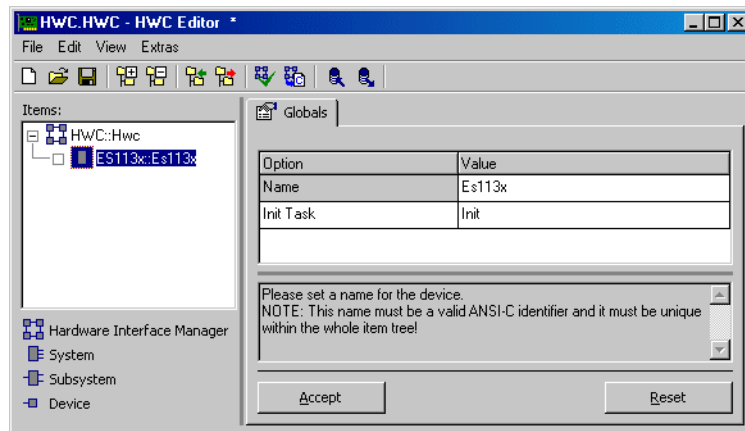
- In the "Items" list, select the `ES113x` item.

  On the "Globals" tab, the `Init Task` option is given the default task name `Init`. An Init task with the same name exists in the OS editor of the sample project; therefore, no other Init task need be selected here.

**How to integrate and configure ES1222:**

- Select **Edit** → **Add Item** to open the list of available items of the next hierarchy level.
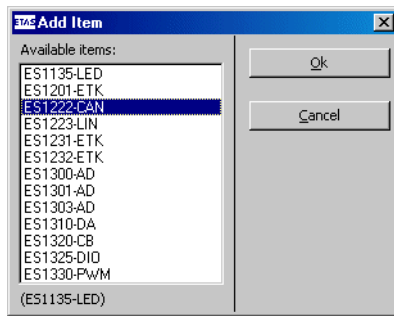


- Select the `ES1122-CAN` entry.

  This entry is used to describe the ES1222 CAN and K Line interfaces.

- Click **OK**.

  The `ES1222-CAN` item is added to the
  "Items" list.
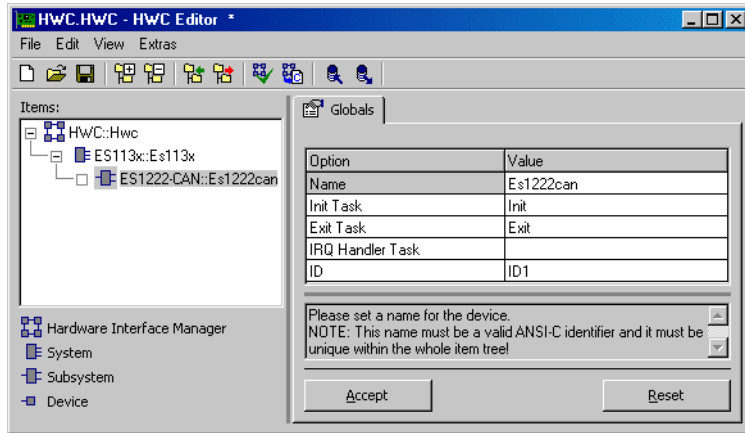


- In the "Items" list, select the `ES1222-CAN`
  item.

On the "Globals" tab, the task names `Init` and `Exit` are predefined for the
`Init Task` and `Exit Task` options. Init tasks with these names exist in the
OS editor of the sample project; therefore, no other tasks need be selected
here. For the `IRQ Handler Task` option, no default name is given.

- Double-click in the empty field next to the
  `IRQ Handler Task` option.

  The "Task selection" window opens.
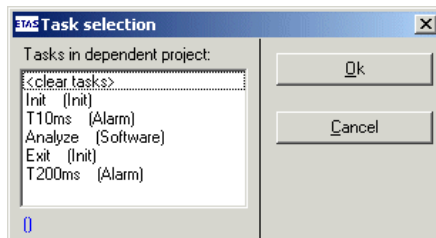
- Select the `Analyze` software task and click **OK**.

  The board number in the `ID` option is set automatically. Since the `ES1222-CAN` item is the first of its type, the value is set to `"ID1."`

- Save the changes with **Accept**.

**How to integrate the CAN channels:**

- In the HWC editor, select the `ES1222-CAN` item.

- Select **Edit** → **Add Item** to open the list of available items of the next hierarchy level.
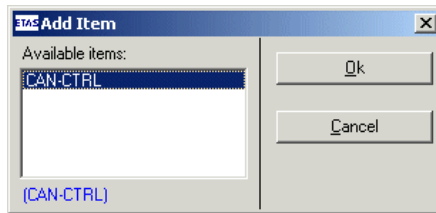


- Select the `CAN-CTRL` entry and click **OK**.

  This entry is used to describe the CAN controller.

- In the "Globals" tab, click on the value 1000 next to the "Baud Rate [kBaud]" option.

  The table field changes to a combo box containing the available baud rates.

- From the combo box, select a rate of 500 kBaud.

- Accept the settings with **Accept**.

- In the "Items" list, select the `CAN-CTRL` item.
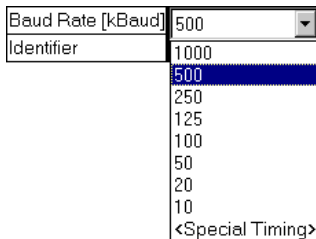
- Select **Edit → Add Item** to open the list of available items of the next hierarchy level.
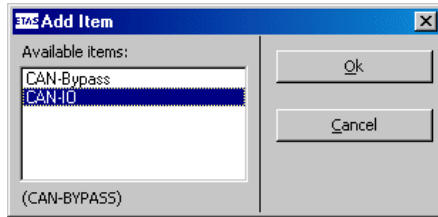


- Select the `CAN-IO` entry and click **OK**.
- Add a second CAN controller with CAN-IO device for the second CAN channel.

The item tree for the description of the sample system is now fully specified.
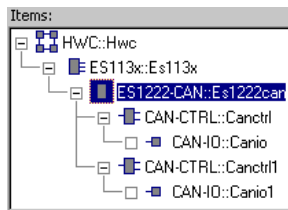


**To select a CAN channel:**

The second CAN connector shall use the *third* CAN channel.

- In the HWC editor, select the second CAN controller.
- In the "Globals" tab, enter the value `CAN3` for the "CAN Connector" option.
- Accept the settings with **Accept**.

## 11.2.4 HWC Settings for the ES1222 (CAN-IO)

Next, the two CAN channels must be configured. The first channel (`CAN-IO::Canio` item in the HWC editor) should send the signals which are received by the second channel (`CAN-IO::Canio1`).
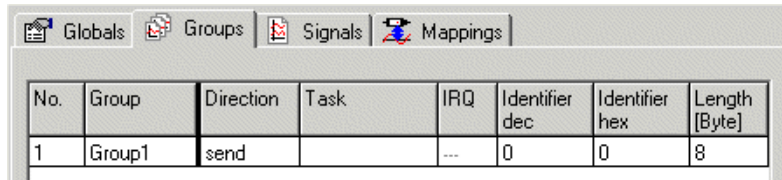
No settings need to be made in the example for the `CAN-IO` item on the "Globals" tab.

**"Groups" tab:**    The CAN messages are specified in form of signal groups on this tab.

**How to perform the settings for group 1:**

- In the "Items" list, select the `CAN-IO::Canio` item.
- Select the "Groups" tab.

| No. | Group | Direction | Task | IRQ | Identifier dec | Identifier hex | Length [Byte] |
|-----|-------|-----------|------|-----|----------------|----------------|---------------|
| 1 | Group1 | send | | --- | 0 | 0 | 8 |

The `Group1` group is always present.

- Double-click in row 1 of the "Group" column.
  The table field changes to an input field.
- Name the group `MSG_100`.
- Double-click in row 1 of the "Task" column.
  The "Task selection" window opens.
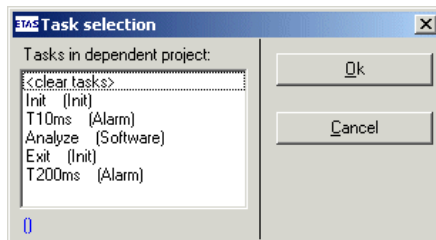
- Select the `T10ms` alarm task and click **OK**.
- Click in row 1 of the "Identifier dec" column.
- Enter the value `100` for the identifier.
  The value `64` is automatically entered in the "Identifier hex" column.
- Save the settings with **Accept**.

**How to add a group:**

Next, add a second group which is transferred in another task.

- Click in row 1of the "No." column, to select the existing group.

  Functions for adding signal groups are activated in the **Edit** menu.

- Choose **Edit → Add Row After**

*or*
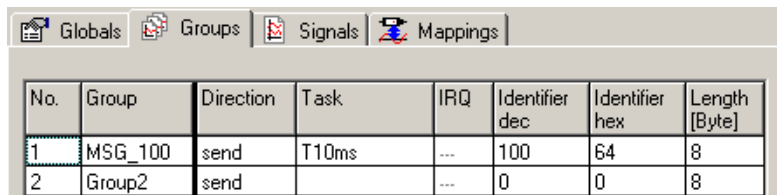
- select **Add Row After** from the context menu.

  - If you did not save the changes to the first group, you will be asked to do so now.
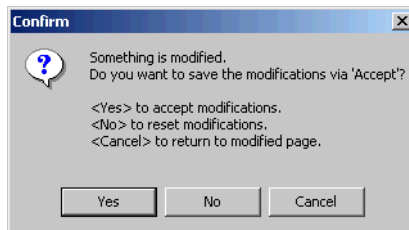


  - Confirm the reminder with **OK**.

  - Open **Add Row After** again.

  A new row is inserted after the first one.



**How to perform the settings for group 2:**

- Name the group `MSG_101`.

- In the "Task" column, select the `T200ms` alarm task.

- In the "Identifier dec" column, enter the value `101`.

  The value `65` is automatically entered in the "Identifier hex" column.

- Save the settings with **Accept**.

**"Signals" tab:**   The CAN signals are specified on this tab, whereby one signal is always present. The example uses two numeric and one logical signal; first, you will add two signals.

**How to add signals:**

- Select the "Signals" tab.



- Click in row 1 of the "No." column, to select the existing signal.

  Functions for adding signals are activated in the **Edit** menu.

- Select **Edit → Add Multiple Rows After**

*or*

- select **Add Row After** from the context menu.

  The "New Value" window opens.
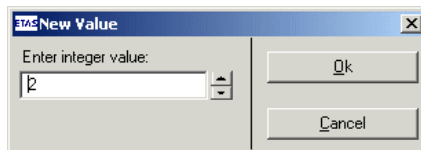


- Enter the number 2 in the entry field.

  The number can also be set using the arrow buttons.

- Click **OK**.

  Two rows are inserted after the specified row. The group from the selected row (`MSG_100`) is preset; the `Signal<n>` standard name is counted up.



(To provide a better overview, the columns of the bit matrix (see section "7654321.. (Bit matrix)" on page 187) that are not required in the example were minimized.)

The logical and one numeric signal are transferred in the first group, the other numeric signal is transferred in the second group.

**How to set up a logical signal:**

- Double-click in row 1 of the "Signal" column.
- Name the signal `vcansend1log`.
- Double-click in row 1 of the "Signal-Type" column.

  The table field changes to a combo box which lists the available types.

- From the combo box, select `bool`.
- Double-click in row 1 of the column "0 0" (last column of the bit matrix).

  The field is marked with 1. This specifies that the signal is transferred in byte 0, bit 0, of the first CAN message.

**How to set up the first numeric signal:**

- Name the signal `vcansend2cont`.
- In row 2 of the "Signal-Type" column, select the `uint` type.
- Double-click in row 2 of the column "1 0."

  The field is marked with 1.

- Double-click in row 2 of the column "1 7."

  The field is marked with 1.

- Press the <SHIFT> key and click again in row 2 of the column "1 7."

  All fields lying between the columns are marked with 1.
  This specifies that the signal is transferred in byte 1, bits 0 – 7, of the first CAN message.

**How to set up the second numeric signal:**

- Name the signal `vcansend3IRQ`.

- In row 3 of the "Group" column, select the `MSG_101` group.

- In row 3 of the "Signal-Type" column, select the `uint` type.

- Mark the columns "0 0" to "0 7" in row 3 as described above.

  This specifies that the signal is transferred in byte 0, bits 0 - 7, of the second CAN message.

- Save the entries by clicking **Accept**.

After setting up the three signals, the table should look as follows:
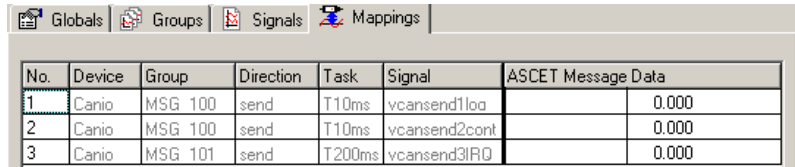
| No. | Group | Signal | Formula | Signal-Type | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 0 7 | 0 6 | 0 5 | 0 4 | 0 3 | 0 2 | 0 1 | 0 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | MSG_100 | vcansend1log | f(phys):=phys | bool | | | | | | | | | | | | | | | | 1 |
| 2 | MSG_100 | vcansend2cont | f(phys):=phys | uint | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | |
| 3 | MSG_101 | vcansend3IRQ | f(phys):=phys | int | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**"Mappings" tab:** The CAN signals and the ASCET messages from the project are assigned to each other on this tab. A selection dialog can be opened by clicking the desired cell in the "ASCET Message" column. This dialog contains only messages featuring the *Exported* attribute and corresponding to the transfer direction of the signal group (*send* or *receive*); that is:

- Direction = receive → receive-messages,
- Direction = send → send-messages

**How to assign an ASCET message manually:**

- Select the "Mappings" tab.



- Double-click in row 3 of the "ASCET Message" column.

  The "Message selection" window opens.
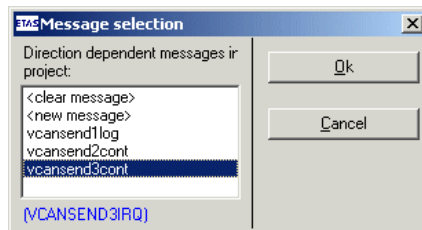


  It contains all send-messages from the ASCET project.

- Select the vcansend3cont message and click **OK**.

- Save your entries by clicking **Accept**.

The signals in rows 1 and 2 have the same names as two send-messages in the project. You can assign the messages manually as described above or assign them automatically.

**How to assign an ASCET message automatically:**

• Select the "Globals" tab.
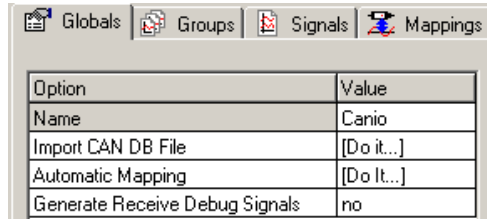


• Double-click in the "Automatic Mapping" row of the "Value" column.

The "Automatic Mapping" window opens.
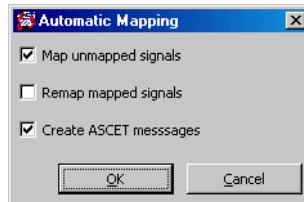


A detailed description can be found in the section "Automatic Mapping" on page 182.

• Activate the **Map unmapped signals** and **Create ASCET messages** options.

The vcansend1log and vcansend2cont signals are assigned to the ASCET messages of the same name.
The vcansend3IRQ signal was already assigned to a message manually so that it is not affected.

The "Mappings" tab now shows that a message is assigned to every signal.

This completes the configuration of the first CAN channel.

*Channel 3*

**"Groups" tab:**   The CAN messages are specified in form of signal groups on this tab.

**How to perform the settings for group 1:**

- In the "Items" list, select the `CAN-IO::Canio1` item.
- Select the "Groups" tab.

  The regularly generated contents of the tab is identical with that of the first channel.
- Name the first group `MSG_100` (see page 298).

Since the third channel *receives* the signals, you must change the transfer direction of the group.

- Double-click in row 1 of the "Direction" column.

  The table field changes to a combo box which lists the available directions.
- Select the `receive` direction.
- In row 1 of the "Task" column, select the `T10ms` alarm task (see page 298).
- In row 1 of the "Identifier dec" column, enter the value `100`.

  The value `64` is automatically entered in the "Identifier hex" column.
- Save the settings by clicking **Accept**.

Next, add a second group which is transferred in interrupt mode as described on page 299.

**How to perform the settings for group 2:**

- Name the second group `MSG_101`.
- In the "Direction" column, select the `receive` direction.
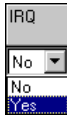
- Double-click in row 2 of the "IRQ" column.

  The table field changes to a combo box which lists the available options.

- Select the `yes` option.

  The "Task" column is automatically reset and blocked. The value 1 is automatically entered in the "Prescaler" column.

| No. | Device | Group | Direction | Task | IRQ | Identifier dec | Identifier hex | Length [Byte] | Activated Task | Pre-scaler |
|-----|--------|-------|-----------|------|-----|----------------|----------------|---------------|----------------|------------|
| 1 | Canio1 | MSG_100 | receive | T10ms | no | 100 | 64 | 8 | | ... |
| 2 | Canio1 | MSG_101 | receive | ... | yes | 0 | 0 | 8 | | 1 |

- In the "Identifier dec" column, enter the value `101`.

  The value `65` is automatically entered in the "Identifier hex" column.

- Save the settings by clicking **Accept**.

**"Signals" tab:**  The CAN signals are specified on this tab, whereby one signal is always present. First, add two signals as described on page 300.

The logical and one numeric signal are transferred in the first group, the other numeric signal is transferred in the second group.

### How to set up a logical signal:

- Name the first signal `vcanrec1log`.
- In row 1 of the "Signal-Type" column, select the `bool` type.
- Mark the "0 0" column in row 1 (see page 301).

  This specifies that the signal is transferred in byte 0, bit 0, of the first CAN message.

### How to set up the first numeric signal:

- Name the signal `vcanrec2cont`.
- In row 2 of the "Signal-Type" column, select the `uint` type.

- Mark the columns "1 0" to "1 7" in row 2 (see page 301).

  This specifies that the signal is transferred in byte 1, bits 0 - 7, of the first CAN message.

**How to set up the second numeric signal:**

- Name the signal `vcanrec3cont`.
- In row 3 of the "Group" column, select the `MSG_101` group.
- In row 3 of the "Signal-Type" column, select the `uint` type.
- Mark the columns "0 0" to "0 7" in row 3.

  This specifies that the signal is transferred in byte 0, bits 0 - 7, of the second CAN message.
- Save your changes by clicking **Accept**.

After setting up the three signals, the table should look as follows:

| No. | Group | Signal | Formula | Signal-Type | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 0 7 | 0 6 | 0 5 | 0 4 | 0 3 | 0 2 | 0 1 | 0 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | MSG_100 | vcanrec1log | f(phys):=phys | bool | | | | | | | | | | | | | | | | 1 |
| 2 | MSG_100 | vcanrec2cont | f(phys):=phys | uint | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | |
| 3 | MSG_101 | vcanrec3cont | f(phys):=phys | uint | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

(Tabs: Globals | Groups | Signals | Mappings)

The three signals have receive-messages under the same name in the project so that all messages can be assigned automatically.

**How to assign ASCET messages automatically:**

- Select the "Globals" tab.
- Click in the "Automatic Mapping" row of the "Value" column.

  The "Automatic Mapping" window opens.

  A detailed description can be found in the section "Automatic Mapping" on page 182.
- Click the **All** button.

  The signals are assigned the ASCET messages of the same name.

Of course, the messages can also be assigned manually on the "Mappings" tab as described on page 303.

The "Mappings" tab now shows that every signal is assigned the message of the same name.

| No. | Device | Group | Direction | Task | Signal | ASCET Message | Data |
|-----|--------|-------|-----------|------|--------|---------------|------|
| 1 | Canio1 | MSG 100 | receive | T10ms | vcanrec1log | vcanrec1log | --- |
| 2 | Canio1 | MSG 100 | receive | T10ms | vcanrec2cont | vcanrec2cont | --- |
| 3 | Canio1 | MSG 101 | receive | | vcanrec3cont | vcanrec3cont | --- |

This completes the configuration of the second CAN channel.

11.2.5    Saving the Hardware Configuration

The created hardware configuration can be saved in the File container of the project, or as a DOS file (*.HWX extension).

**To save the hardware configuration in the File container:**

- In the HWC editor, click on the **Save** button

  *or*

- select **File → Save** to save the hardware configuration.

  If this is the first time you save the configuration, the "Save Configuration File" opens.
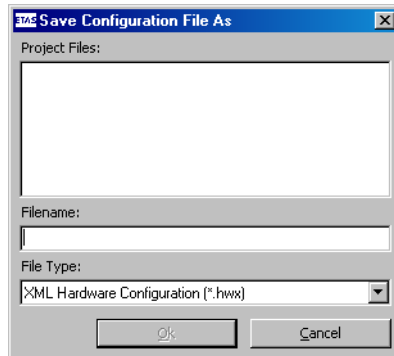
- In the "File Type" combo box, select the XML Hardware Configuration (*.hwx) entry.

- Enter a name for the configuration.

**Note**

*The solution of the example is stored under the name of* `ES1222.hwx`. *Make sure that you **do not overwrite** this file.*

- Click **OK**.

  The hardware configuration is saved to the File container of the project. It is visible in the "Files" tab.

**To save the hardware configuration as a DOS file:**

- In the HWC editor, choose the **File → Export**.
- In the "Export Hardware Configuration" window, enter the file type, path and file name for the hardware configuration.

  The hardware configuration is saved to the specified file. It can be imported to the HWC editor at a later time via **File → Import**.

### 11.2.6 Generating Code for the HWC Module

Subsequently, the generating sequence for the HWC module has to be started.

**To generate code for the HWC module:**

- In the HWC editor, select **Extras → Generate Code → For Current Experiment**

*or*

- click on the **Generate Code for Current Experiment** button.
- Confirm overwriting the HWC module.
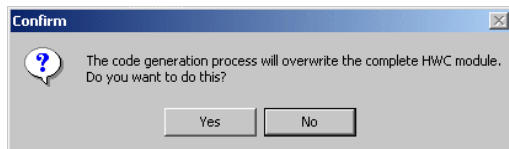
- The C code for the HWC module is generated for the experiment currently selected in the project's code generation options.

You can also generate code for other experiments:

- In the HWC editor, select **Extras → Generate Code → For Phys. and Quant. Experiment** to generate code for both the Physical and the Quantization experiment.

- In the HWC editor, select **Extras → Generate Code → For Phys., Quant. and Impl. Experiment** to generate code for the Physical, Quantization, and Implementation experiment.

### 11.2.7  Experimenting with the Sample Project

Because all RTIO-specific actions are now finished, you can close the HWC editor. The next step is to start the regular code generation for the experimental target from within the project editor.

> **Note**
>
> *It is not recommended to include the HWC module in the graphical display of the Project Editor. Since this module changes with each RTIO generation process, it would result in an unfavorable representation of the HWC module.*

**How to generate code for the experimental target:**

- In the project editor, choose **Component → Build** to generate code for the entire project.

- Select **Component → View Generated Code** to view the generated code.

**How to experiment online:**

- Connect the CAN 1 and CAN 3 inputs of the ES1222 with the appropriate cables and a terminator.
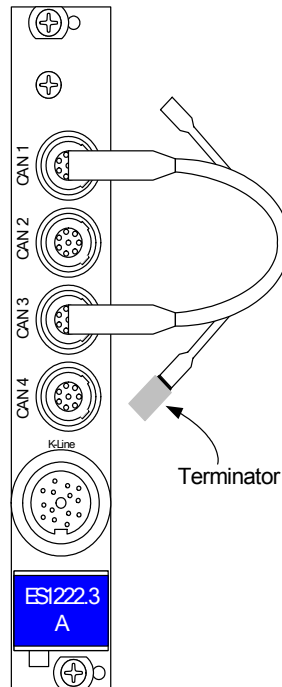


Terminator

**Fig. 11-3**  ES1222 with cable and terminator

**Note**

*If you omit the terminator, no communication will take place.*

- Select `Online (RP)` from the "Experiment Target" combo box.

  `Offline (RP)` is intended for offline experiments on the Target.

- Select **Component → Open Experiment**

*or*

Open Experiment for selected Experiment Target

- click the **Open Experiment for selected Experiment Target** button.

  The "Physical Experiment" window and the predefined experiment environment consisting of two oscilloscopes open.
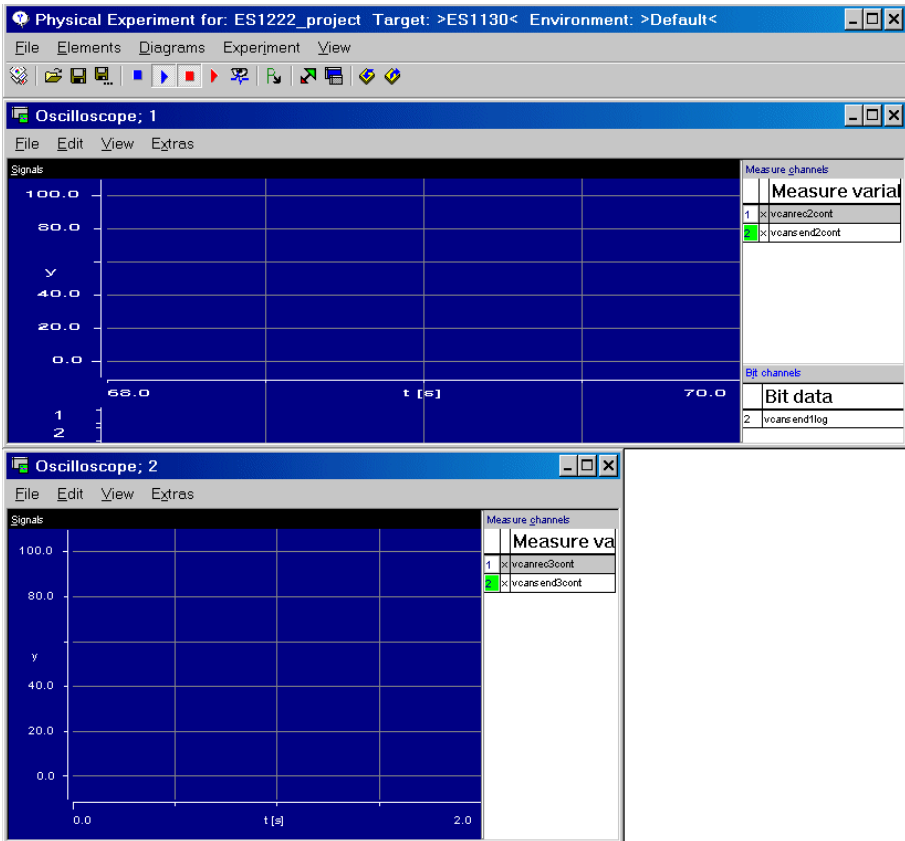


- In the "Physical Experiment" window, select **Experiment → Start ERCOS**

*or*

- click the **Start ERCOS** button.

  The operating system starts and the model is executed.

The top oscillograph shows the `vcansend2cont` send-message (green, left curve) and the `vcanrec2cont` receive-message (white, right curve) in the "Signals" area. The bit display shows the `vcanrec1log` receive-message (top curve) and the `vcansend1log` send-message (bottom curve).
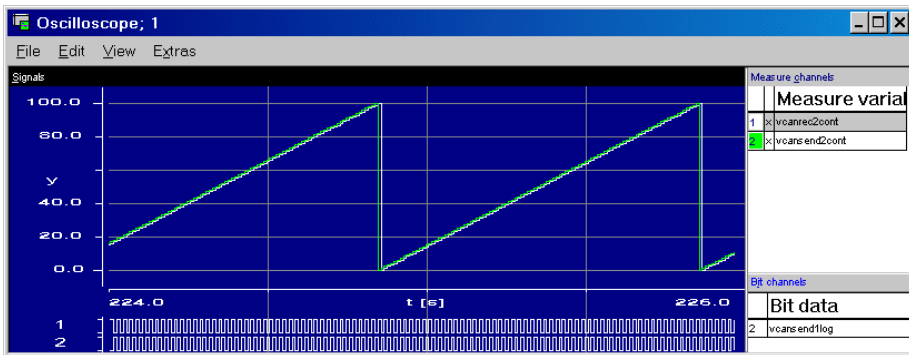


Characteristic and offset of the curves are generated as follows:

- The values for the send-messages are processed (`calcCANLoop` process in the `T10ms` task) and sent (`CanioMSG100_T10ms_HWCL` process in the `T10ms` task) in the 10-ms raster.

- The processed value is read in again via receive-message (`Canio1MSG100_T10ms_HWCF` process in the `T10ms` task), but delayed by one time unit.

The bottom oscillograph shows the `vcansend3cont` send-message (green curve with small steps) and the `vcanrec3cont` receive-message (white curve with large steps).
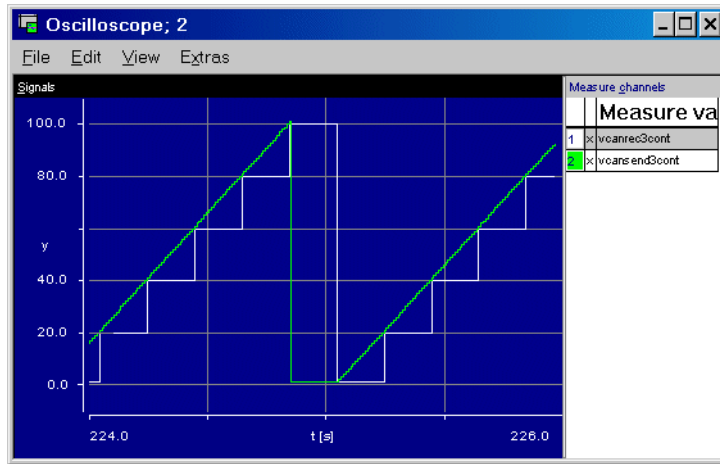
The `vcansend3cont` send-message is processed in the 10-ms raster (`calc-CANLoop` process in the `T10ms` task), which causes the green curve to respond as smoothly as the curve of `vcansend2cont`. The value is sent in the 200-ms raster (`CanioMSG101_T200ms_HWCL` process in the `T200ms` task) and received again in the interrupt mode via receive-message after being triggered by the `Analyze` task. The graduated curve for `vcanrec3cont` responds accordingly.

## 11.3    Tutorial – ES1303

This section describes the integration of an ES1303 in an ASCET project using the RTIO package. Prerequisites for the integration are the steps that are described in chapter 6 "Preparatory Measures". Remember that the mapping between the RTIO channel and the model can only be specified by messages that are declared as *Exported* in the remaining model.

Since the ES1303 board can only receive, but not dend, signals, this example does not include a final experimental demonstration.

### 11.3.1 The ES1303 Hardware

The figure below shows the front panel of the ES1303 analog/digital converter card:



The inputs A to D each bundle four input channels of the ES1303. The input TRIG is used for the trigger signals.

*Input voltage ranges of the ES1303*

The input voltage range of the ES1303 board can be set in the HWC editor either to ±10 V or ±60 V.

*Use of Several ES1303 Boards*

Up to 4 ES1303 boards can be used in one ES1000 system.

### 11.3.2 Sample Project

**How to open the exercise example:**

- In the Component Manager, select the `ASCET_RP\RTIOTutorial\ES1303` folder.
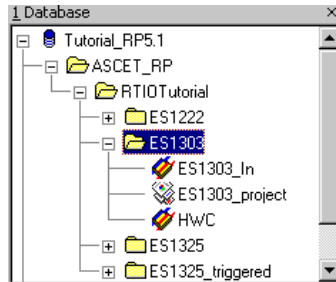


- Open the `ES1303_project` project.
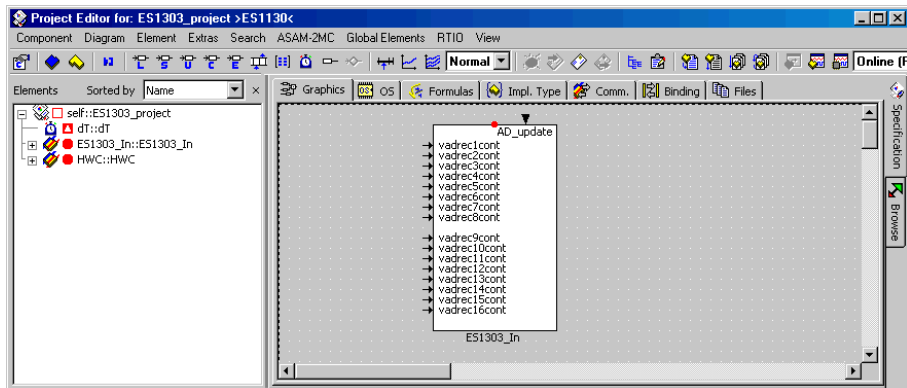
- Click the **Project properties** button.
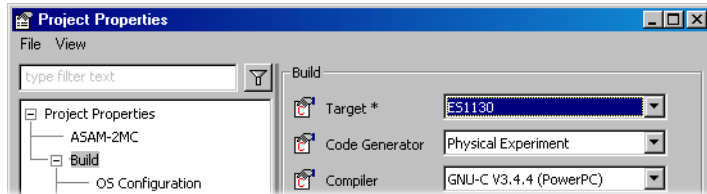
  The "ProjectProperties" window opens.



- In the "Build" node, select the options

  ```
  Target:    >ES1130< or >ES1135<,
  Compiler: GNU-C * (PowerPC).
  ```

**Note**

*Only messages declared as "Exported" are available for the RTIO communi-cation.*

In the sample project, the preparations required for the RTIO integration of the ES1303 (tasks, messages and HWC module) are already completed.

### 11.3.3   Creating the Hardware Configuration

**Note**

*Normally, you must create the C code module* HWC, *and insert it into the project, before you edit the hardware configuration. In the tutorial, however, this step has been performed for you.*

**How to open the HWC editor:**

- In the project editor, choose **RTIO Æ Open Editor**.

  The HWC editor is opened.

### How to create the hardware configuration (HWC):

The hardware must be described as a tree structure in the items list. The HWC item always exists and forms the root of the tree.

- In the HWC editor, select **Edit → Add Item**

  *or*

- click the **Add Item** button.

  The "Add Item" window is displayed.



**Add Item...** always opens the list of available items of the next hierarchy level.

- Select the `ES113x` entry.

  This entry is used to describe the ES1000.x system with integrated ES1130 or ES1135 PowerPC processing node.

- Click **OK**.

  The `ES113x` item is added to the "Items" list.



- In the "Items" list, select the `ES113x` item .

  On the "Globals" tab, the `Init Task` option is given the default task name `Init`. An Init task with the same name exists in the OS editor of the sample project, therefore, no other Init task needs to be selected here.

- Next, select **Hardware → Add Hardware Item** to open the list of available items of the next hierarchy level.
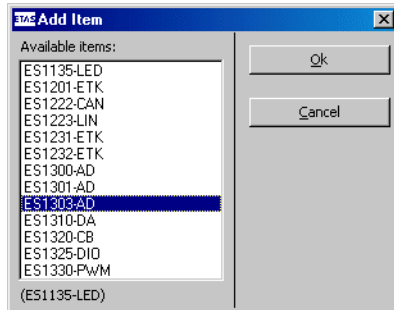


- Select the `ES1303-AD` entry, which is used to describe the ES1303 analog-digital interface.
- Click **OK**.

The item tree for the description of the sample system is now fully specified.

### 11.3.4 HWC Settings for the ES1303

The task settings on the "Globals" tab have to be specified now.

**How to specify the "Globals" settings:**

- In the items list, select the `ES1303-AD` item.
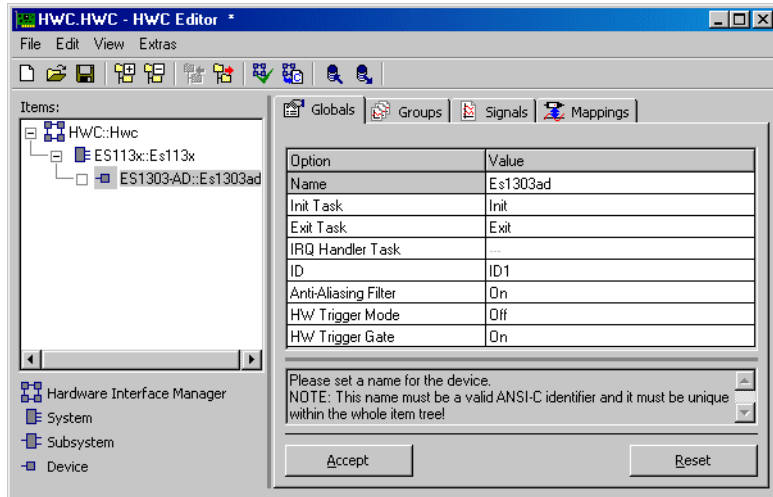
On the "Globals" tab, the task names `Init` and `Exit` are predefined for the `Init Task` and `Exit Task` options. Tasks with these names exist in the OS Editor of the sample project; therefore, no other tasks need to be selected here.

For the `IRQ Handler Task` option, no default name is given. When you use the hardware trigger (`HW Trigger Mode`), you can select a task.

The board number in the ID option is set automatically. Since the ES1303-AD item is the first of its type, the value is set to "`ID1`".

• If you want to limit the bandwidth of the input signal, set the `Anti-Aliasing Filter` option to `On`.

• In the `HW Trigger Mode` option, select the edge of the first trigger signal to start the measurement (e.g., `Falling Edge`).

- Use the `HW Trigger Gate` option to determine whether the second trigger signal is used (`On`) or not (`Off`).

<blockquote>

**Note**

*For polling mode ("Groups" tab, IRQ=No), the options* `HW Trigger Mode` *and* `HW Trigger Gate` *are irrelevant.*

</blockquote>

- Double-click in the empty field next to the `IRQ Handler Task` option.

  The "Task selection" window opens.
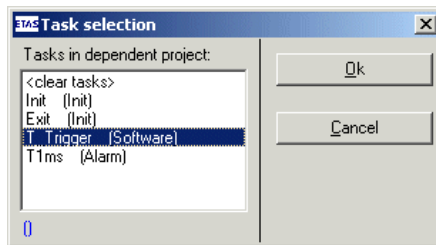


- Select the `T_Trigger` software task.

  The table below shows a sample configuration.

| Option | Value |
|---|---|
| Name | ES1303ad |
| Init Task | Init |
| Exit Task | Exit |
| Trigger Task | T_Trigger |
| ID | ID1 |
| Anti-Aliasing Filter | On |
| Trigger Mode | Falling Edge |
| Trigger Gate | On |

- When you have made all necessary settings, click on the **Accept** button in the HWC editor to save the changes.

The "Groups" tab is used to specify the signal group-specific settings.

The ES1303 provides a fixed signal group (input) for each signal. You select the task assignment, input voltage range and receive mode (interrupt-driven or polling).

**How to specify the "Groups" Settings:**

- Select the "Groups" tab.
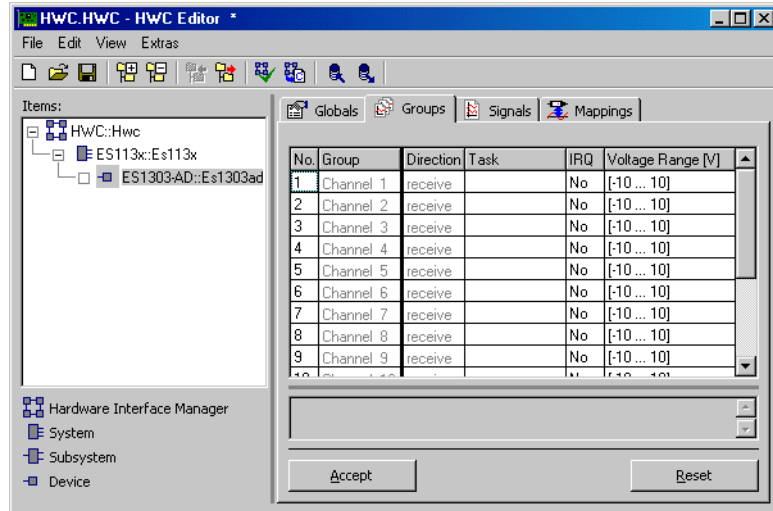


Perform the following steps for each signal to be used.

1. Select the receive type:

- When you want to receive a signal in interrupt mode, double-click into the "IRQ" field of the respective line.

   A combo box appears.

- From the combo box, select Yes.

   The modified setting is shown in the "IRQ" field. At the same time, the "Task" field is reset and blocked.

| No. | Group | Direction | Task | IRQ | Voltage Range [V] |
|-----|-----------|-----------|------|-----|-------------------|
| 1 | Channel 1 | receive | --- | Yes | [-10 ... 10] |
| 2 | Channel 2 | receive | | No | [-10 ... 10] |

- If you want to receive a signal in polling mode, use the default setting No of the "IRQ" column.

2. Assign a task (only for polling mode):

- In the desired line, double-click into the "Task" field to open the "Task selection" window.

- In the task selection list, select the T1ms task. This task should be used for data transfer.
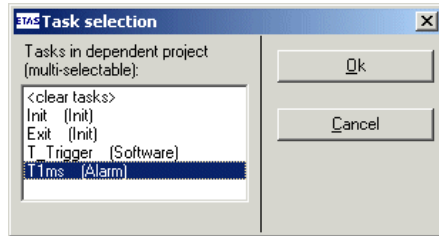


You can also select more than one task in the task selection list. For the ES1303, these are generally the alarm tasks.

- Click **OK**.

The selected task(s) are shown in the "Task" column.

3. Select the voltage range:
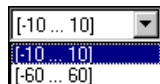
- To determine the input voltage range of a signal, double-click into the "IRQ" field of the respective line.

A combo box appears.



- Select a value for the voltage range from the combo box.

- The selection is shown in the "Voltage Range" column.

| No. | Group | Direction | Task | IRQ | Voltage Range [V] |
|-----|-----------|-----------|------|-----|-------------------|
| 1 | Channel 1 | receive | --- | Yes | [-10 ... 10] |
| 2 | Channel 2 | receive | | No | [-60 ... 60] |

- When you have made all necessary settings, click on the **Accept** button in the HWC editor to save the changes.

The "Signals" tab contains signal-specific settings. Since the ES1303 card has no special options here, no actions have to be taken on this tab. You can select the tab to view the current settings.
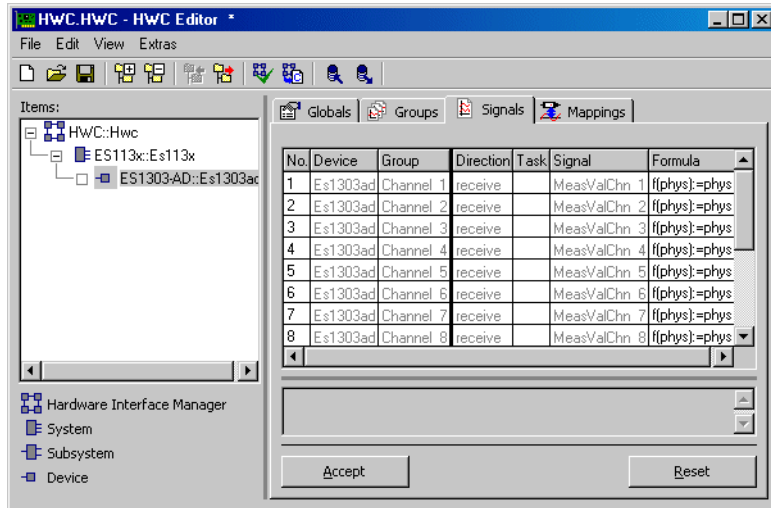


The "Mappings" tab is used to specify the mapping between the signals and ASCET messages. Click the desired cell in the "ASCET Message" column to open a selection dialog for the located messages of the residual model. The selection dialog lists only messages featuring the Exported attribute set and whose transmission direction matches that of the signal group (send or receive), i.e.:

Direction = receive → Receive-Messages

Direction = send → Send-Messages (irrelevant for ES1303)

**How to specify "Mappings" settings:**

- Select the "Mappings" tab.
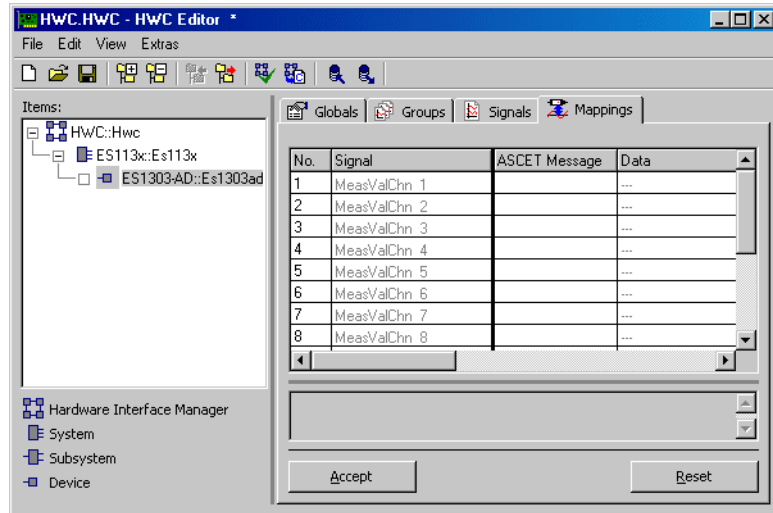


- In the desired line, double-click on the "ASCET Message" field to open the "Message selection" window.
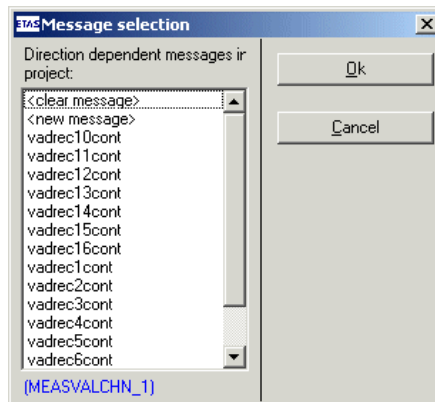
- Use the "Message selection" window to map the sixteen receive messages one by one to the corresponding signals.
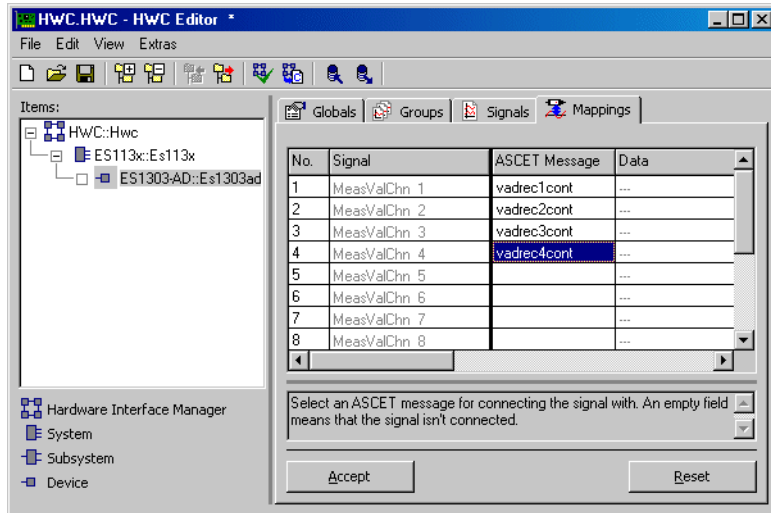


- In the HWC editor, click the **Accept** button.

### 11.3.5 Saving the Hardware Configuration

The created hardware configuration can be saved in the File container of the project, or as a DOS file (`*.HWX` extension). The procedure is described in chapter 11.2.5 on page 308.

> **Note**
>
> *The solution of the example is stored under the name of `ES1303.hwx`. Make sure that you do not overwrite this file.*

### 11.3.6 Generating Code for the HWC Module

Subsequently, the generating sequence for the HWC module has to be started. The procedure is described in chapter 11.2.6 on page 309.

### 11.3.7    Final Actions

Because all RTIO-specific actions are now finished, you can close the HWC editor. The next step is to start the regular code generation for the experimental target from within the project editor.

> **Note**
>
> *It is not recommended to include the HWC module in the graphical display of the Project Editor, because this module changes with each RTIO generation process. This would result in an unfavorable representation of the HWC module.*

**How to generate code for the experimental target:**

- In the project editor, choose **Component → Build** to generate code for the entire project.
- Choose **Component → View Generated Code** to look at the generated code.

**How to experiment online:**

- In the project editor, select `Online (RP)` from the "Experiment Target" combo box.

  `Offline (RP)` is intended for offline experiments on the Target.
- Select **Component → Open Experiment**.
- In the Environment Browser, select the `Tutorial` environment.
- In the "Physical Experiment" window, select **Experiment → Start ERCOS**.
- In the "Physical Experiment" window, select **Experiment → Start Measurement**.

  The analog measured values of the ES1303 are displayed on an ASCET oscilloscope.

## 11.4 Tutorial – ES1325 (without Trigger)

The ES1325 sample project without using the trigger contains models of all important applications of the board. The models are shown in graphic hierarchies to keep the block diagram clear.
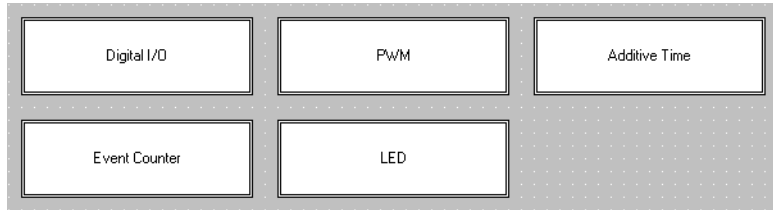


**Fig. 11-4** ES1325 – Model Overview

**Digital I/O:** Input channel 3 is read in and transferred to output channel 3. The signal is inverted by assigning the "active" state to the low level on reading, whereas on sending, the high level is assigned the "active" state (see "Active State" on page 255).
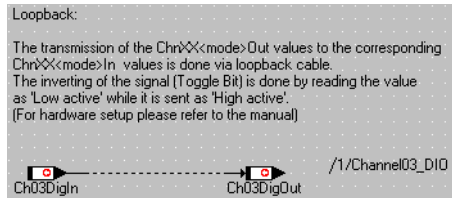


**Fig. 11-5** ES1325 – Digital I/O

**PWM:**   Channel 4 has a static PWM signal with fixed active and inactive times. Channel 5 has a dynamic PWM signal whose duty cycle is determined by a sawtooth signal.
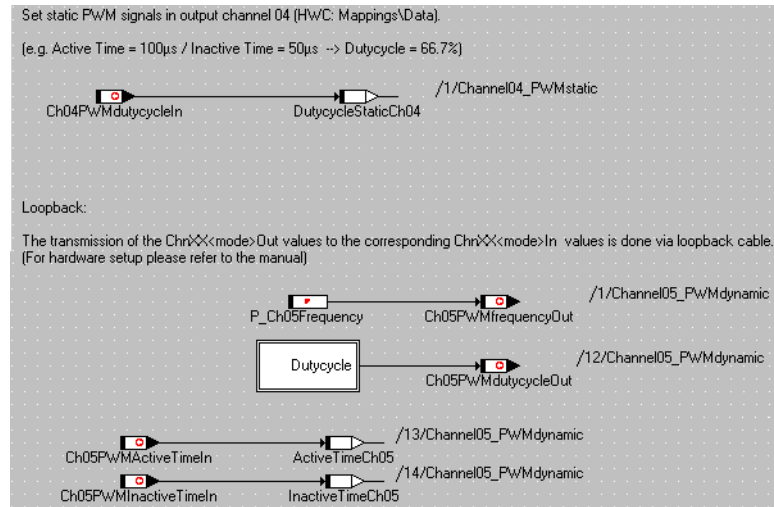


**Fig. 11-6**   ES1325 – PWM

**Additive Time:**   The PWM output signal of channel 5 is also transmitted by output channel 6. In this case it is useful for additive time measuring, i.e. the totaling of the active phases of the signal during task runtime.
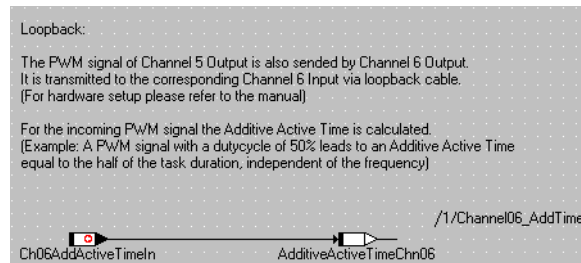


**Fig. 11-7**   ES1325 – Additive Time

**Event Counter:** Channel 7 counts every rising edge of an event; channel 8 counts every edge of an event. Both channels are controlled by the `P_Ch0708DigOut` parameter.
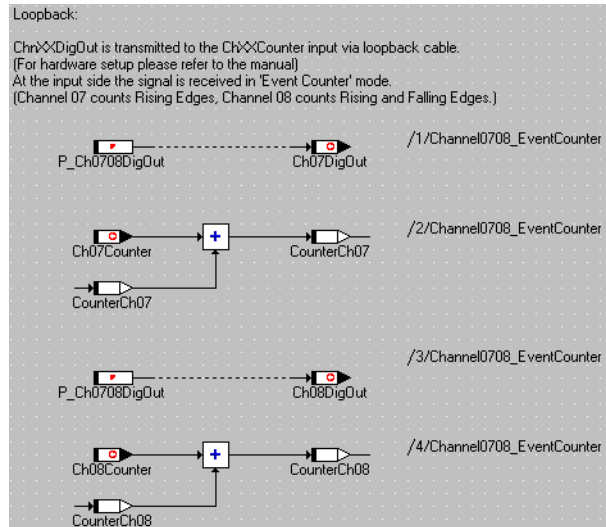


**Fig. 11-8** ES1325 – Event Counter

**LED:** This is where the input signals for LEDs 1 to 3 are set. LED1 is controlled by a parameter of type `log`, LED2 via a parameter of type `sdisc`, and LED3 via a parameter of type `cont`.

---

**Note**

`cont` *and* `sdisc` *values can be used to control the LEDs, but we **do not recommend** this at all.*
*In the interests of good modeling/programming style, mapping to a logical variable, e.g. with case differentiation, is to be preferred.*

---

LED4 is controlled via the `Ch03DigOut` message.
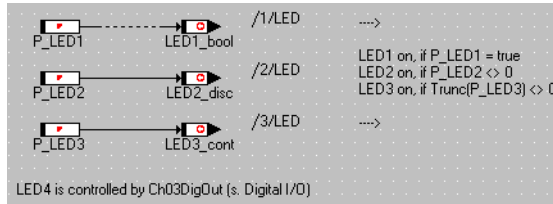


**Fig. 11-9** ES1325 – PWM

## 11.4.1 The ES1325 Board

The following figure shows the front panel of the ES1325 Board:
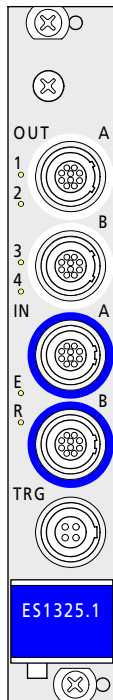


Ports Out-A and OUT-B each contain 8 output channels; ports IN-A and IN-B each contain 8 input channels. TRG contains 2 trigger inputs. LEDs 1 – 4 are available for display purposes.

*Connections*

To ensure that the experiment works, make sure you have the right wiring. In the tutorial you will be using ports Out-A, IN-A and (in chapter 11.5) TRG; the wiring specified here applies to both tasks on the ES1325.
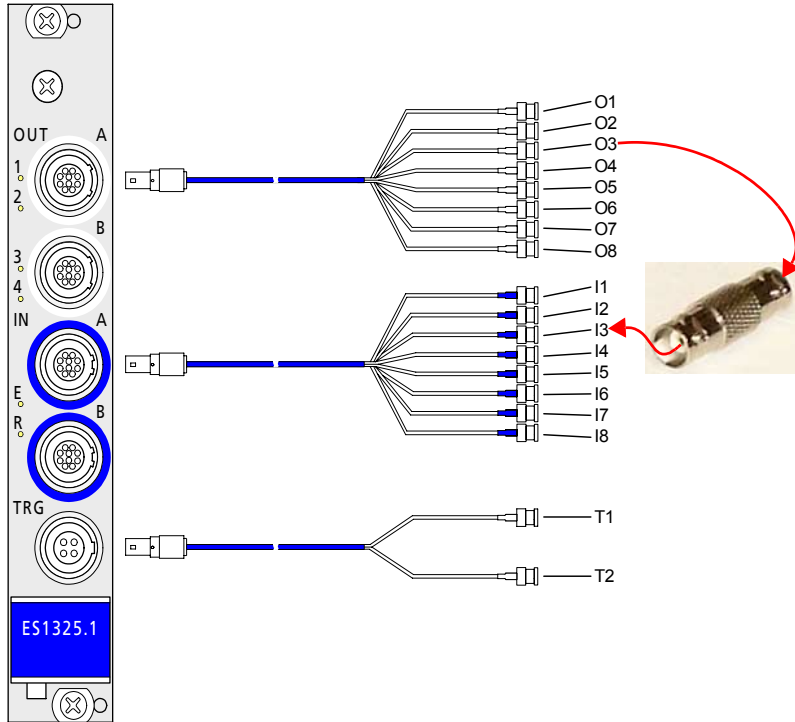


**Fig. 11-10**   Wiring of the ES1325

The following table specifies which outputs are connected with which inputs. Inputs I1 and I2 remain empty.

| **Output** | O1 | O2 | O3 | O4 | O5 | O6 | O7 | O8 |
|---|---|---|---|---|---|---|---|---|
| **Input** | | | I3 | I4 | I5 | I6 | I7 | I8 |
| **Trigger** | T1 | T2 | | | | | | |

---

**Note**

*The user has to provide the connecting pieces for the inputs and outputs.*

---

*Working with Several ES1325 Boards*

Up to 4 ES1325 Boards can be operated in an ES1000.

## 11.4.2 Sample Project

**To open the exercise example:**

- Select the `ASCET_RP\RTIOTuto-rial\ES1325` folder from the Component Manager.
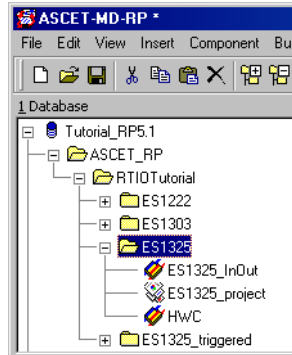


- Select the `ES1325_project` project.
- Open the project.
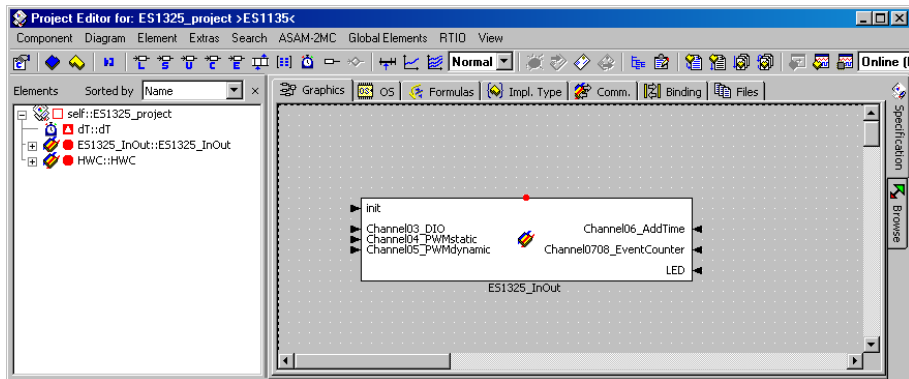
- Click the **Project properties** button.

  The "ProjectProperties" window opens.



- In the "Build" node, select the options

  ```
  Target:   >ES1130< or >ES1135<,
  Compiler: GNU-C * (PowerPC).
  ```
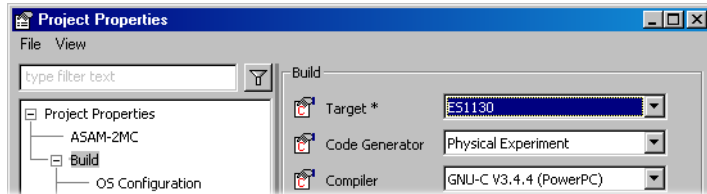
**Note**

*Only messages declared as "Exported" are available for RTIO communica-tion.*

11.4.3    Creating the Hardware Configuration

**Note**

*Normally you have to create the C-Code module* HWC *and link it to the project before you edit the hardware configuration. This step has already been taken care of in the tutorial.*

**To open the HWC Editor:**

- Select **RTIO → Open Editor** in the Project Editor.

  The HWC Editor opens.

**To create the hardware configuration (HWC):**

The hardware has to be described as a tree-like structure in the items list. The HWC item is always available and forms the roots of the tree.

- In the HWC Editor select **Edit** → **Add Item**

*or*

- click the **Add Item** button.

  The "Add Item" window is displayed.



**Add Item** always opens the list of available items of the next hierarchy level.

- Select `ES113x`.

  This is used to describe the ES1000.x system with an integrated ES1130 or ES1135 PowerPC computer node.

- Click **OK**.

  `ES113x` is added to the "Items" list.



- Select `ES113x` from the "Items" list.

  In the "Globals" tab, the task name `Init` is specified for the `Init Task` option. In the OS-Editor of the sample project, there is an init task with the same name which means that no other init task has to be selected here.

**To link and set up the ES1325:**

- Open the list of available items of the next hierarchy level using **Edit → Add Item**.



- Select `ES1325-DIO`.

  This is used to describe the ES1325 interfaces.

- Click **OK**.

  `ES1325-DIO` is added to the "Items" list.

- Select `ES1325-DIO` from the "Items" list.

  In the "Globals" tab, the task names `Init` and `Exit` are specified for the options `Init Task` and `Exit Task`. In the OS-Editor of the sample project, there are init tasks with the same name which means that no other tasks have to be selected here.

  The board number is automatically set in the `ID` option. As the `ES1325-DIO` item is the first of this type, `"ID1"` is set.

- Accept the settings using **Accept**.

**To create devices:**

- Select `ES1325-DIO` from the "Items" list.

- Open the list of available items of the next hierarchy level using **Edit → Add Item**.
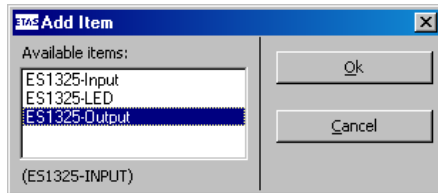


- Select `ES1325-Input` and click **OK**.

  This is used to describe the input channels of the ES1325.

- Also add the devices `ES1325-Output` and `ES1325-LED`.

  `ES1325-Output` is used to describe the output channels of the ES1325 and `ES1325-LED` to describe the LEDs.

  For this example, use the default settings in the "Globals" tab for all three devices.

The item tree for the description of the sample system is now completely specified.
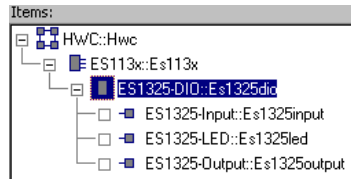


### 11.4.4 Making HWC Settings for the ES1325

The inputs and outputs as well as the LEDs now have to be configured. The outputs provide the signals which the inputs receive. The LEDs are used for display purposes.

**"Globals" Tab:** You do not have to make any settings for any of the three devices in the "Globals" tab.

*Outputs –* `ES1325-Output` *Device*

**"Groups" Tab:** The signal-group-specific settings are made in the "Groups" tab.

The `ES1325-Output` device has a specified signal group for every output signal (`Ch<n>Gen`, `<n>` = 01 – 16). Here, you set the mode, task assignment, the generated signals as well as the levels and edges for each group used.

**To make settings in the "Groups" tab:**

- Select the "Groups" tab.
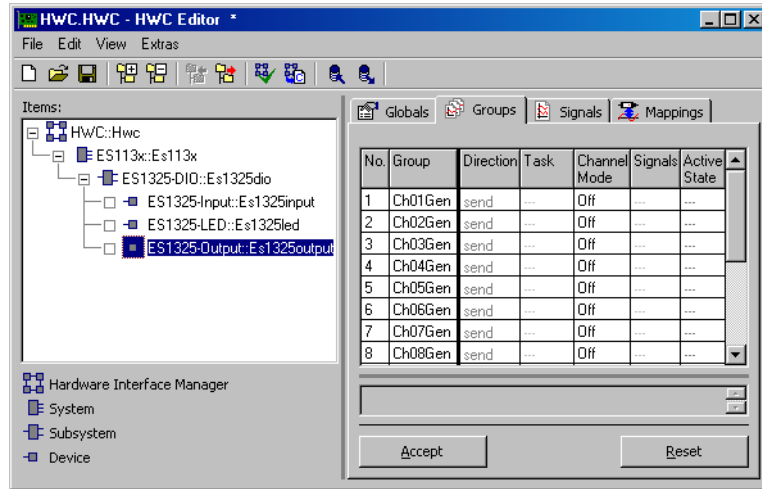


Execute the following steps for signal groups 3 to 8.

1. Selecting the channel mode



- Select the following modes for the different groups in the "Channel Mode" column:

| Group | Channel Mode |
|---|---|
| Ch03Gen, Ch07Gen, Ch08Gen | Digital Output |
| Ch04Gen, Ch05Gen, Ch06Gen | PWM Periodic Output |

The default value for the relevant mode is entered in the "Active State" column.
In the "Task" column, you can now assign a task to the groups used in which signal transfer is to take place.

2. Assigning a task

- Double-click the "Task" column in the relevant row to open the "Task selection" dropdown list.
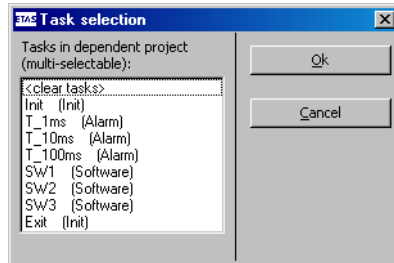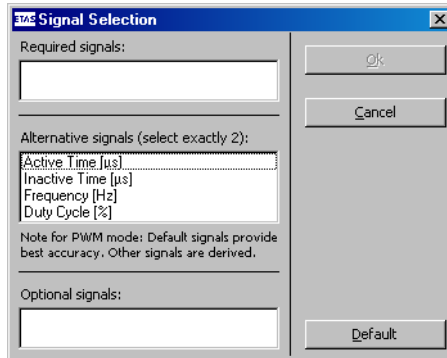


- Select the following tasks for data transfer in the task selection list.

| Group | Task |
|-------|------|
| Ch03Gen, Ch04Gen | T_100ms |
| Ch05Gen, Ch06Gen, Ch07Gen, Ch08Gen | T_10ms |

- Click **OK**.

  The selected tasks are displayed in the "Task" column.

3. Selecting signals

- Double-click the "Signals" column in the relevant row to open the "Signal Selection" window.



For a description of the signals, please refer to the section "Signals" on page 260.

- Select the following signals.

| Group | Signals |
|-------|---------|
| Ch03Gen, Ch07Gen, Ch08Gen | Accept default setting |
| Ch04Gen | Active Time [µs], Inactive Time [µs] |
| Ch05Gen, Ch06Gen | Frequency [Hz], Duty Cycle [%] |

- Confirm your selection with **OK**.

4. Selecting levels

As all groups use the default setting High, you do not have to make any changes to the "Active State" column.

- Once you have made all the necessary settings, click **Accept** in the HWC Editor to save the settings.

Once you have set up the signal groups, the tab should look as follows:



**"Signals" tab:** The signals you generated in the "Groups" tab are contained in this tab.



You can edit the names and formulae of the signals in this tab. *No* changes are necessary, however, for the sample project.
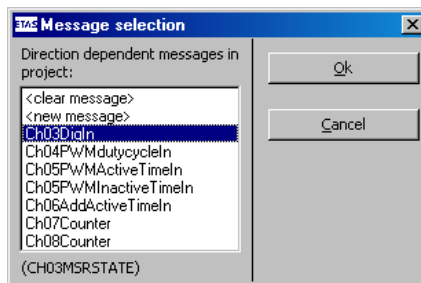
**"Mappings" Tab:** The signals and the ASCET messages from the project are assigned to each other in this tab. A selection dialog opens when you click the required cell in the "ASCET Message" column. This only contains messages which have the attribute *Exported* and correspond to the transfer direction of the signal group (here: *send*); i.e.:

- Direction = send → send messages



**To assign an ASCET message manually:**

- Select the "Mappings" tab.
- Double-click the "ASCET Message" column in the relevant row.
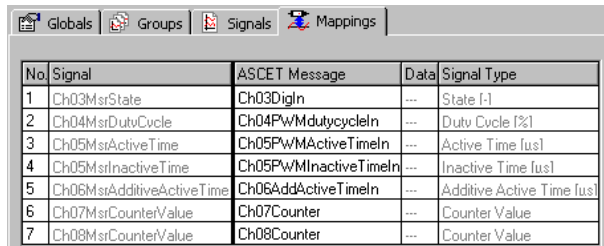
  The "Message selection" window opens.



It contains all send messages from the ASCET project.

- Select the following ASCET messages for the signals.

| Signal | ASCET Message |
|---|---|
| Ch03GenState | Ch03DigOut |
| Ch05GenFrequency, Ch06GenFrequency | Ch05PWMfrequencyOut |
| Ch05GenDutyCycle, Ch06GenDutyCycle | Ch05PWMdutycycleOut |
| Ch07GenState | Ch07DigOut |
| Ch08GenState | Ch08DigOut |

No message is assigned to the signals Ch04GenActiveTime and Ch04GenInactiveTime; instead a fixed value is assigned manually.

- Click **OK**.
- Then click **Accept** in the HWC Editor to save the settings.

**To enter a value manually:**

If no ASCET message has been assigned to a signal, you can enter a fixed value in the "Data" column.

- Double-click the "Data" column in the relevant row.

  The field becomes the input box.
- Enter the following values.

| Signal | Data |
|---|---|
| Ch04GenActiveTime | 100.0 |
| Ch04GenInactiveTime | 50.0 |

Once you have assigned messages or values to all signals, the tab should look as follows:



*Inputs –* `ES1325-Input` *Device*

**"Groups" Tab:**   The signal-group-specific settings are made in the "Groups" tab.

The `ES1325-Input` device has a specified signal group for every input signal (`Ch<n>Msr, <n>` = 01 – 16). Here, you set the mode, task assignment, the generated signals as well as the levels and edges for each group used.

**To make settings in the "Groups" tab:**

- Select the "Groups" tab.
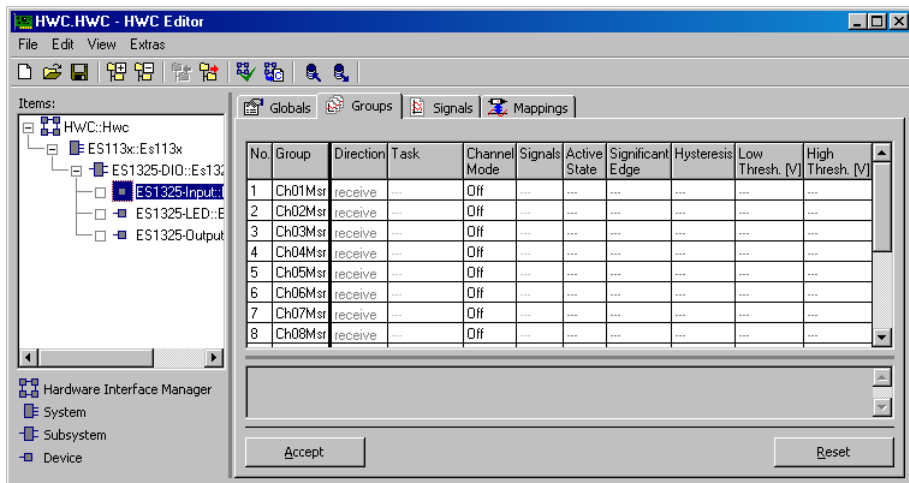


Execute the following steps for signal groups 3 to 8.

1.  Selecting the channel mode

    - Double-click the "Channel Mode" column in the relevant row.

      A dropdown list opens.

    

    - Select the following modes for the different groups:
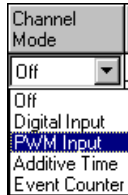
      | Group | Channel Mode |
      | --- | --- |
      | Ch03Msr | Digital Input |
      | Ch04Msr, Ch05Msr | PWM Input |
      | Ch06Msr | Additive Time |
      | Ch07Msr, Ch08Msr | Event Counter |

      The default values for the relevant modes are entered in the columns "Active State", "Significant Edge", "Hysteresis", "Low Thresh. [V]" and "High Thresh. [V]".
      In the "Task" column, you can now assign a task to the groups used in which signal transfer is to take place.

2.  Assigning a task

    - Double-click the "Task" column in the relevant row to open the "Task selection" dropdown list.
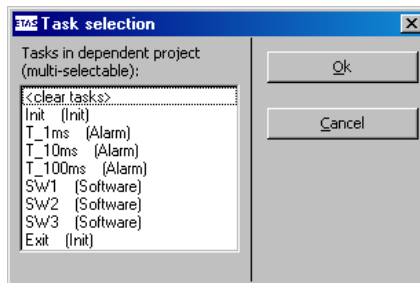
• Select the following tasks in which data transfer is to take place from the task selection list.

| Group | Task |
|-------|------|
| Ch03Msr, Ch04Msr | T_100ms |
| Ch05Msr, Ch06Msr, Ch07Msr, Ch08Msr | T_10ms |

• Click **OK**.

The selected tasks are displayed in the "Task" column.

3. Selecting signals

• Double-click the "Signals" column in the relevant row to open the "Signal Selection" window.
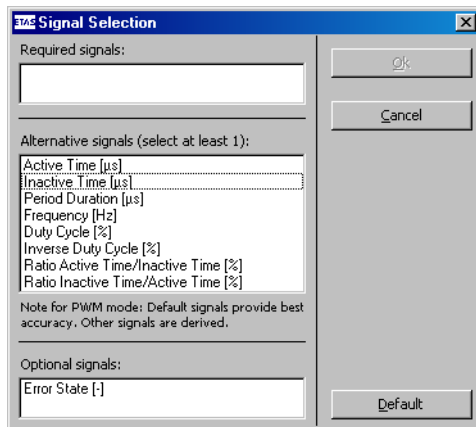


In this window, you specify which signals are generated for the group. For a description of the signals, please refer to the section "Signals" on page 253.

- Select the following signals.

| Group | Signals |
|---|---|
| `Ch04Msr` | `Duty Cycle [%]` |
| `Ch05Msr` | `Active Time [µs],` `Inactive Time [µs]` |
| `Ch03Msr, Ch06Msr,` `Ch07Msr, Ch08Msr` | Accept default setting |

- Confirm your selection with **OK**.

4. Selecting levels

- Double-click the "Active State" column in the relevant row (see page 255) to assign the active state to a level of the input signal.

  A dropdown list opens.

- Select the following states.

| Group | Active State |
|---|---|
| `Ch03Msr` | `Low` |
| `Ch04Msr – Ch08Msr` | `High` |

  The selection is displayed in the "Active State" field.

5. Selecting an edge (only `PWM Input` and `Event Counter` modes)

- Double-click the "Significant Edge" column in the relevant row (see page 255) to assign an event to an edge.

  A dropdown list opens.

- Select the following options.

| Group | Significant Edge |
|---|---|
| `Ch04Msr, Ch05Msr,` `Ch07Msr` | `Inactive-` `Active` |
| `Ch08Msr` | `Both` |

  The selection is displayed in the "Significant Edge" field.

- Once you have made all the necessary settings, click **Accept** in the HWC Editor to save the settings.

Once you have set up the signal groups, the tab should look as follows:

| No. | Group | Direction | Task | Channel Mode | Signals | Active State | Significant Edge | Hysteresis | Low Thresh. [V] | High Thresh. [V] |
|-----|-------|-----------|------|--------------|---------|--------------|------------------|------------|-----------------|------------------|
| 1 | Ch01Msr | receive | --- | Off | --- | --- | --- | --- | --- | --- |
| 2 | Ch02Msr | receive | --- | Off | --- | --- | --- | --- | --- | --- |
| 3 | Ch03Msr | receive | T_100ms | Digital Input | [Select] | Low | --- | TTL | 1.728 | 2.304 |
| 4 | Ch04Msr | receive | T_100ms | PWM Input | [Select] | High | Inactive-Active | TTL | 1.728 | 2.304 |
| 5 | Ch05Msr | receive | T_10ms | PWM Input | [Select] | High | Inactive-Active | TTL | 1.728 | 2.304 |
| 6 | Ch06Msr | receive | T_10ms | Additive Time | [Select] | High | --- | TTL | 1.728 | 2.304 |
| 7 | Ch07Msr | receive | T_10ms | Event Counter | [Select] | High | Inactive-Active | TTL | 1.728 | 2.304 |
| 8 | Ch08Msr | receive | T_10ms | Event Counter | [Select] | High | Both | TTL | 1.728 | 2.304 |

**"Signals" Tab:** The signals you generated in the "Groups" tab are contained in this tab.



You can edit the names and formulae of the signals in this tab. *No* changes are necessary, however, for the sample project.

**"Mappings" Tab:** The signals and the ASCET messages from the project are assigned to each other in this tab. A selection dialog opens when you click the required cell in the "ASCET Message" column. This only contains messages which have the attribute *Exported* and correspond to the transfer direction of the signal group (here: *receive*); i.e.:

• Direction = receive → receive messages



**To assign an ASCET message manually:**

• Select the "Mappings" tab.
• Double-click the "ASCET Message" column in the relevant row.

The "Message selection" window opens.



It contains all receive messages from the ASCET project.

- Select the following ASCET messages for the signals.

| Signal | ASCET Message |
|---|---|
| Ch03MsrState | Ch03DigIn |
| Ch04MsrDutyCycle | Ch04PWMdutycycleIn |
| Ch05MsrActiveTime | Ch05PWMActiveTimeIn |
| Ch05MsrInactiveTime | Ch05PWMInactiveTimeIn |
| Ch06MsrAdditiveActiveTime | Ch06AddActiveTimeIn |
| Ch07MsrCounterValue | Ch07Counter |
| Ch08MsrCounterValue | Ch08Counter |

- Click **OK**.
- Then click **Accept** in the HWC Editor to save the settings.

Once you have assigned messages to all signals, the tab should look as follows:

| No. | Signal | ASCET Message | Data | Signal Type |
|---|---|---|---|---|
| 1 | Ch03MsrState | Ch03DigIn | --- | State [-] |
| 2 | Ch04MsrDutyCycle | Ch04PWMdutycycleIn | --- | Duty Cycle [%] |
| 3 | Ch05MsrActiveTime | Ch05PWMActiveTimeIn | --- | Active Time [us] |
| 4 | Ch05MsrInactiveTime | Ch05PWMInactiveTimeIn | --- | Inactive Time [us] |
| 5 | Ch06MsrAdditiveActiveTime | Ch06AddActiveTimeIn | --- | Additive Active Time [us] |
| 6 | Ch07MsrCounterValue | Ch07Counter | --- | Counter Value |
| 7 | Ch08MsrCounterValue | Ch08Counter | --- | Counter Value |

*LEDs –* `ES1325-LED` *Device*

**"Groups" Tab:**   The signal-group-specific settings are made in the "Groups" tab.

The `ES1325-LED` device has a specified signal group (`GroupLED`). This is where you set the task assignment for this group.

**To make settings in the "Groups" tab:**
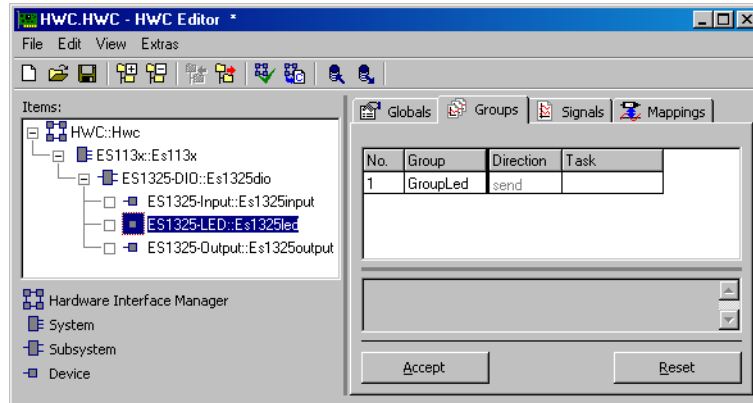
- Select the "Groups" tab.



- Double-click the "Task" column in the relevant row to open the "Task selection" dropdown list.
- Select the `T_1ms (Alarm)`.task from the task selection list.
- Click **OK**.

  The task is displayed in the "Task" column.
- Click **Accept** in the HWC Editor to save the settings.

The tab should now look as follows:

**"Signals" Tab:** This tab contains the signals which belong to the signal group GroupLED. There is one signal for each LED on the board.



You can edit the names and formulae of the signals in this tab. *No* changes are necessary, however, for the sample project.

**"Mappings" Tab:** The signals and the ASCET messages from the project are assigned to each other in this tab. A selection dialog opens when you click the required cell in the "ASCET Message" column. This only contains messages which have the attribute *Exported* and correspond to the transfer direction of the signal group (here: *send*); i.e.:

- Direction = send → send messages



**To assign an ASCET message manually:**

- Select the "Mappings" tab.

- Double-click the "ASCET Message" column in the relevant row.

  The "Message selection" window opens.



  It contains all send messages from the ASCET project.

- Select the following ASCET messages for the signals.

  | Signal | ASCET Message |
  |--------|---------------|
  | LED1   | LED1_bool     |
  | LED2   | LED2_disc     |
  | LED3   | LED3_cont     |
  | LED4   | Ch03DigOut    |

- Click **OK**.

- Then click **Accept** in the HWC Editor to save the settings.

The tab should now look as follows:
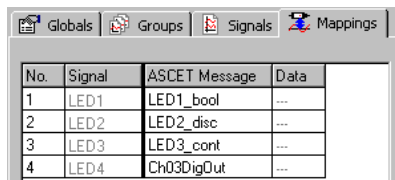
### 11.4.5 Saving the Hardware Configuration

The created hardware configuration can be saved in the file container of the project or as a DOS file (extension `*.HWX`). How this is done is explained in section 11.2.5 on page 308.

> **Note**
>
> *The solution of the example is saved as* `ES1325.hwx`*. Make sure you do not overwrite this file.*

### 11.4.6 Creating Code for the HWC Module

The generation sequence for the HWC module then has to be started. How this is done is explained in section 11.2.6 on page 309.

### 11.4.7 Experimenting with the Sample Project

All RTIO-specific actions have now been completed; this means that the HWC Editor can be closed. A further subsequent step is the normal code generation for the experimental target which is started from the Project Editor.

> **Note**
>
> *We would recommend that you do not include the HWC module in the graphic representation of the project editor as this module is modified in every RTIO generation process. This leads to an unfavorable graphic representation of the HWC module.*

**Code generation for the experimental target:**

- In the Project Editor, select **Component** → **Build** to generate code for the entire project.
- Select **Component** → **View Generated Code** to view the generated code.

**To experiment online:**

- Connect the inputs and outputs of the ES1325 in accordance with the section "Connections" on page 333, if this has not already taken place, and switch on the power supply of the ES1000.x.

- In the project editor, select `Online (RP)` from the "Experiment Target" combo box.

  `Offline (RP)` is intended for offline experiments on the Target.

- Select **Component** → **Open Experiment**.

  The "Physical Experiment" window and the predefined experiment environment consisting of five oscilloscopes, a numeric display and four calibration windows open.
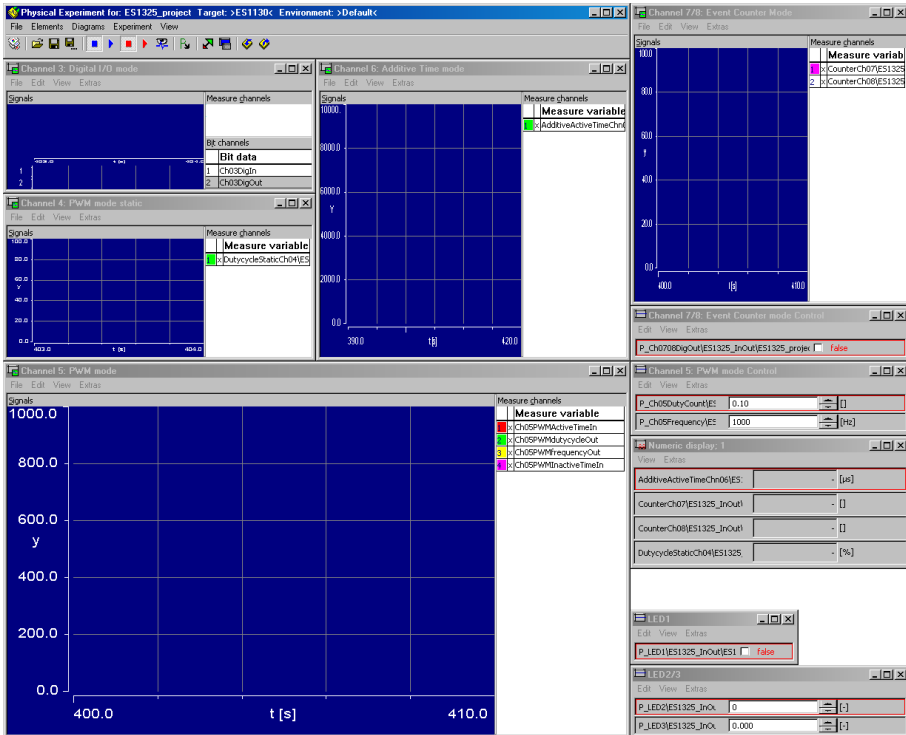


- Select **Experiment** → **Start ERCOS** in the "Physical Experiment" window

  *or*

- click the **Start ERCOS** button.

  The operating system is launched; the model is run.

- Select **Experiment** → **Start Measurement** in the "Physical Experiment" window

*or*

- click the **Start Measurement** button.

  The values of the ASCET messages are displayed in the oscilloscopes and in the numerical display.

The displays and calibration possibilities of the individual model blocks are described below.

**Digital I/O:** The "Channel 3: Digital I/O" oscilloscope shows the messages `Ch03DigIn` and `Ch03DigOut`.
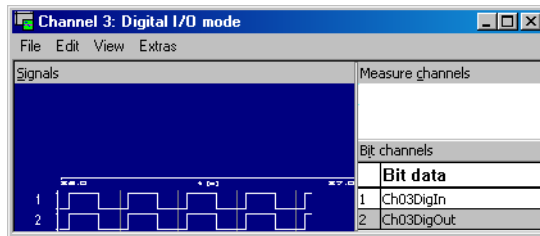


**PWM:** The "Channel 4: PWM mode static" oscilloscope shows the `DutycycleStaticCh04` variable which records the duty cycle of the PWM signal. As active and inactive time are both specified as fixed values, the value of the variables is constant.
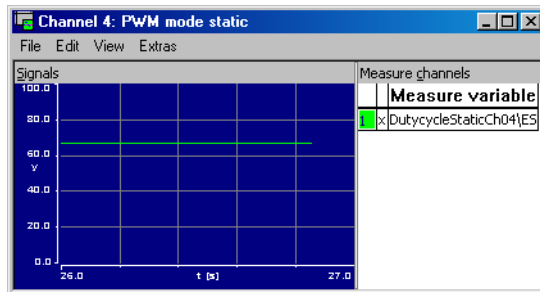
The "Channel 5: PWM mode" oscilloscope shows the input messages `Ch05PWMActiveTimeIn` (red curve) and `Ch05PWMInactiveTimeIn` (violet curve) which contain the active and inactive time of the PWM signal. The output messages `Ch05PWMdutycycleOut` (green curve) and `Ch05PWMfrequencyOut` (yellow curve) are displayed which contain the frequency (in Hz) and duty cycle (in %) of the PWM signal. When the experiment is started, the Y-axis of `Ch05PWMActiveTimeIn` is displayed.
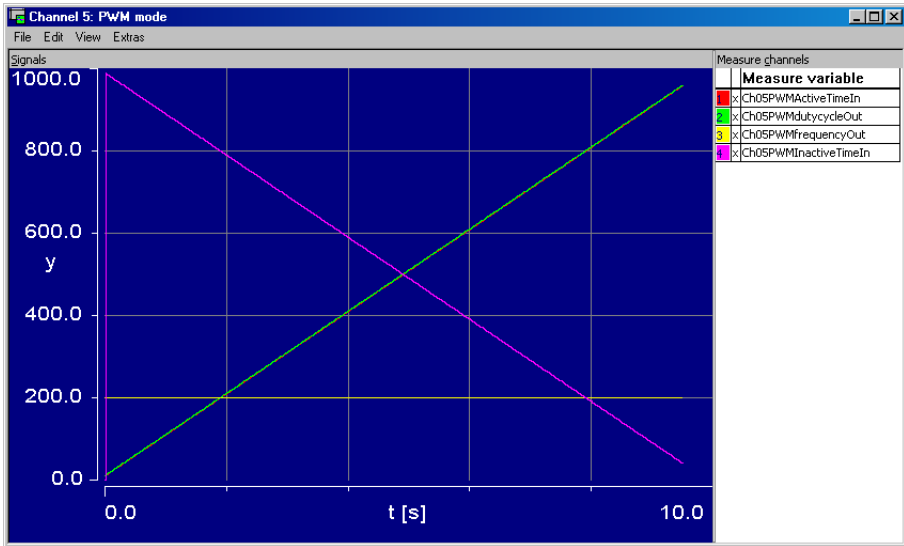


If you modify the `P_Ch05Frequency` parameter at the bottom of the "Channel 5: PWM mode Control" calibration window, the frequency changes and thus the active and inactive time. The duty cycle remains the same.

If you modify the parameter `P_Ch05DutyCount` at the top of the "Channel 5: PWM mode Control" calibration window, the course of the duty cycle changes and thus the active and inactive time. The frequency remains the same.
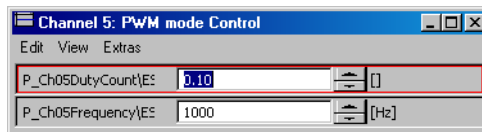
At point 1 in the following diagram, a frequency of 1000 Hz was set to 2000 Hz so that the curves for `Ch05PWMActiveTimeIn`, `Ch05PWMInactiveTimeIn` and `Ch05PWMfrequencyOut` show discontinuity. `P_Ch05DutyCount` was not changed; the curve for `Ch05PWMdutycycleOut` retains its slope.
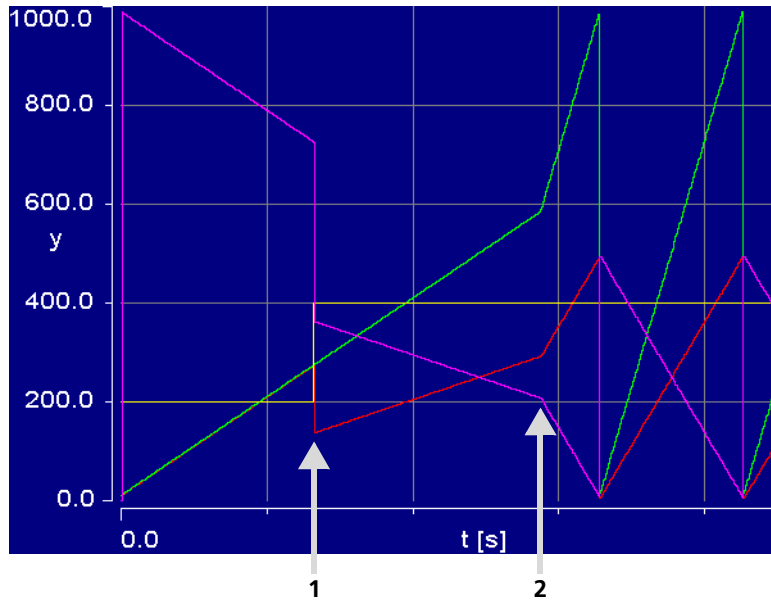


At point 2, the `P_Ch05DutyCount` parameter was set from `0.1` to `0.5` so that the curves for `Ch05PWMdutycycleOut`, `Ch05PWMActiveTimeIn` and `Ch05PWMInactiveTimeIn` break off. The frequency remains unchanged at this point.

**Additive Time:**  The "Channel 6: Additive Time mode" oscilloscope shows the `AdditiveActiveTimeChn06` variable which accepts the additive active time of the signal.
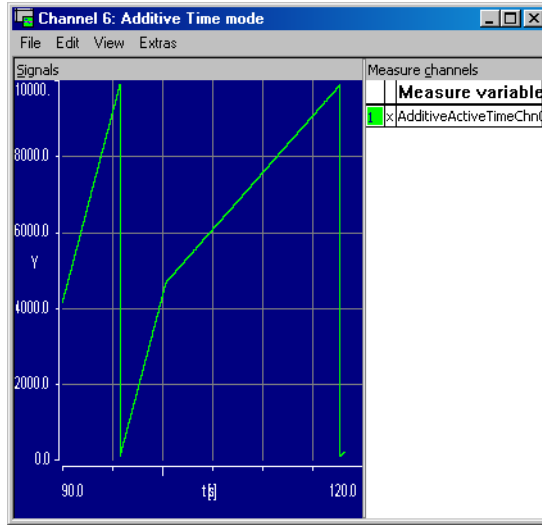


You can influence the slope of the sawtooth by modifying the `P_Ch05DutyCount` parameter at the top of the "Channel 5: PWM mode Control" calibration window. A value of `0.1` was set for the first part of the curve shown and for the second a value of `0.01`.
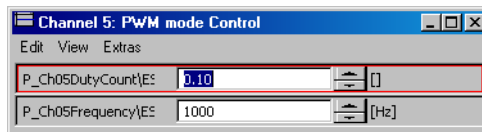


The `AdditiveActiveTimeChn06` signal is independent of the frequency set as the additive active time remains unaffected by it during task runtime.

**Event Counter:** The "Channel 7/8: Additive Time mode" oscilloscope shows the variables `CounterCh07` (lower curve) and `CounterCh08` (upper curve) which accept the additive active time of the signal.
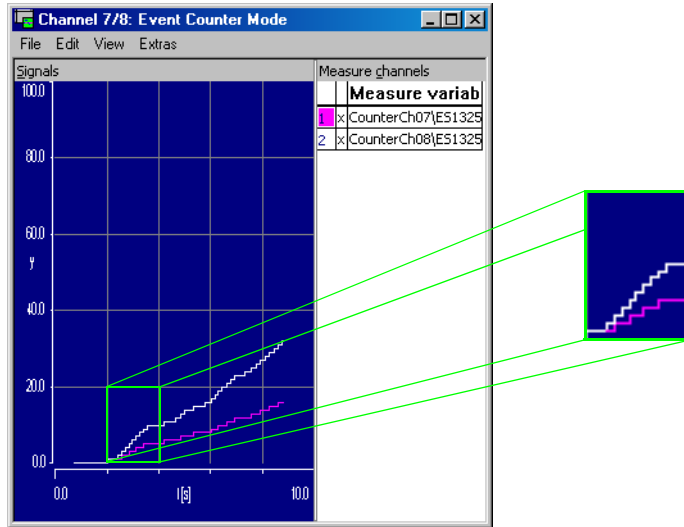


You can add an increment to both variables by setting the `P_Ch0708DigOut` parameter in the "Channel 7/8: Event Counter mode Control" calibration window alternately to `true` and `false`.
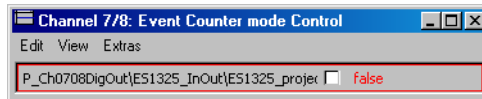


`CounterCh07` grows half as quickly as `CounterCh08` because channel 8 counts every change whereas channel 7 only counts changes from `false` to `true`.

**LED:** In the "LED1" and "LED2/3" windows, you control the LEDs on the front panel of the ES1325 via the parameters `P_LED1`, `P_LED2` and `P_LED3`.
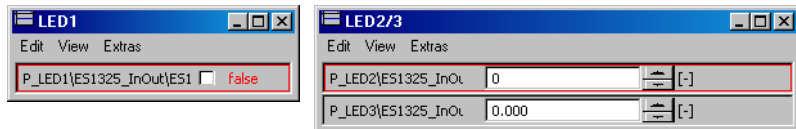
LED1 lights up when P_LED1 is true, LED2 lights up when P_LED2 is not equal to 0, and LED3 lights up when the value of P_LED3 is not equal to 0 after the decimal places are cut off.

## 11.5    Tutorial – ES1325 (with Trigger)

The ES1325 sample project using the trigger contains a model with a dynamic PWM signal (PWM block, see Fig. 11-12), whose duty cycle is determined by a sawtooth signal.
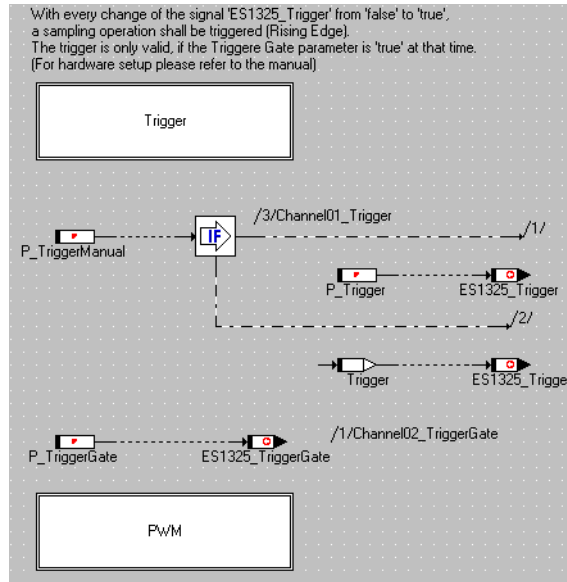


**Fig. 11-11**    ES1325 (with Trigger) – Model
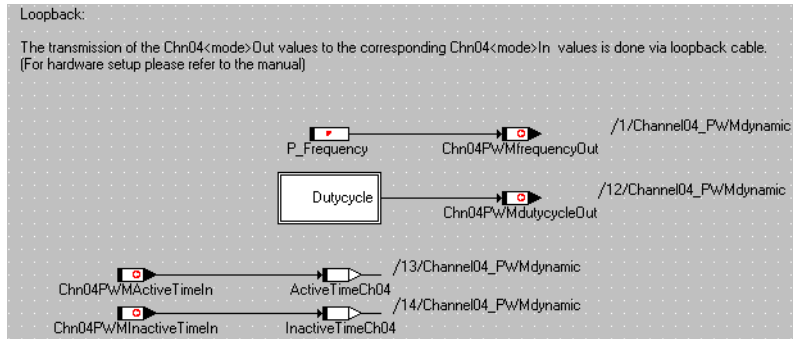
**Fig. 11-12** ES1325 (with Trigger) – PWM

The output signals (`Chn04PWMfrequencyOut` and `Chn04PWMdutycycleOut` messages) are sent continuously but the reading of the input signals (`Chn04PWMActiveTimeIn` and `Chn04PWMInactiveTimeIn` messages) is controlled by a hardware trigger. This can either be triggered manually or automatically (`P_TriggerManual` parameter).

With automatic triggering, the first trigger signal `ES1325_Trigger` in the `Trigger` block (Fig. 11-13) is calculated; with manual triggering the trigger signal is determined via the `P_Trigger` parameter.

The second trigger signal `ES1325_TriggerGate` is always determined via the `P_TriggerGate` parameter.
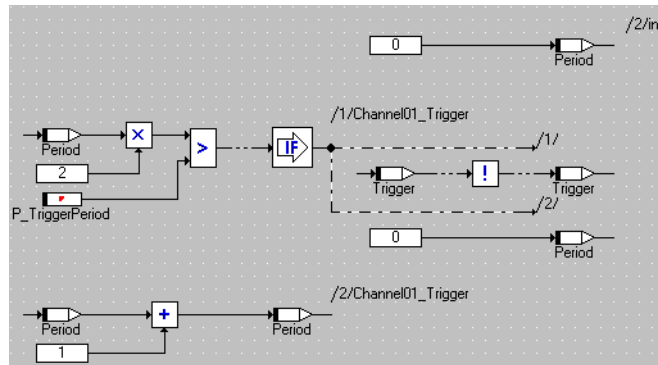


**Fig. 11-13** ES1325 – Trigger

## 11.5.1 The ES1325 Board

For details on the board and the necessary wiring please refer to section 11.4.1 "The ES1325 Board" on page 332.

## 11.5.2 Sample Project

**To open the exercise example:**

- Select the `ASCET_RP\RTIOTutorial\ES1325_triggered` folder from the Component Manager.
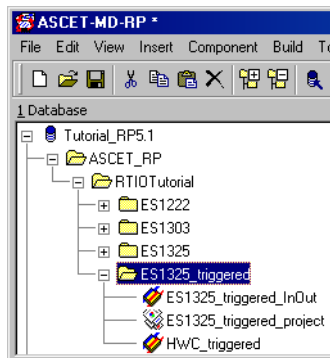


- Select the `ES1325_triggered_project` project.
- Open the project.
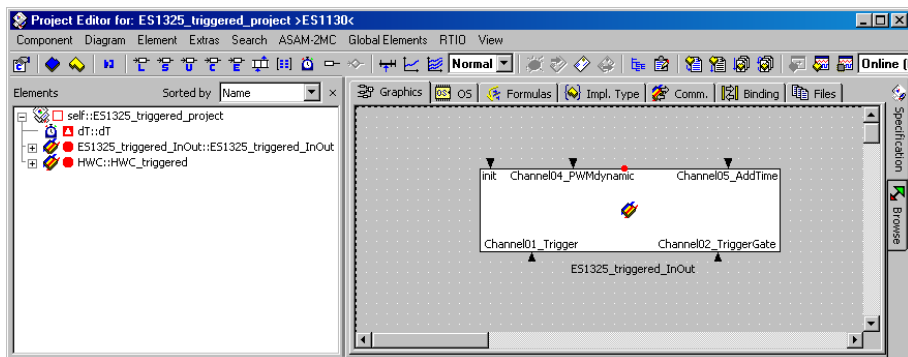
• Click the **Project properties** button.

The "ProjectProperties" window opens.
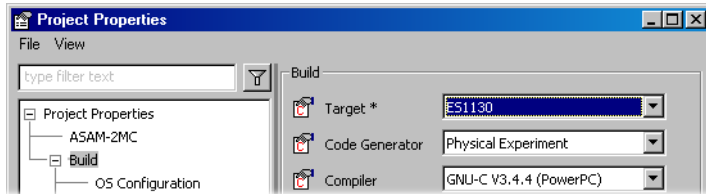


• In the "Build" node, select the options

  Target:    >ES1130< or >ES1135<,
  Compiler: GNU-C * (PowerPC).

**Note**

*Only messages declared as "Exported" are available for RTIO communication.*

### 11.5.3 Creating the Hardware Configuration

**Note**

*Normally you have to create the C-Code module* HWC *and link it to the project before you edit the hardware configuration. This step has already been taken care of in the tutorial.*

**To open the HWC Editor:**

• Select **RTIO → Open Editor** in the Project Editor.

The HWC Editor opens.

**To create the hardware configuration (HWC):**

The hardware has to be described as a tree-like structure in the items list. The HWC item is always available and forms the roots of the tree.

- In the HWC Editor select **Edit → Add Item**

*or*

- click the **Add Item** button.

  The "Add Item" window is displayed.



**Add Item** always opens the list of available items of the next hierarchy level.

- Select `ES113x`.

  This is used to describe the ES1000.x system with an integrated ES1130 or ES1135 PowerPC computer node.

- Click **OK**.

  `ES113x` is added to the "Items" list.



- Select `ES113x` from the "Items" list.

  In the "Globals" tab, the task name `Init` is specified for the `Init Task` option. In the OS-Editor of the sample project, there is an init task with the same name which means that no other init task has to be selected here.

**To link and set up the ES1325:**

- Open the list of available items of the next hierarchy level using **Edit → Add Item**.



- Select `ES1325-DIO`.

  This is used to describe the ES1325 interfaces.

- Click **OK**.

  `ES1325-DIO` is added to the "Items" list.



- Select `ES1325-DIO` from the "Items" list.

For the options `Init Task` and `Exit Task` the specified task names `Init` and `Exit` can be used as in section "To link and set up the ES1325:" on page 338. The board number is automatically set in the `ID` option. As the `ES1325-DIO` item is the first of this type, "`ID1`" is set. You still have to make the settings for the hardware trigger.

- Double-click the "Value" column in the "HW Trigger Mode" row (see page 245).

  A dropdown list opens.

- Select `Rising edge`.

- Double-click the "Value" column in the "HW Trigger Gate" row (see page 247).

  A dropdown list opens.

- Select `On`.

  The type of hardware trigger used is now determined.

- Accept the settings using **Accept**.

The tab should now look as follows:

| Option | Value |
|---|---|
| Name | Es1325dio |
| Init Task | Init |
| Exit Task | Exit |
| ID | ID1 |
| HW Trigger Mode | Rising edge |
| HW Trigger Gate | On |
| Zero Transition [°] | --- |
| Tooth Width [°] | --- |

**To create devices:**

- Select `ES1325-DIO` from the "Items" list.

- Open the list of items of the next hierarchy level using **Edit → Add Item**.
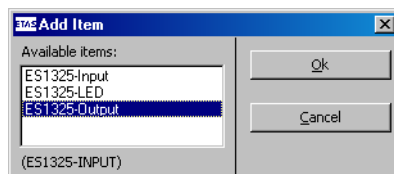
- Select `ES1325-Output` and click **OK**.

  This is used to describe the ES1325 output channels.

- Add the `ES1325-Input` device.

  `ES1325-Input` is used to describe the input channels of the ES1325.

The item tree for the description of the sample system is now completely specified.
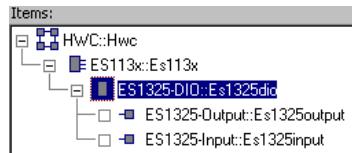


## 11.5.4 Making HWC Settings for the ES1325

The inputs and outputs now have to be configured. The outputs provide the signals which the inputs receive.

*Outputs –* `ES1325-Output` *Device*

**"Globals" Tab:**  You do not have to make any settings in the "Globals" tab.

**"Groups" Tab:**  The signal-group-specific settings are made in the "Groups" tab.

The `ES1325-Output` device has a specified signal group for every output signal (`Ch<n>Gen, <n>` = 01 – 16). Here, you set the mode, task assignment, the generated signals as well as the levels and edges for each group used.

**To make settings in the "Groups" tab:**

- Select the "Groups" tab.
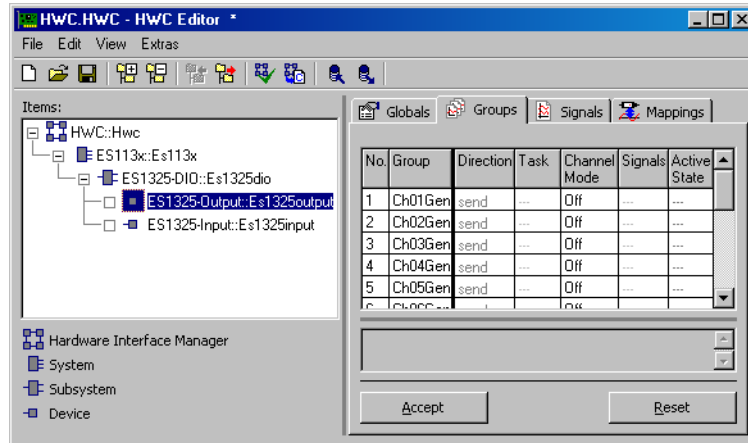


Execute the following steps for signal groups 1, 2 and 4.

1. Selecting the channel mode
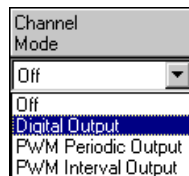
- Select the following modes for the different groups in the "Channel Mode" column:

| Group | Channel Mode |
|---|---|
| Ch01Gen, Ch02Gen | Digital Output |
| Ch04Gen | PWM Periodic Output |

The default value for the relevant mode is entered in the "Active State" column.
In the "Task" column, you can now assign a task to the groups used in which signal transfer is to take place.

2.  Assigning a task

- Double-click the "Task" column in the relevant
  row to open the "Task selection" dropdown
  list.
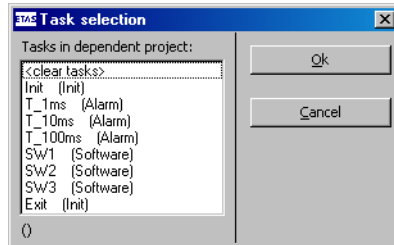


- Select the following tasks in which data trans-
  fer is to take place from the task selection list.

| Group | Task |
|---|---|
| Ch01Gen, Ch01Gen | T_1ms |
| Ch04Gen | T_10ms |

- Click **OK**.

  The selected tasks are displayed in the "Task"
  column.

3. Selecting signals

- Double-click the "Signals" column in the relevant row to open the "Signal Selection" window.



For a description of the signals, please refer to the section "Signals" on page 260.

- Select the following signals.

| Group | Signals |
|---|---|
| Ch01Gen, Ch02Gen | Accept default setting |
| Ch04Gen | Frequency [Hz], Duty Cycle [%] |

- Confirm your selection with **OK**.

4. Selecting levels

As all groups use the default setting `High`, you do not have to make any changes to the "Active State" column.

- Once you have made all the necessary settings, click **Accept** in the HWC Editor to save the settings.

Once you have set up the signal groups, the tab should look as follows:



**"Signals" Tab:** The signals you generated in the "Groups" tab are contained in this tab.



You can edit the names and formulae of the signals in this tab. *No* changes are necessary, however, for the sample project.

**"Mappings" Tab:** The signals and the ASCET messages from the project are assigned to each other in this tab. A selection dialog opens when you click the required cell in the "ASCET Message" column. This only contains messages which have the attribute *Exported* and correspond to the transfer direction of the signal group (here: *send*); i.e.:

- Direction = send → send messages



**To assign an ASCET message manually:**

- Select the "Mappings" tab.
- Double-click the "ASCET Message" column in the relevant row.

  The "Message selection" window opens.



  It contains all send messages from the ASCET project.
- Select the following ASCET messages for the signals.

| Signal | ASCET Message |
|---|---|
| Ch01GenState | ES1325_Trigger |

| Signal | ASCET Message |
|--------|---------------|
| `Ch02GenState` | `ES1325_TriggerGate` |
| `Ch04GenFrequency` | `Chn04PWMfrequencyOut` |
| `Ch04GenDutyCycle` | `Chn04PWMdutycycleOut` |
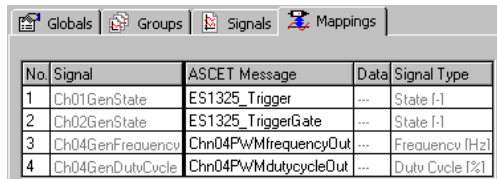
- Click **OK**.
- Then click **Accept** in the HWC Editor to save the settings.

Once you have assigned messages or values to all signals, the tab should look as follows:



*Inputs –* `ES1325-Input` *Device*

**"Globals" Tab:**   The settings for the device are made in the "Globals" tab.

**To make settings in the "Globals" tab:**

- Select the "Globals" tab.
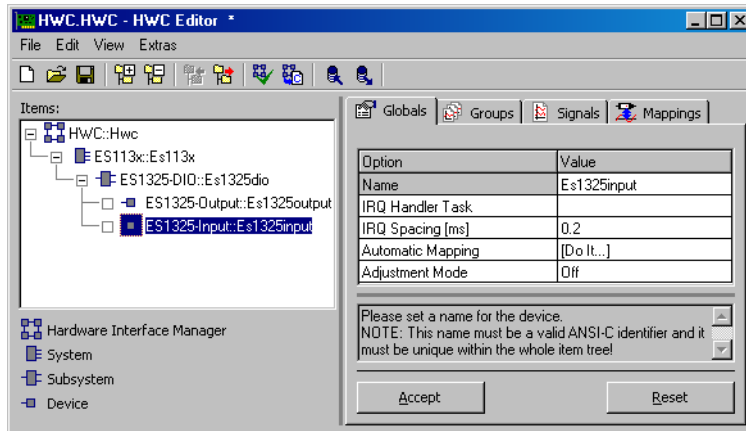
- Double-click the empty box next to the option `IRQ Handler Task`.

  The "Task selection" window opens.
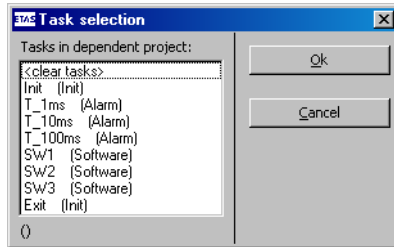


- Select the software task `SW1`.

  You can use the default settings for the other options.

- Accept the change using **Accept**.

The tab should now look as follows:
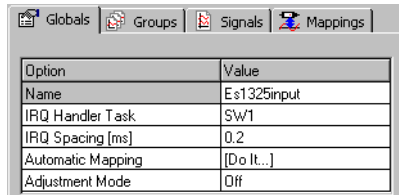


**"Groups" Tab:**  The signal-group-specific settings are made in the "Groups" tab.

The `ES1325-Input` device has a specified signal group for every input signal (`Ch<n>Msr`, `<n>` = 01 – 16). Here, you set the mode, task assignment, the generated signals as well as the levels and edges for each group used.

**To make settings in the "Groups" tab:**

- Select the "Groups" tab.
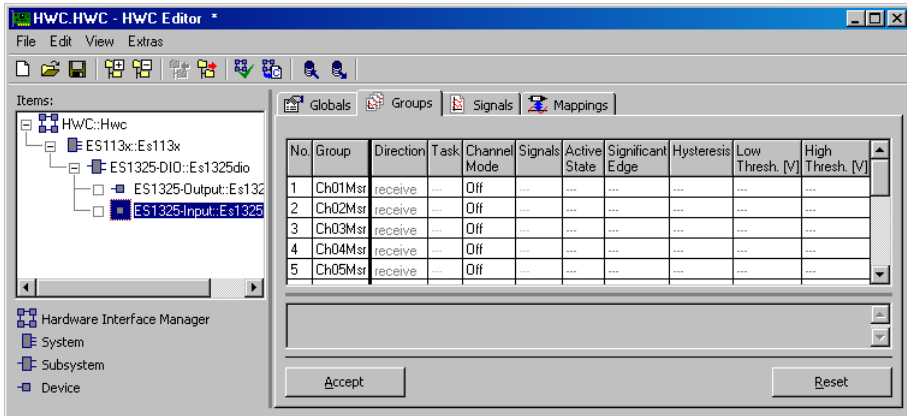


Execute the following steps for signal group 4.

1. Selecting the channel mode



- Double-click the "Channel Mode" column in row 4.

  A dropdown list opens.

- Select the PWM Input mode for the Ch04Msr group.

  The default values for this mode are entered in the other columns.

2. Setting the use of the hardware trigger
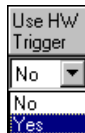


- Double-click the "Use HW Trigger" column in row 4.

  A dropdown list opens.

- Select Yes.

- Double-click the "IRQ" column in row 4.
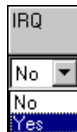
  A dropdown list opens.



- Select Yes.

  The modified selection is displayed in the "IRQ" field. At the same time, the "Task" field is reset and locked.

3. Selecting signals

- Double-click the "Signals" column in row 4 to open the "Signal Selection" window.
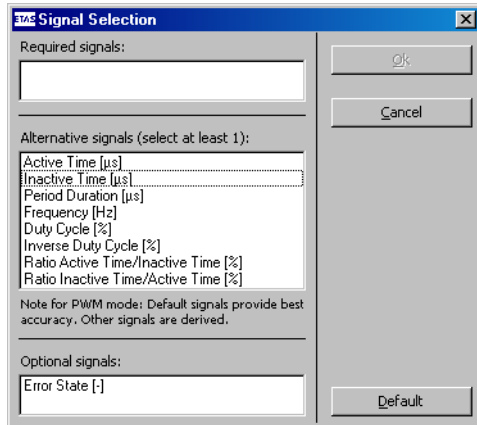


In this window, you specify which signals are generated for the group. For a description of the signals, please refer to the section "Signals" on page 253.

- Select the signals `Active Time [µs]` and `Inactive Time [µs]`.

  No further settings are necessary as you use default values in the columns "Active State" and "Significant Edge".

- Once you have made all the necessary settings, click **Accept** in the HWC Editor to save the settings.

The tab should now look as follows:

**"Signals" Tab:** The signals you generated in the "Groups" tab are contained in this tab.



You can edit the names and formulae of the signals in this tab. *No* changes are necessary, however, for the sample project.

**"Mappings" Tab:** The signals and the ASCET messages from the project are assigned to each other in this tab. A selection dialog opens when you click the required cell in the "ASCET Message" column. This only contains messages which have the attribute *Exported* and correspond to the transfer direction of the signal group (here: *receive*); i.e.:

- Direction = receive → receive messages



**To assign an ASCET message manually:**

- Select the "Mappings" tab.

- Double-click the "ASCET Message" column in the relevant row.

  The "Message selection" window opens.



  It contains all receive messages from the ASCET project.

- Select the following ASCET messages for the signals.

| Signal | ASCET Message |
|---|---|
| Ch04MsrActiveTime | Chn04PWMActiveTimeIn |
| Ch04MsrInactiveTime | Chn04PWMInactiveTimeIn |

- Click **OK**.
- Then click **Accept** in the HWC Editor to save the settings.

Once you have assigned messages to all signals, the tab should look as follows:



- Click **OK**.
- Then click **Accept** in the HWC Editor to save the settings.

### 11.5.5 Saving the Hardware Configuration

The created hardware configuration can be saved in the file container of the project or as a DOS file (extension `*.HWX`). How this is done is explained in section 11.2.5 on page 308.

> **Note**
>
> *The solution of the example is saved as* `ES1325.hwx`. *Make sure you do not overwrite this file.*

### 11.5.6 Creating Code for the HWC Module

The generation sequence for the HWC module then has to be started. How this is done is explained in section 11.2.6 on page 309.

### 11.5.7 Experimenting with the Sample Project

All RTIO-specific actions have now been completed; this means that the HWC Editor can be closed. A further subsequent step is the normal code generation for the experimental target which is started from the Project Editor.

> **Note**
>
> *We would recommend that you do not include the HWC module in the graphic representation of the ProjectEditor as this module is modified in every RTIO generation process. This leads to an unfavorable graphic representation of the HWC module.*

**Code generation for the experimental target:**

- In the project editor, select **Component → Build** to generate code for the entire project.
- Select **Component → View Generated Code** to view the generated code.

**To experiment online:**

- Connect the inputs and outputs of the ES1325 in accordance with the section "Connections" on page 333, if this has not already taken place, and switch on the power supply of the ES1000.x.

- Select `Online (RP)` from the "Experiment Target" combo box.

  `Offline (RP)` is intended for offline experiments on the Target.

- Select **Component → Open Experiment**.

  The "Physical Experiment" window and the predefined experiment environment consisting of one oscilloscope, a numeric display and three calibration windows open.
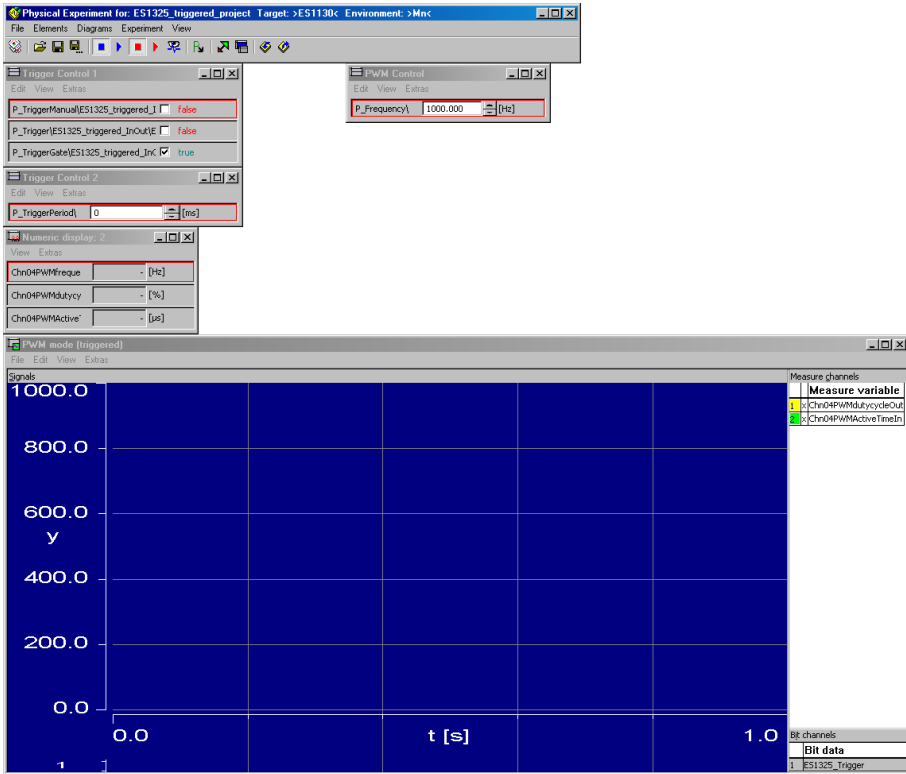


The default setting assumes automatic calculation of the first trigger signal and a constant second trigger signal (parameter `P_TriggerManual` is `false`, parameter `P_TriggerGate` is `true`).

The "PWM mode (triggered)" oscilloscope shows the output message `Chn04PWMdutycycleOut` (yellow curve) and the input message `Chn04PWMActiveTimeIn` (green curve) as well as the `ES1325_Trigger` message, containing the first trigger signal, in the lower section.
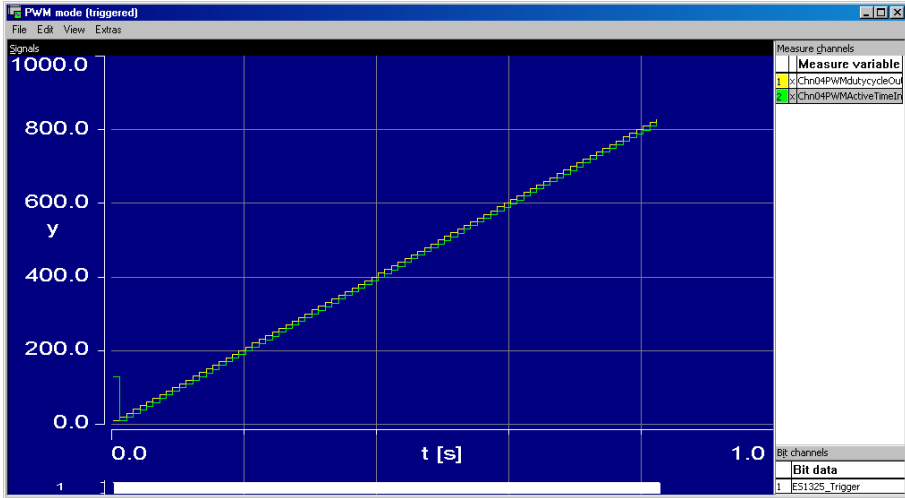


When you start measuring with the defined values, you have two curves offset by one step.

If you increase the value of the `P_TriggerPeriod` parameter in the "Trigger Control 2" calibration window, the time between two rising edges increases. `Chn04PWMActiveTimeIn` is thus read in correspondingly less frequently.
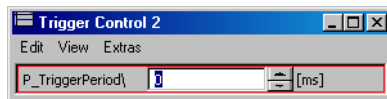
At the point marked in the following diagram, the `P_TriggerPeriod` parameter was set from 0 ms to 250 ms. The trigger signal at the bottom of the display area of the oscilloscope only delivers a rising edge every 250 ms; the `Chn04PWMdutycycleOutIn` message is read in accordingly rarely and the relevant curve has large steps.
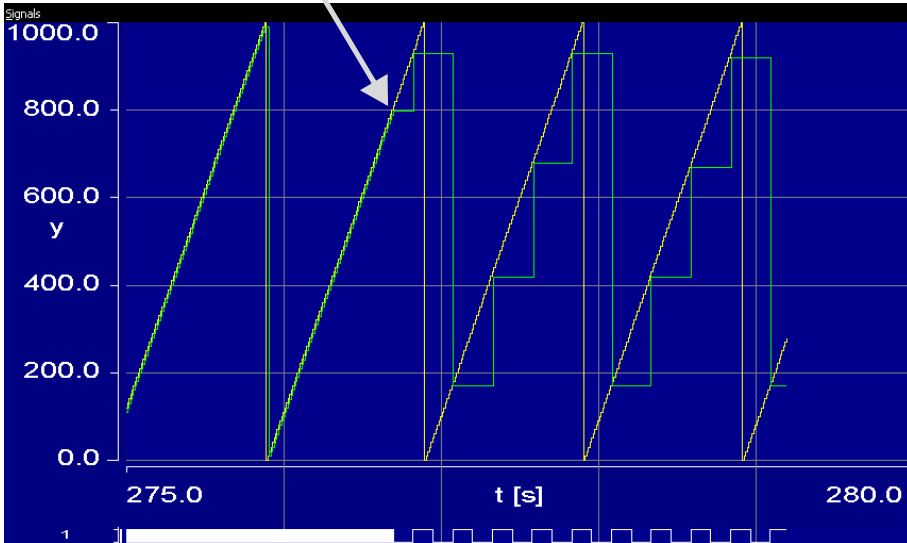


If you modify the parameter `P_Frequency` in the "PWM Control" calibration window, the frequency changes and hence the active and inactive time. The duty cycle remains the same.
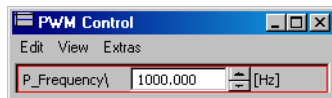
At the point marked in the following diagram, the frequency was set from 1000 Hz to 1500 Hz. The curve for `Chn04PWMdutycycleOut` remains unchanged but the curve for `Chn04PWMActiveTimeIn` now has a lower maximum.
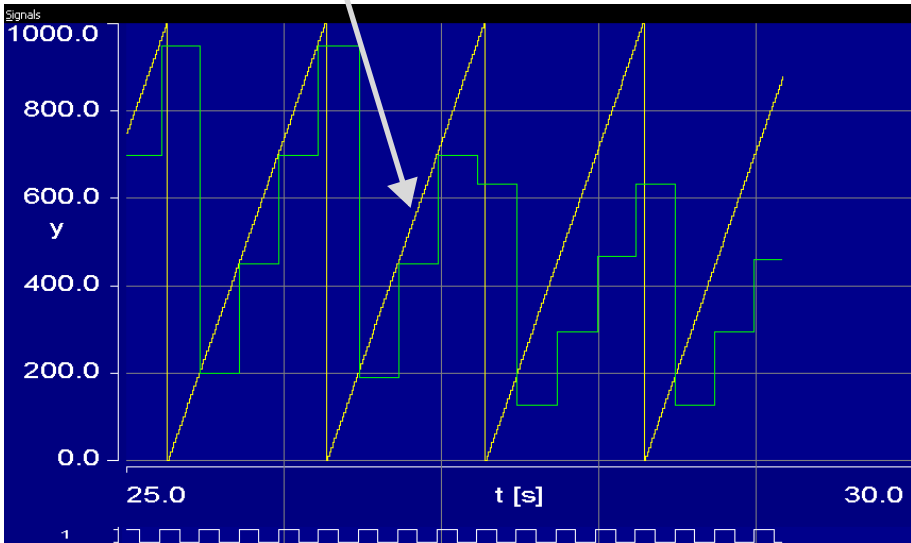


For this figure, the range of the x-axis in the "Extent" field of the "Display setup" window (to be reached from the oscilloscope using **Extras → Setup**) was set to 5 seconds.

You can change the trigger control in the "Trigger Control 1" calibration window.
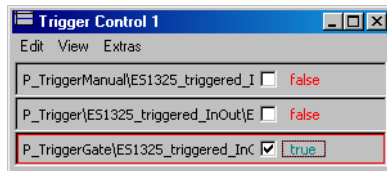
If, for example, you set the `P_TriggerGate` parameter to `false`, `Chn04PWMActiveTimeIn` is no longer read in as the second trigger signal is `false`. The curve remains at the last value. The curve for `Chn04PWMdutycycleOut` again remains unchanged.
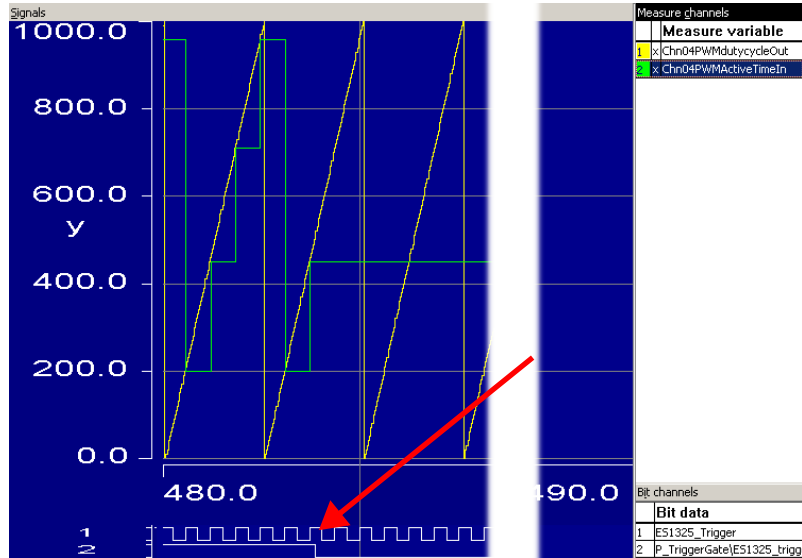


For this figure, the range of the x-axis was set to 10 seconds, and the variable `P_TriggerGate` was added to the oscilloscope.

Set `P_TriggerGate` back to `true` to continue measuring `Chn04PWMActiveTimeIn`.

If you set the `P_TriggerManual` parameter in the "Trigger Control 1" window to `true`, activate the manual control of the first trigger signal. Each time you change the `P_Trigger` parameter from `false` to `true`, a trigger signal is generated and the `Chn04PWMActiveTimeIn` message read in.

Manual triggering was activated in the following diagram at point 1. The course of the `ES1325_Trigger` signal shows how `P_Trigger` has been changed. `Chn04PWMActiveTimeIn` was read in with every rising edge of the trigger signal.
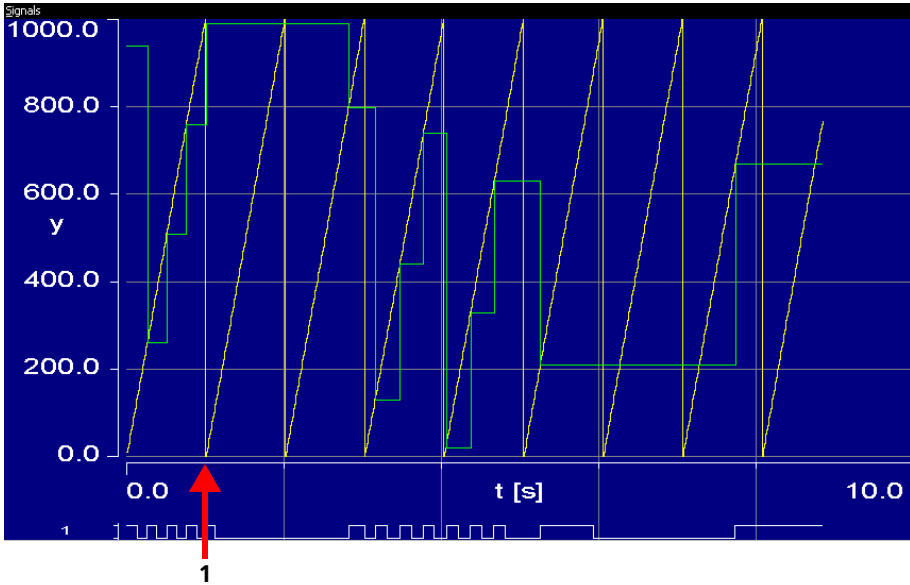


For this figure, the range of the x-axis in the "Extent" field of the "Display setup" window (to be reached from the oscilloscope using **Extras → Setup**) was set to 10 seconds.

# 12 ETAS Network Manager

This chapter describes the configuration of the ETAS network via the *ETAS Network Manager*, as well as the respective network topology.

## 12.1 Overview

ASCET-RP 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**

*With ASCET-RP V5.6, you no longer need 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. This requires administrator rights (see the documentation for your operating system).*

## 12.2 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 ASCET-RP draws from an address pool to assign the IP address to the connected devices.

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

## 12.3 Network Adapter Addressing

### 12.3.1 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 NT | yes | yes | no | no |
| Windows 98 SE | yes | yes | yes | no |
| Windows 2000 | yes | yes | yes | no |
| Windows XP | 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 NT | yes | yes | no | no |
| Windows 98 SE | yes | no | yes | no |
| Windows 2000 | yes | no | yes | no |
| Windows XP | 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 (exception: Windows NT). DHCP can be used only in combination with APIPA or an alternative IP address.

### 12.3.2 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.

### 12.3.3 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 is there no DHCP server (as in the ETAS network), the network adapter has not been configured.

In this instance, each operating system (except Windows NT) has an automatic feature that automatically assigns the network adapter an IP address:

*Windows 98 SE*

Windows 98 SE automatically uses the DHCP address in the company network and the APIPA address in the ETAS network. If the DHCP network was used to boot the PC, the APIPA reconfiguration is kicked off when the ETAS software (e.g., ASCET-RP) is started. This takes at least 60 seconds. If the PC is reconnected to the DHCP network after using the ETAS software, the network adapter is reconfigured to a DHCP address. This reconfiguration may only take 5 minutes. To shorten this time, the user has the option of executing the reconfiguration manually using the **ipconfig /renew** command in an MS-DOS command window.

Addressing a network adapter via DHCP without an APIPA address is not supported.

*Windows 2000*

Windows 2000 automatically checks whether there is a connection to the DHCP server. If there is none, it automatically assigns the IP address via APIPA. In the ETAS network, the APIPA address is always used. 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. 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 an APIPA address is not supported.

*Windows XP*

Windows XP automatically checks 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.

*Windows NT*

In Windows NT, the DHCP address is assigned for a certain time, the so-called lease time. The network adapter is no longer configured after this period of time has elapsed. The Network Manager allows you to extend this lease time so you can use the network adapter's IP address even if you have been "idle" in the company network for an extended period of time. The lease time is kick started only after the ETAS software (e.g., ASCET-RP) has accessed the hardware for the first time. Increasing the lease time takes approximately 20 seconds. When exiting the ETAS software, the lease time is reset to the original value.

**Note**

*Changing the lease time makes sense only when you are not working in the DHCP network for a longer period of time (longer than the existing lease time). To exclude IP address conflicts, the PC should not be connected to the DHCP network for 20 seconds after leaving the ETAS software.*

## 12.4 User Interface

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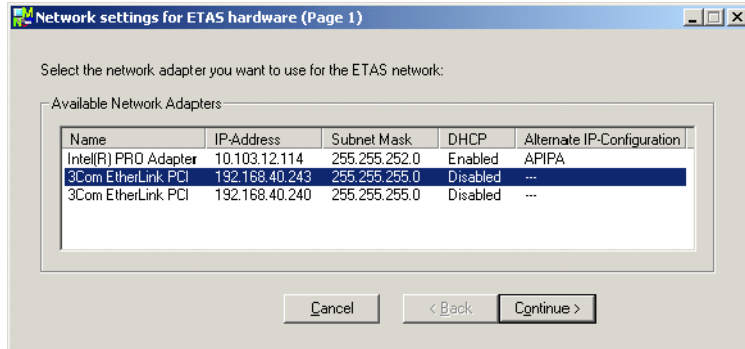


The following information on the available network adapters is displayed:

- **Name**

  Name of the network adapter. This entry cannot be edited in this window.

- **IP-Address**

  IP address of the network adapter. This entry cannot be edited in this window.

- **Subnet Mask**

  Setting for the subnet mask. This entry cannot be edited in this window.

- **DHCP**

  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**

  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 or its use is not supported by the operating system (Windows NT).

– User Defined

The user can define a user-specific alternative IP address (Windows XP).

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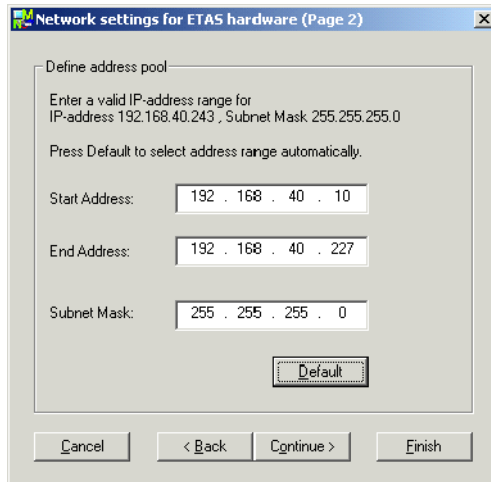


In general, all values can be modified by directly typing them in the corresponding field, or by selecting the default setting from a list box.

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.

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

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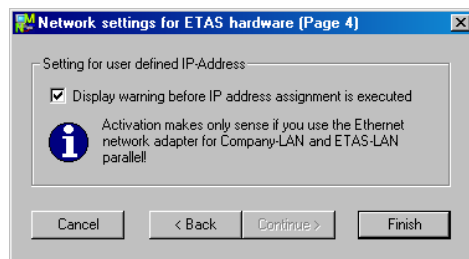


The following parameters can be set:

- **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.*

### 12.4.4 "Network settings for ETAS hardware (Page 5)" Dialog Window

This dialog window is displayed only if Windows NT is the operating system and the selected network adapter is configured for DHCP.
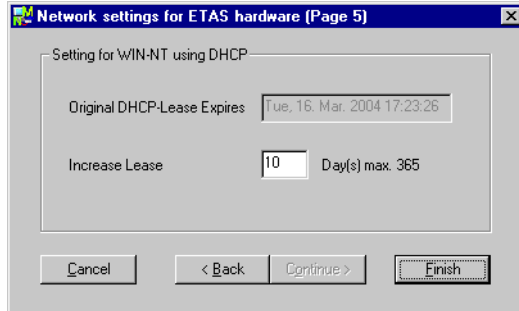


The following information on the available network adapters is displayed:

- **Original DHCP Lease Expires**

  Indicates when the DHCP lease time will expire. This field cannot be edited.

- **Increase Lease**

  Here you can specify the number of days by which to extend the lease time. You can assign a positive integer up to 365.

### 12.5 Configuring Network Addresses for ETAS Hardware

The first steps apply to the configuration of network adapters with fixed IP address and network adapters in a DHCP environment.

**To start the Network Manager:**

- In the Windows Start menu, select **Start → Programs → ETAS → ASCET5.1 → ETAS Network Settings**.

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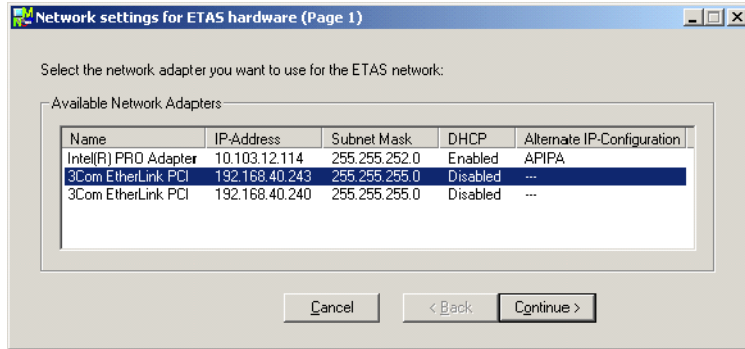


**To select the network adapter:**

- 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 those network adapters whose addressing type the ETAS network supports.

  Use **Cancel** to abort the procedure.

- Click **Continue** to configure the selected network adapter.

  The "Network settings for ETAS hardware (Page 2)" dialog window opens.
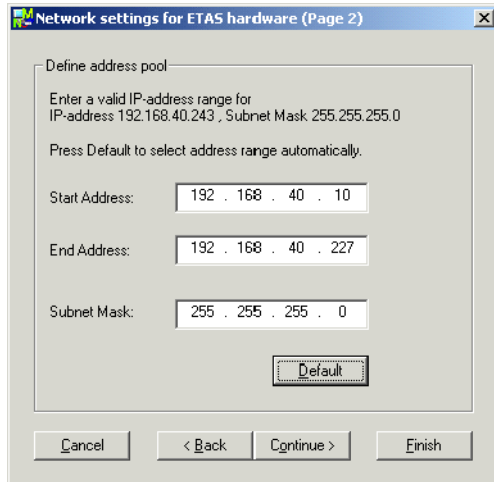


**Note**

*The **Finish** button is only available when you are configuring a network adapter in a DHCP environment.*

*Configuring Network Addresses: Adapter with Fixed IP Address*

**To define the address pool:**

- In the "Network settings for ETAS hardware (Page 2)" window, click the entry you want to modify ("Start Address," "End Address" or "Subnet Mask").
- Edit the value directly (text input)

  *or*

- click the **Default** button.

  The address range and the setting for the subnet mask are entered automatically by the Network Manager. You may accept these settings or overwrite them.

  Use **Cancel** to close the dialog window without saving the changes. Use **Back** to return to the previous dialog window.

  The **"**Network settings for ETAS hardware (Page 4)**"** dialog window opens.
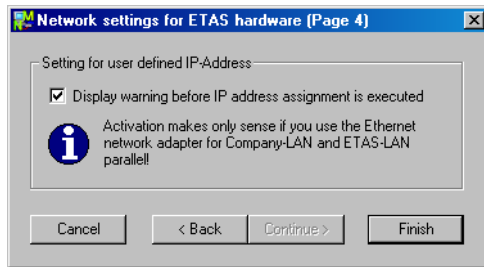


**To set a user-defined IP address:**

- Activate the **Display warning before IP address assignment is executed** option 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.

  Use **Cancel** to close the dialog window without saving the changes. Use **Back** to return to the previous dialog window.

- Restart ASCET-RP and all other ETAS software applications to make the changes effective.

  Restarting is necessary only if ASCET-RP did not automatically invoke the configurator during a hardware search or initialization.

*Configuring Network Addresses: Adapter in DHCP Environment*

**To define the address pool:**

- In the "Network settings for ETAS hardware (Page 2)" window, click the entry you want to modify ("Start Address," "End Address" or "Subnet Mask").

- 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.

1. If you address the network adapter via DHCP using an APIPA or alternative IP address, proceed as follows to finish the configuration:

   - Click the **Finish** button.

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

   - Restart ASCET-RP and all other ETAS software applications to make the changes effective.

     Restarting is necessary only if ASCET-RP did not automatically invoke the configurator during a hardware search or initialization.

2. If you address the network adapter via DHCP and use the Windows NT operating system, proceed as follows to continue the configuration:

• Click the **Continue** button.

  The **"**Network settings for ETAS hardware (Page 5)" dialog window opens.
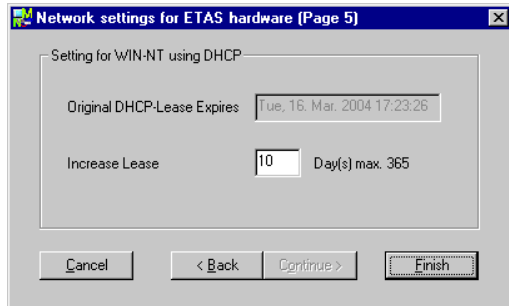


**To set up DHCP in Windows NT:**

• In the "Increase Lease" field, enter the number of days by which you want to extend the lease time.

  Use **Cancel** to close the dialog window without saving the changes. Use **Back** to return to the previous dialog window.

• Click the **Finish** button.

  Configuring the network adapter is finished and the dialog window closes. The changes are saved.

• Restart ASCET-RP to make the changes effective.

  Restarting is necessary only if ASCET-RP did not automatically invoke the configurator during a hardware search or initialization.

## 12.6    Troubleshooting Ethernet Hardware Access

This section gives an overview of known problems accessing ETAS hardware via the Ethernet interface.

### 12.6.1 APIPA disabled on Windows 98 SE, 2000 or XP

The alternative mechanism for IP addressing (APIPA) is usually enabled on all Windows 98 SE, 2000 and XP 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:**

- In the Windows Start menu, select **Start → Run**.
- In the "Run" window, enter `regedit` and click **OK**

  The registry editor opens.
- In the directory tree of the registry editor, open the folder `HKEY_LOCAL_MACHINE\ SYSTEM\CurrentControlSet\ Services\`.
- Select **Edit → Find** to search for the key `IPAutoconfigurationEnabled`.
- Set the value for all instances of this key to `1` to enable the APIPA mechanism.

  You may find several instances of this key in the Windows registry because the APIPA mechanism can be disabled both for the TCP/IP service in general and for specific network adapters individually.
- Close the registry editor.

If you cannot find any instances of the registry key mentioned, the APIPA mechanism has not been disabled on your system.

### 12.6.2 Personal Firewalls

Windows XP comes with a built-in personal firewall. On many other systems it is very common to have personal firewall software from third party vendors, such as Symantec, McAffee or BlackIce installed.

Personal firewalls may interfere with access to Ethernet hardware using ASCET-RP. The automatic search for hardware typically cannot find any Ethernet hardware at all, although the configuration parameters are correct. In that case, you may have firewall software installed on your system.

You should either disable the firewall software while working with ASCET-RP, or open it for IP address 192.168.40.240 (ports 18000—18005). For details, please refer to the user documentation of your personal firewall software.

## 13    Annex

This annex contains remarks to target-specific external C code (chapter 13.1 "Compiler Switches for External C Code"), as well as the API funcions ASCET-RP provides for the ES113x expperimental target. These functions define the interfaces between ASCET-RP and the following applications:

- ERCOS[EK] (chapter 13.2)
- NVRAM (chapter 13.3)
- Watchdog (chapter 13.4)
- LEDs (chapter 13.5)
- Miscellaneous (chapter 13.6)

## 13.1    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, ASCET-RP V5.6 provides the following switches:

- ES910
- ES1130
- ES1135
- ES113x (für ES1130 und 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
```

## 13.2    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<sup>EK</sup> commands supported by ASCET-RP 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. | 409 |
| SetNextAppMode | Switches to the specified application mode after processing all active tasks. | 410 |
| | | |
| **Tasks** | | |
| DeclareTask | Serves as an external declaration of a task. | 411 |
| ActivateTask | Activates a SW task. | 411 |
| | | |
| **System Time** | | |
| GetSystemTime | Gets the current system time. | 412 |

| Command | Function | Page |
|---|---|---|
| GetSystemTimeLow | Gets the low-order part of the current system time. | 412 |
| GetSystemTimeHigh | Gets the high-order part of the current system time. | 413 |

| **Interrupt Handling** | | |
|---|---|---|
| EnableAllInterrupts | Globally enables all interrupts. | 413 |
| DisableAllInterrupts | Globally disables all interrupts. | 413 |

| **dT Query** | | |
|---|---|---|
| GetDeltaT | Returns the value of dT. | 414 |

### 13.2.1  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 | `#define DeclareAppMode(AppID)`<br>`extern AppModeType AppID` |

| 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 | `extern uint excCtr;`<br>`extern uint randx;`<br>**`DeclareAppMode`**`(idleMode);` |
| See also | DeclareTask |

## SetNextAppMode

| Function | Switches to the specified application mode after processing all active tasks. |
|---|---|
| Syntax | StatusType SetNextAppMode(AppModeType appMode) |
| 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); |
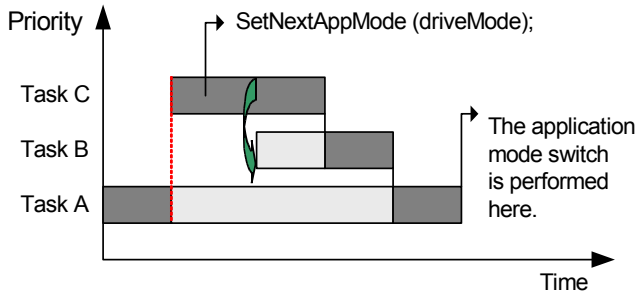


See also      –

## 13.2.2 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)`<br>`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()**.<br>The function and use of this service are similar to that of the external declaration of variables. |
| Example | `extern uint excCtr;`<br>`extern uint randx;`<br>**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.<br>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.<br>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 been reached already 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 | – |

### 13.2.3    System Time

A discrete system time is the time base of ERCOS^EK. 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 ERCOS^EK 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;<br>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 ERCOS^EK 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;<br>lowPartOfNow = **GetSystemTimeLow()**; |
| See also | GetSystemTime, GetSystemTimeHigh |

| GetSystemTimeHigh | |
| --- | --- |
| Function | Gets the high-order part of the current system time. |
| Syntax | `TickType GetSystemTimeHigh(void)` |
| 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 | `TickType highPartOfNow;`<br>`highPartOfNow = GetSystemTimeHigh();` |
| See also | `GetSystemTime, GetSystemTimeLow` |

### 13.2.4    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 | `void EnableAllInterrupts(void)` |
| 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 | `DisableAllInterrupts` |

| DisableAllInterrupts | |
| --- | --- |
| Function | Disables all interrupts globally. |
| Syntax | `void DisableAllInterrupts(void)` |

| Description | **DisableAllInterrupts()** disables all interrupts globally and stores the state of nested calls. |
|---|---|
| Return code | None |
| See also | EnableAllInterrupts |

### 13.2.5 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.
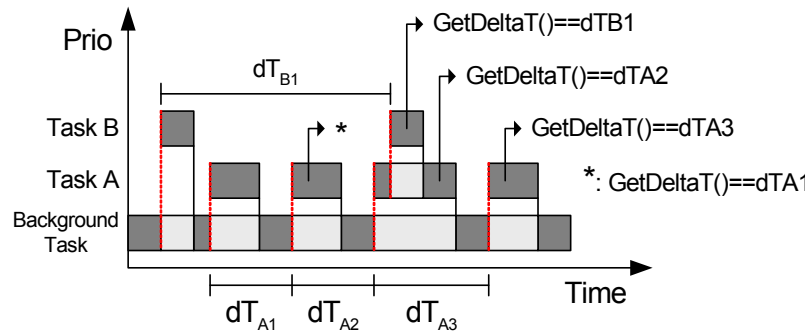


The dT returned by GetDeltaT() is very useful for mathematical calculations e.g. an integration:

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

| **GetDeltaT** | |
|---|---|
| Function | Returns the value of dT. |
| Syntax | TickType GetDeltaT(void) |

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

## 13.3    API Functions (NVRAM)

The default behavior of the NVRAM manager described in chapter 6.2.1 can be altered from within an ASCET model (C code component) via the following interfaces:

| **nvramInitModelVars** | | |
|---|---|---|
| Function | Initializes the NV variables. | |
| Syntax | `uint32 nvramInitModelVars(void)` | |
| 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 | `EC_NVRAM_SUCCESS` | Success |
| | `EC_NVRAM_NO_NV_VARIABLES` | No NV variables inside the model |
| | `EC_NVRAM_INADMISSIBLE_US E` | Function has been already called |
| | `EC_NVRAM_INTERNAL_ERROR` | An internal error occurred |
| | `EC_NVRAM_NO_MATCH` | The NVRAM content does not match the current model |
| Example | – | |
| See also | `nvramCheckForInitializedVars` | |

## nvramSetUpdateInterval

| | |
|---|---|
| Function | Sets the automatic NVRAM update interval. |
| Syntax | `uint32 nvramSetUpdateInterval(uint32 interval_sec)` |
| 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 | `EC_NVRAM_SUCCESS`    Success |
| | `EC_NVRAM_INVALID_ARG`  `Interval_sec > 30` |
| Parameter | `interval_sec`    Update interval in seconds. Must be a value between 0 and 30 (0: no periodical update). |
| Example | – |
| See also | `nvramGetUpdateInterval` |

## nvramGetUpdateInterval

| | |
|---|---|
| Function | Gets the automatic NVRAM update interval. |
| Syntax | `uint32 nvramGetUpdateInterval(void)` |
| 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 | `interval_sec`    Update interval in seconds. |
| Example | – |
| See also | nvramSetUpdateInterval |

## nvramSetConsistencyLevel

| | |
|---|---|
| Function | Sets the level of NV variable data consistency. |
| Syntax | `uint32 nvramSetConsistencyLevel (T_consistencyLevel level)` |
| Description | Sets the level of NV variable data consistency. *No consistency*: NVRAM update is done without respect to consistency inside NV variables and between individual NV variables. *Low level consistency*: data consistency within NV variables (scalars, vectors and matrices but not characteristics) is guaranteed. *High level consistency*: all NV variables are updated without interruption by the model, out of the idle task. |

| Return Value | `EC_NVRAM_SUCCESS` | Success |
|---|---|---|
| | `EC_NVRAM_INVALID_ARG` | Invalid level argument |
| Parameter | `level` | `NVRAM_NO_CONSISTENCY` `NVRAM_LOW_CONSISTENCY` `NVRAM_HIGH_CONSISTENCY` |
| Example | – | |
| See also | `nvramGetConsistencyLevel` | |

## nvramGetConsistencyLevel

| | |
|---|---|
| Function | Gets the level of NV variable data consistency. |
| Syntax | `T_consistencyLevel nvramGetConsistencyLevel(void)` |
| Description | Gets the level of NV variable data consistency. *No consistency*: NVRAM update is done without respect to consistency inside NV variables and between individual NV variables. *Low level consistency*: data consistency within NV variables (scalars, vectors and matrices but not characteristics) is guaranteed. *High level consistency*: 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_CONSISTENC Y` | 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 | `uint8 nvramCheckForAutoUpdate(void)` | |
| Return Value | `true` | Auto update mode is enabled |
| | `false` | Auto update mode is disabled |
| Example | – | |
| See also | `nvramEnableAutoUpdate` `nvramDisableAutoUpdate` | |

## nvramManualUpdateExit

| | |
|---|---|
| Function | Ensures a final update of the NVRAM content. |
| Syntax | `void nvramManualUpdateExit (void)` |
| 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 | `nvramManualUpdateBackground`<br>`nvramManualUpdateBlocked` |

## nvramManualUpdateBackground

| | | |
|---|---|---|
| Function | Starts a manual update of the NVRAM content. | |
| Syntax | `uint32 nvramManualUpdateBackground(void)` | |
| 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 `nvram-CheckRunningUpdate()`.<br>**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. |

| **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 timeout 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 timeout 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. |

**nvramManualUpdateBlocked**

|  | EC_NVRAM_UPDATE_RUNNING | Other Update process (manual or cyclic) is currently running. Start of manual update failed. |
|---|---|---|
| Parameter | `timeoutUs` | Timeout period in µs |
| Example | – | |
| See also | `nvramManualUpdateBlocked,`<br>`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 `nvramStart-ManualUpdateBlocked` 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`<br>`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 |
| 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 | `uint32 nvramClear(void)` |
| Return Value | `EC_NVRAM_SUCCESS`   Success |
| Description | This function erases the NVRAM contents. The memory is initialized with zeros. |
| Example | – |
| See also | – |

## 13.4 API Functions (Watchdog)

The ES1135 Simulation Controller has a hardware watchdog. the watchdog functionality is summarised in chapter 6.2.2. The following interfaces are provided by the firmware.

### 13.4.1    Watchdog Configuration

**`wdSetSafetyMode`**

| | |
|---|---|
| Function | Sets the Safety Mode. |
| Syntax | `uint32 wdSetSafetyMode`<br>`(uint32 event, uint32 period)` |
| Description | 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.<br>The parameter **event** selects the action which is to be done when the watchdog expires.<br>`WD_EVENT_DISABLE` disables the watchdog.<br>`WD_EVENT_PPC750_RESET` resets the IBM 750GX simulation processor.<br>`WD_EVENT_PPC750_INT` triggers an interrupt to the simulation processor.<br>The parameter **period** (time period after that the watchdog expires) can be configured in the range from 0.25 ms up to 4096 ms. |

| Return Value | | |
|---|---|---|
| | `EC_CFW_SUCCESS` | Success |
| | `EC_CFW_WD_SAFETY_MODE` | Watchdog is already in safety mode |
| | `EC_CFW_INVALID_ARG` | Invalid event or period value |

| Parameter | | |
|---|---|---|
| | event | `WD_EVENT_DISABLE`<br>`WD_EVENT_PPC750_RESET`<br>`WD_EVENT_PPC750_INT` |
| | period | `WD_PERIOD_4096MS`<br>`WD_PERIOD_1024MS`<br>`WD_PERIOD_256MS`<br>`WD_PERIOD_64MS`<br>`WD_PERIOD_16MS`<br>`WD_PERIOD_4MS`<br>`WD_PERIOD_1MS`<br>`WD_PERIOD_0_25MS` |

| | |
|---|---|
| Example | ```uint32 period;``` |
| | ```uint32 event;``` |
| | ```uint32 retVal;``` |
| | ```event = WD_EVENT_DISABLE;``` |
| | ```period = WD_PERIOD_4096MS;``` |
| | ```retVal = wdSetSafetyMode(event, period);``` |
| See also | `wdSetPeriod, wdSetEvent` |

**wdSetReducedSafetyMode**

| | |
|---|---|
| Function | Sets the Reduced Safety Enhanced Function Mode. |
| Syntax | `uint32 wdSetReducedSafetyMode(void)` |
| 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 ASCET-RP use, because the model starts with the watchdog in RSEF mode.<br>The loader disables also the watchdog events. Afterwards, watchdog period and event can be modified via `wdSetPeriod` and `wdSetEvent`. |
| Return Value | `EC_CFW_SUCCESS`　　Success |
| | `EC_CFW_WD_SAFETY_MODE`　　Watchdog is in safety mode. This cannot be undone. |
| | `EC_CFW_WD_RSEF_MODE`　　Watchdog is already in RSEF mode. |
| Example | - |
| See also | `wdSetPeriod, wdSetEvent` |


**wdSetPeriod**

| | |
|---|---|
| Function | Sets the Watchdog Period. |
| Syntax | `uint32 wdSetPeriod(uint32 period)` |
| 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 | `EC_CFW_SUCCESS`　　Success |
| | `EC_CFW_WD_SAFETY_MODE`　　Watchdog is in safety mode. No period modification possible. |
| | `EC_CFW_INVALID_ARG`　　Invalid period value |
| | `EC_CFW_WD_PRE_OP_MODE`　　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 | `uint32 wdSetEvent(uint32 event)` |
| Description | The function selects the action which should be done when the watchdog expires.<br>`WD_EVENT_DISABLE` disables the watchdog.<br>`WD_EVENT_PPC750_RESET` resets the IBM 750GX simulation processor.<br>`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. |

| wdSetEvent | |
|---|---|
| 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 |

## 13.4.2  Watchdog Service

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

| wdEnableAutoService | |
|---|---|
| Function | Enables automatic servicing. |
| Syntax | `void wdEnableAutoService (void)` |
| 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 | `wdEnableAutoService();` |
| See also | wdService, wdDisableAutoService |

**wdDisableAutoService**

| | |
|---|---|
| Function | Disables automatic servicing. |
| Syntax | `void wdDisableAutoService(void)` |
| Description | This function disables the watchdog automatic servicing feature.<br><br>**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 | `wdDisableAutoService();` |
| See also | `wdService, wdEnableAutoService` |

### 13.4.3 Interrupt Control

**wdIntEnable**

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

**wdIntDisable**

| | |
|---|---|
| Function | Disables Watchdog interrupt handling. |
| Syntax | `void wdIntDisable(void)` |
| Description | This function disables the watchdog interrupt handling. |
| Example | `wdIntDisable();` |
| See also | `wdIntEnable` |

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

| **wdIntAck** | |
|---|---|
| Function | Acknowledges Watchdog interrupt. |
| Syntax | `void wdIntAck(void)` |
| 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 `wdService()` before. |
| Example | `if(wdIntPend() == true)`<br>`{`<br>    `intPollCount++;`<br>    `/* Reset Interrupt */`<br>    `wdIntAck();`<br>`}` |
| See also | `wdSetEvent`<br>`wdIntDisable`<br>`wdIntPend` |

### 13.4.4    Watchdog Status

| **wdCheckReducedSafetyMode** | |
| --- | --- |
| Function | Checks if Watchdog is in RSEF mode. |
| Syntax | `uint8 wdCheckReducedSafetyMode(void)` |
| 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 | `false`      Watchdog is running in safety mode |
| | `true`       Watchdog is running in RSEF mode |
| Example | `asdWriteUserDebug("Active = %u ReducedSafety = %u \n", wdCheckActive(), wdCheckReducedSafetyMode());` <br> For `asdWriteUserDebug`, refer to chapter 13.6. |
| See also | `wdSetSafetyMode, wdCheckActive` |


| **wdCheckActive** | |
| --- | --- |
| Function | Checks if Watchdog is active. |
| Syntax | `uint8 wdCheckActive(void)` |
| 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 | `false`      Watchdog is currently disabled |
| | `true`       Watchdog is currently enabled |
| Example | `asdWriteUserDebug("Active = %u ReducedSafety = %u \n", wdCheckActive(), wdCheckReducedSafetyMode());` <br><br> For `asdWriteUserDebug`, refer to chapter 13.6. |
| See also | `wdSetSafetyMode, wdSetEvent, wdCheckReducedSafetyMode` |

## 13.5 API Functions (ES1135 LEDs)

The ES1135 Simulation Controller has three configurable LEDs. They are briefly described in chapter 6.2.3. The following interfaces to the LEDs are provided.

| **userLed[n]On** | |
| --- | --- |
| Function | Switches LED [n] on. |
| Syntax | `void userLed1On(void)`<br>`void userLed2On(void)`<br>`void userLed3On(void)` |
| Description | These functions switch the respective LEDs on. |
| Example | `userLed1On();` |
| See also | `userLed[n]Off, userLed[n]Toggle` |

| **userLed[n]Off** | |
| --- | --- |
| Function | Switches LED [n] off. |
| Syntax | `void userLed1Off(void)`<br>`void userLed2Off(void)`<br>`void userLed3Off(void)` |
| Description | These functions switch the respective LEDs off. |
| Example | `userLed1Off();` |
| See also | `userLed[n]On`<br>`userLed[n]Toggle` |

| **userLed[n]Toggle** | |
| --- | --- |
| Function | Toggles LED [n]. |
| Syntax | `void userLed1Toggle(void)`<br>`void userLed2Toggle(void)`<br>`void userLed3Toggle(void)` |
| Description | These functions toggle the respective LEDs. |
| Example | `userLed1Toggle();` |
| See also | `userLed[n]Off`<br>`userLed[n]On` |

## 13.6 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 `printf` |
| Description | This function displays user messages in the ASCET monitor window. |
| Example | `uint8 number = 1;`<br>`asdWriteUserError("Example %u \n", number);` |
| See also | `asdWriteUserDebug` |

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

# Index

Task
    Activation 411
    general description 411
TCP/IP protocol 70
Toggle 430

## W

Watchdog 79
    API Functions 422
    interrupt control 80
    modes 80
    period 79
    service 79, 80
    service register 79
wdCheckActive 429
wdCheckReducedSafetyMode 429
wdDisableAutoService 427
wdEnableAutoService 426
wdIntAck 428
wdIntDisable 427
wdIntEnable 427
wdIntPend 428
wdService 426
wdSetEvent 425
wdSetPeriod 424
wdSetReducedSafetyMode 424
wdSetSafetyMode 423