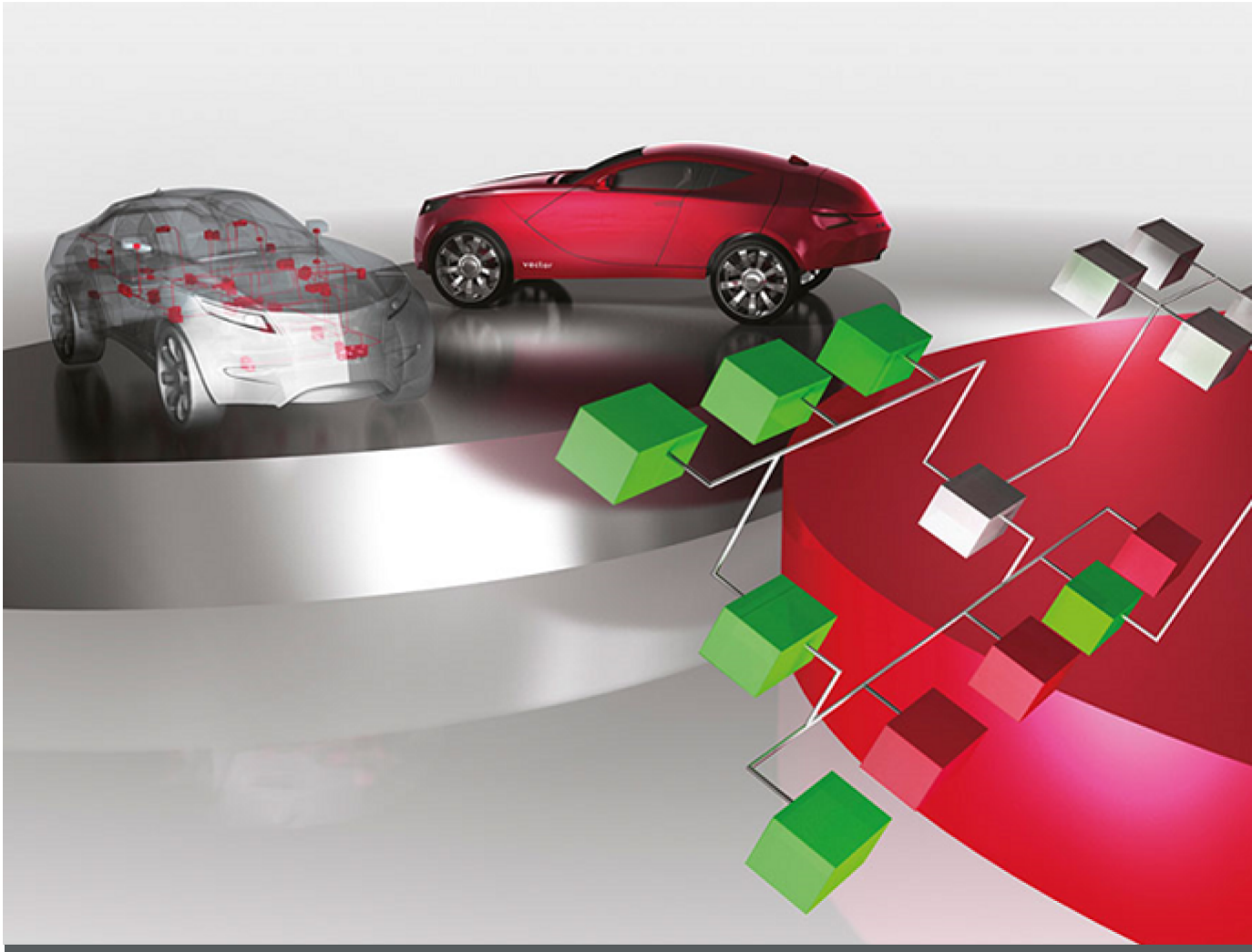
VECTOR



User Manual
VT System FPGA Manager
User Programmable FPGA

Version 1.9
English

Imprint

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

Table of Contents

1 Introduction

This chapter contains the following information:

1.1 VT System FPGA Manager at a Glance

Overview

For some VT System modules Vector offers a special version equipped with a user programmable FPGA. This allows for implementing custom functionality with direct access to the VT System module's hardware and very high performance. This makes the VT System modules even more flexible and applicable for almost any kind of application.

To aid you in the process of developing custom FPGA functionality for your FPGA enabled VT System modules Vector provides the VT System FPGA Manager. This utility helps to manage your projects and simplifies the process of compiling and downloading them to VT System modules. However the VT System FPGA Manager is no HDL code editor or modeling tool. Instead you can use your favorite IDE or alternatively Simulink® to actually write or model the HDL code implementing your desired functionality.

1.2 About this User Manual

1.2.1 Navigational Aids and Conventions

To find information quickly

This user manual provides you with the following navigational aids:

> At the beginning of each chapter you will find a summary of the contents
> The header shows which chapter and paragraph you are located in
> The footer shows which version the user manual refers to

Conventions

The following two charts show the spelling and symbol conventions used in this manual.

Style	Utilization
bold	Fields, interface elements, window and dialog names in the software. Accentuation of warnings and notes. **[OK]** Buttons are denoted by square brackets **File** | **Save** Notation for menus and menu commands
CANoe	Legally protected proper names and side notes.
`Source code`	File name and source code.
Hyperlink	Hyperlinks and references.
<Ctrl>+<S>	Notation for shortcuts.

Symbol	Utilization
	You can obtain supplemental information here.
	This symbol calls your attention to warnings.
	You can find additional information here.
	Here is an example that has been prepared for you.
	Step-by-step instructions provide assistance at these points.
	Instructions on editing files are found at these points.
	This symbol warns you not to edit the specified file.

1.2.2 Latest Information

Additional technical information

You may find additional technical information about your VT System:

- in the CANoe online help,
- on the Vector website www.vector.com (e.g. application notes), and
- in your CANoe installation.

Reference: You may find the **latest version of this manual** in your CANoe installation (start menu ⇨ CANoe ⇨ Help).

1.2.3 Certification

Certified Quality Management System

Vector Informatik GmbH has ISO 9001:2008 certification.
The ISO standard is a globally recognized quality standard.

CE Compliance

All VT System products comply with CE regulations.

1.2.4 Warranty

Limitation of warranty

We reserve the right to change the contents of the documentation and the software as well as the hardware design without notice. Vector Informatik GmbH assumes no liability for correct contents or damages which are resulted from the usage of the VT System FPGA Manager user manual. We are always grateful for references to mistakes or for suggestions for improvement, so as to be able to offer you even better-performing products in the future.

1.2.5 Support

Need support?

You can get through to our hotline by calling

+49 (711) 80670-200

or you can send a problem report to CANoe Support.

1.2.6 Registered Trademarks

Registered trademarks

All trademarks mentioned in this user manual, including those registered to third parties, are governed by the respective trademark laws and are the property of their respective owners. All trademarks, trade names or company names are or can be trademarks or registered trademarks of their particular owners. All rights which are not expressly allowed are reserved. Failure to explicitly note any given trademark within this user manual does not imply that a third party does not have rights to it.

- Windows is a trademark of the Microsoft Corporation.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc.
- Altera® and Quartus® are registered trademarks of Altera Corporation.

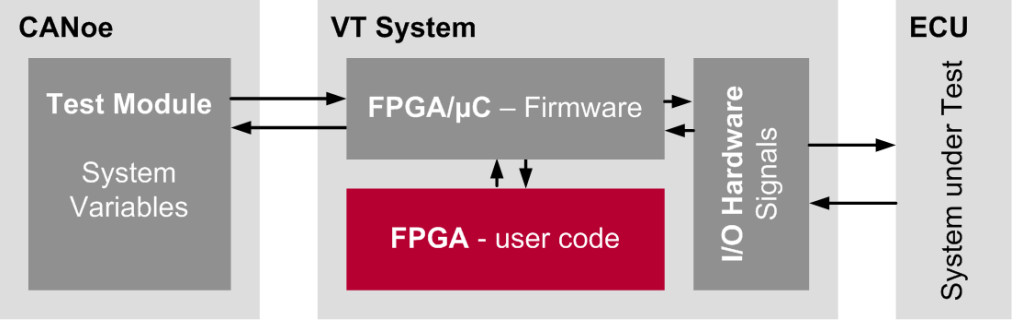
2 Basic Concepts

This chapter contains the following information:

2.1 Hardware Concept

User FPGA

On the VT System modules basically FPGAs from Altera® are used. Usually on the processor board there is one main FPGA which is responsible for the data and signal processing and accessing the I/O hardware. On the special FPGA processor board there is a second User FPGA which is user programmable as the following figure shows:



The User FPGA has no direct connection to CANoe or the I/O Hardware on the VT System module and therefore to the ECU. So the focus during creating your own functionality has to be set only to the data processing but not how to access the I/O hardware or CANoe. The controlling of the I/O hardware and the preparing of the raw data will be done by the main FPGA. The User FPGA will get the prepared raw measurement data cyclic from the main FPGA and the main FPGA will also output the data from the User FPGA via the I/O hardware when the signal changes. The communication with CANoe will be handled by the main FPGA in the same way. As the User FPGA is only signal based and has no direct hardware access for example switching relays or setting measurement or output ranges have to done as usual in your test sequence within CANoe.

Programming the User FPGA will be done very convenient by the VT System FPGA Manager via the firmware without the need of additional programming hardware or tools. The User FPGA communicates with the following signals:

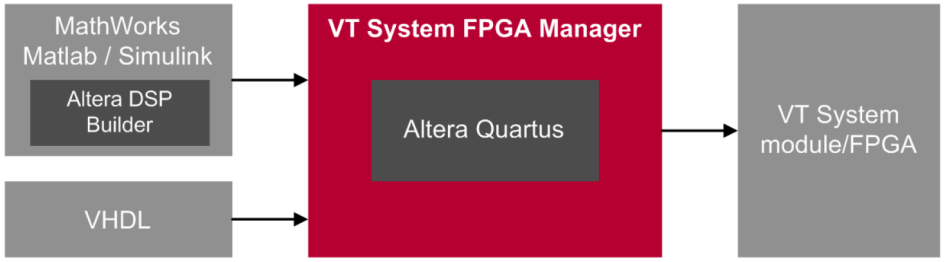
- Communication with CANoe will be done via system variables. They can be defined with the VT System FPGA Manager. Signals to CANoe will be updated cyclic.
- Input signals from the I/O hardware will be available cyclic at the User FPGA.
- Output signals to the I/O hardware will be updated on signal change.

2.2 Tool Chain

Supported design entries

As shown in the following figure the VT System FPGA Manager supports two design entries for the User FPGA:

- Graphical modeling with Simulink® with a special block set provided from Altera®
- Description with VHDL



The compilation of the project will be done with Altera® Quartus®, the design environment for Altera® FPGAs. This will be done automatically in the background because Quartus® is completely controlled by the VT System FPGA Manager. Downloading the compiled code will be done by the firmware also controlled by the VT System FPGA Manager.

Main tasks

So the main tasks of the VT System FPGA Manager are:

- **Project Management**:
  Add, change, and delete projects
- **Configuration**:
  Add, change, and delete System Variables
- **Compilation**:
  Translating the user code to gate level
- **Programming**:
  Download the compiled user code to the User FPGA temporarily or permanent

3 The VT System FPGA Manager

This chapter contains the following information:

3.1 Prerequisites

Overview

Before using the VT System FPGA Manager please make sure the following applications are installed correctly:

- Quartus® by Altera® (in most cases the free Quartus® Prime Lite suffices)
- Device files for Cyclone IV FPGAs by Altera® (automatically included for older versions of Quartus®)

If you plan to create your FPGA code by modeling it in Simulink® you also need the following software products:

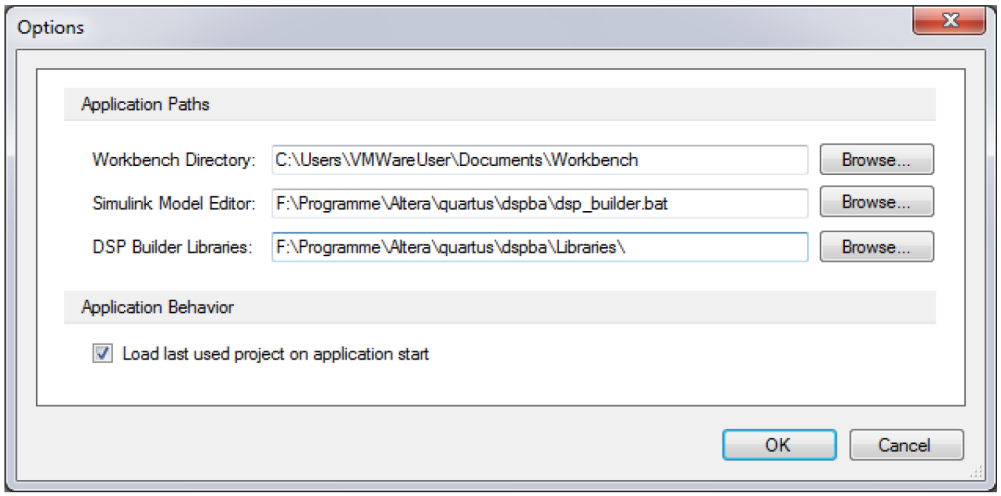
- MATLAB® with Simulink® by Mathworks
- DSP Builder by Altera®

Note: Vector recommends to use MATLAB® version 2011b or later. Although earlier versions work in most cases, too, there might be stability problems when using the DSP Builder Simulink® block set.

3.2 Configuring the VT System FPGA Manager

First configuration

Before using the VT System FPGA Manager a quick configuration is necessary. When the VT System FPGA Manager is started for the first time the configuration dialog is automatically shown. If you wish to change the settings later on you can always open the configuration dialog manually via **Tools|Options…**.



3.2.1 Setting the Workbench Directory

Workbench directory

The workbench directory is the folder in which all newly created projects will be stored. It is recommended to create a new, empty folder for this purpose. Please also make sure that you have write access to the folder and that enough hard disk space is available to store your projects.

3.2.2 Setting up the Simulink® Model Editor and DSP Builder® Libraries

Modeling in Simulink

If you wish to create your FPGA code by modeling it in Simulink® this setting is important. It specifies the path to the executable that will be used to open your .mdl files. Usually this is not MATLAB® but rather a batch file called **dsp_builder.bat**. This file is supplied by Altera® to open Simulink® with correctly set up block sets.

The DSP Builder Libraries path points to the location where the DSP Builder libraries are stored on your hard drive. Usually this is a subfolder in the Quartus® installation directory as shown in the image above.

3.2.3 Setting up the Quartus® Compiler

Compiling FPGA projects

The VT System FPGA Manager uses the Quartus® compiler to compile your FPGA projects. There is no extra setting in the options dialog for specifying a path to this compiler. Instead the path is read from the system's environment variable **PATH**. This variable should be modified automatically when installing Quartus®.

3.2.4 Example Projects

Examples in workbench directory

When setting up the VT System FPGA Manager for the first time two example projects will be copied to your workbench directory. The first project is created with MATLAB Simulink while the second project consists of manually written HDL code. The purpose of these projects is to demonstrate the basic functionalities of VT System FPGA modules and to serve as a starting point for your own projects. A detailed description of every example project can be found in the project's files, either the Simulink model or the main HDL code file.

3.3 Managing Projects

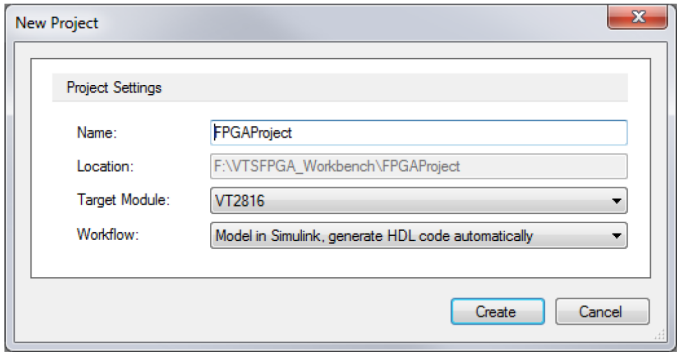
Overview

The VT System FPGA Manager allows you to create new projects, open projects for editing and delete existing projects.

3.3.1 Creating new Projects

Project creation dialog

To create a new project choose **File|New Project…** or click the corresponding button in the toolbar. Alternatively perform a right click into the Project Explorer and click on **New Project…**. This opens the project creation dialog.



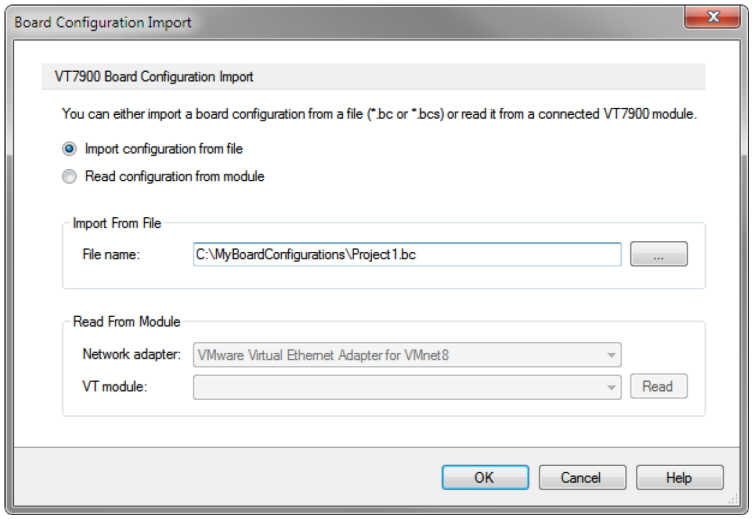
In this dialog the following settings have to be made:

- **Name**:
  Specifies the name of the new project. Please note that the name has to meet the naming conventions and must be unique.
- **Location**:
  This shows where the new project will be located on your hard drive. This setting can only be changed by changing the project's name or your workbench directory.
- **Target module**:
  Here the VT System module you are developing for has to be selected.
- **Workflow**:
  This setting lets you choose between two workflows. On the one hand you can write the HDL code for your project manually. On the other you can choose to create a Simulink® model and generate the HDL code from that model automatically. Both workflows are explained in detail in the following chapters.

3.3.2 Creating a New Project For The VT7900A

Board configuration import

When creating a User FPGA project for the VT7900A it is required to import a board configuration file. This file contains information about the application board mounted to the VT7900A. In addition it defines which features of the VT7900A User FPGA will be available for programming. The board configuration files have the extension .bc or .bcs and can be created by using the VT System Application Board Designer. This tool is shipped with CANoe, just like the VT System FPGA Manager.



In this dialog you have two options:

- **Import configuration from file**:
  This allows you to load a .bc or .bcs file previously saved with the VT System Application Board Designer.
- **Read configuration from module**:
  This allows you to read the configuration from a connected VT7900A module. Please make sure that CANoe is not running in the background. Otherwise the connected module may not be detected.

3.3.3 Opening Projects

Opening projects

To open a project for editing, simply double click on the project in the Project Explorer. You may open multiple projects at once. A tab will be created for each project allowing you to quickly switch between them. On application start-up the last used project will be automatically re-opened. This behavior can be changed via the options dialog.

3.3.4 Deleting Projects

Deleting projects

To delete a project click on **Edit| Delete Project…** or click the corresponding button in the toolbar. Alternatively you may perform a right click on the project in the Project Explorer and choose **[Delete]**. The project's folder will then be removed from your workbench directory.

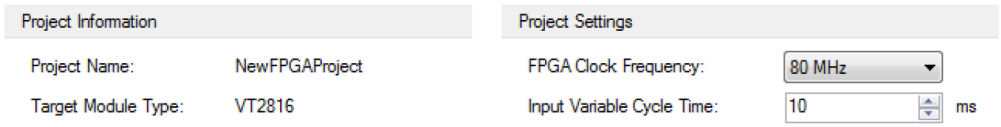
3.3.5 Editing Projects

Editing projects

After a project has been opened in the VT System FPGA Manger it can be edited.

Project settings

The top of the project editor shows some general project information, like the project's name and the targeted VT System module.



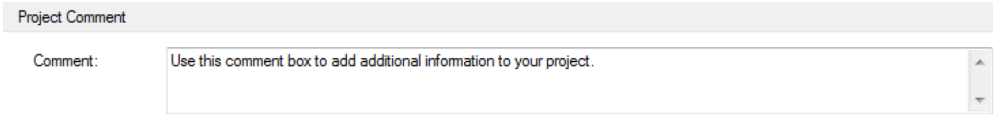
Here you can also make general settings that are available in every type of project, no matter which workflow you are using:

> **FPGA Clock Frequency**:
This is the clock frequency to be used by the FPGA. The more complex your HDL code gets the lower the frequency should be. If the frequency is too high for your project's complexity timing errors may arise. In that case the Quartus® compiler will generate adequate warning messages.

> **Input Variable Cycle Time**:
This setting specifies the cycle time in ms at which the signals from the User FPGA to CANoe will be updated.

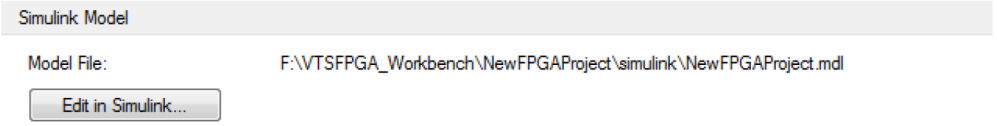
Project comment

The comment box allows you to add metadata to your project. For example it can be used to describe the functionality of your project or store information like the creation date or the author.



Simulink® workflow

This section only applies if you have created a project using the Simulink® workflow.

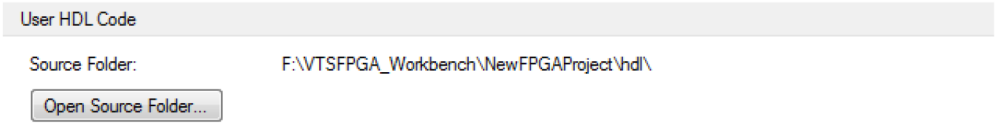


The second group in the project editor shows the path to your project's Simulink® model. Via the button **[Edit in Simulink®…]** you can quickly open the model file from within the VT System FPGA Manager. Please note that the button **[Edit in Simulink®…]** is disabled as long as unsaved changes exist in the project.

For detailed information on how to create a VT System FPGA project with Simulink® please refer to the corresponding chapter Modeling the Behavior of VT System FPGA Modules.

Manual workflow

This section only applies if you have created a project using the manual HDL coding workflow.

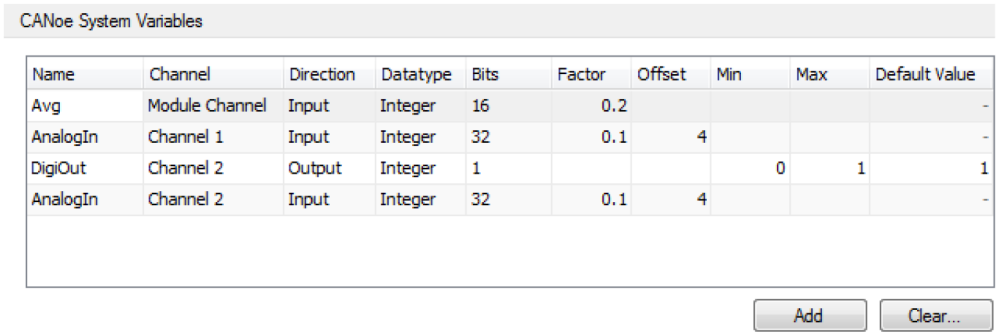


The second group in the project editor shows the path to your project's HDL source files. Via the button **[Open Source Folder…]** you can quickly open this folder from within the VT System FPGA Manager.

For detailed information on how to create a VT system FPGA project by manually writing the HDL code please refer to the corresponding chapter Writing HDL Code for VT System FPGA Modules.

System variables

The third group in the project editor lets you manage the CANoe system variables for your project.



On the one hand these variables can be used to send measurement values from the User FPGA to CANoe. On the other hand stimulation values can be transferred from CANoe to the User FPGA.

Adding/removing system variables

To add and remove variables in your project you can use the corresponding buttons in the lower right corner of the project editor. Alternatively you can open a shortcut menu by right clicking into the System Variable table.

Naming system variables

All system variables' names must conform to the CANoe naming conventions. For example names must not start with a cypher or contain blanks. Please see the CANoe online help for a full overview of the naming rules. Additionally the system variables' names may not use any reserved keywords. These include the names of automatically generated inputs and outputs (e.g. "debug_LED_", "in_signal_" or "out_dac_").

Configuring system variables

Each of the system variables offers the following settings:

Name	This is the name that will be displayed in CANoe. The name is also used in the HDL code as well as the Simulink® model. On every channel each system variable name must be unique and meet certain naming constraints.
Channel	In CANoe each system variable is assigned to one specific channel of the VT System module. That channel is specified via this setting.
Direction	Here you can determine if the variable shall be an input (measurement) variable or an output (stimulation) variable.
Data Type	This setting specifies if the variable shall contain floating point or integer numbers.
Bits	Specifies the number of bits used to store this variable's values. The value range and the resolution of the system variable are depending on this setting.
Factor	Every value that is transmitted from the User FPGA to CANoe is multiplied by this factor. Values transmitted from CANoe to the User FPGA are divided by this factor. See the explanation of factor and offset below for more details. This is an optional setting. If no value is entered a default value of 1.0 is assumed.
Offset	Every value that is transmitted from the User FPGA to CANoe is increased by this offset. Values transmitted from CANoe to the User FPGA are decreased by the offset. See the explanation of factor and offset below for more details. This is an optional setting. If no value is entered a default value of 0.0 is assumed.

Min	This setting specifies the minimum value that is displayed in CANoe for this system variable. This is an optional setting. If no value is entered no minimum value will be displayed.
Max	This setting specifies the maximum value that is displayed in CANoe for this system variable. This is an optional setting. If no value is entered no maximum value will be displayed.
Default Value	For output (stimulation) variables a default value can be specified with this setting. Until the system variable has been set for the first time in CANoe, it has this default value.
Time Stamp	By default all input (measurement) variables will be updated in CANoe with the same time stamp. This time stamp is computed by the VT System module automatically every time the variable values are transmitted to CANoe. By using this setting you can assign your variables to time stamps other than the default one. All variables assigned to a custom time stamp will then use this time stamp's value when being updated in CANoe. The non-default time stamps will appear as signals in the VHDL code or as blocks in the Simulink model. You can then write your own values to these time stamp signals/blocks.

Note: The **factor** and **offset** settings are applied in two directions. Values that are transferred from the User FPGA to CANoe are multiplied by the factor and then increased by the offset. Values transferred from CANoe to the User FPGA are decreased by the offset and then divided by the factor.

Note: All **system variables** will be automatically made available in the Simulink® model (if used) or the HDL code every time the project is saved. Please do not attempt to add, remove or modify system variables directly in the model or HDL code files. Always use the VT System FPGA Manager to perform these operations.

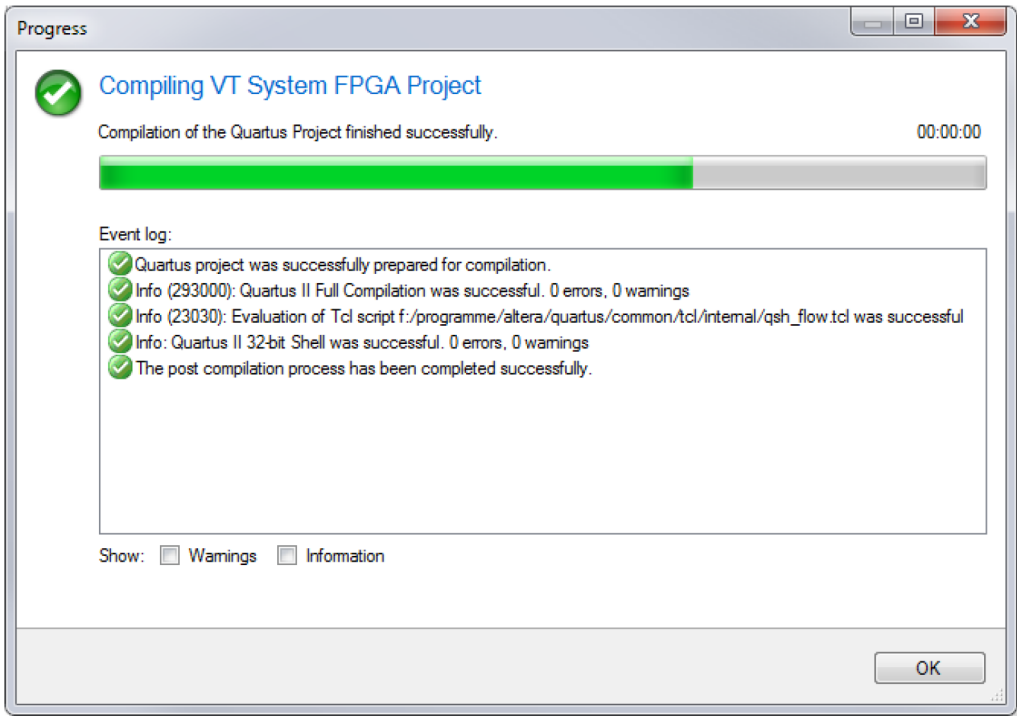
3.3.6 Compiling Projects

Compiling projects

To compile a project select **Tools|Compile Project…** or press the corresponding button in the toolbar. This will open up the compilation dialog.

Compilation dialog

The compilation dialog uses the Quartus® compiler to compile your FPGA project. All of the compiler's outputs will be displayed in the event log.



By default only success and error messages are displayed. To show warnings and additional information simply activate the corresponding checkboxes below the event log. Above the event log the current status of the compilation process and the time taken is displayed.

Troubleshooting

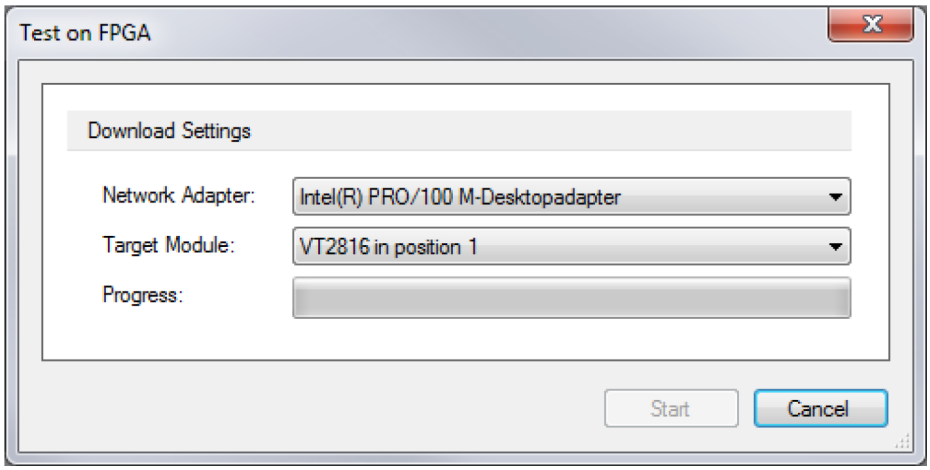
If the compilation of your project fails according error messages will be shown in the event log. These should help you fix any problems preventing a successful compilation.

In some cases it might also be helpful to open your project directly with Quartus® to gain additional information on the error. To do so, navigate to your project's folder in the workbench. Open the subfolder **quartus/SYN** where you can find the **.qpf** project file that can be directly opened with Quartus®.

3.3.7 Testing and Persisting Projects

Overview

After your project has been successfully compiled it can be downloaded to the VT System module. The VT System FPGA Manager offers two ways of downloading the project, one for testing and one for persisting the project.



Testing projects

When you choose to download a project for testing the project will only be stored temporarily on the VT System module. That means that it will be gone after the next power cycle. This function is intended to be used during active development of your project since it allows quickly performing development/testing iterations.

To download a project for testing purposes click on **Tools|Test Project…** or click the corresponding button in the toolbar. Alternatively you may perform a right click on the project in the Project Explorer and choose **[Test…]**.

Persisting projects

When you choose to persist a project the project will be stored permanently on the VT System module. This function is intended to be used when the project has been completed and is ready to be used.

To persist a project click on **Tools|Persist Project…** or click the corresponding button in the toolbar. Alternatively you may perform a right click on the project in the Project Explorer and choose **[Persist…]**.

3.3.8 Working with Project References

Overview

The VT System FPGA Manager allows you to work with projects that are not located in the workbench directory. This might be useful in cases where the project has to be stored externally, e.g. to be managed by a version control system like SVN.

Adding a reference

To add an externally stored project as a reference choose **File|Add Project Reference…** from the menu bar. Alternatively you can issue the same command from the Project Explorer's shortcut menu. Then navigate to the project folder and open the project.xml file located in that folder. The project is now added as a reference to your workbench. This is indicated by a slightly different icon in the Project Explorer.

Working with a reference

Referenced projects are treated exactly like regular projects. This means that all of the above information on project editing, compiling and persisting apply here as well.

Deleting a reference

When you delete a referenced project in the VT System FPGA Manager only the reference will be removed. The actual files of your project won't be deleted from your hard drive.

3.3.9 Clearing the User FPGA

Overview	With the VT System FPGA Manager you can easily clear the User FPGA. This will remove the project currently downloaded to the FPGA and thus disable any custom functionality. **Warning**: Clearing the User FPGA cannot be undone.
Clearing the FPGA	To clear the User FPGA click on **Tools| Clear User FPGA…**.. Then select the network adapter your VT System is connected to. Now all available FPGA enabled VT System modules will be shown. Select the one you want to clear and press **Start**.
Restoring the User FPGA's functionality	As mentioned above the clearing of the User FPGA cannot be undone. If you want to restore its previous functionality you have to re-download the corresponding project.

3.3.10 Updating a VT7900A Configuration

Overview	It may occur that you have to make changes to your VT900A board configuration after you have created an User FPGA project for your application board. In that case you have to update the board configuration in your User FPGA project.
Importing the new configuration	To update the board configuration oft he current project use the button **Update board configuration…** under **Project Information**. This will open up a dialog where you can either choose the updated .bc or .bcs file, or read the updated configuration from a connected VT7900A module.
Warning	**Warning**: Importing a new configuration will save all unsaved changes you have made to your project. In addition most oft he code files and/or your Simulink® model will be modified (e.g. some signals may be added or removed). Thus it is highly recommended to create a backup of your project before updating the board configuration.

3.3.11 Exporting and Importing Projects

Overview	The VT System FPGA Manager allows you to export your projects as .zip files. These files can then be imported back into the VT System FPGA Manager. This allows you to easily archive your projects or exchange them with collegues.
Exporting projects	Select one or more projects in the Project Explorer. The right click on the project and choose **Export…**. Now you can browse for a folder, where you want the zipped projects tob e stored. An even faster way of exporting projects ist o drag them from the Project Explorer to any folder in the Windows Explorer.
Importing projects	Importing projects works just like exporting them. Simply right click into the Project Explorer and then choose Import **Projects…**. Alternatively you can drag zipped projects from the Windows Explorer into the Project Explorer.

4 Writing HDL Code for VT System FPGA Modules

This chapter contains the following information:

4.1 Working with the FPGA Manager

FPGA Manager

If you want to write your VHDL code manually you must chose this workflow in the VT System FPGA Manager when you create a new project. In the source folder which is directly accessible from the VT System FPGA Manager a VHDL file with the top level entity will be created. Starting from this top level entity you can write your VHDL code with the editor of your choice. After VHDL coding is finished the project can be compiled and downloaded onto the User FPGA with the VT System FPGA Manager.

4.2 Coding with VHDL

Coding with VHDL

If you want to write your own VHDL files please remember to follow proper coding guidelines to assure that your code is working correctly with the used hardware. We recommend reading the Quartus® **handbook** by Altera® before writing your own code. This concerns especially the following chapters: **Recommended Design Practices** and **Recommended HDL Coding Styles**.

4.3 Folder Structure

Working folders

The work folder of your project consists of different subfolders.
These are:

- **bin**
  This folder contains the file for programming the FPGA and does therefore not need to be changed.
- **hdl**
  This folder contains the VHDL file created by the VT System FPGA Manager. If you create additional VHDL files for your project they should also be stored in this folder to guarantee that the VT System FPGA Manager can find them.
- **quartus**
  This folder folder contains the Quartus® II files which are created for your project. These files are usually used in the background by the VT System FPGA Manager without opening Quartus®. This folder contains the subfolders **hdl** and **syn**.
  - **hdl**
    This folder contains basic hdl files which should not be changed in any way.
  - **syn**
    This folder contains the Quartus® files of your project.

The **.qpf** file is the corresponding Quartus® project file.

4.4 Template File

VHDL template

By creating a new project with the VHDL workflow the VT System FPGA Manager creates a VHDL template file. This will be stored in the subfolder: **…\hdl** and must remain in this folder for the VT System FPGA Manager to use it properly. The template file contains the basic structure needed for a correct usage of the User FPGA.

Example file

```
-- ExampleProject_GN VHDL code file

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;
use IEEE.MATH_REAL.all;


ENTITY ExampleProject_GN IS
  PORT (

    -- @CMD=SYSVARSTART
    Sysvar1_ch0 : out std_logic_vector(31 downto 0) := (others => '0');
    Sysvar2_ch0 : in std_logic_vector(31 downto 0);
    -- @CMD=SYSVAREND

    -- @CMD=TIMESTAMPSTART
    -- @CMD=TIMESTAMPEND

    clk         : IN  std_logic;                                        -- Clock.clk
    areset      : IN  std_logic;                                        -- .reset
    h_areset    : IN  std_logic;

    in_adc_1    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_1.wire
    in_adc_2    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_2.wire
    in_adc_3    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_3.wire
    in_adc_4    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_4.wire
    in_adc_5    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_5.wire
    in_adc_6    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_6.wire
    in_adc_7    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_7.wire
    in_adc_8    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_8.wire
    in_adc_9    : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_9.wire
    in_adc_10   : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_10.wire
    in_adc_11   : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_11.wire
    in_adc_12   : IN  std_logic_vector(31 DOWNTO 0);                    -- in_adc_12.wire

    out_dac_1   : OUT std_logic_vector(31 DOWNTO 0) := (OTHERS => '0');  -- out_dac_1.wire
    out_dac_2   : OUT std_logic_vector(31 DOWNTO 0) := (OTHERS => '0');  -- out_dac_2.wire
    out_dac_4   : OUT std_logic_vector(31 DOWNTO 0) := (OTHERS => '0');  -- out_dac_4.wire
    out_dac_3   : OUT std_logic_vector(31 DOWNTO 0) := (OTHERS => '0');  -- out_dac_3.wire

    in_adc_update   : IN  std_logic_vector(12 DOWNTO 1);
    in_invar_update : IN  std_logic;
    in_timestamp    : IN  std_logic_vector(63 DOWNTO 0);

    in_ibc_1   : IN std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- in_ibc_1.wire
    in_ibc_2   : IN std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- in_ibc_2.wire
    in_ibc_3   : IN std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- in_ibc_3.wire
    in_ibc_4   : IN std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- in_ibc_4.wire

    out_ibc_1   : OUT std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- out_ibc_1.wire
    out_ibc_2   : OUT std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- out_ibc_2.wire
    out_ibc_3   : OUT std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- out_ibc_3.wire
    out_ibc_4   : OUT std_logic_vector(0 DOWNTO 0) := (OTHERS => '0');  -- out_ibc_4.wire

    debug_LED_1 : OUT std_logic_vector(0 DOWNTO 0)  := (OTHERS => '0');  -- debug_LED_1.wire
    debug_LED_2 : OUT std_logic_vector(0 DOWNTO 0)  := (OTHERS => '0');  -- debug_LED_2.wire
    debug_LED_3 : OUT std_logic_vector(0 DOWNTO 0)  := (OTHERS => '0');  -- debug_LED_3.wire
    debug_LED_4 : OUT std_logic_vector(0 DOWNTO 0)  := (OTHERS => '0');  -- debug_LED_4.wire
    debug_LED_5 : OUT std_logic_vector(0 DOWNTO 0)  := (OTHERS => '0')   -- debug_LED_5.wire
    );
END;

ARCHITECTURE rtl OF ExampleProject_GN IS


BEGIN


END ARCHITECTURE rtl;  -- of  ExampleProject_GN
```

The picture above shows the VHDL template file for the VT2816 with two system variables (**Sysvar1_ch0** & **Sysvar2_ch0**) created according to the settings in the VT System FPGA Manager. The other input and output signals will be created automatically by the VT System FPGA Manager and depend on the used VT System module. Manually created system variables will always be restored to the settings in the VT System FPGA Manager when the project will be saved the next time. So it is not recommended to change them directly in the VHDL file. This should also be

considered for comments behind the signals. So additional comments should be placed outside the entity declaration. All output signals will be initialized with zeros so that they have a predetermined output value in case they aren't used in the project.

The VT System FPGA Manager adds suffixes to the name of the project and also to the names of the created system variables. The name of the entity always consists of your chosen name plus the suffix **_GN**. System variables always get a suffix based on the channel they are declared for. This means if you create a system variable and chose channel 1 for it the suffix will be **_ch1**. The suffix **_ch0** is used if the signal is mapped to the module instead of a single channel. The direction and the number of bits for the system variables are adjusted at the FPGA Manager. Please remember that variables with only one bit are not declared as **std_logic** but as **std_logic_vector(0 downto 0)**.

The other signals (including clock and reset signal) are a connection to the I/O hardware and will always be created automatically by the VT System FPGA Manager. These signals are dependent from the used VT System module and are described in the chapter Signal Reference.

The architecture of the top level entity is empty. Here is the entry to start writing VHDL code for the User FPGA.

4.5 Adding/Removing/Changing System Variables

Working with system variables

You can add, remove or change the settings of the system variables from your VHDL file by using the VT System FPGA Manager. In all cases you have to save your project afterwards in the VT System FPGA Manager to adapt these changes to the VHDL file. Manually editing the system variables in the VHDL file directly is not recommended as the complete top level entity will overwritten with the current settings in the VT System FPGA Manager by the next save of the project.

4.6 Using Quartus®

Quartus®

It is not intended and therefore not recommended to open Quartus®, because Quartus® is controlled completely by the VT system FPGA Manager. But if it is necessary to work with Quartus® directly you have to compile your project before downloading to the FPGA always with the VT system FPGA Manager, because the VT system FPGA Manager creates the final file which is needed to program the FPGA. Also downloading to the User FPGA is only possible with the VT system FPGA Manager. The basic VHDL code of the User FPGA is stored encrypted and not accessible. So using some functions like the RTL Viewer is not possible. In this case we recommend therefore creating a separate Quartus® project and including the code you want to analyze there.

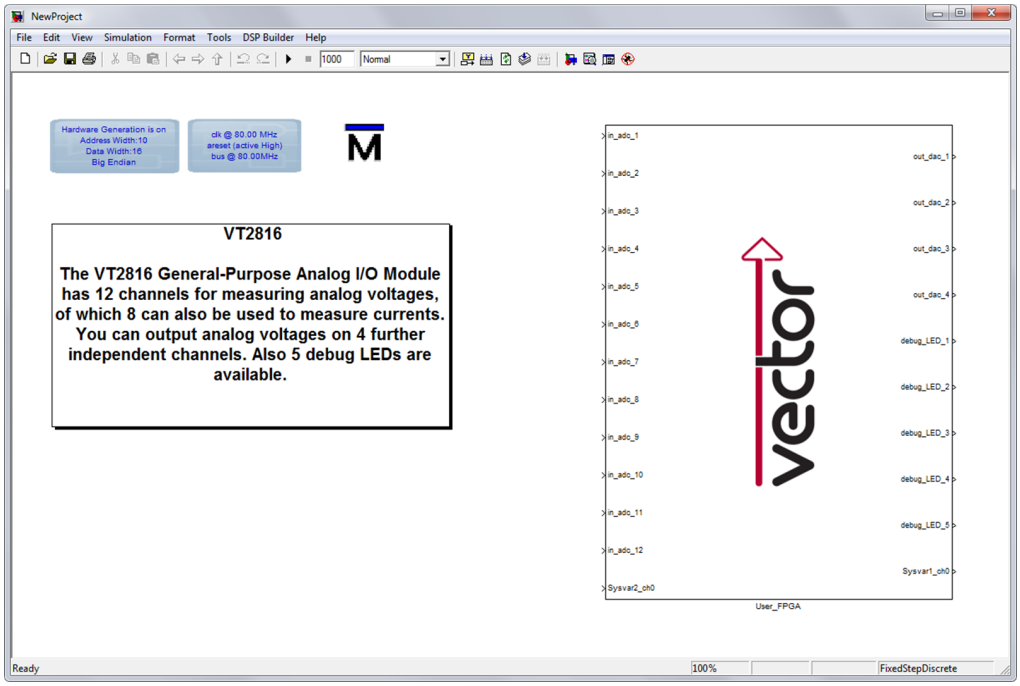
5 Modeling the Behavior of VT System FPGA Modules

This chapter contains the following information:

5.1 Using the Model File template

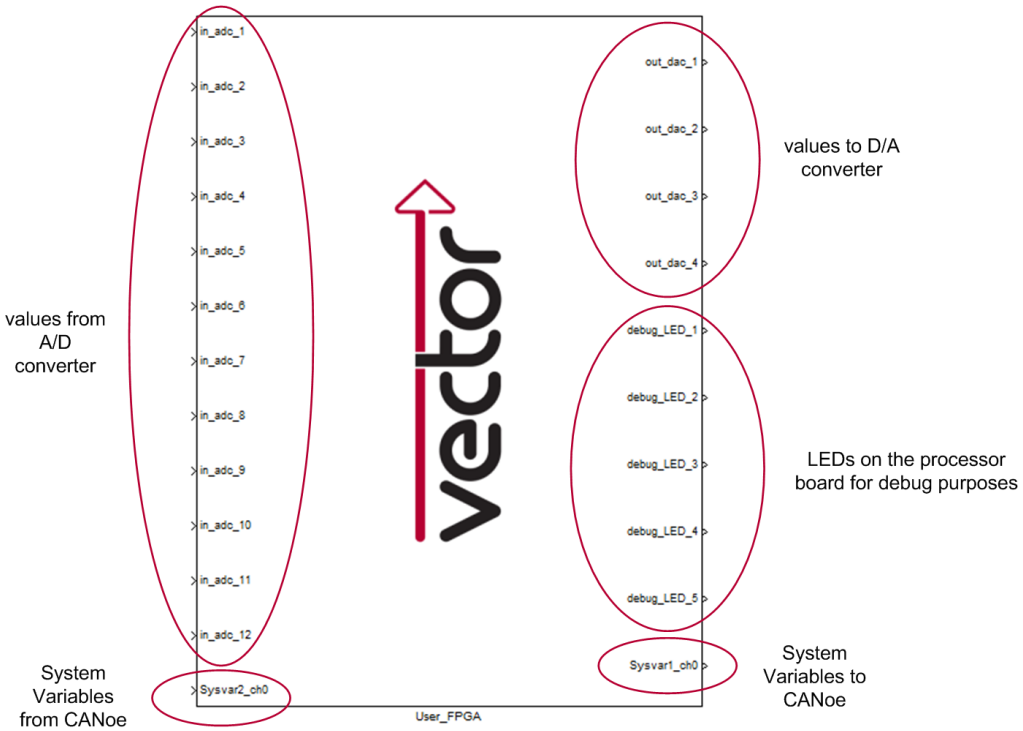
Simulink® model file

While creating a new FPGA project and choosing the Simulink® workflow, the VT System FPGA Manager will generate a Simulink® model file (.mdl). You have to open the Simulink® model file always with the VT system FPGA Manager and not directly with Simulink®. Otherwise the Altera® DSP Builder is not included correctly. After opening this file with the VT system FPGA Manager, dependent on the chosen VT System module the template appears:

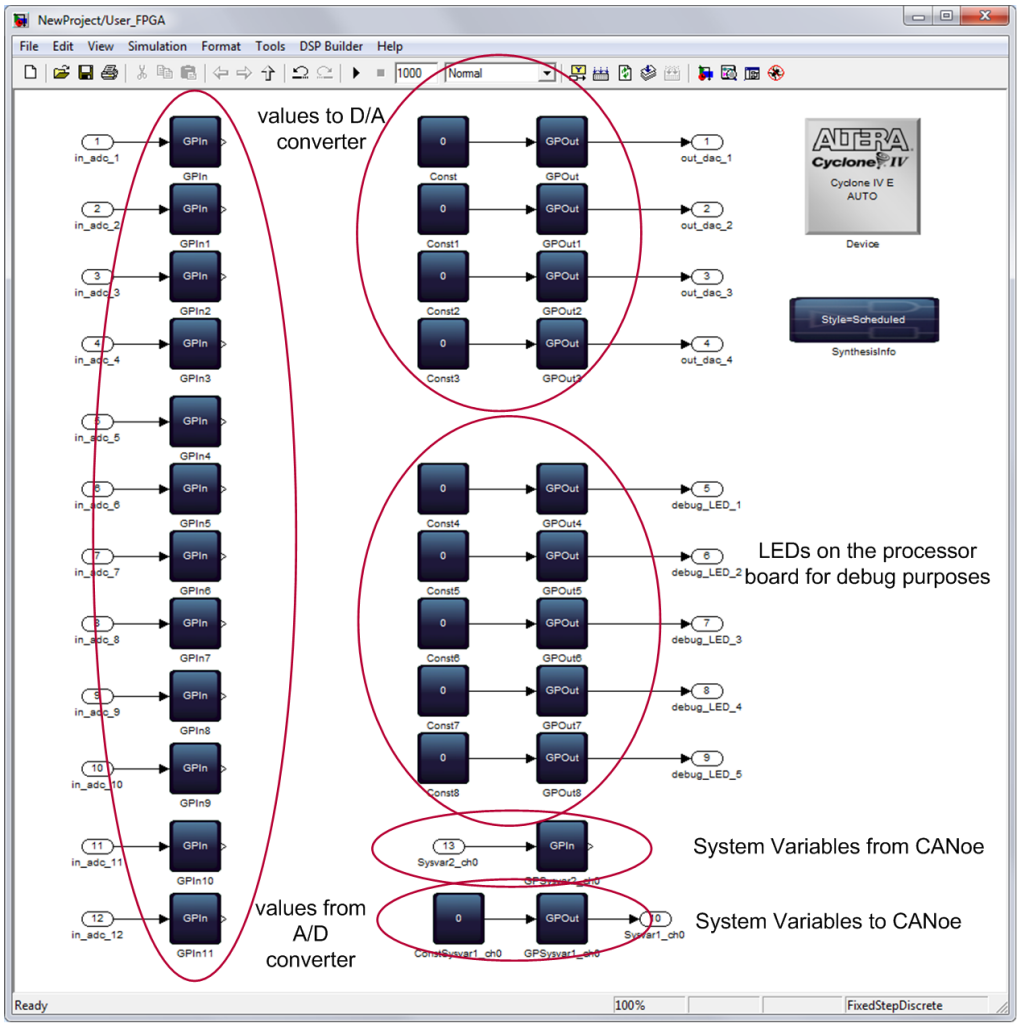


On the top left corner there are two blocks to adjust some basic settings like the used clock frequency or reset signal. These settings will be done by the VT System FPGA Manager and therefore it is not necessary to set these parameters manually.

The block on the right side includes the Simulink® model of the User FPGA. At this top level block all the available signals are listed. The following figure shows a detailed view of the top level block for the VT2816.



By double-clicking on this block the next hierarchy level appears.



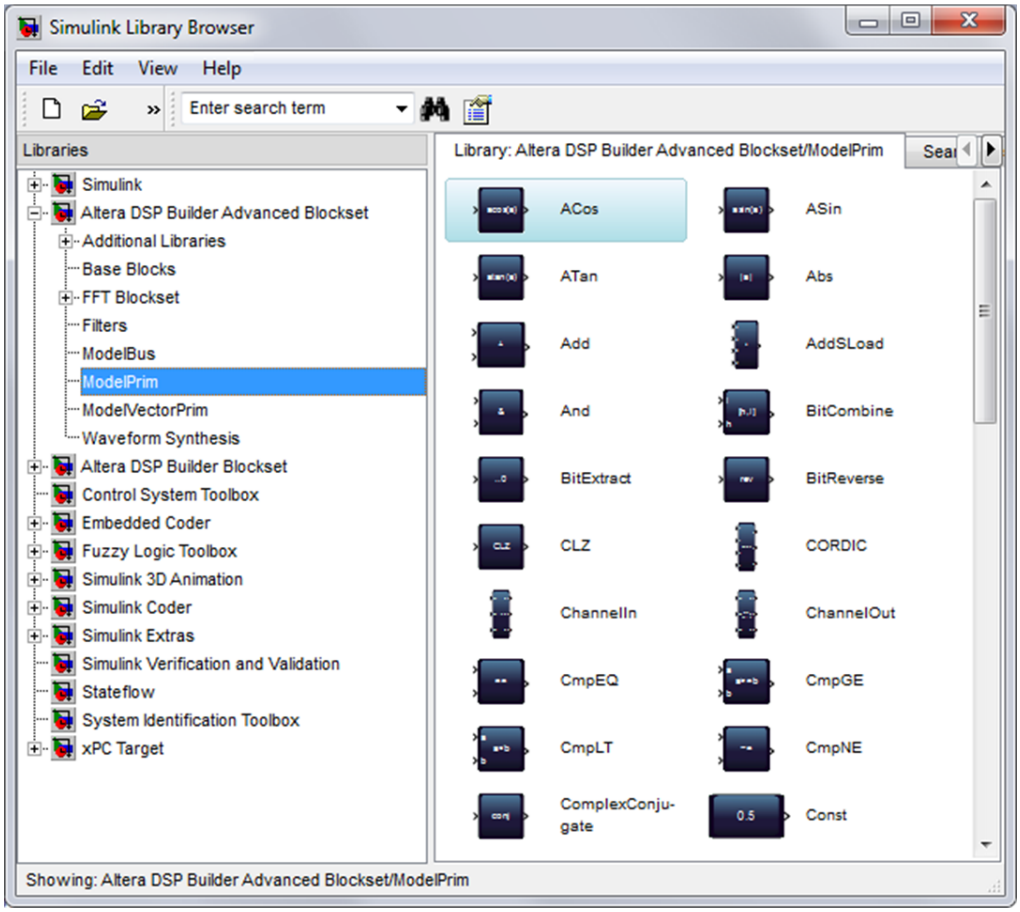
There are mainly two signals types available. System variables for communication with CANoe which are defined with the VT System FPGA Manager will automatically appear in the User FPGA model template. Signals for accessing the I/O hardware will appear also automatically dependent on which VT System module was chosen. In this figure the VT2816 is shown.

Basic signals like clock and reset are wired automatically and therefore have not to be considered in the User FPGA model. For debug purposes there are five LEDs on the processor board. All output signals are initial set to zero. You can find a detailed description of the signals in the chapter Signal Reference.

Altera® DSP Builder Advanced Blockset

At this level you can use blocks from the Altera® DSP Builder Advanced Blockset to create a model. The connection to the in- or output signals has to be done with the special in- and output blocks GPin and GPout which are also included in the Altera® DSP Builder Advanced Blockset and automatically placed by the template. If Altera® DSP Builder is installed correctly the Altera® DSP Builder Advanced Blockset is available in the Simulink® Library Browser which can be opened under **View|Library Browser**:

Simulink® Library Browser



There is also another block set available, the Altera® DSP Builder Blockset. As Altera® does not recommend using this block set for new developments, this block set is not supported by the VT System FPGA Manager.

For more detailed information about the Altera® DSP Builder and Simulink® it is recommended to read the documentation at www.altera.com and www.mathworks.com.

5.2 Simulation

After modeling the User FPGA with the Altera® DSP Builder Advanced Blockset the model can be simulated cycle accurate in Simulink® like normal Simulink® models by setting up the simulation and then pressing the start simulation button. Therefore also the normal Simulink® block sets can be used on the top level of the User FPGA model to create a test bench. Simulink® blocks can be used for simulation but they will not be translated to VHDL code.

5.3 Compiling and Download to FPGA

When pressing the start simulation button the corresponding VHDL Code for the User FPGA model is automatically generated and put in the folder where also the manually written VHDL files are stored. After code generation the compiling of the project and downloading to the FPGA will be done with the VT System FPGA Manager again.

5.4 Adding/Removing/Changing System Variables

Working with system variables

When system variables have to be added, removed or changed this has always to be done with the VT System FPGA Manager and not directly in Simulink®. The modifications do not only affect the User FPGA model, but also some settings in the firmware and CANoe. So the model file has to be closed and the modifications have to be done with the VT System FPGA Manager. After saving the VT System FPGA Manager the changes will be adapted to the .mdl file and the User FPGA model can be opened again. If the system variables will be changed in the Simulink® model directly, the changes will be overwritten with the system variables settings in the VT System FPGA Manager when the model file is opened again after saving the project in the VT System FPGA Manager.

6 Signal Reference

In this chapter you find the following information:

6.1 Clock and Reset Signal

Signals

The clock and the reset signal (**clk** and **areset**) are only available for VHDL coding. For the modeling with Simulink® the clock and reset signal are used automatically. The clock frequency can set to 10, 40 or 80 MHz in the VT System FPGA Manager; the reset is always high active and can't be changed.

6.2 Additional Signals

The VT2004A and VT2816 FPGA modules don't support the following signals.

ADC update signal

This signal exists only for modules that are able to measure analog signals. The ADC update signal (**in_adc_update**) is set to high for one clock cycle when new values from the ADC are received. This allows to recognize a new value even if it is identical to the previous value.

Variable update

The variable update signal (**in_invar_update**) is set to high for one clock cycle when new signals (excluding the ADC update signals) are received. This includes signals send from CANoe.

Timestamp

The timestamp signal (**in_timestamp**) is a 64bit value that shows the current time of the module. This value can be send as a timestamp to CANoe and is explained in the chapter „Time Stamps".

6.3 Debug LEDs

Debugging

For debug purposes it is possible to use five LEDs located on the FPGA processor board. The FPGA processor board is a stacked PCB located near to the front panel on the VT System module.

The LEDs are controlled with simple Boolean signals, named **debug_LED_1** to **debug_LED_5**. These signals can also be watched with an oscilloscope by connecting a cable at the provided connector on the processor board. The configuration of this male connector is:

PIN	Configuration
1	LED1
2	GND
3	LED2
4	GND
5	LED3
6	GND
7	LED4
8	GND
9	LED5
10	GND

6.4 Time Stamps

Default time stamps

By default all input (measurement) variables will be updated in CANoe with the same time stamp. This time stamp is computed by the VT System module automatically every time the variable values are transmitted to CANoe.

Custom time stamps

In the System Variables table in the VT System FPGA Manager you can assign your variables to time stamps other than the default one. All variables assigned to a custom time stamp will then use this time stamp's value when being updated in CANoe.

The non-default time stamps will appear as signals in the VHDL code or as blocks in the Simulink model. You can then write your own values to these time stamp signals/blocks.

Name	Direction	Data Type	Description
out_time_stamp_group1 … out_time_stamp_group32	OUT	uint64	Holds the current time stamp for all measurement System Variables associated with this time stamp group.

6.5 VT1004A

VT1004A channels

The VT1004A has 4 analog/digital input channels.

The input data of the ADC is forwarded to the User FPGA. The ADC has a sample rate of 250 kHz and a resolution of 16 Bit. This resolution will be converted for the User FPGA, so that a signed int32 value is available cyclic every 4 µs with a resolution of 1 µV. Negative values are represented by a two's complement.

The digital input state of a channel is also available at the User FPGA with some delay.

Name	Direction	Data Type	Resolution	Cycle Time	Description
in_adc_1 … in_adc_4	IN	int32	1 µV	typ. 4 µs (delay typ. 20 µs)	data from ADC input channel 1 to 4

Name	Direction	Data Type	Delay Time	Description
In_signal_1 … In_signal_4	IN	boolean	typ. 1 µs	data from digital input channel 1 to 4

6.6 VT2004A

VT2004A channels

The VT2004A has 4 analog output channels.

The data to the 14 Bit DAC is only forwarded when the output signal changes. For outputting a signal, the curve type has to be set to User FPGA in CANoe. More information can be found in the CANoe help.

Name	Direction	Data Type	Resolution	Cycle Time	Description
out_dac_1 … out_dac_4	OUT	int32	1 µV	on change (delay typ. 1 µs)	data to DAC output channel 1 to 4

6.7 VT2516A

VT2516A channels

The VT2516A has 16 analog/digital input channels and 16 analog output channels.

The input data of the ADC is forwarded to the User FPGA. The ADC has a resolution of 12 Bit and works in multiplexed mode. The resolution will be converted for the User FPGA, so that a signed int32 value is available cyclic with a resolution of 1 µV. Negative values are represented by a two's complement. Because of the multiplexed sampling of the channels, the cycle time of the input signals varies.

The data to the 8 Bit DAC is only forwarded when the output signal changes. For outputting a signal, the curve type has to be set to User FPGA in CANoe. More information can be found in the CANoe help.

The digital input state of a channel is also available at the User FPGA with some delay.

Name	Direction	Data Type	Resolution	Cycle Time	Description
in_adc_1 … in_adc_16	IN	int32	1 µV	min.20 µs max.40 µs	data from ADC input channel 1 to 16
out_dac_1 … out_dac_16	OUT	int32	1 µV	on change (delay typ. 1 µs)	data to DAC output channel 1 to 16

Name	Direction	Data Type	Delay Time	Description
In_signal_1 … In_signal_16	IN	boolean	typ. 1 µs	data from digital input channel 1 to 16

6.8 VT2710

VT2710 channels

The VT2710 has 17 channels that can be used to map user defined system variables. These channels include one module channel, four piggy channels, two SPI channels, two UART channel, two RS-485 channels, two I2C channels, two LVDS channels and two DIO channels.

For each of the two topmost connectors of the VT2710 the same set of variables exists. Their functionality depends on the User FPGA configuration in CANoe. The following table shows the variables for connector 1 (left). The variables for connector 2 look identical, but use "con2" instead of "con1" in their names. In addition there are four variables to control the two LVDS channels of the VT2710.

Name	Direction	Description
io_in_con1_dio1	IN	Digital input 1 from connector 1.
io_in_con1_dio2_rs232_rx	IN	Digital input 2 from connector 1. Can also be used as RS-232 RX.
io_in_con1_dio3	IN	Digital input 3 from connector 1.
io_in_con1_dio4_rs485_rx	IN	Digital input 4 from connector 1. Can also

Name	Direction	Description
		be used as RS-485 RX.
io_in_con1_dio5	IN	Digital input 5 from connector 1.
io_in_con1_dio6	IN	Digital input 6 from connector 1.
io_in_con1_dio7_i2c_sda	IN	Digital input 7 from connector 1. Can also be used as I2C SDA.
io_in_con1_dio8_i2c_scl	IN	Digital input 8 from connector 1. Can also be used as I2C SCL.

Name	Direction	Description
io_out_con1_dio1_rs232_tx	OUT	Digital output 1 from connector 1. Can also be used as RS-232 TX.
io_out_con1_dio2	OUT	Digital output 2 from connector 1.
io_out_con1_dio3_rs485_tx	OUT	Digital output 3 from connector 1. Can also be used as RS-485 TX.
io_out_con1_dio4	OUT	Digital output 4 from connector 1.
io_out_con1_dio5_rs485_oe	OUT	Digital output 5 from connector 1. Can also be used as RS-485 output enable.
io_out_con1_dio6	OUT	Digital output 6 from connector 1.
io_out_con1_dio7_i2c_sda	OUT	Digital output 7 from connector 1. Can also be used as I2C SDA.
io_out_con1_dio8_i2c_scl	OUT	Digital output 8 from connector 1. Can also be used as I2C SCL.

Name	Direction	Description
lvds_in_1 and lvds_in_2	IN	LVDS input from LVDS connector 1/2.
lvds_out_1 and lvds_out_2	OUT	LVDS output to LVDS connector 1/2.

6.9 VT2816

VT2816 channels

The VT2816 has 12 analog input channels and 4 analog output channels.

The input data of the ADC is forwarded to the User FPGA. The ADC has a sample rate of 250 kHz and a resolution of 16 Bit. This resolution will be converted for the User FPGA, so that a signed int32 value is available cyclic every 4 µs with a resolution of 1 µV. Negative values are represented by a two's complement.

The data to the 14 Bit DAC is only forwarded when the output signal changes. For outputting a signal, the curve type has to be set to User FPGA in CANoe. More information can be found in the CANoe help.

Name	Direction	Data Type	Resolution	Cycle Time	Description
in_adc_1 … in_adc_12	IN	int32	1 µV	typ. 4 µs (delay typ. 20 µs)	data from ADC input channel 1 to 12
out_dac_1 … out_dac_4	OUT	int32	1 µV	on change (delay typ. 1 µs)	data to DAC output channel 1 to 4

6.10 VT2848

VT2848 channels

The VT2848 has 48 digital input channels and 48 digital output channels.

It is recommended to use channels 1 to 16 for input purposes and channels 33 to 48 for output purposes.

The input state of a channel is available at the User FPGA with some delay.

A change of the output state is forwarded also with some delay. For outputting a signal, the curve type has to be set to User FPGA in CANoe. More information can be found in the CANoe help.

Name	Direction	Data Type	Delay Time	Description
in_signal_1 … in_signal_48	IN	boolean	typ. 1 µs	data from digital input channel 1 to 48
out_signal _1 … out_signal_48	OUT	boolean	typ. 1 µs	data to digital output channel 1 to 48

6.11 VT7900A

VT7900A channels

The number of channels as well as their names are not fixed for the VT7900A. Instead they can be freely configured by using the VT System Application Board Designer.

The available signals also depend on the board configuration used for this project.

Name	Direction	Data Type	Description
In_i2c_scl1	IN	boolean	I²C 1 Clock (Pin State)
out_i2c_scl1	OUT	boolean	I²C 1 Clock (Output)
in_i2c_sda1	IN	boolean	I²C 1 Data (Pin State)
out_i2c_sda1	OUT	boolean	I²C 1 Data (Output)

Name	Direction	Data Type	Description
inout_databus_d0 …	IN/OUT	boolean	Databus Data

Name	Direction	Data Type	Description
inout_databus_d15			
out_databus_d0_7_out_active	OUT	boolean	Databus Output Active Data 0-7
out_databus_d8_15_out_active	OUT	boolean	Databus Output Active Datat 8-15

Name	Direction	Data Type	Description
in_spi_miso	IN	boolean	SPI MISO
in_spi_n_mirq	IN	boolean	SPI Interrupt Input (Low Active)
out_spi_spck	OUT	boolean	SPI Clock
out_spi_mosi	OUT	boolean	SPI MOSI
out_spi_n_mcs0 ... out_spi_n_mcs3	OUT	boolean	SPI Chip Select (Low Active)

Name	Direction	Data Type	Description
out_dout_0 ... out_dout_3	OUT	boolean	Digital Output

Name	Direction	Data Type	Description
in_din_0 ... in_din_3	IN	boolean	Digital Input

Name	Direction	Data Type	Description
out_databus_a1 ... out_databus_a5	OUT	boolean	Databus Address
out_databus_n_wr	OUT	boolean	Databus Write Enable (Low Active)
out_databus_n_rd	OUT	boolean	Databus Read Enable (Low Active)
out_databus_n_bhe	OUT	boolean	Databus Reserved
out_databus_n_pclk	OUT	boolean	Databus Clock
out_databus_n_conv	OUT	boolean	Databus Conversion Start (Low Active)
out_databus_n_reset	OUT	boolean	Databus Reset (Low Active)
in_databus_n_sirq	IN	boolean	Databus Additional Interrupt Input (Low Active)
in_databus_n_wait	IN	boolean	Databus Wait (Low Active)
in_databus_n_busy	IN	boolean	Databus Busy (Low Active)
in_databus_n_irq0	IN	boolean	Databus Interrupt Input 0 (Low Active)

Name	Direction	Data Type	Description
in_databus_n_irq1	IN	boolean	Databus Interrupt Input 1 (Low Active)

7 Troubleshooting

In this chapter you find the following information:

7.1 Problems and Solutions

FAQ: The VT System FPGA Module is not Recognized

Problem

When trying to test or persist a project in the VT System FPGA Manager your VT System module does not appear in the target module selection box.

Solution

> Make sure the module is correctly inserted into the VT System rack
> Check that the VT System is turned on and properly connected to the computer
> Verify that no other applications that may access the VT System (e.g. CANoe) are running
> Check your network adapter's settings in the Windows control panel (see CANoe online help for details)

FAQ: MATLAB® Crashes When Using the DSP Builder Block Set

Problem

When opening the Altera® DSP Builder standard block set in the Simulink® library browser MATLAB® crashes.

Solution

> The crashes should only occur when opening the DSP Builder standard block set. For programming the User FPGA this block set is not needed. Instead use the advanced block set as described in the chapters above.
> This problem seems to occur only in DSP Builder 12.0. So if possible try to install more up to date DSP Builder versions.

FAQ: DSP Builder does not Create HDL Code due to a Missing License

Problem

When trying to create HDL code from a Simulink® model the DSP Builder shows an error message regarding a missing license.

Solution

Please make sure you have a valid DSP Builder license file. Usually this is a .**dat** file. Store this file in some safe place on your hard drive. Then make sure the environment variable **LM_LICENSE_FILE** exists and is pointing to your license file. After that reboot your computer. On the next start-up DSP Builder should find the license file and generate the HDL code properly.

FAQ: Demo projects are broken or missing

Problem

The VT System FPGA Manager is shipped with several sample projects for all supported VT System modules. But how can you restore such a sample to its original state after you have modified or deleted it?

Solution

Go to the options dialog and look for a section called “Demo Projects”. Here you can restore each of the sample projects. Please be careful if you have made any modifications that you would like to keep. These changes will get lost when you restore the corresponding project.

FAQ: Quartus® does not Compile HDL Code due to a Missing License

Problem

When trying to compile your project from the VT System FPGA Manager you get an error about a missing license.

Solution

Quartus® needs a license file to compile your projects. This license file is part of your CANoe installation and can be found in **\Exec32\VTSFPGAAssets\lic_VT_User_FPGA.dat**.

To allow Quartus® finding this license file the user environment variable **LM_LICENSE_FILE** has to be set accordingly. Usually this is done automatically by the VT System FPGA Manager. In some cases a reboot of the computer may be required to make the changes visible to Quartus®.

If this variable is not created and set automatically, you may do so manually. To do so, make sure the user environment variable **LM_LICENSE_FILE** exists (if not, create it). Then set its value to the absolute path of the file **lic_VT_User_FPGA.dat** in your CANoe installation. If the variable already exists, append a semicolon and the file path to the variable’s value.

In most cases the should solve the problem. If the error still occurs, start the Quartus application. Go to **Tools | License Setup**. Insert the contents of the computer’s environment variable **LM_LICENSE_FILE** into the field **License file**. Now you should see a new entry called **Vector Informatik** in the list **Licensed AMPP/MegaCore functions**.

More Information

- News
- Products
- Demo Software
- Support
- Training Classes
- Addresses

www.vector.com