Rev. 7 - March 14th, 2025

Sci-Compiler Quick Start

Graphical Programming Language for CAEN Open FPGA Boards

SCI-COMPILER

CAEN
Tools for Discovery

User Guide
GD6520

Purpose of this Guide

This document is the Sci-Compiler Quick Start Guide. It contains information about the compatibility of the software with CAEN programmable boards, the installation and configuration of the software, its main features and some guided examples of usage

Change Document Record

Date	Revision	Changes
September 11th, 2018	00	Initial release.
February 22nd, 2019	01	Modified Technical Support.
May 17th, 2019	02	Added support to R5560 board.
June 20th, 2019	03	Added example for CITIROC ASIC. Now removed.
March 25th, 2021	04	General modifications to add x5560 Digitizer family compatibility
November 2nd, 2022	05	General and complete review of the Guide following new Sci-Compiler features and license management
February 14th, 2023	06	Added Chap. Firmware Simulation and SciSDK. Revised boards compatibility for Remote Customization Service
March 14th, 2025	07	Chap. 1,2,3 revised

Symbols, Abbreviated Terms and Notation

FPGA	Field Programmable Gate Array
OTP	One Time Password
IDE	Integrated Development Environment
GUI	Graphical User Interface
HDL	Hardware Description Language

Reference Document

[RD1] UG973 - Vivado Design Suite User Guide (available on Xilinx website)

[RD2] MNL-1065 - Intel FPGA Software Installation and Licensing (available on Altera website)

[RD3] UM5175 - V2495/VX2495 User Manual

[RD4] UM6506 – DT5550 User Manual

[RD5] 00106/03:V1718.MUTx/09 – Mod. V1718/VX1718 Manual

[RD6] 00106/03:A2818.MUTx/01 – Mod. A2818 PCI Optical Link

[RD7] 00102/08:A3818.MUTx/10 – Mod. A3818 PCI Express Optical Link

[RD8] UG908 - Vivado Design Suite User Guide, Programming and Debugging (available on Xilinx website)

[RD9] UG-USB81204 - Intel FPGA USB Download Cable User Guide

[RD10] UM6952 – R5560 User Manual

[RD11] UM7970 – DT5560SE User Manual

[RD12] UM8412 – R5560SE User Manual

[RD13] UM8717 – 2740/45 User Manual

[RD14] GD8712 – Sci-Compiler SMART kit Quick Start

[RD15] https://www.xilinx.com/products/intellectual-property/axi-amm-bridge.html#overview

[RD16] https://www.caen.it/products/caen-wavedump2/

[RD17] https://www.caen.it/products/compass/

Manufacturer contact

CAEN S.p.A.
Via Vetraia, 11 55049 Viareggio (LU) - ITALY
Tel. +39.0584.388.398 Fax +39.0584.388.959
www.caen.it | info@caen.it

Limitation of Responsibility

If the warnings contained in this manual are not followed, CAEN will not be responsible for damage caused by improper use of the device. The manufacturer declines all responsibility for damage resulting from failure to comply with the instructions for use of the product. The equipment must be used as described in the user manual, with particular regard to the intended use, using only accessories as specified by the manufacturer. No modification or repair can be performed.

Disclaimer

The information contained herein has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies. CAEN spa reserves the right to modify its products specifications without giving any notice; for up to date information please visit www.caen.it.

Made in Italy

We remark that all our boards have been designed and assembled in Italy. In a challenging environment where a competitive edge is often obtained at the cost of lower wages and declining working conditions, we proudly acknowledge that all those who participated in the production and distribution process of our devices were reasonably paid and worked in a safe environment (this is true for the boards marked "MADE IN ITALY", while we cannot guarantee for third-party manufactures).

Table of Contents

List of Figures

List of Tables

1 Introduction

Overview

The increasing use of programmable logic devices in trigger and data acquisition systems makes clear that having a general-purpose platform and technicians dedicated to the firmware development is becoming more and more important. The advantage of employing programmable logic devices with respect to standard logic modules (like NIM logic modules) is remarkable: a single programmable logic device includes the potentiality of hundreds of thousands of standard logic modules. However, the use of specific programming languages like VHDL or Verilog for the firmware development could represent a limitation in the spread of these powerful devices.

We introduce an innovative method to **simplify the firmware development** on **CAEN Open FPGA boards**. This method is based on a **block-diagram-based** software, called **Sci-Compiler**, that uses a set of high-level functionalities (blocks) to mask the real firmware coding. In this way, it is possible to obtain a VHDL firmware code starting from a rather simple block diagram structure.

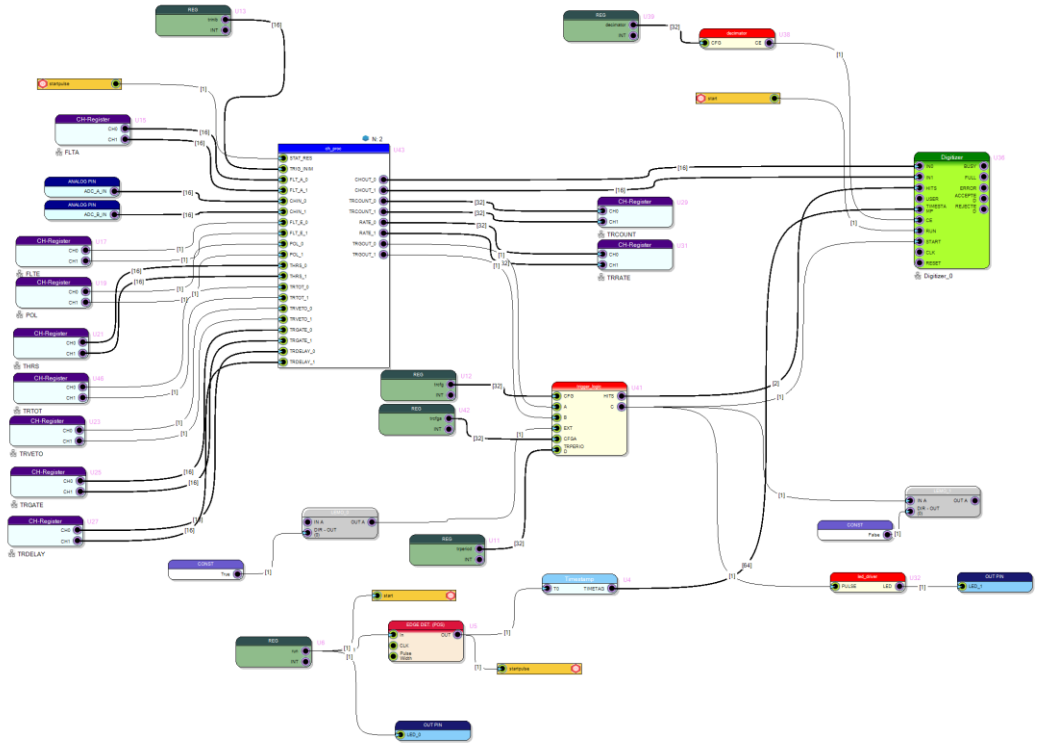


Figure 1.1: example of a block diagram in Sci-Compiler

Sci-Compiler improves and speeds-up the R&D phase. Placing and interconnecting the available blocks (e.g. oscilloscope, TDC, MCA, charge integration, etc.) on a diagram, Sci-Compiler is able to **automatically generate a VHDL piece of code** that implements the required function and deploy it to the FPGA. In this way, even a non-expert user can write his own firmware code, focusing on the functional blocks of the application to be implemented more than on the VHDL/Verilog code to achieve the goal.

Moreover, Sci-Compiler installs a **Software Development kit** (SciSDK) compatible with the custom firmware for any supported board. The SDK is made of drivers, libraries and example codes in C++ and Python, so that it is straightforward to build a custom DAQ software that can run on Windows or Linux.

The Sci-Compiler software allows to develop both purely **digital applications**, exploiting blocks like logic gates, scaler, state machine, and **analog processing applications**, such as custom Multichannel Analyzers, charge integration, Pulse Shape Discrimination, Waveform recording with custom trigger logic, and many others. Additionally, the Sci-Compiler provides the blocks to easily read and test the CITIROC and PETIROC ASICs by Weeroc, enabling the user to develop a custom sequencer for the ASIC control.

Sci-Compiler is able to **automatically** generate the VHDL firmware code implementing the functions requested in the block diagram. The compilation is highly accessible thanks to the optional **Remote Customization Service**, which allows you to compile the firmware even without having a full FPGA compiler software running locally on your PC, thus simplifying a lot the software setup. Using the Remote Customization Service, Sci-Compiler will take care of adding to your project the hard-coded parts that remain fixed in each firmware and will makes the final firmware file into the Remote Center and in the MyCAEN+ area as well, ready to be deployed onboard.

CAEN makes the **Remote Customization Service** and **Yearly Upgrade Plan** available, to offer a high-quality software product, with the needed flexibility to meet the users' requirements.

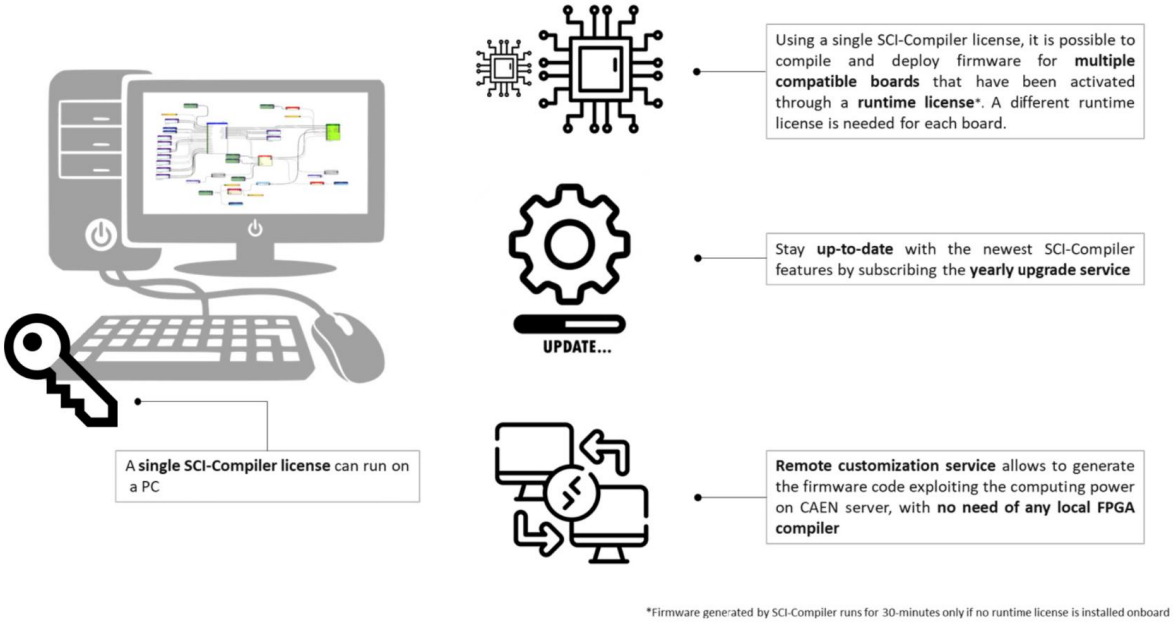


Figure 1.2: Summary of Sci-Compiler operative scenario, with description of the additional services available.

Supported boards

Sci-Compiler fully supports the following hardware devices.

Note: only the units for which a *Runtime License* has been installed can fully run a firmware generated by Sci-Compiler. A different Runtime License code is needed for each board. If a runtime is not detected onboard, generated custom firmware runs for a limited period (30 minutes). Contact CAEN to purchase a Runtime License.

Note: The *Runtime License* is included with the purchase of 5560 Digitizer Family units, DT5550, DT5550W, x495 boards and Sci-Compiler SMART kit.

Supported hardware	References
2730 Digitizer Family - 32 Channel 14-bit 500MS/s	https://www.caen.it/subfamilies/2730-digitizer-family/
2740/2745 Digitizer Family - 64 Channel 16-bit 125 MS/s	https://www.caen.it/subfamilies/2740-digitizer-family/ https://www.caen.it/subfamilies/2745-digitizer-family/
5560 Digitizer Family - 32/128 Channel 14-bit 125 MS/s	https://www.caen.it/subfamilies/55xx-digitizer-family/
DT5550 - 32 Channel Programmable Readout System	https://www.caen.it/products/dt5550/
DT5550W - 32/64/128 Channel SiPM readout system	https://www.caen.it/products/dt5550w/
x495 Family - Programmable Logic Unit	https://www.caen.it/products/v2495/ https://www.caen.it/products/dt5495/
Sci-Compiler SMART kit	https://www.caen.it/products/Sci-compiler-smart/

Table 1.1: table of the boards supported by Sci-Compiler.

Sci-Compiler PRO vs. Sci-Compiler SMART

CAEN provides two different software licenses:

- Sci-Compiler **PRO** is compatible with all supported hardware units.
- Sci-Compiler **SMART** is a license available within the Sci-Compiler SMART kit only (including DT1260 , 2 ch. 12bit @ 65MS/s) and it cannot work for the other hardware units. Refer to **[RD14]** for more information about the SMART kit.

Note: Sci-Compiler SMART is provided exclusively in a bundle kit with DT1260 unit.

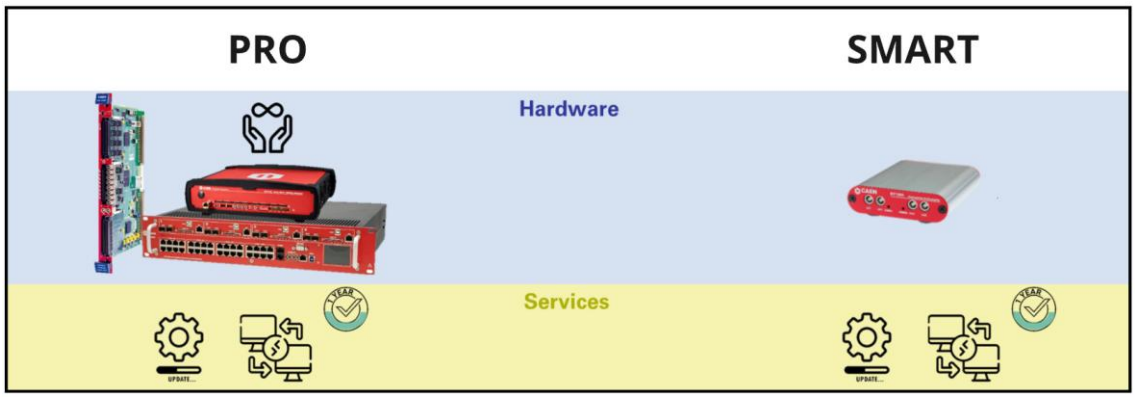


Figure 1.3: Sci-Compiler PRO vs. Sci-Compiler SMART

What do I receive?

	Part
	Sci-Compiler License Coupon: **PRO** – works for **all** Sci-Compiler supported board **SMART** – works **only** for Sci-Compiler **SMART kit unit**

Table 1.1: delivered kit.

What is included in my license?

	Description
	Sci-Compiler license is a **lifetime-working** license. Its expiration date (1 year after activation by default) refers to the possibility to access optional services (see below *Yearly Upgrade Service* and *Remote Customization Service*) E.g.: if your license expires *2024, November 30th*, it means that you can access software upgrades up to that date. If you do not renew your license, you won't be able to download latest software releases after that date. You will still be able to work lifetime with any Sci-Compiler version released before *2024, November 30th*
	The Sci-Compiler PRO license supports compilation for any compatible hardware available at the time of purchase. The Sci-Compiler SMART license supports compilation for DT1260 unit only (*) (*) DT1260 is sold only in a bundle kit with the Sci-Compiler SMART license

	The *Yearly Upgrade Service* gives access to the latest software releases, including new features and enhancements. **One year** of free upgrades is included by default at the time of purchase of the USB Dongle and can be renewed at the USB Dongle expiration date.
	The *Remote Customization Service* allows the user to compile the firmware even without having a full FPGA compiler software running locally on the PC, thus simplifying the software setup. **One year** of service is included by default at the time of purchase of the USB Dongle and can be renewed at the USB Dongle expiration date.

Table 1.2: description of USB Dongle factory status when you receive it.

Ordering Options

Product	Description	Product Code
Sci-Compiler PRO License	SW555 - Sci-Compiler PRO License	WSW555PROXAA
Runtime License	Sci-Compiler runtime license for Digitizers 2.0 Series	WSW55RUNTIME
Remote Customization Service	1-year remote customization service + upgrade for Sci-Compiler	WSW55RCSXAAA
Remote Customization Service – 5-year package	5 years remote customization service + upgrade for Sci-Compiler	WSW55RCSX5YA
Sci-Compiler SMART kit	Sci-Compiler SMART kit	WKSCISMARTXA

Table 1.2: Table of ordering options

2 Requirements & Installation

System Requirements

The Sci-Compiler is compliant with Windows 10/11 OS. The software requires Microsoft .NET 4.0 or higher. If the framework is not available on your PC, it can be downloaded from Microsoft© website.

The firmware compilation and FPGA programming require **Xilinx Vivado** (for DT5550, DT5550W and x5560 family) or **Intel Quartus** (for V2495/DT5495) software installed on the user computer. The minimum versions for which the compatibility is guaranteed are indicated in **Table** 2.1. Alternatively, it is possible to use the Remote Customization Service to generate the firmware code exploiting the computing power on CAEN serves, with no need of any local FPGA compiler installed on your PC.

Note: The Remote Customization Service and local compilers can be used at the same time. Sci-Compiler can be configured at any time to perform compilation on your PC or use the Remote Customization Service, when available for your license

To exploit the features of FPGA programming and firmware testing, the needed drivers should be installed (refer to the instructions on the correspondent User Manual).

OS	Windows Framework	Supported Boards	Local option (*)		Remote Customization Service	
			Compilation	Simulation	Compilation	Simulation
10/11	.NET Framework 4.0 or higher	V2730 DT2730 V2740 VX2740 DT2740 V2745 VX2745 DT2745	VIVADO 2020.2– Full edition (**)		✓	
		R5560A R5560SEA DT5560SE	VIVADO 2017.4 – Webpack			
		R5560B R5560SEB	VIVADO 2017.4 – Full edition	VIVADO 2017.4 – Webpack		
		DT5550 DT5550W	VIVADO 2017.4 – Webpack	COMING SOON		
		SMART kit	VIVADO 2020.2– Webpack			
		V2495 DT5495	Quartus Prime 18.0 – Lite Edition	COMING SOON		

(*) Third-party software not required if using remote customization service

(**) Vivado and JESD204 licenses are required. If needed, 10Gbps Ethernet license is also necessary

Table 2.1: Sci-Compiler system requirements.

Sci-Compiler License Activation

The full version of Sci-Compiler requires a valid license for operation. The license must be purchased separately and has a specific expiration date. During its validity period, the license allows unrestricted use of Sci-Compiler, while software updates are limited to a specific release version, which is indicated when running the software. All available software releases, up to the latest version compatible with your license, can be downloaded from the MyCAEN+ area.

Note: in order to extend your license for further upgrade, you should enable the Yearly Upgrade Service. Contact CAEN for more information

Note: in order to be able to activate your license, you need to register as a MyCAEN+ user at https://www.caen.it/become-mycaenplus-user/

Once the license has been purchased, the user must activate it by following these steps:

- Open the e-mail received from ***My Sci-Compiler***
- Locate and copy the alphanumeric license coupon provided in the e-mail

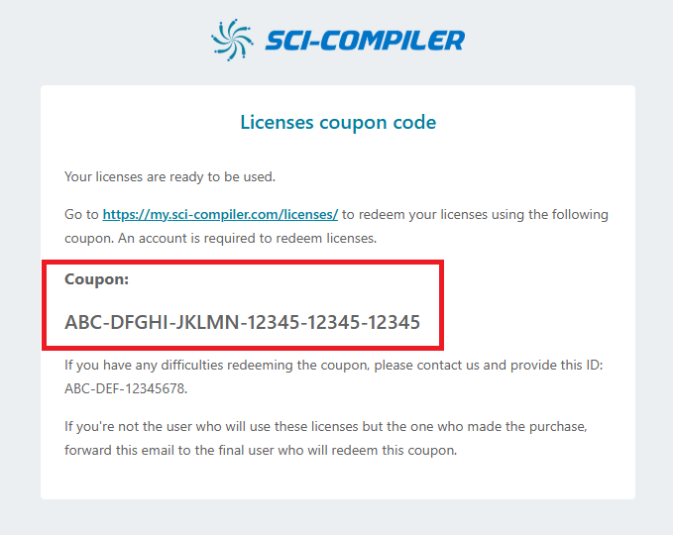


Figure 2.1: Alphanumeric license coupon in Sci-Compiler e-mail.

- Login into your **MyCAEN+** area: [*https://www.caen.it/become-mycaenplus-user/*]

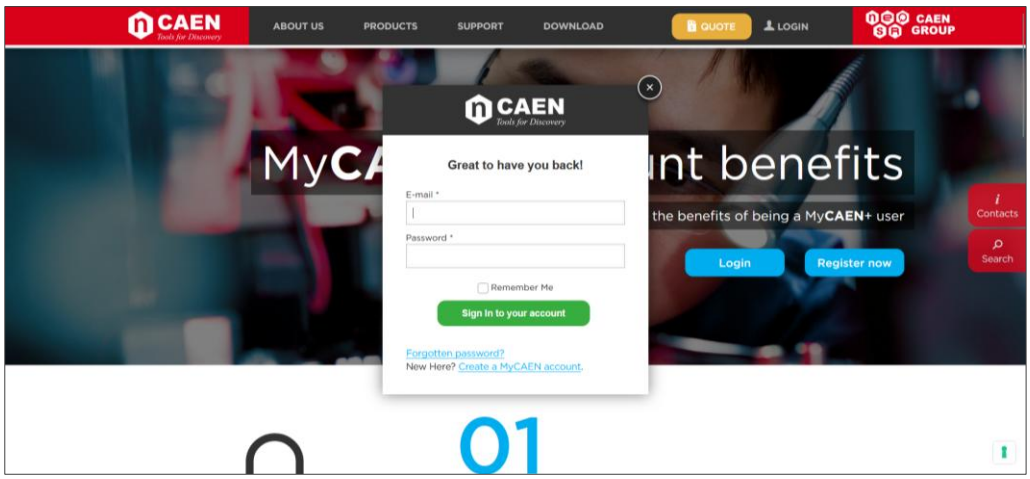


Figure 2.2: MyCAEN+ login web page.

- Surf into the **Sci-Compiler tab**

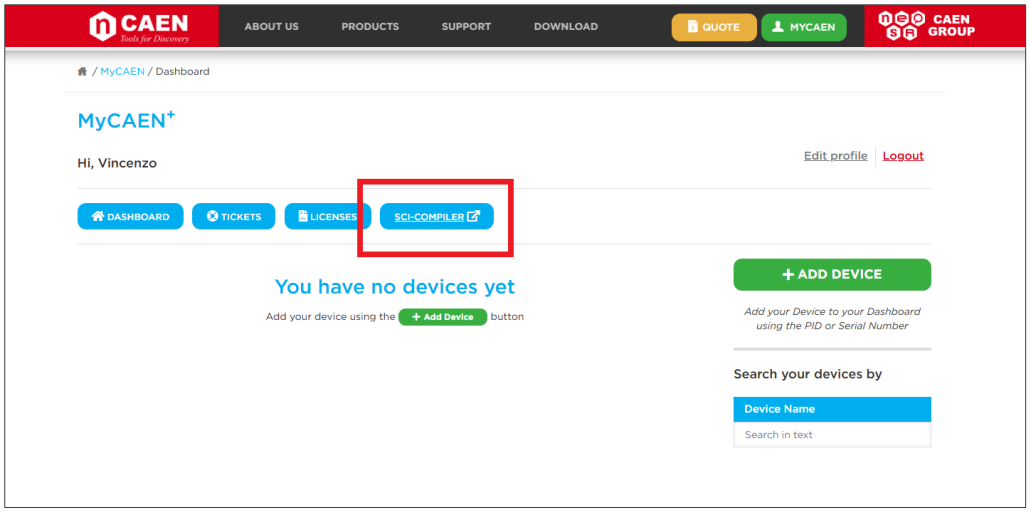


Figure 2.3: Sci-Compiler tab button in the MyCAEN+ page.

- A web page on the Sci-Compiler website will open. The user must log in using either **Sci-Compiler** account credentials (if previously registered) or **MyCAEN+** account credentials

Figure 2.4: Sci-Compiler's login page.

- Once logged in, the user will be redirected to his **Sci-Compiler portal**. From the main dashboard, click on the ***Licenses*** section to proceed

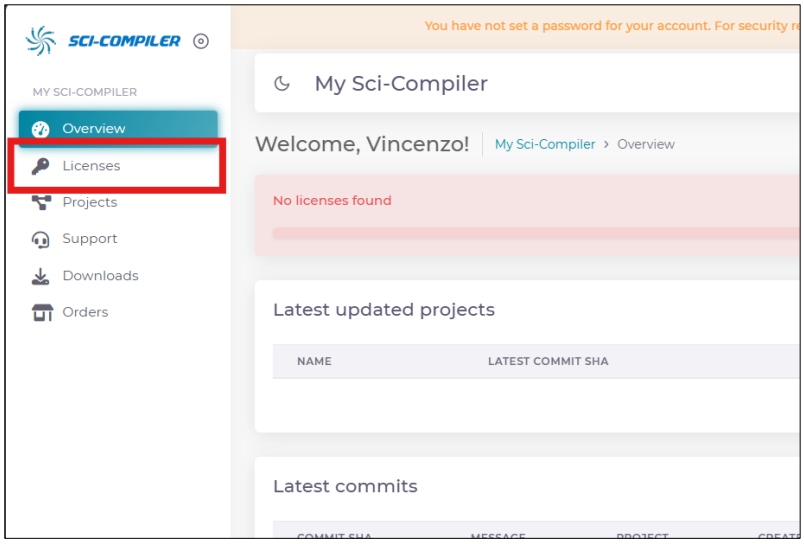


Figure 2.5: Licenses section in Sci-Compiler portal

- On the right side of the screen, click on the **"+"** button. Since the license is being activated using a coupon, select ***Redeem Coupon*** from the available options. The ***Request Trial*** option allows users to apply for a 30-day trial; however, this trial does not include access to the **Remote Customization** service. Any activated license can be imported in multiple MyCAEN+ account via ***Add new license*** button

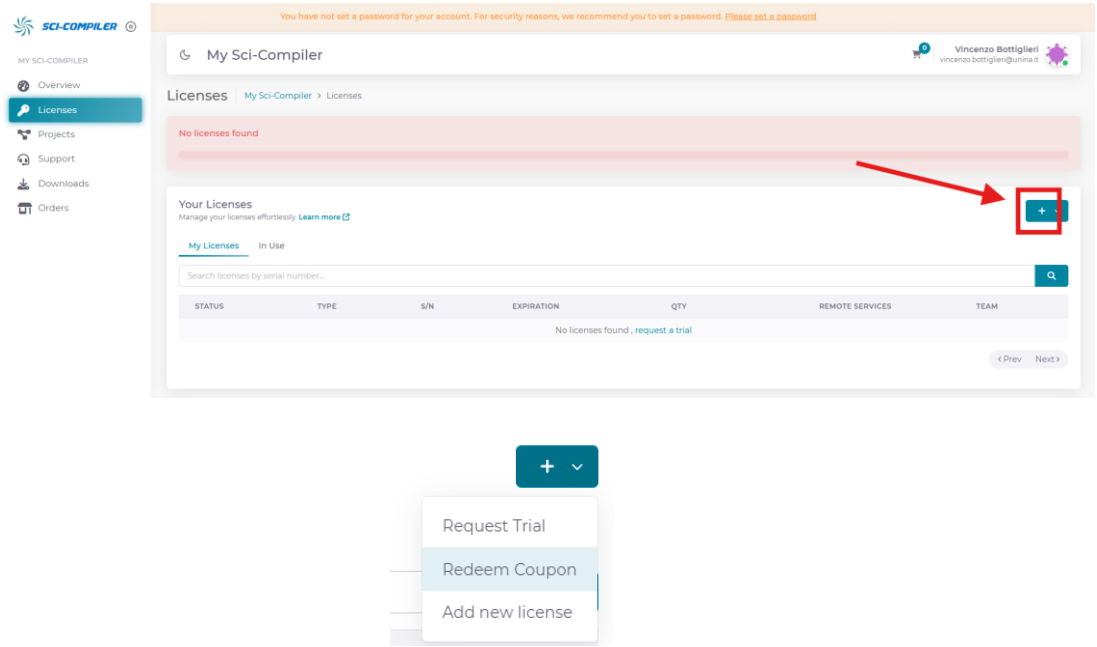


Figure 2.6: Redeeming a coupon in Sci-Compiler portal.

- After entering the alphanumeric coupon, the license will be successfully activated. The **Sci-Compiler** portal will display a "***Success!"*** banner confirming the activation. In the ***Your Licenses*** table, users can view detailed information about the newly activated license, including its validity period and associated features

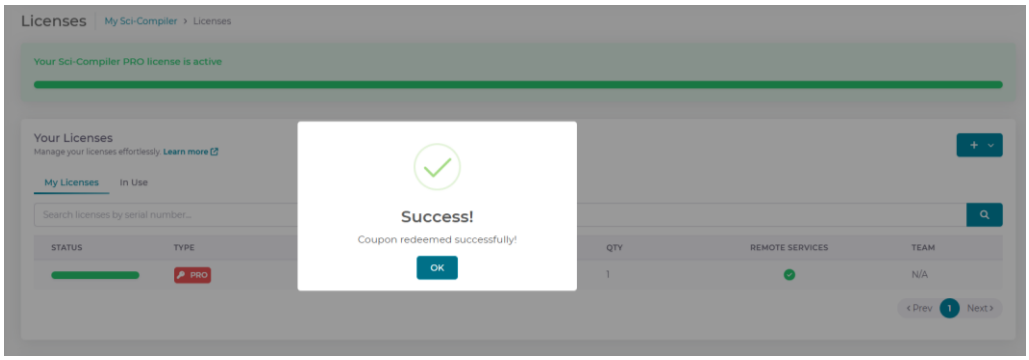


Figure 2.7: coupon redeemed successfully message.

Sci-Compiler Installation

Once a license has been activated (or added), the next step is to download the software. On the left side of the screen, navigate to the **"Downloads"** section. To download the latest version, click on **"Download"** under the **"Latest Release"** section. All available releases can be found in the **"All Sci-Compiler Releases"** section, where older versions can also be downloaded.

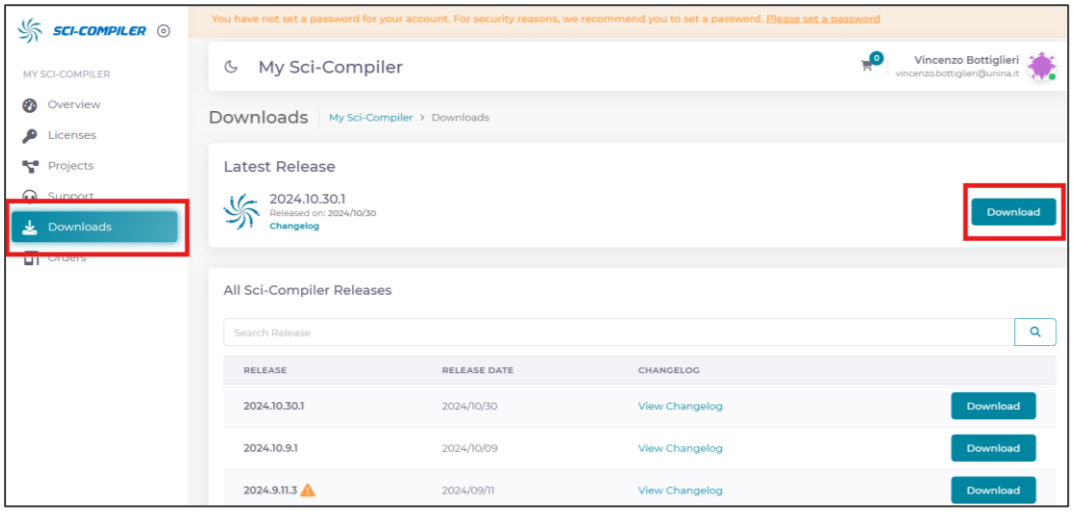


Figure 2.8: Downloads section in Sci-Compiler portal.

Note: *All Sci-compiler Releases* tab shows all the available software releases but gives access only to the ones compatible with your license expiration date.

Perform the following steps to install Sci-Compiler:

- Activate your license as explained in **Sci-Compiler License Activation** paragraph
- **Run the setup file**
- **Accept the agreement** and press “Next”
- A setup wizard will start. **Press “Next” to continue**
- Choose the installation folder and Start Menu Folder and **press “Next”**

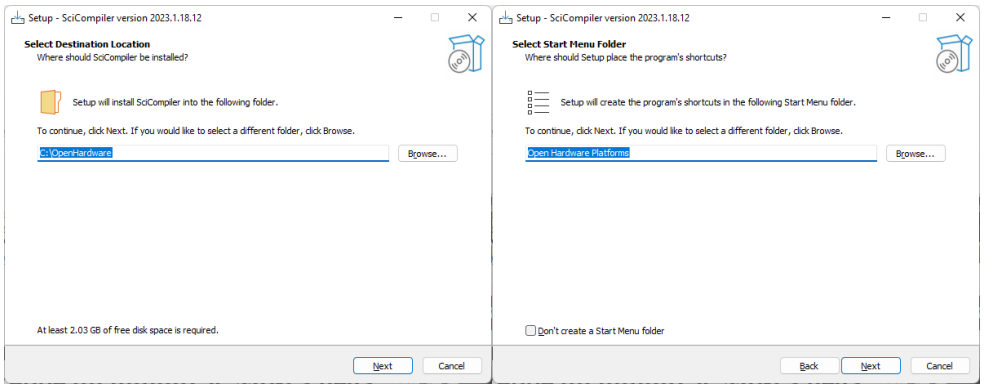


Figure 2.9: Sci-Compiler installation folder**.**

- **Click "Install"** to complete software installation

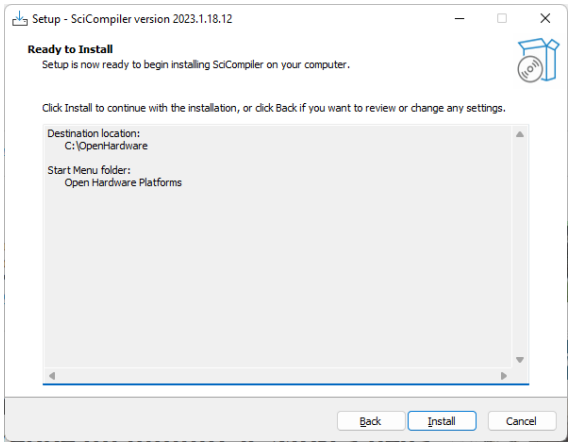


Figure 2.10: Sci-Compiler Installation tab.

- Wait until installation is completed

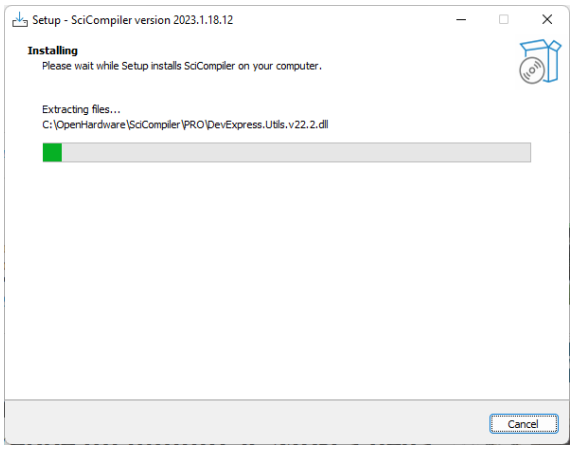


Figure 2.11: installation progressing bar**.**

- Congratulations! The installation is now complete, and you can finally launch **Sci-Compiler**

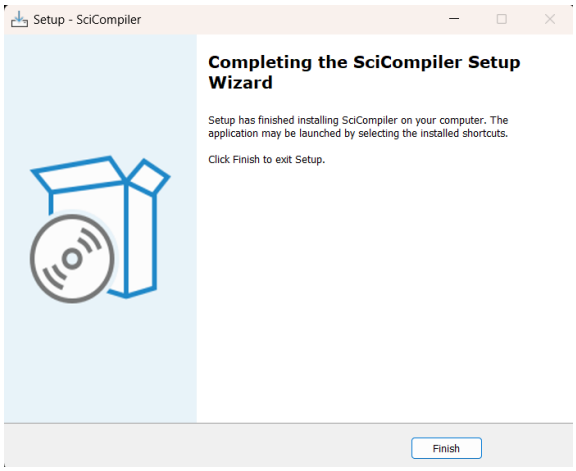


Figure 2.12: installation last tab.

- To launch **Sci-Compiler**, search for **SciCompiler** in the list of installed programs on your local disk and run the application

Note: in some cases, certain Sci-Compiler features may require administrative privileges to function properly. To ensure full functionality and avoid potential restrictions, it is recommended to always run Sci-Compiler as an Administrator. To do so, right-click on the SciCompiler icon and select "Run as administrator".

- Once **Sci-Compiler** is launched, a banner message may appear stating, **"Ops... we have a trouble with your license."** This is normal at this stage, as the active license has not yet been associated with the computer running **Sci-Compiler**. The next step is to link the license to the system to enable full software functionality.

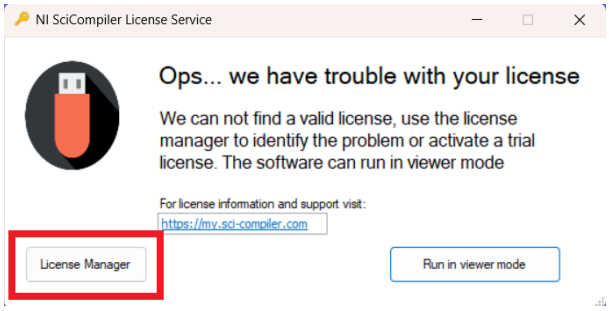


Figure 2.13: Sci-Compiler trouble license pop-up.

- The **Licenses** tab will open. Here, the user must log in using their **MyCAEN+** credentials. Once logged in, the previously redeemed license will appear under the **"Available Licenses for this Sci-Compiler version"** section. To complete the activation, the user must click on **"Activate License"** and confirm to use the selected licenseon this computer. It is recommended to check **"Remember license selection"** to ensure automatic license selection at startup, especially if only one active license is available.

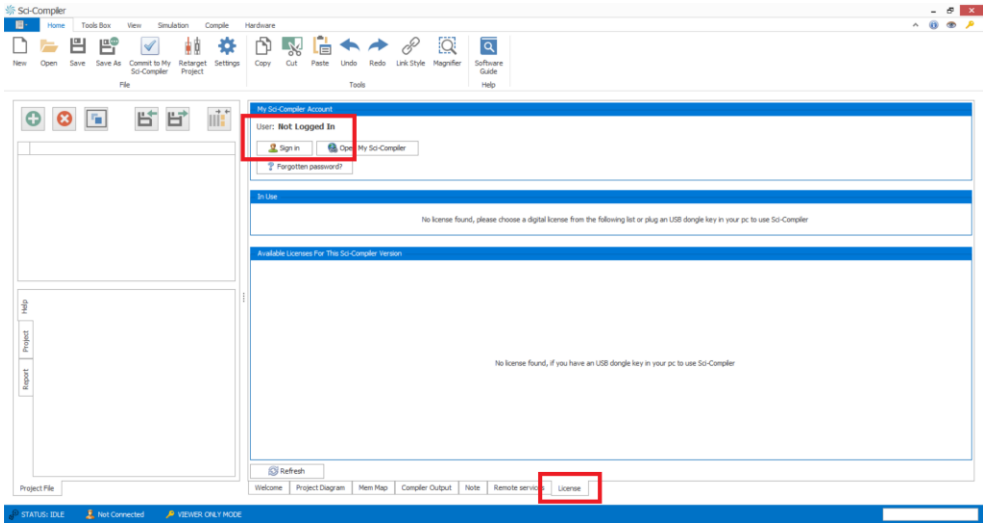


Figure 2.14: Sci-Compiler license tab.

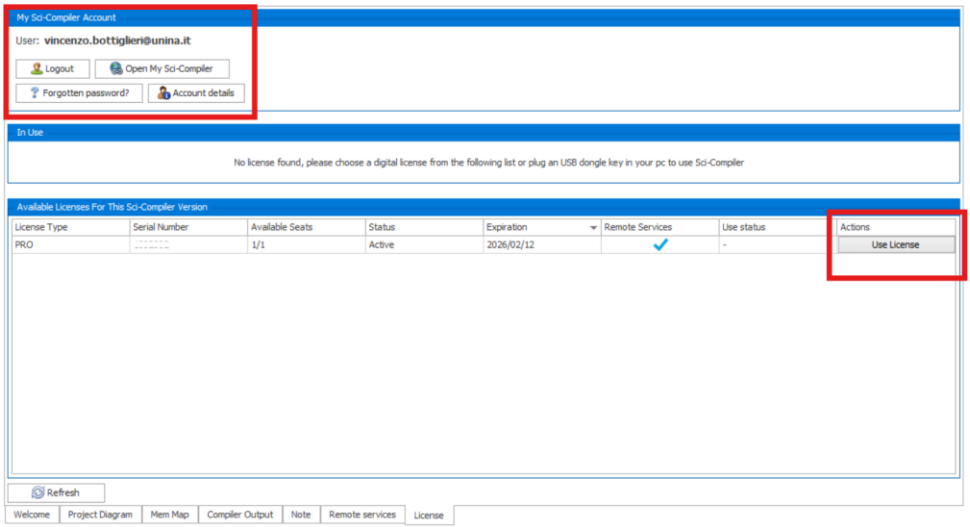


Figure 2.15: "Use License" location in Sci-Compiler tab.

- If a popup appears with the message **"Success! The license has been successfully bound to this machine!"** and your license is listed under the **"In Use"** section, congratulations! The license association is complete, and **Sci-Compiler** is now ready to use.

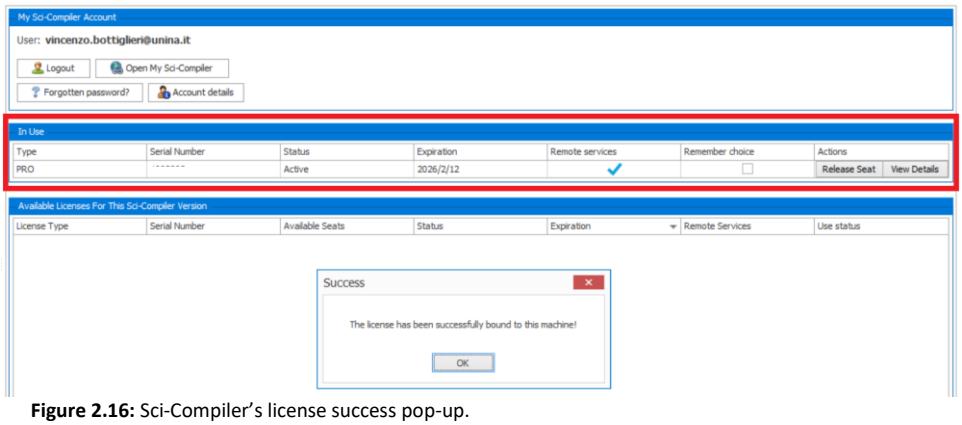


Figure 2.16: Sci-Compiler's license success pop-up.

Note: If the user has a physical ***USB DONGLE*** license, it can be converted into an equivalent digital license by clicking ***"Convert to a Digital License"*** in the license tab of Sci-Compiler.

Note: If you need to use Sci-Compiler on a computer without an Internet connection (which is required for proper license recognition), please contact CAEN support at:

https://www.caen.it/support-services/getting-started-with-mycaen-portal/

3 Welcome in Sci-Compiler

In this chapter we give a brief overview of the software GUI.

At start-up, the main window appears, with a “Welcome” tab with few information about the latest Sci-Compiler features, together with some online documentation.

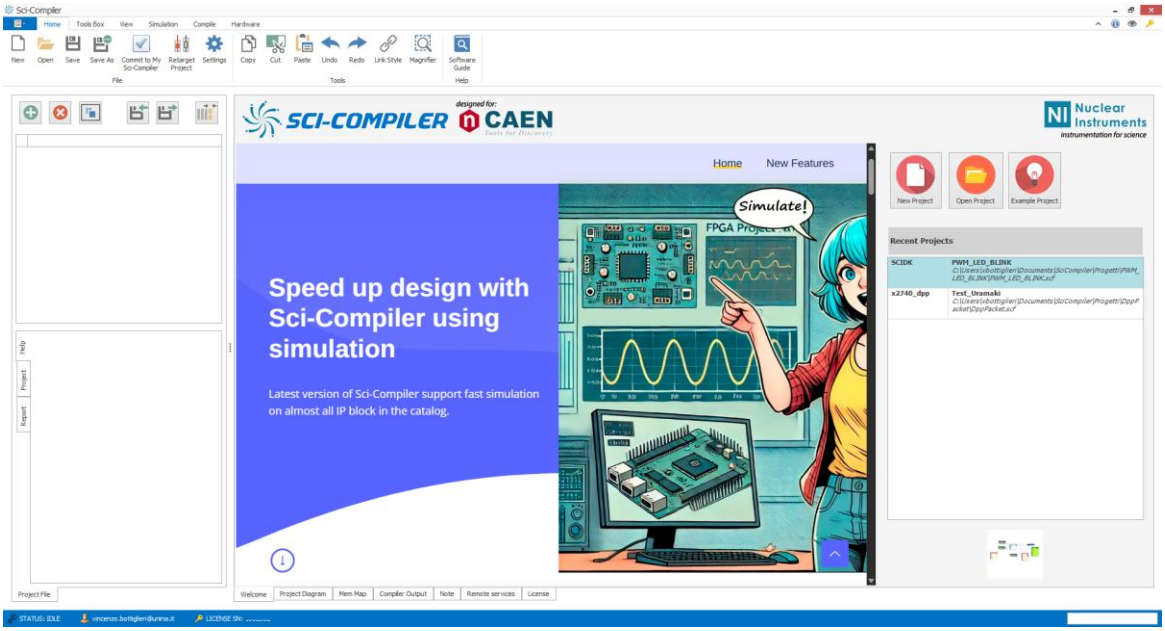


Figure 3.1: the *Welcome* tab.

The “Welcome” tab can be used to create a **new project**, to **open a project** or to explore the **example projects** that are provided with the software, click on the correspondent button on the right-side of the *Welcome* tab. The **Recent Projects**, if available, are listed in the right-side box.

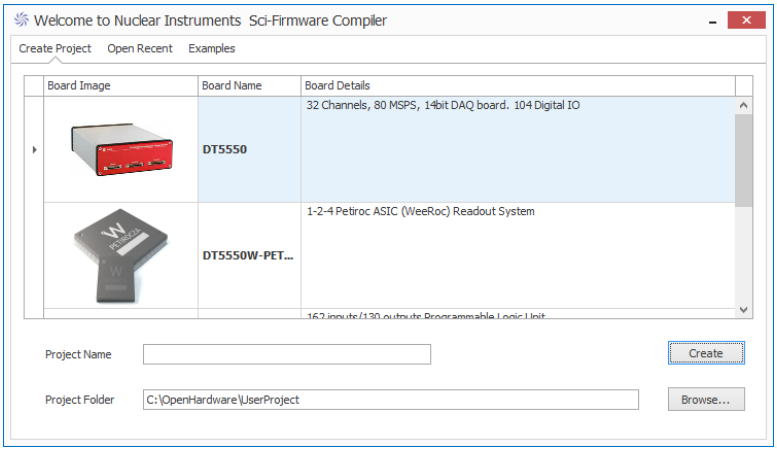


Figure 3.2: *New Project* window.

The "New Project" button allows the user to create a new project. The user has to click one of the line to choose the board model of interest. Then, the name of the project can be specified in the "Project Name" field, while the folder in which the project will be saved has to be written in the "Project Folder" field. The "Browse..." button allows to choose the project folder by looking at the system tree.

Note: a “Project Folder” path containing one or more spaces in the folder names is not recognized by Quartus or Vivado, causing compilation errors. Please use a path without spaces.

The "Create" button will then create a folder at the desired path with the same name of the project. Inside this folder the following files/folders will be created:

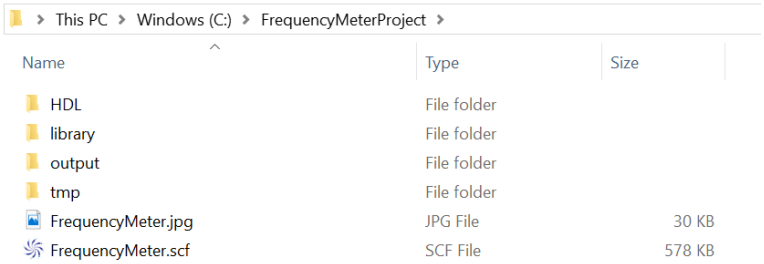


Figure 3.3: files and folders created in the Sci-Compiler project directory.

- a **.scf file**, the file format specific of the Sci-Compiler, containing all the information of the project
- a **.jpg image of the diagram** (at the start it is empty)
- four folders (*HDL*, *library*, *output* and *tmp*) that will be filled during the compilation process with generated VHDL code, FPGA programming files, libraries and example code.

The "Open Recent" tab allows to open a project that the user has previously created and saved. The box on the left shows in chronological order the recent project files list. By clicking on a file, the image of the diagram contained in the project will be displayed on the right and the field in the bottom will show the .scf file path. By pressing the "Browse..." button it will be possible to choose a project file (with extension .scf) by searching through the computer folders. The "Open" button allows to open the selected project file and load the correspondent diagram in the "Diagram" section of the GUI.

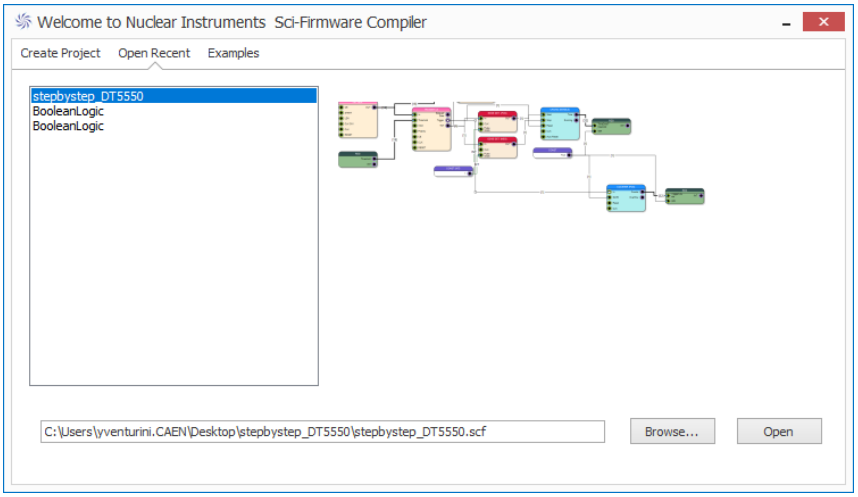


Figure 3.4: the “Open Project” tab in the “Starting” window of Sci-Compiler.

The "Examples" tab allows the user to create a project by starting from an available example provided with the program. The examples are pre-compiled projects files (.scf) that the user can open, explore and compile to obtain the compilation output file that can be uploaded on the board FPGA. The box on the left shows all the example folders organized by board type and category. By clicking on an item, the name of the project file will appear in the "Project Name" field, the image of the project diagram will be displayed on the right and the description of the example functionalities is reported below the image. Then, the user can set the folder in which the project will be created by writing the path in the "Create Project in folder" field or by using the "Browse" button. The user can also decide to change the project name by using the correspondent field. The "Create" button will create a project containing the blocks and the connections of the selected example with the name and the destination folder specified by the user.

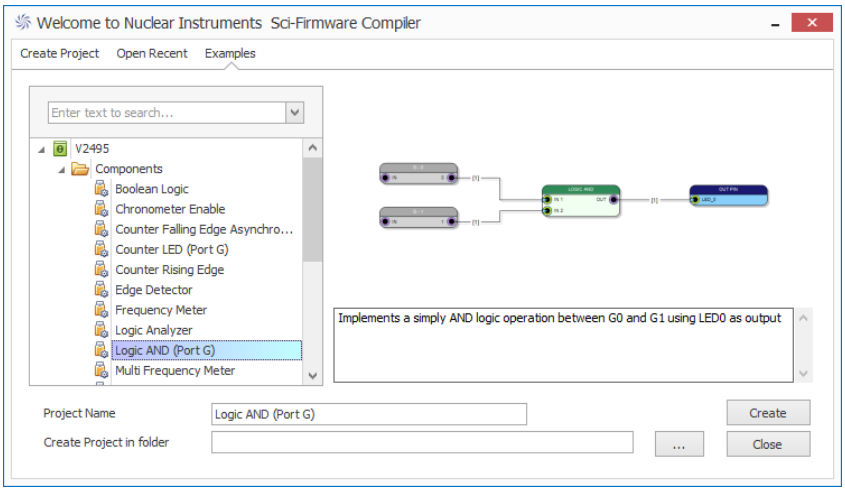


Figure 3.5: the *Examples* tab.

Sci-Compiler main working area

The main Sci-Compiler working area is the Diagram sheet, where the user can place the blocks available in the *Tools Box* bar menus.

Note: the specific documentation for each block is available in the Online Help by clicking on the block of interest after placing it in the diagram.

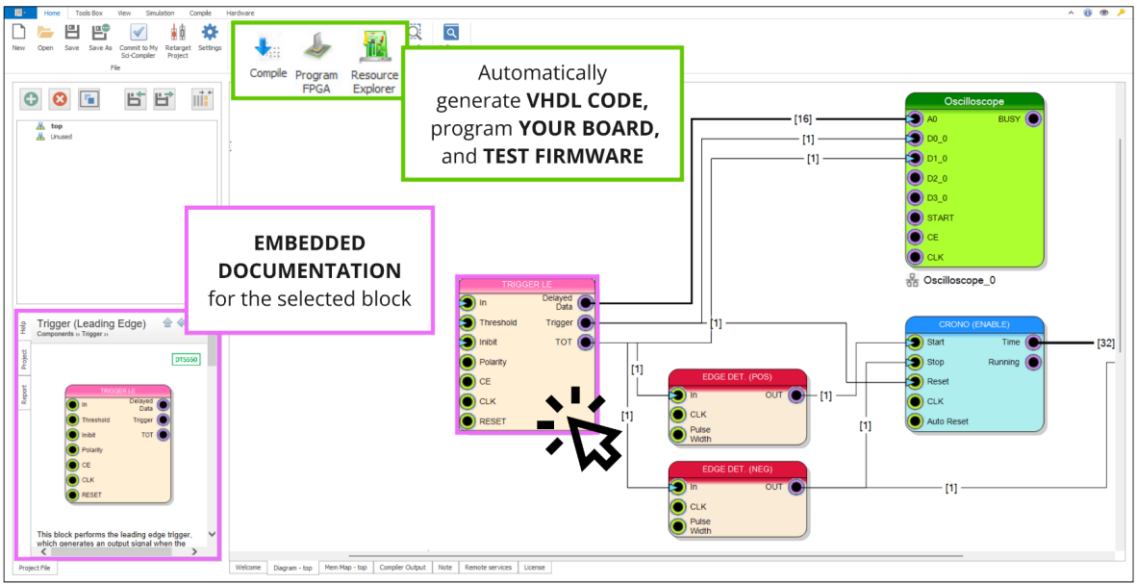


Figure 3.6: Sci-Compiler main tools are highlighted.

The main interface is organized as follows:

- Toolbars: the toolbars contain the library of blocks available to draw the diagram and the commands to start compilation, simulation and FPGA programming.
- Resources and Settings: this section has two tabs; the *Project File* tab allows to manage the files of the project and to see a block information through the online "Help", while the *Hardware Settings* tab can be used to select the board connectors and registers settings to be configured.
- Development GUI: this section allows to graphically design the FPGA firmware exploiting and interconnecting the available blocks (*Diagram* tab), to configure the board connectors and registers settings (*Editor* tab) and to read the output messages of the compilation process (*Compiler output* tab).
- Status: this bar shows, on the left, the license information and the program status and, on the right, a bar to indicate the progression of the current process.

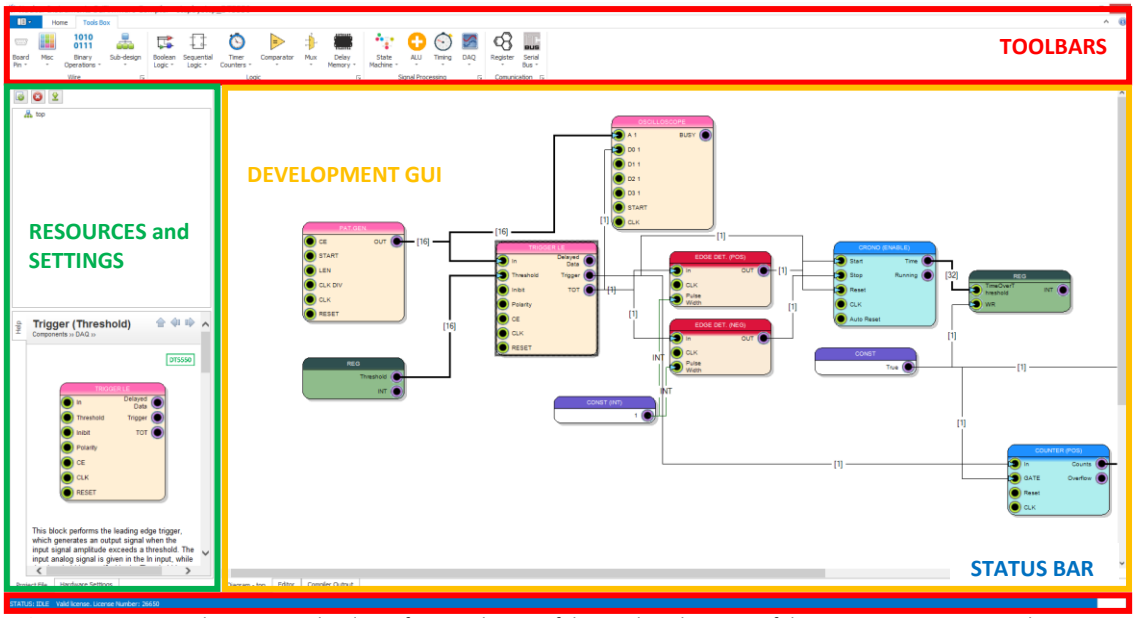
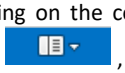


Figure 3.7: Sci-Compiler main graphical interface. At the top of the window the name of the current open project is shown.

Toolbars

Six toolbars are available in Sci-Compiler – *Home, Tools Box, View, Simulation, Compile, Hardware* toolbars. The user can switch between the toolbars clicking on the correspondent labels. Moreover, a *File Menu* is available and can be visualized by clicking on the button  , in the top left corner of the software window. This menu contains the following items:

- "New": opens, when clicked, the *Starting Window* shown in **Figure 3.2** and allows to create a new project.
- "Open": enables the user to browse the computer files and to open an already created project file with the *.scf* extension.
- "Save": saves the current project and updates the .jpg image of the diagram and the *.scf* file.
- "Save As": gives the possibility to change the name of the current project and to save it in a different location.
- "Exit": closes the software.

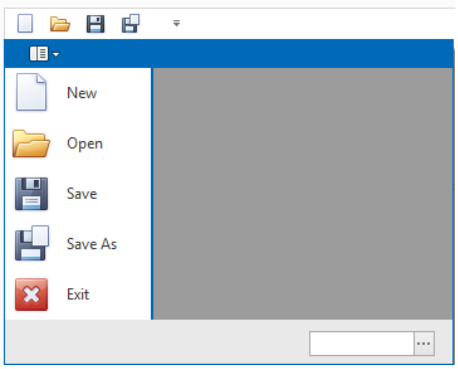


Figure 3.8: the "File Menu" of Sci-Compiler

Home Toolbar

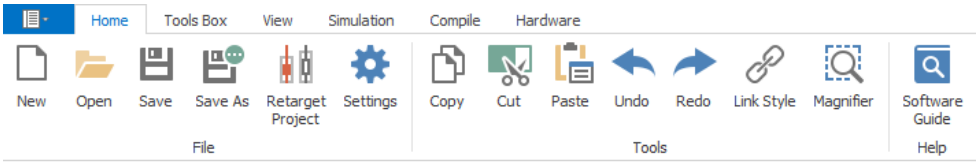


Figure 3.9: the Home Toolbar of Sci-Compiler

The "Home" toolbar contains some controls that are organized in sub-groups dedicated to different types of actions.

The "File" sub-group contains the same buttons of the "File Menu", with the same functionalities described above. Moreover, the Settings button allows to set the Vivado and Quartus compilers configuration. The *Retarget Project* allows to transform a previously created design for a certain board into a similar project for another target board.

The "Tools" sub-group has self-explicative controls that can be used when drawing the diagrams.

The "Help" allows to access a complete software guide with descriptions of all functional blocks available.

Tools Box Toolbar

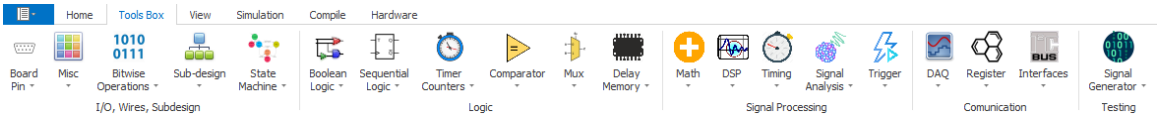


Figure 3.10: the Tools Box Toolbar of Sci-Compiler

The "Tools Box" toolbar contains all the **pre-compiled implemented blocks** for the target board, divided in groups depending on their category. The available groups and the content of the sub-menus of each group, i.e. the available blocks, depend on the board selected for the project.

Note: all the blocks are described in detail in the **online *Help***, available in the *Resources and Settings* area of the software. Clicking on a block of the diagram, its description will be displayed, together with labels to indicate the boards for which that block is specifically designed.

The "Wire" group contains all the single wire blocks, organized in sub-menus:

- the "Board Pin" allows to use the I/O board pins and implement the readout modes for ASICs (for DT5550W);
- the "Misc" contains constant values, Boolean variables, clocks, and conversion between binary and decimal values; it also contains the "User HDL" function which allows the user to create a graphical block based on his custom HDL code.
- the "Bitwise Operations" allows to implement the main bitwise binary operations.
- the "Sub-design" enables to select a sub-design file and synchronize it in order to insert it in the main diagram file.

The "Logic" group contains the following sub-menus:

- the "Boolean Logic" allows to implement all the Boolean logic basic operations.
- the "Sequential Logic" has edge detectors, latches and flip-flops.
- the "Timer Counters" contains scalers, chronometers, timers, counters and frequency meters.
- the "Comparator" has blocks that can be used to compare signals;
- the "Mux" allows to implement multiplexer and demultiplexer.
- the "Delay Memory" contains blocks to implement delays, serialization and deserialization, RAM, ROM, various type of buffers and a pattern generator tool.

The "Signal Processing" group contains:

- the "State Machine" menu to implement a finite state machine.
- the "ALU" menu containing the basic Arithmetic Logic Units operations.

- the "Timing" menu with blocks implementing the Time to Digital Converter, the Single Channel Analyzer and the Time over Threshold.
- the "DAQ" menu which allows to implement the processing of the signal using blocks like the oscilloscope, the baseline restorer, the charge integration, the trapezoidal filtering, …

The "Communication" group is divided in the following sub-menus:

- the "Register" allows to read and write directly on the board register.
- the "Interfaces" offers the possibility to choose between the I2C, the SPI or the UART communication protocols.

The "Testing" group contains the “Signal Generator” menu, to add testing input signals to the diagram.

View Toolbar

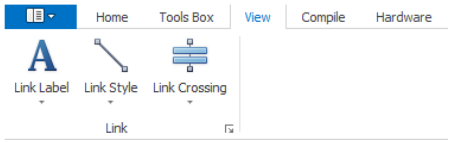


Figure 3.11: the View Toolbar of Sci-Compiler

The *View* Toolbar contains general options for the block diagram visualization, like enabling/disabling the label on the link and choosing the link style.

Simulation Toolbar

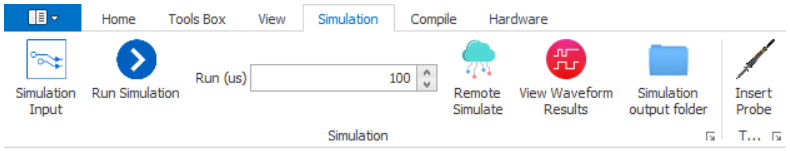


Figure 3.12: the Simulation Toolbar of Sci-Compiler

The *Simulation* Toolbar contains the command and options to start firmware simulation. *Run Simulation* allows to start simulation through local compilers, while *Remote Simulate* uses the Remote Customization Service. It is possible to set the simulation time, choose some *Simulation Input* (like external signals) and *Insert Probe* in the block diagram in order to monitor the electrical signal in that point. Sci-Compiler gives access to a tool to *View Waveform Results* in order to monitor the logic design operation. The output files of a simulation are saved in the *Simulation output folder*. For more details, please see the **Firmware Simulation** paragraph.

Compile Toolbar

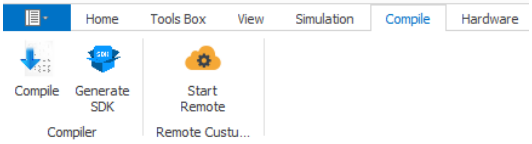


Figure 3.13: the Compile Toolbar of Sci-Compiler

The *Compiler* toolbar contains the controls related to the compilation procedure.

The “Compiler” sub-group contains the commands for local compilation options (active when available for the specific target board). The "Compile" button starts the compilation on the user computer: a .vhdl file is generated starting from the user graphically designed firmware. A .tcl file is created and the Vivado or Quartus program, depending on the target board chosen for the project, will be executed from the shell. The output of the program execution is redirected to the "Compilation Output". For more details, please see the **Firmware Compilation** paragraph. The "Program FPGA" button allows to program the board FPGA by using the bitstream file generated during the firmware compilation process, connecting the board to the computer with a specific programmer cable (Vivado or Quartus are called automatically from the shell). Also, in this case the output of the process is redirected on the "Compiler Output". For further details, please see the **FPGA Programming** paragraph.

The “Remote Customization” sub-group allows to access the Remote Customization Service and monitor the status of compilations through the Remote Center (active when available for the specific target board)

Hardware Toolbar

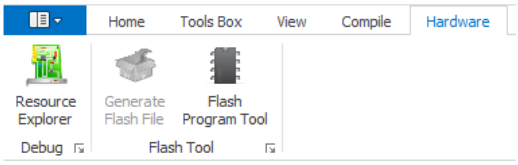


Figure 3.14: the Hardware Toolbar of Sci-Compiler

The *Hardware* toolbar gives access to Debug and FPGA flashing tools.

The "Resource Explorer" button opens a window that allows to connect with the hardware and test the FPGA firmware. The functionalities of the "Resource Explorer" are briefly explained in Par. **How to test your firmware: the Resource Explorer**, while a complete description can be found in the online *Help* of the software.

The ”Flash Tool” sub-group contains the commands needed to generate the firmware flash file (active when firmware is compiled locally) and launch the firmware flashing tool for DT5550x series and x5560 boards.

Resources and Settings

The “Resources and Settings” section (on the left-hand side of the main GUI) allows to manage the files of the project, view the project properties and read a block explanation through the *Help*.

The *Project File* tab is organized in two sections.
The section on top shows the files contained in the project organized in a tree view:

-  represents the main file project.
-  denotes the sub-design file project.

By clicking on a file, the correspondent content will be shown in the "Diagram" and the "Tools Box" toolbar will be displayed.

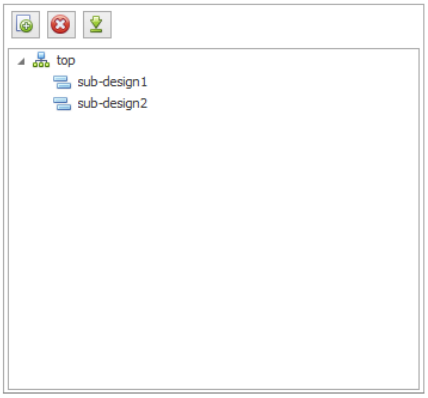


Figure 3.15: the top section of the “Project File” tab.

 This button allows to create a new sub-design file.

 This button deletes the selected sub-design file.

 This buttons allow to synchronize a selected sub-design file in order to insert it as a block in the diagram of the main project.

The section on the bottom part of the *Project File* tab is organized in two sub-tabs:

- the *Help* sub-tab reports the name of the selected block and its category, with arrows that allows to navigate the various sections. The *Help* shows a label indicating if the block is available for a specific board (V2495 and/or DT5550), the image of the block and its description. This explanation is followed by a summary of all the input and output signals of the block, each one with the signal name, the label specifying the signal type (integer INT or logic vector VECT), the label indicating the signal direction (input or output), the signal size and the signal brief description.
- the *Project* sub-tab reports the properties of the open project and allows to enable/disable precompiled IP cores. If this latter option is enabled, the compiling time can be significantly reduced.

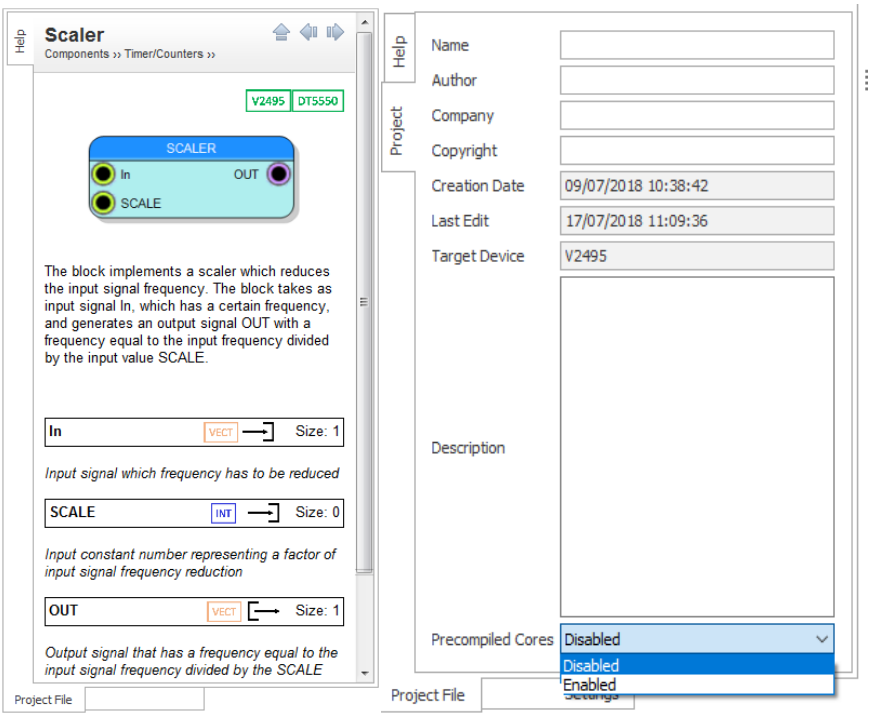


Figure 3.16: the bottom section of the *Project File* tab, with the *Help* and the *Project* sub-tabs.

Status Bar

Figure 3.17: The "Status" bar of Sci-Compiler

The "Status" bar on the bottom part of the GUI is composed by three elements:

- the status label, on the left corner of the status bar, indicates the process that is executed by the software: it is "COMPILING" if the compilation process is in execution, "PROGRAMMING" if the FPGA programming process is running and it is "IDLE" if there is no Vivado or Quartus process in execution.
- the e-mail used to login to the MyCAEN+ account, if connected online
- the progress bar on the right corner of the status bar shows the progression of the current process (the firmware compilation or the FPGA programming).
- the license label shows the information related to the Sci-Compiler license, which is provided through the USB dongle or free trial.

4 General project structure

How does a typical project look like?

Sci-Compiler uses block diagrams to convert the concept of a Digital Pulse Processing algorithms into a full working firmware. Typically, a project is made of three main parts:

- **Inputs**, that are blocks mapping the actual hardware input of the hardware.
- **Processing,** i.e., blocks interconnecting each other and making the Realtime Signal Processing. For instance, in **Figure 4.1,** the signal from the input feeds a *Derivative Trigger*, which in turn activates a *Trapezoidal Filter*. The flat top of the trapezoid is sampled by the *Energy Sampler* block and energy of the incoming signal is then measured.

Note: refer to the software *online Help* for the accurate description of the Inputs and Processing blocks available for each compatible board.

- **Readout**, i.e., blocks to read the results of the Processing and transfer them to the PC as data. For instance, in **Figure 4.1,** a *Spectrum* block is used to read data coming from the *Energy Sampler.* Data are read by the *Spectrum* block when the *Energy Sampler* and an external digital signal on pin IO_A_0_A of the board are in coincidence (*Logic AND*).

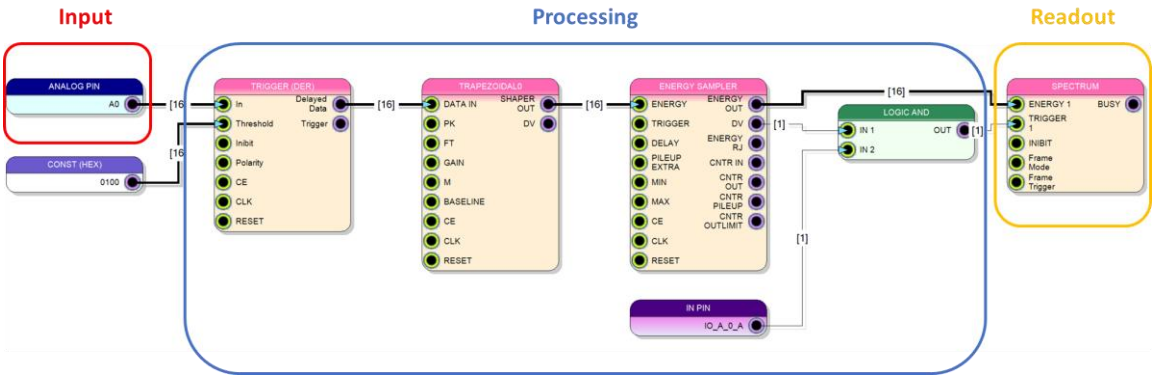


Figure 4.1: the typical structure of a Sci-Compiler project for Digital Pulse Processing.

Readout structure and Resource Explorer

Sci-Compiler uses different readout structures in order to adapt the acquisition to the different compatible hardware.

There are two main readout structures (see **Figure** 4.3):

- **Local Bus** readout (available for all compatible boards): data from the processing blocks and configuration registers are sent through a AXI Local Bus Bridge (refer to **[RD15]**) and can be directly read by the **Resource Explorer**, that is a built-in tool available in Sci-Compiler GUI (see **Figure** 4.2). The Resource Explorer gives the possibility to read and write the configuration registers (i.e. Signal Processing parameters) and shows the readout instruments like *Spectrum*, *Oscilloscope*, etc. in a GUI, so that it is possible to change the signal processing parameters and test the results of acquisition live, with no need to write any software code. In practice, the Resource Explorer is a first-use debug interface to test the firmware and check the behaviour of the Processing algorithm specified through the block diagram.

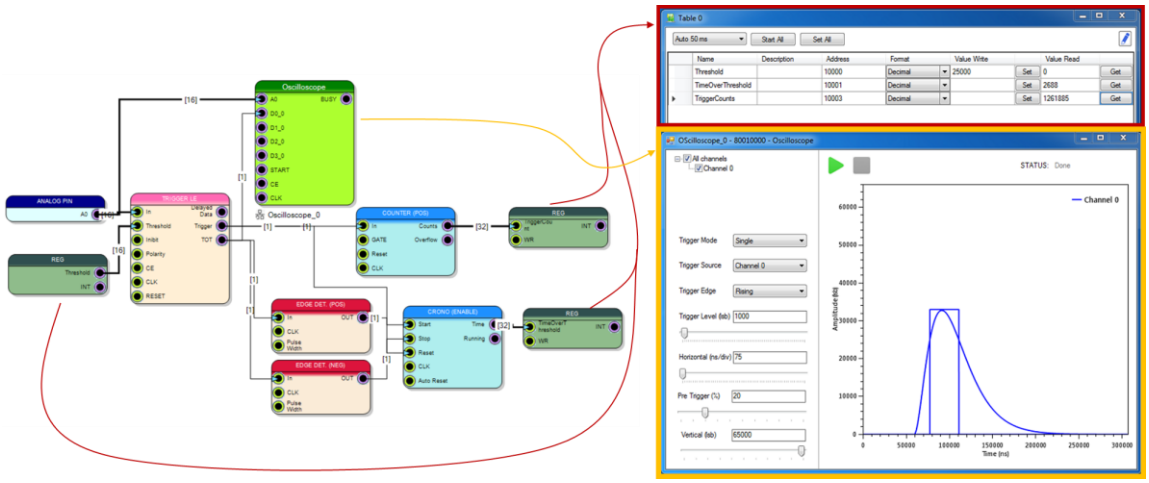


Figure 4.2: the Sci-Compiler Resource Explorer (right), showing an oscilloscope acquisition and Digital Pulse Processing parameters for a given firmware block diagram. The Read/Write registers are reported in the table, the signal acquired by the *Oscilloscope* block are shown in a user-friendly GUI.

- **DMA/DDR** readout (available for 2730 - 2740/45 Family only): special readout blocks are available for boards supporting DMA/DDR architecture. In this case, the acquired events flow into a large DDR4 memory (5 GB) for buffering, waiting for a DMA that transfers the data into the DDR4 memory of the ARM processor. Finally, upon request of the DAQ software running in the host PC, the data are transferred to the computer and can be readout through the CAEN software WaveDump2 **[RD16]** and CoMPASS **[RD17]**, or alternatively using the Sci-Compiler Software Development Kit (SciSDK).

  Special Readout blocks are available for **Waveform** and **DPP** modes of the 2740/45 Digitizer family, called respectively *x2740/45 Digitizer* and *CAEN List*

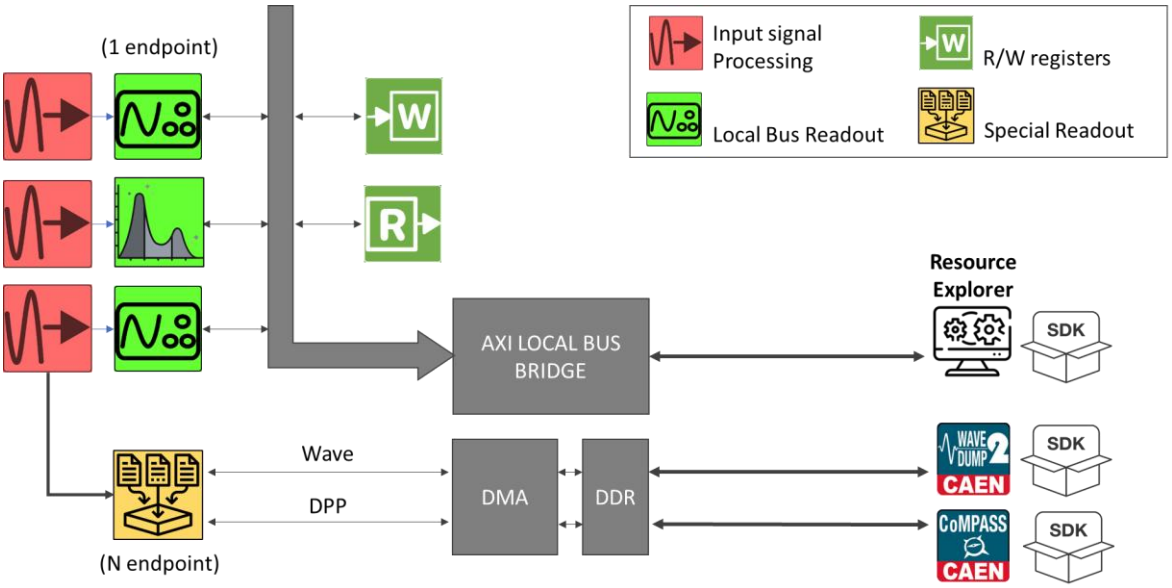


Figure 4.3: sketch of the Sci-Compiler readout modes.

Local Bus and DMA/DDR Readout blocks can be used together in the same project. For instance, it is possible to readout data from the same *Charge Integration* block using both the *Custom Packet* (feeding the Local Bus) and the *CAEN LIST* (feeding the DMA/DDR path), as shown in **Figure 4.4**. In particular, data coming from the *Custom Packet* or from the *CAEN LIST* would be the same but in different formats. With respect to the Local Bus blocks, the Special Readout ones are faster and more performing because they use the DDR4 memory instead of the FPGA RAM.

Note: *x2740/45 Digitizer* block and *CAEN List* block cannot be instantiated in the same project. In fact, the user has to decide at the creation of the project if using the Wave or DPP mode. Wave mode should be used for common trigger acquisitions and DDR4 is mainly used for waveform samples storage. DPP mode should be used for independent channel trigger acquisitions, where a complex Signal Processing is needed to extract a list file. In DPP mode, the DDR4 is feed with the list coming from the *CAEN List* block.

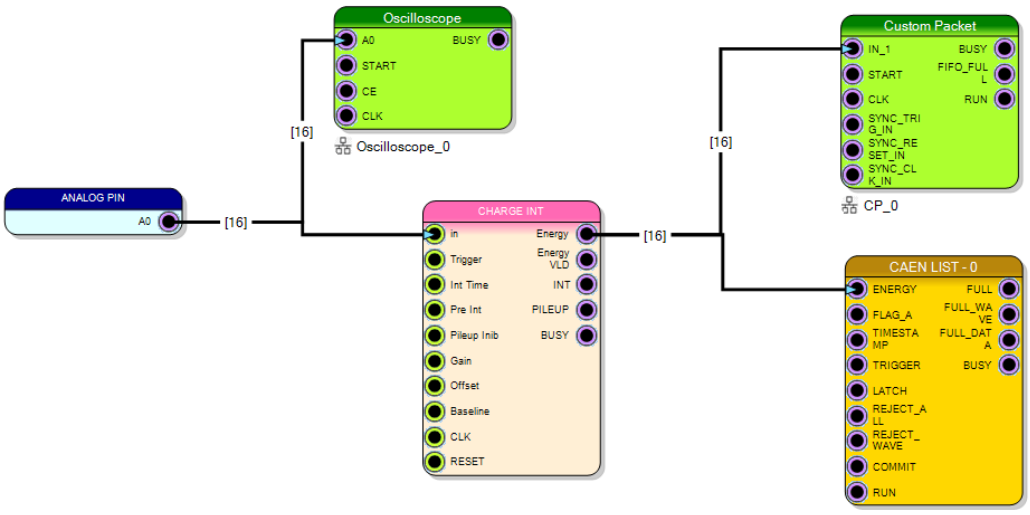


Figure 4.4: example of use of Local Bus blocks (Oscilloscope and Custom Packet) together with Special Readout blocks (CAEN List) in the same project.

5 Firmware Simulation

Firmware code simulation is a well-known best practice during FPGA programming flow. Since compiling a firmware may require a lot of time, it is possible to simulate it in order to reproduce its behaviour under some external stimulus.

Sci-Compiler allows to simulate the firmware code either using local Vivado and/or Quartus installation, either through the Remote Customization Service.

The simulation process is much faster than compilation and it allows to have a quick response on how the Signal Processing Algorithm implemented in the firmware is working. In fact, it is possible to add some input to the simulation (like an external signal) and set probing points in the block diagram. In this way it is possible to monitor the electrical signal in that point. Moreover, Sci-Compiler gives access to an external tool (GTKWave) to monitor the overall Waveform Results.

A simulation example

Two main tools can be used to add simulation stimulus and probes to a given block diagram: the *Simulation Input* and *Insert Probe* available under the Simulation toolbar.

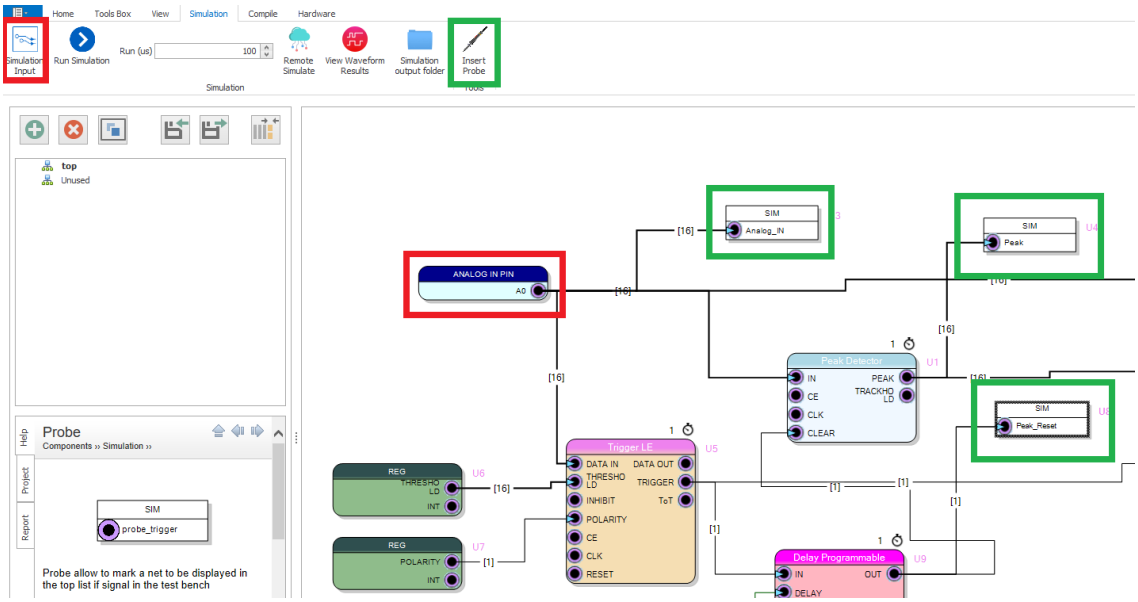


Figure 5.1: management of simulation stimulus and probes in Sci-Compiler. The "Simulation Input" sections are highlighted in red, while the "Insert Probe" sections are highlighted in green.

The *Simulation Input* allows to generate a signal of interest at each *Board Pin* available into the block diagram. For example, in **Figure 5.1**, it is possible to emulate a specific signal at the Analog Input A0. After placing the input in the diagram, right click on it and select Add/Edit to simulation Stimulus. The given input will be added to the *Simulation Input* table and, from there, it is possible to choose the kind of signal to be generated as a stimulus of the simulation. As shown in **Figure 5.2**, the Simulation Input menu allows to select the characteristics of the stimulus to be generated at a certain input (in the example A0 of a V2740 board). As a more advanced use mode, it is possible to go into the *Registers* tab and write a script to generate values of interest for the Registers available in the block diagram.

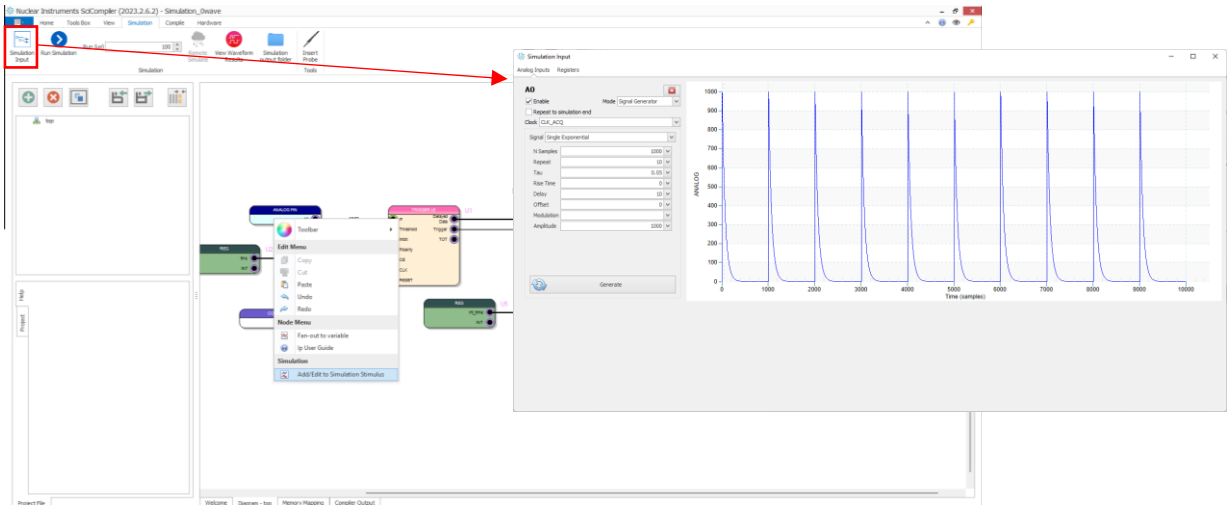


Figure 5.2: generating an exponential stimulus at a board analog input in Sci-Compiler.

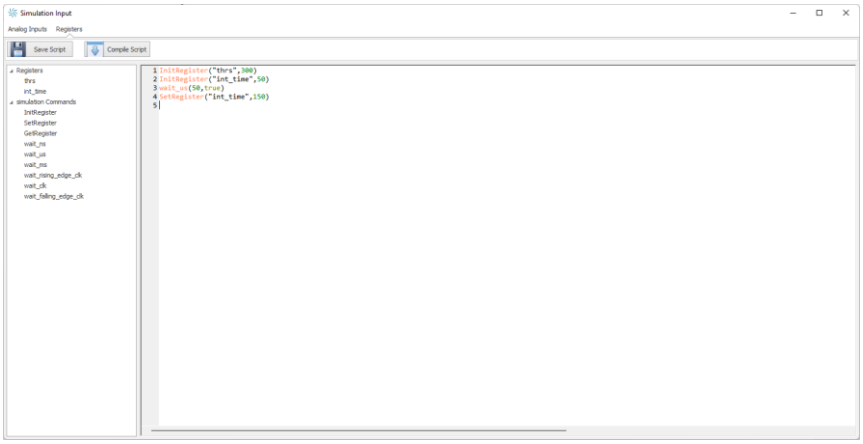


Figure 5.3: using a script to generate stimulus for the registers available in the block diagram.

The other tool is the *Insert Probe*, that allows to add a SIM block to the diagram and connect it to whatever block pin in order to monitor some signals of interest. Those signals will be available, together with the input stimulus, through the *View Waveform Results* after simulation is completed.

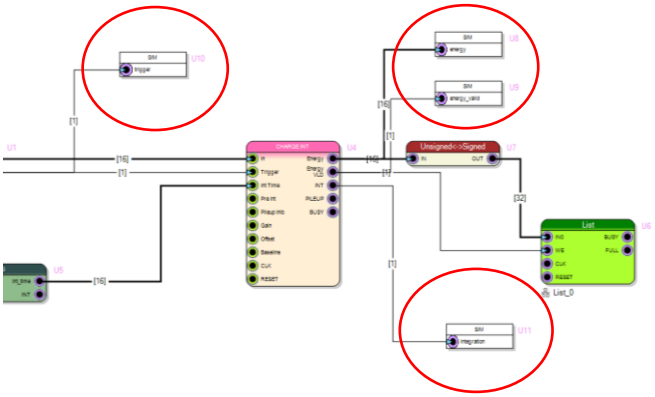


Figure 5.4: the SIM probes placed in a block diagram to monitor output signals coming from other blocks of the diagram.

Note: an extensive tutorial for simulation is given in the Sci-Compiler **online help**.

6 Firmware Compilation

When the block diagram design is finished, it is possible to compile the firmware with local Vivado and/or Quartus installations or through the Remote Customization Service.

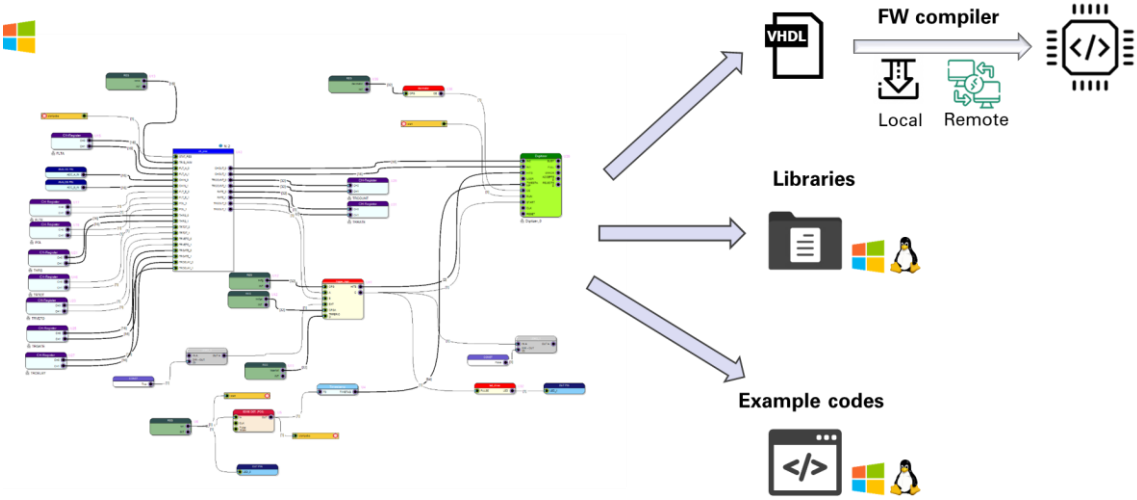


Figure 6.1: sketch of the Sci-Compiler generated files.

Local compilation

The firmware compilation on the user's computer starts by pressing the "*Compile*" button in the "Home" toolbar. The button is immediately transformed in the "*Stop Compilation*" button in order to stop the compilation process if needed. Automatically, the "Compiler Output" section is displayed in order to show the messages describing the processes that are occurring. At the same time, the status label on the left corner of the status bar changes from "IDLE" to "COMPILING" and the progress bar on the right corner of the status bar starts to be updated to show the progression of the compilation process.

Figure 6.2: the status bar during firmware compiling.

The firmware compilation is composed by various phases: the analysis and synthesis of all the components, their mapping, their placing and routing, where the optimization of the space and time constraints is considered.

The first compilation step is executed by the Sci-Compiler and consists in the generation of the VHDL code from the user-designed diagram. The various operations are described and reported in the "Compiler Output" in blue, while eventual errors will be displayed in red. In particular, the registers and the signals map are generated and mapped on the correct addresses. Then, for each block used in the diagram is generate an HDL code in the *"HDL/pcores"* folder of the current project. At this point, the HDL code can be assembled for all the blocks used in the main diagram and all the sub-designs, and the correspondent **.vhd files** are generated in the *"HDL"* folder of the current project.

The second compilation step involves the creation by the Sci-Compiler of **C++/Phyton libraries** (C libraries can be compiled in VC++, gcc, XCODE) and their relative **example code** in the "*library*" folder of the current project. This code can be exploited by the user to test the board communication. In addition, a **.json file** is created in the same folder: it contains all the defined registers, oscilloscopes, list modules and logic analyzers blocks in order to transmit all the necessary information to the "Resource Explorer" tool, which allows to test the functionalities of the designed firmware (see later for more details).

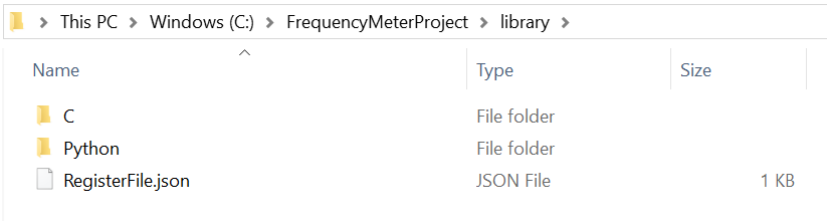


Figure 6.3: overview of the files created in the “library” folder during firmware compiling.

In the third compilation step the Sci-Compiler generates a .tcl file in the "*HDL*" folder that is used to execute from the shell the Vivado or Quartus program, depending on the target board chosen for the project. In fact, Vivado and Quartus are, respectively, the Xilinx and the Intel Altera software allowing the FPGA firmware compilation and bitstream generation.

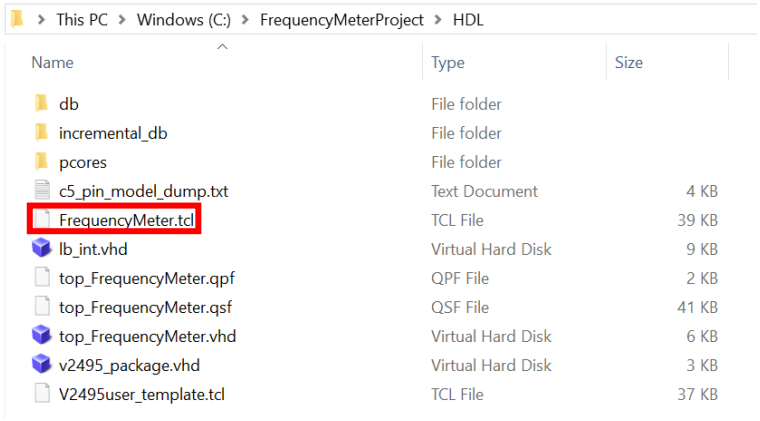


Figure 6.4: overview of the files created in the “HDL” folder during firmware compiling.

The output of the compiler software execution is redirected to the "Compiler Output" tab and visualized in black, with the warning messages displayed in orange and the error messages reported in red. The messages describing the operations executed by the Sci-Compiler itself, as the creation of a new project folder, the opening of a project, the creation of all the files and folders necessary for the firmware compilation, are displayed in blue. The output of the Vivado or Quartus process that are redirected in this console are visualized in black. In addition, all the warning massages are displayed in orange, while all the error messages are reported in red.

Figure 6.5: the “Compiler Output” during firmware compilation.

The final compilation step is represented by the generation of the bitstream file that is required to program the FPGA, i.e. a configuration file compatible with the supported platform: in the case of the DT5550x/x5560 it is a .bit file, while

in the case of the V2495/DT5495 it is a .rpd file. All the output files are generated in the "*output*" folder of the current project.

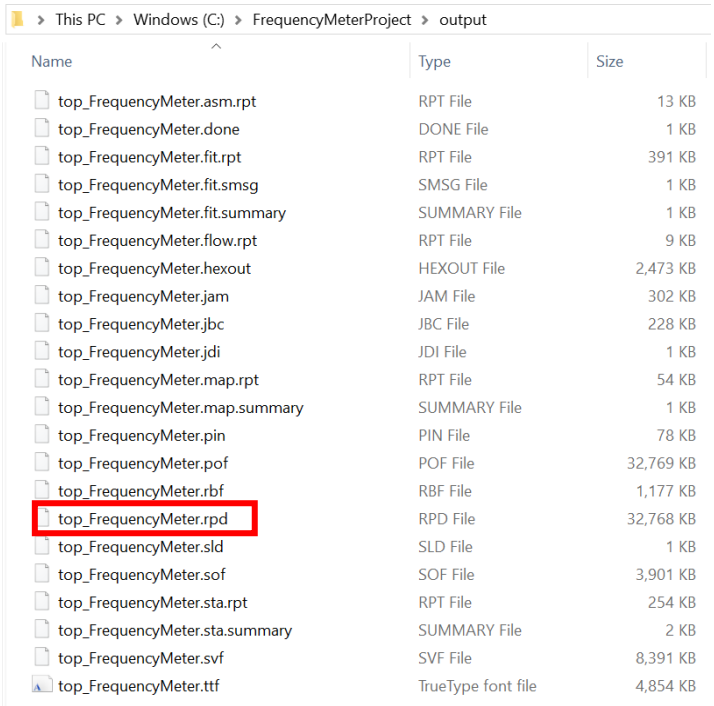


Figure 6.6: overview of the files created in the “output” folder during firmware compiling. The .rpd flash firmware file created for a V2495 is highlighted.

In case of errors during the firmware compilation, the process is stopped, the "Stop Compilation" button is transformed again in the "Compilation" button to start a new compilation, the progress bar is reset, and the status bar returns in the "IDLE" state. The user can then use the error messages to solve the issues and start a new compilation. If no errors occur, the compilation process continues till the end and the Sci-Compiler sends the message "Successful compilation!". Also in this case, the progress bar is brought back to the start, the status bar returns in the "IDLE" state and the "Stop Compilation" button changes to "Compile", in order to allow a new firmware compilation process.

Remote Customization Service

The firmware compilation through the Remote Customization Service starts by pressing the "*Start Remote*" button in the "Home" toolbar. The button is immediately transformed in the "*Stop Compilation*" button in order to stop the compilation process if needed. Automatically, the "Compiler Output" section is displayed in order to show the messages describing the processes that are occurring. At the same time, it is possible to check the status of the compilation in the *Remote Center*. At the end of compilation, it is possible to download the final build of the firmware from the *Remote Center* and the file can be loaded directly on the compatible board.

7 FPGA Programming

Once the firmware compilation process is terminated successfully and the bitstream file has been generated, the user can program the FPGA. The firmware file can be generated by pressing *Generate Flash File* or it is available through the Remote Center, depending on the compilation being local or through the Remote Customization Service.

Note: refer to the board specific User Manual for the exact upgrade procedure of each board compatible with Sci-Compiler.

Note: procedure for the FPGA permanent upgrade of V2495/DT5495 and 2740/45 is given in **[RD3]** and **[RD13]** respectively.

Moreover, Sci-Compiler offers an embedded tool to upgrade the firmware flashing permanently the FPGA of a DT5550/DT5550W/x5560/DT1260 board. The "Generate Flash File" button generates a .bin firmware file, compatible with the Xilinx FPGA. The "Flash Program Tool" button launches the Firmware Upgrader.

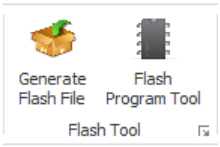
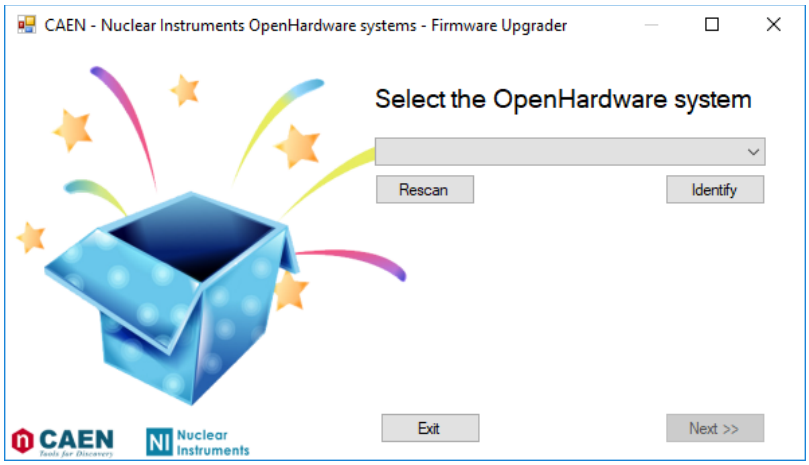


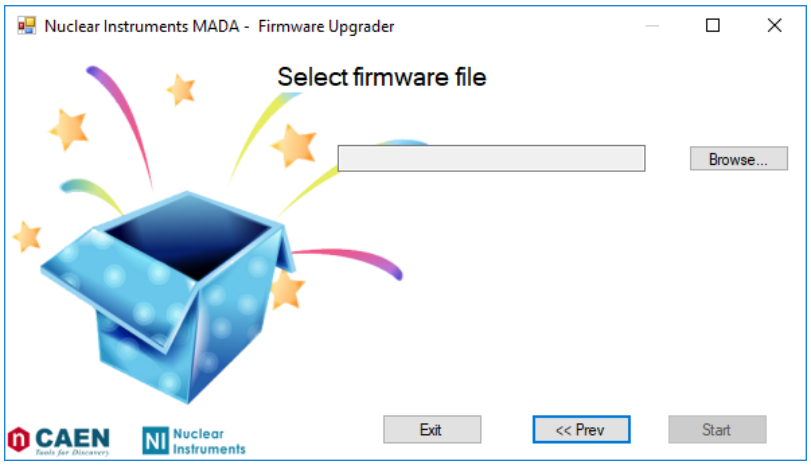
Figure 7.1: the "Flash Tool" sub-group of the *Hardware* Toolbar

The procedure to perform a permanent firmware upgrade is described below:

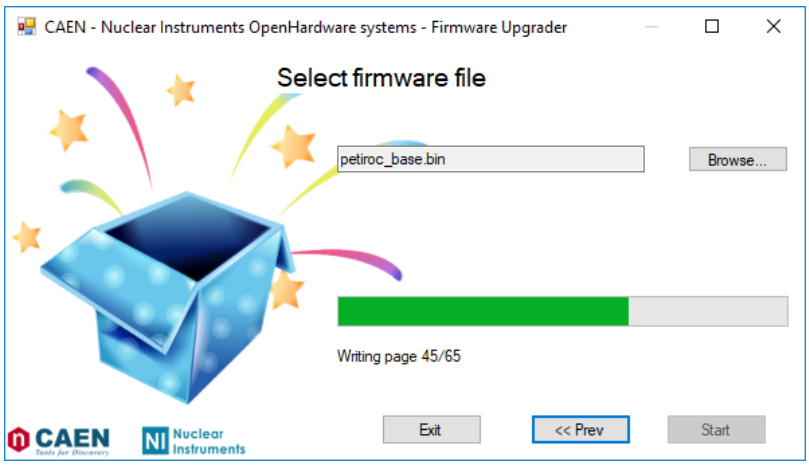
1. **Generate a .bin Flash File,** which contains the firmware code.
2. **Switch the Boot Mode Switch to "Bootloader"** mode with the board switched off.
3. Connect the board to the PC via USB 3.0 communication port and power on.
4. Press the "**Flash Program Tool**" button.
5. A wizard will guide you in the firmware upgrade process. All supported hardware connected by USB to the computer will be listed in the combo box. **Select the hardware** you want to load the firmware on. "Identify" button will make a LED blinks on the selected board.



6. Once the board is chosen from the combo box, press "**Next**" to select the firmware file to load.



7. **Load the .bin file,** containing the firmware code and press the "**Start"** button.



8. When the firmware is fully loaded a message invites the user to **power cycle the board,** in order to load the new firmware.

9. Remember to **switch the Boot Mode Switch to normal operation mode** in order to allow the board to run in standard mode and load the firmware from the flash memory.

WARNING: There is no way for the system to recognize if a firmware is correct for a given hardware. For example, a firmware designed for another board can be loaded on a DT5550, but it can damage the board. User must check the hardware target for a specific firmware BEFORE loading it on the flash memory.

Note: the x5560 firmware can be also permanently upgraded through its dedicated web interface. Refer to **[RD10][RD11][RD12]**for more details about the needed procedure

8 How to test your firmware: the Resource Explorer

Sci-Compiler offers a simple built-in tool, called *Resource Explorer,* to connect one of the supported boards and test the features of the FPGA firmware with no need to write any software code. This tool allows to manage all **Local Bus** readout blocks placed in the firmware diagram(see **Figure 4.2**), therefore it gives the possibility to read and write the configuration registers (i.e. Signal Processing parameters) and shows the readout instruments like *Spectrum* , *Oscilloscope*, etc. in a GUI , so that it is possible to change the signal processing parameters and test the results of acquisition live, with no need to write any software code. In practice, the Resource Explorer is a first-use debug interface to test the firmware and check the behaviour of the Processing algorithm specified through the block diagram.

The *Resource Explorer* can be launched by from the "Home" toolbar. In the “Connection” window that will appear (see **Figure 8.1**), the user can configure the connection to the desired board. In particular, it is possible to choose the board model by clicking on one of the board icons, select one of the options for the "Connection Type" drop down list between USB, Ethernet or VME and insert in the field "IP/SN" the board internet protocol address or the serial number (depending on the chosen connection type). The firmware uploaded on the board can be tested by selecting the .json file located in the "library" folder of the Sci-Compiler project and generated during the compilation process. The user can select the file by clicking the "Browse..." button and search through the computer files. The "Selected Json File" field will show the selected file path. The "Connect" button enables the connection to the board with all the specified settings.

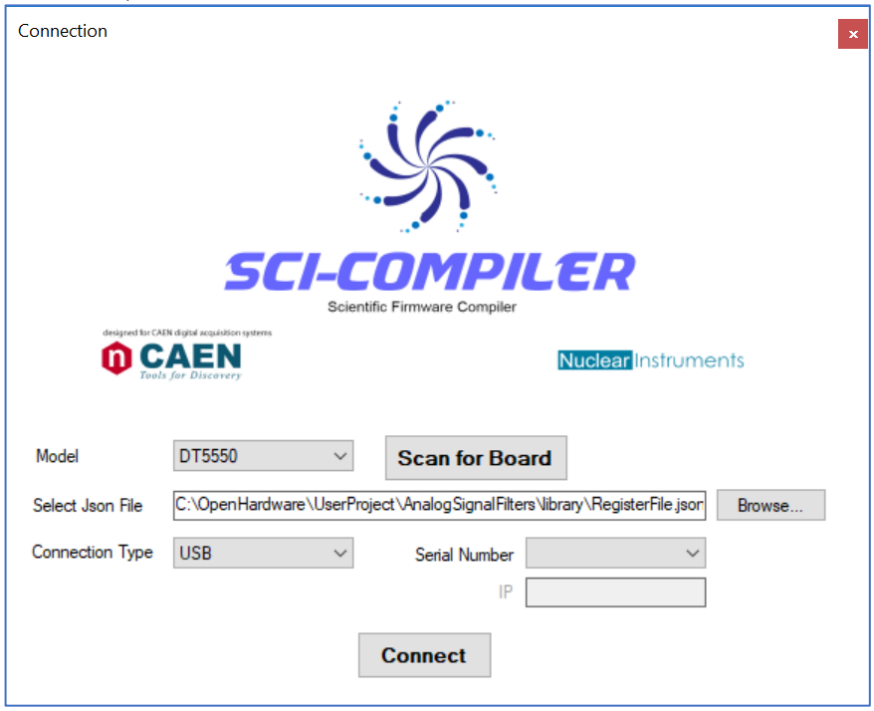


Figure 8.1: the “Connection” window of the Resource Explorer.

The following window will appear, showing the list of all the Registers, List Modules, Logic Analyzers and Oscilloscopes available on the FPGA firmware of the connected board. The items of all the components can be visualized by clicking on the "+" symbol.

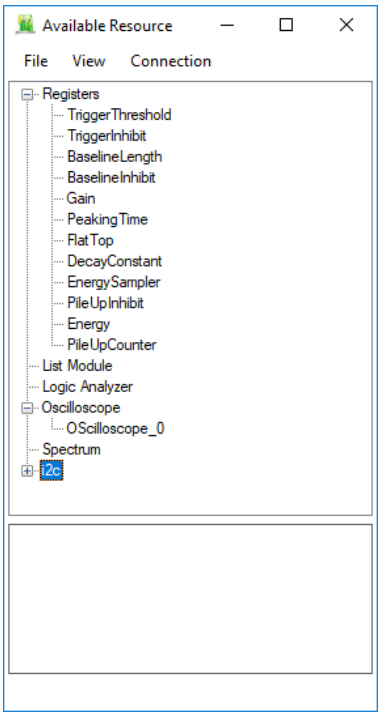


Figure 8.2: example of the “Available Resource” window of the Resource Explorer. All the items available for the considered firmware are listed.

The "File" button opens a menu with the following items:

- the "Open" button allows to load a different .json file by opening a window to search through the computer files. The file should be compatible with the connected board, otherwise an error will notify the issue to the user.
- the "Save" button enables the user to save the .json file related to the registers listed in the “Available Resource”.
- the "Exit" button closes the "Resource Explorer" tool.

The "View" button allows to create a new table or plot that will be used to set or get the values from a set of registers simultaneously.

The "Connection" opens a menu with the following items:

- the "Disconnect" button closes the connection with the board.
- the "New Connection" allows to create a new connection by showing again the Resource Explorer “Connection” window.

Note: for a detailed description of the Resource Explorer functionalities, refer to the **online Help**

9 SciSDK

The SciSDK is a Software Development Kit installed together with Sci-Compiler setup and compatible with any board supported by Sci-Compiler.

Note: for specific documentation of the functions and parameters of the SciSDK, refer to the Sci-Compiler **online Help** or to **https://nuclearinstruments.github.io/SCISDK/**

Note: alternative installation methods for the SciSDK are given in the online documentation

10 Creating your first project

In this Chapter we give two examples to be taken as reference for the procedure to create a project in Sci-Compiler and compile the firmware, in local mode using the Remote Customization Service. We take some specific boards as examples, but the method can be extended to all the compatible hardware.

Step by step example – Remote Customization Service

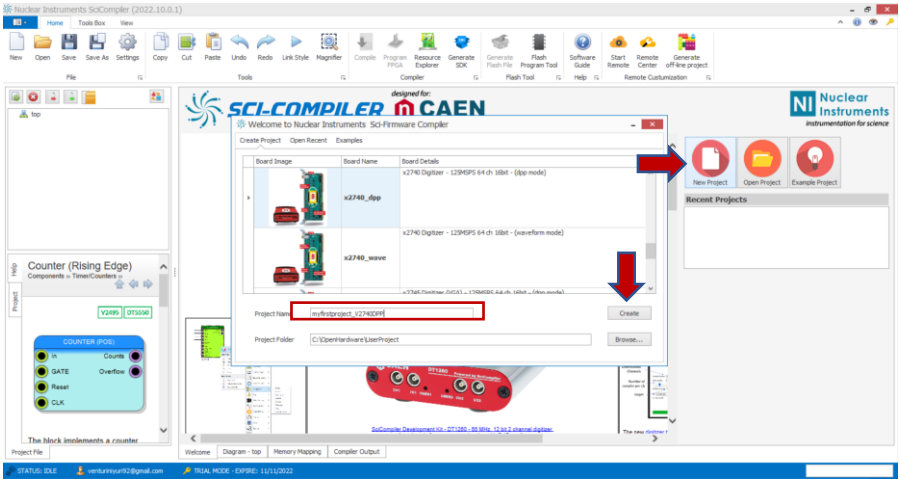
In this step-by-step example we give instructions to design a custom firmware for 2740/2745 Digitizers family to extract counts and charge of an analog exponential signal. **Since we are interested in quantities extracted by the pulse processing algorithm only and not in the full waveform, we create a DPP project.**

Note: in this Paragraph we give instructions based on a DPP project V2740 board, but it can be extended to all 2740/45 Digitizer Family models (V2740, VX2740, DT2740, V2745, VX2745, DT2745). Moreover, the general concepts can be extended to DT5550x, x5560 families as well.

1. Activate your Sci-Compiler license and set up the software installation as described in Par. **Sci-Compiler License Activation**, **Sci-Compiler Installation**, Errore. L'origine riferimento non è stata trovata..

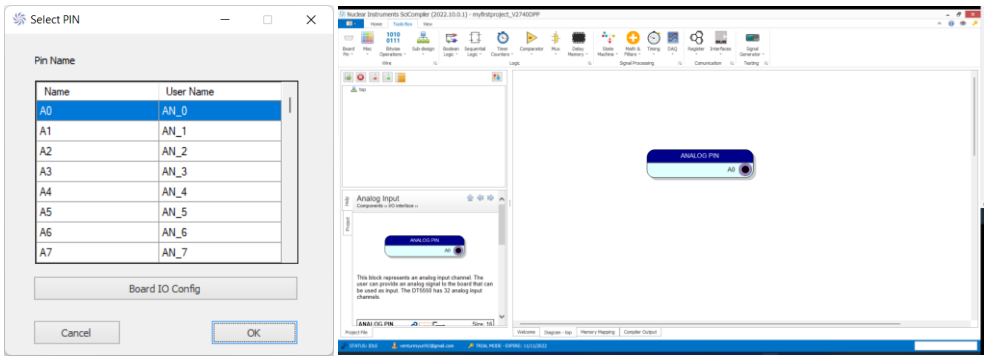
2. Plug the USB Dongle in your PC and launch Sci-Compiler

3. **Create a project** for your V2740 board: press *New Project* to open the selection menu→ Select *x2740_dpp* → Choose a name/folder for your project → Press *Create*.



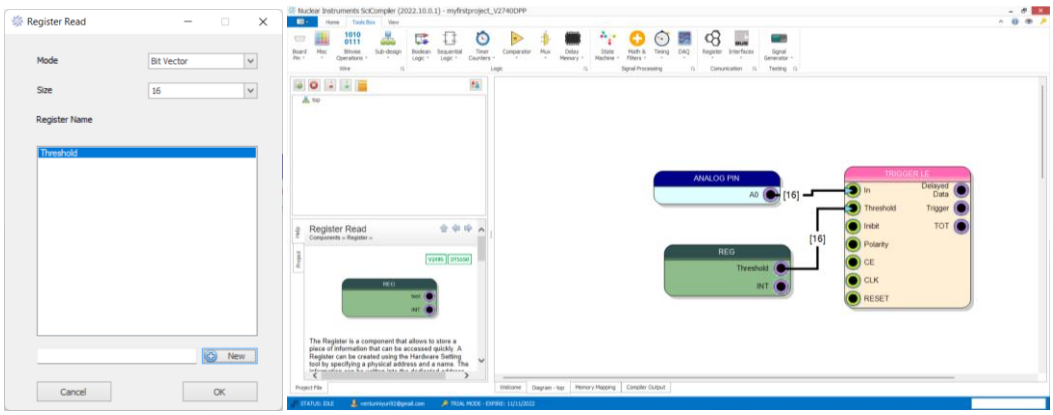
Note: In alternative to creating a project from scratch, Sci-Compiler offers ready-to-use examples that can be opened and modified to speed up the learning curve.

4. First of all, we place an analog input in the diagram. From the Tools Box bar, click Board Pin → Analog In and select, for instance, A0, corresponding to the physical CH0 of the V2740 board.

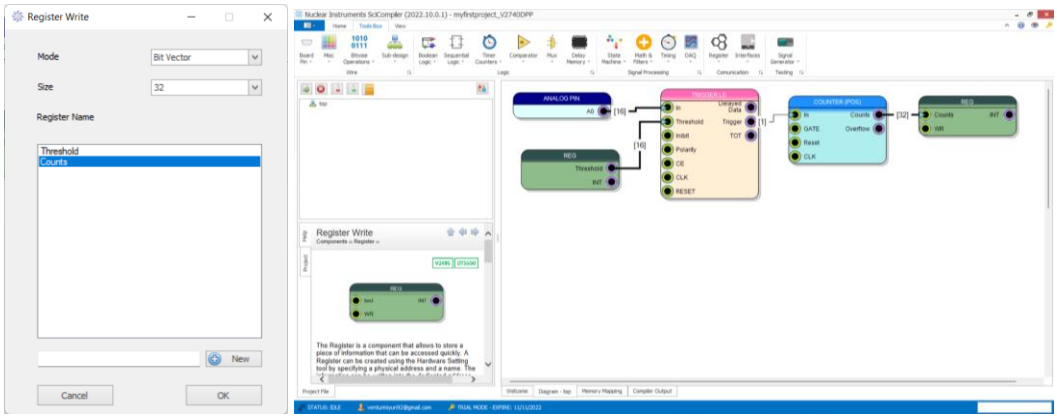


5. To count the input signals, we need first to discriminate them. Therefore, we should place in the diagram a Leading-Edge Discriminator with a programmable threshold via a register read by the discriminator block. Place the discriminator from **DAQ→Trigger Leading Edge** and the register from **Register→ Register (read)**. In the

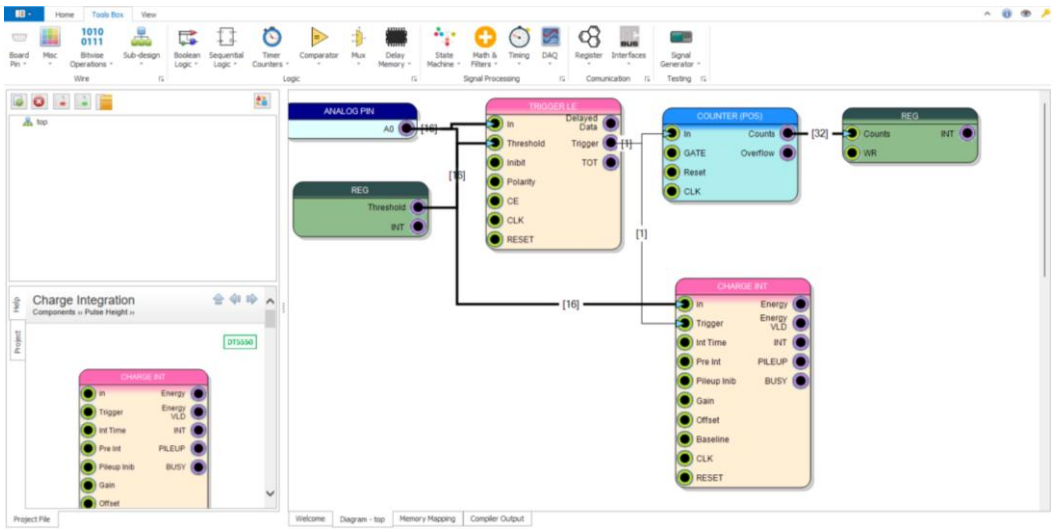
Register window choose Bit Vector mode, size 16, in order to match the Leading-Edge Discriminator block requirements. Give a name to the register (for instance *Threshold*), press New and then select the register name and press OK. Connect the blocks as shown in the figure below.



6. If the input signal has positive polarity, we now have to add a counter triggering on the rising edge of the signal. Place the counter block from **TimerCounter → Counter (Rising Edge)** and connect it as shown in the figure below. In order to store the counts number in a register written by the Counter (POS) block, add a Register Write with name *Counts,* Bit Vector mode, Size 32 and connect it to the *Counts* output of the counter block.

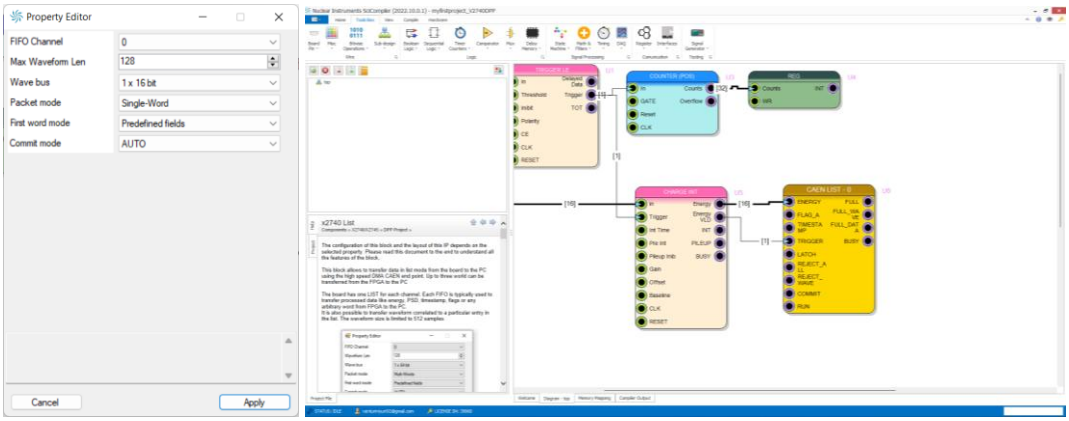


7. The charge of the signal can be calculated with the Charge Integration block. Place it in the diagram from **DAQ→ Charge integration** and connect it as shown in the figure below, so that the Analog In is the input of the Charge Integration block and the trigger is given by the Leading Edge Discriminator.
For the sake of simplicity, leave the the Charge Integration parameters – Integration Time, Pre-Integration, Pileup and Gain - to their default values (see Online Help of the Charge Integration block)



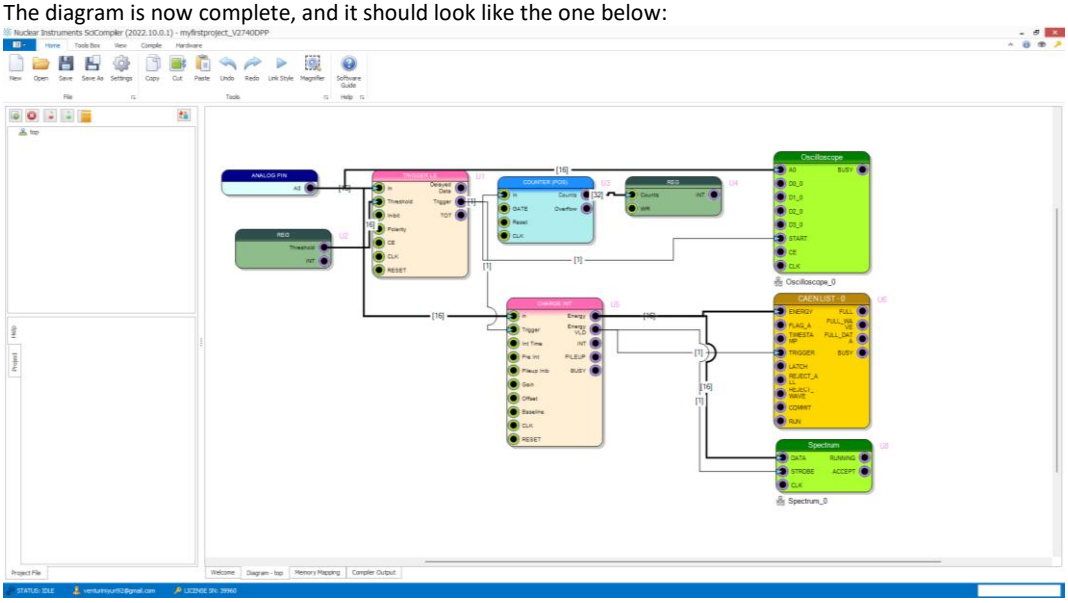
8. To define the Charge Integration parameters, place in the diagram a *Register→ Register File* block.

9. In order to transfer energy data in a packet that can be easily handled by the V2740 readout we place the proper block from **DAQ→x2740 CAEN List**. This is one of the Special Readout blocks that allows to transfer data in list mode from the board to the PC using the high-speed DMA endpoint (refer to **Readout structure and Resource Explorer** for more details). The configuration details of the *CAEN list* block are available in the online help. For this purpose, we choose FIFO Channel = 0, Packet Mode = Single Word and Predefined fields. Connect the blocks as shown in the figure below (Energy→ Energy; EnergyVLD→Trigger)

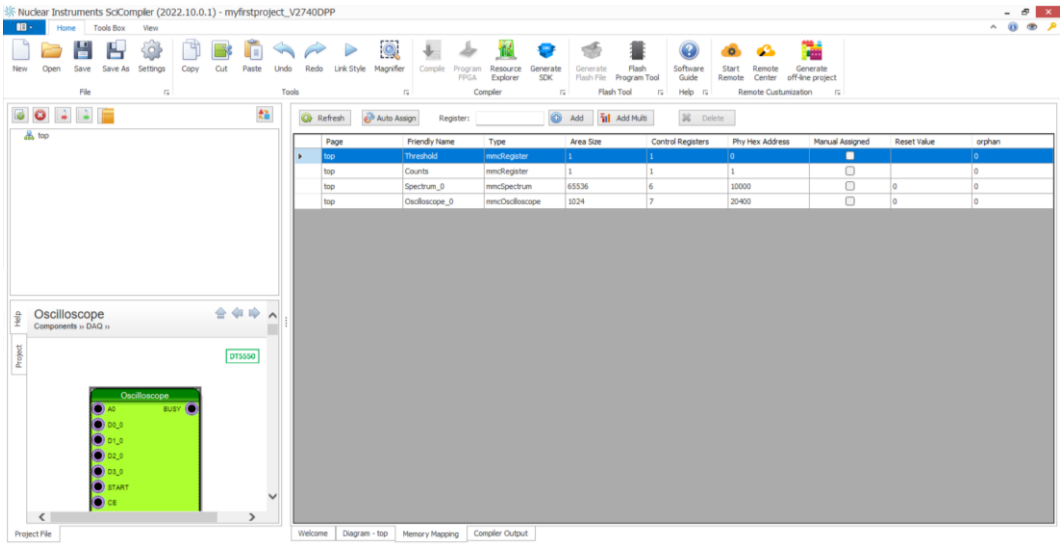


10. In order to easily test the firmware with the Resource Explorer before using more complete DAQ software code, we can place a **DAQ→Oscilloscope** block as debug monitor for the analog input and a **DAQ→Spectrum** block to readout the energy from the Charge Integration block. The Resource Explorer will also allow to set the threshold of the discriminator and read the number of counts detected by the counter.
Connect the Analog Pin A0 to A0 channel of the Oscilloscope, with the Oscilloscope Start given by the trigger output of the Leading-Edge trigger block. Following the same concept, connect the Energy output of the Charge Integration to Data input of the Spectrum block, with the Strobe given by the Energy VLD signal coming from the Charge Integration.

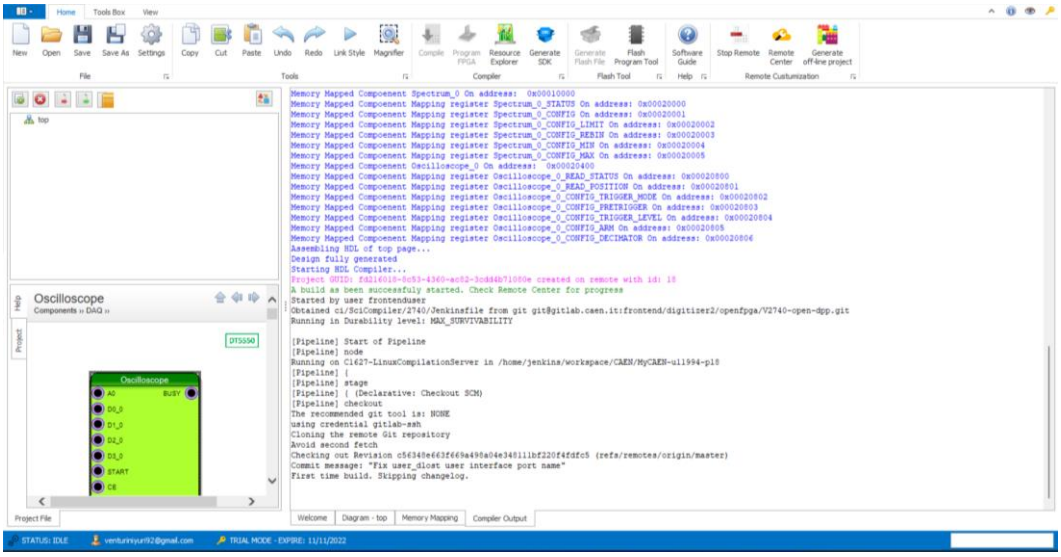
The diagram is now complete, and it should look like the one below:

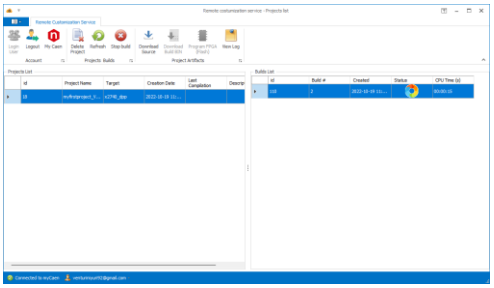


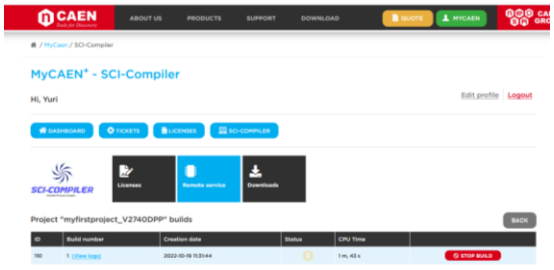
11. Before compilation, you need to set properly the Local Bus Readout blocks (registers, Oscilloscope and Spectrum). To do this, go into the *Memory Mapping* tab, where those blocks are listed, and **press *Auto Assign***. This operation assigns a Physical Hex Address to each block. A manual assignment is also possible.



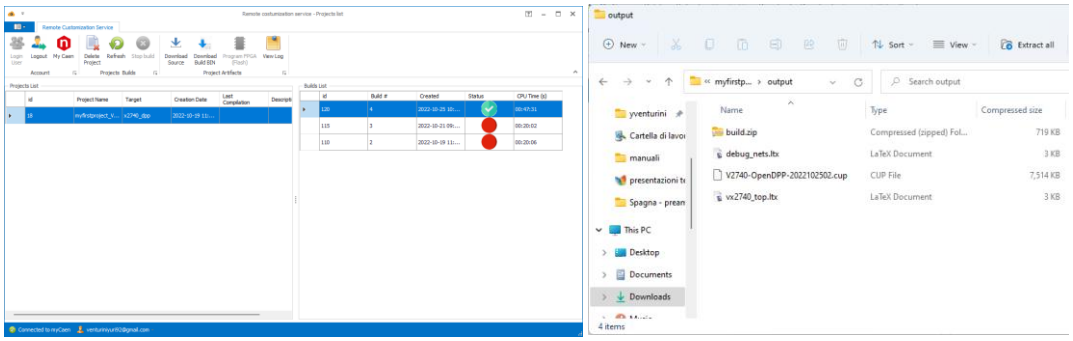
12. You are now ready to start the compilation. If you are logged in the MyCAEN+ (see Status Toolbar at the bottom of the software window), you can use the Remote Customization Service to synthetize the firmware for the V2740, with no need of installing any local Vivado compiler.
From *Compile* toolbar, press **Start Remote.** The software will automatically display the compilation log while compiling, and you can double-check the status of the compilation from the **Remote Center** or directly from your MyCAEN+ area (see figures below).



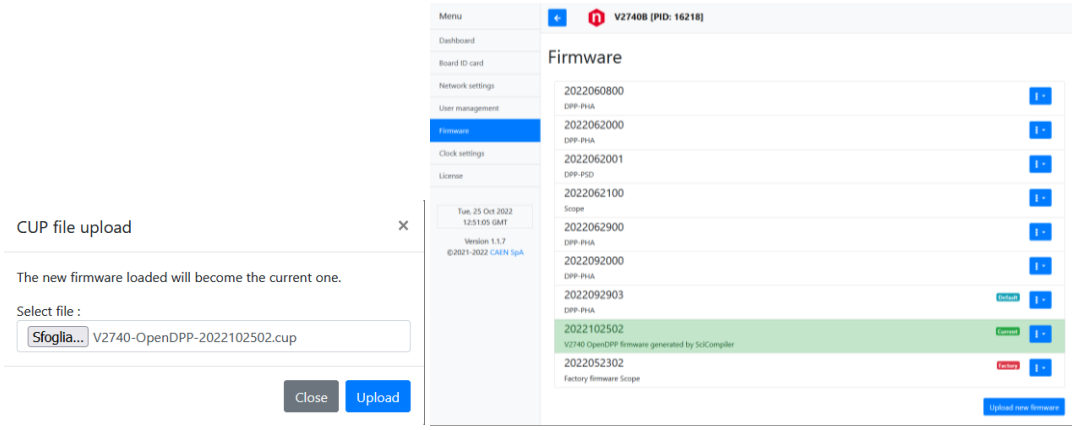




13. After compilation has been successful, the final firmware file (in the case of 2740/45 a .cup file) will be available through the Remote Center. Open the Remote Center, select the line corresponding to the successful compilation and press *Download Build BIN* to obtain the final firmware file. The Remote Center will download a .zip file containing the file to be deployed on the target hardware. In the 2740/45 case, the file is a *.cup* extension file. Extract that file from the .zip archive.

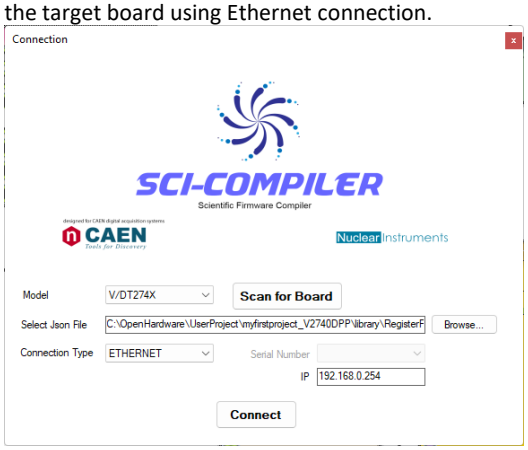


14. Load the .cup file onto the target board, in our case a V2740, following the instructions for firmware upgrade given in **[RD13] – Par.10.8**. Briefly recalling the procedure here, connect to the board Web Interface, surf into the Firmware page and *Upload new firmware.*

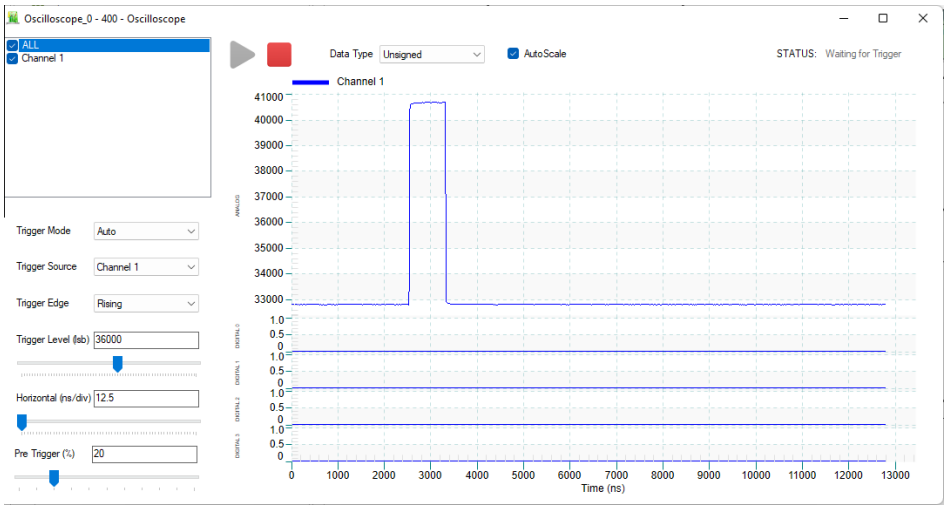


Note: the board can run the firmware generated via Sci-Compiler if a valid runtime license is installed onboard. It is possible to check the Sci-Compiler Runtime status in the *Dashboard* tab of the board Web Interface. If no runtime is installed, the firmware can run for 30-minutes, and it is possible to monitor the timebomb status from the Web Interface.

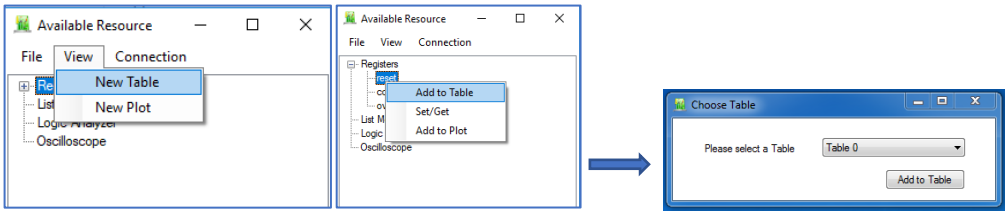
15. After upgrade, the board is running the firmware previously designed in Sci-Compiler and we can test it, for instance using the Resource Explorer tool. Open the Resource Explorer from Hardware toolbar and connect to the target board using Ethernet connection.



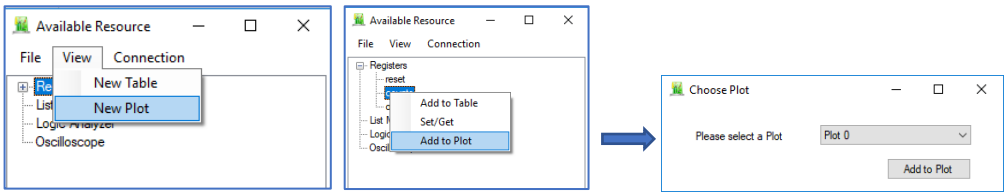
16. In the Resource Explorer, the available resources are listed (in this case Oscilloscope, Spectrum and the two registers to set the threshold and read the counts). As a first test, we want to see a signal at ch0 of the board using the Oscilloscope. Feed the V2740 with a signal (in this case a square pulse, 500mV, 1us width, 1kHz) and double-check the result from the Oscilloscope. From the Resources list right click → Oscilloscope→ View to open the oscilloscope window. Set Trigger Source = Free Running at first, to see the level of the baseline and press start. Once you have an estimate of a proper trigger level, set Trigger Source = Channel 1. In our case we set Trigger Level = 36000 to trigger correctly on the positive edge of the signal.



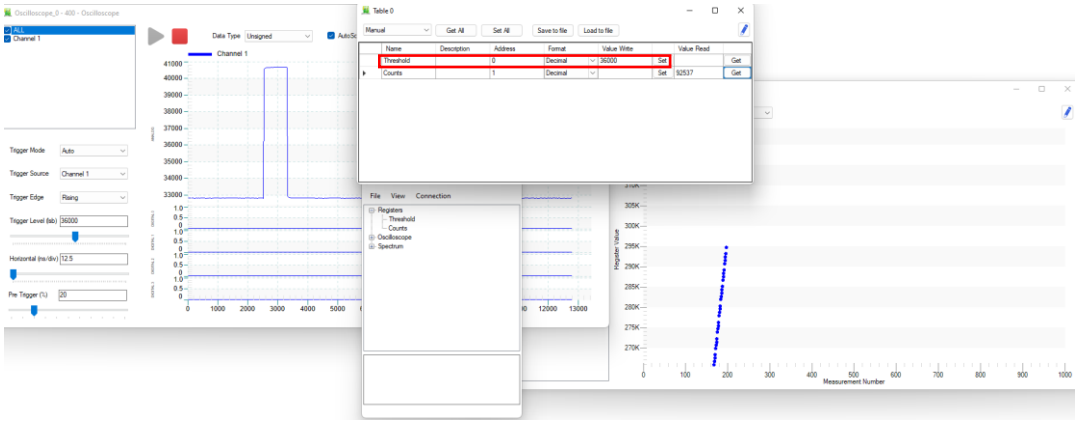
17. Now that we have found the right threshold level from the Oscilloscope instrument, we set the same threshold in the firmware threshold register. Click View→ New Table. Right click on each register name → Add to table.



At the same time, it is possible to add a plot for the counts register. Click ViewàNew Plot. Right-click on "counts" register and Add to Plot



From the table, set Threshold register =36000 and monitor the counts trend in the plot. Increasing the threshold up to 45000, it is possible to notice that the plot will stop, due to the leading edge trigger being above the signal level.



In the same way it is possible to look at the energy calculation result using the Spectrum resource listed among the Available Resources.

Step by step example – Local Compilation

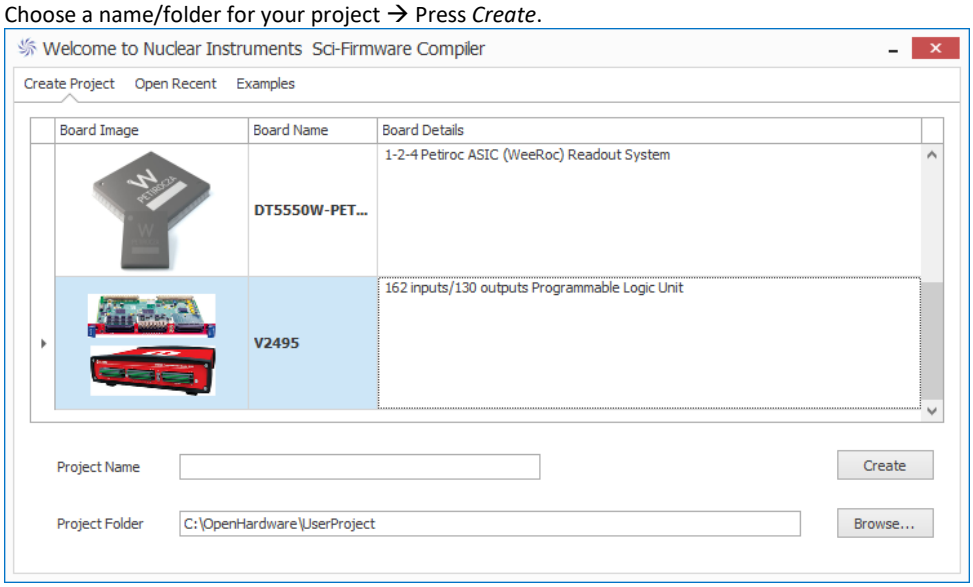
This step-by-step example will show how to design a firmware for the V2495 that implements a counter of rising edge signals fed at G0 input. Some registers will be used to control the counter and LED 0 on the board will light up when the counted edges are more than a certain value.

Note: in this Paragraph we give instructions basing on a V2495 board, but the general concepts can be extended to all boards for which it is possible to use the local compilation.

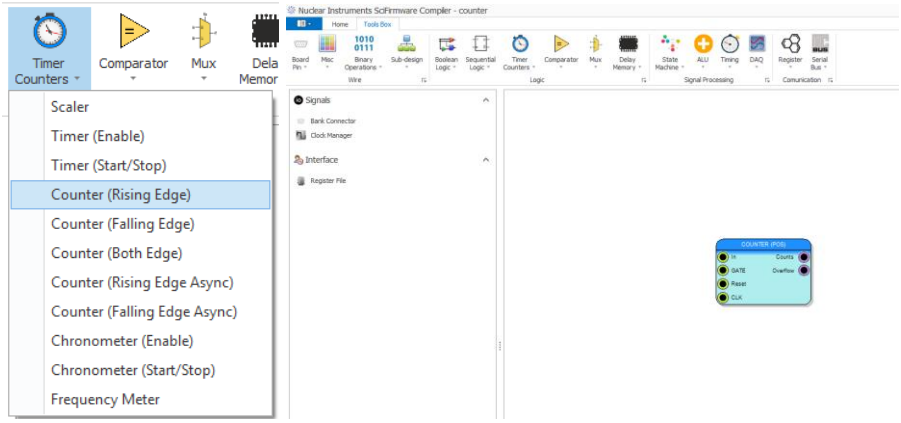
1. Activate your Sci-Compiler license and set up the software installation as described in Par. **Sci-Compiler License Activation**, **Sci-Compiler Installation**, Errore. L'origine riferimento non è stata trovata..

2. Plug the USB Dongle in your PC and launch Sci-Compiler

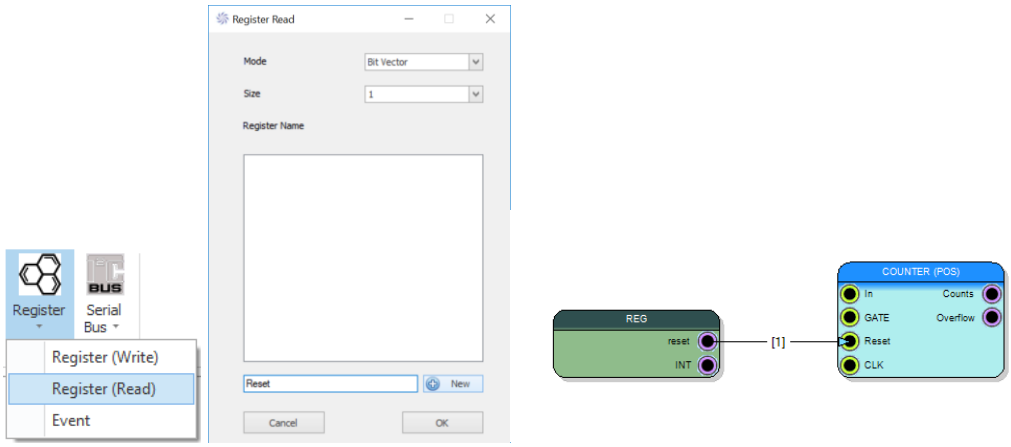
3. **Create a project** for your V2495 board: press *New Project* to open the selection menu→ Select *V2495* → Choose a name/folder for your project → Press *Create*.



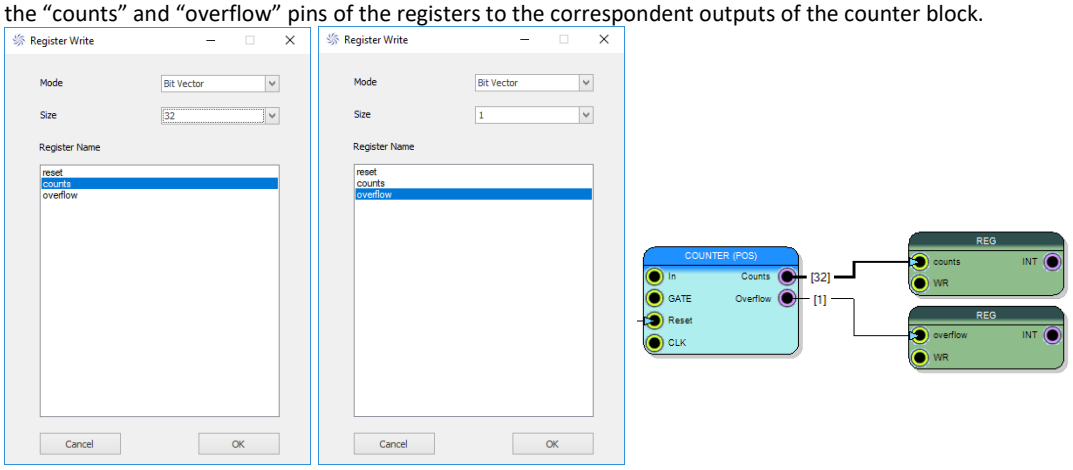
4. Add the Counter (Rising Edge). Select the "Tools Box" toolbar and click on *Timer Counters* à *Counter (Rising Edge)*. The block will be added to the diagram.



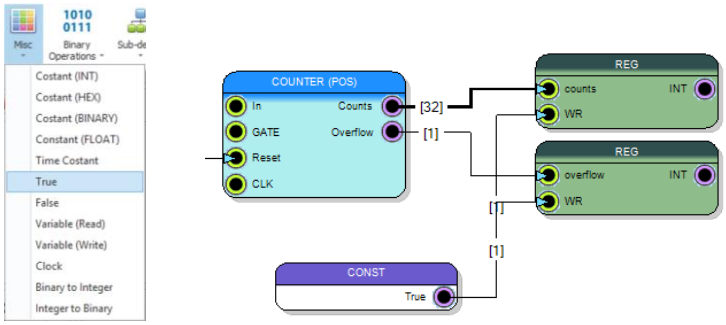
5. Add the *reset* register. In the toolbar, click *Register (Read)* and in the following window create the "reset" register. Set the *Bit Vector* "Mode" and, in the "Size" field, the value 1. Click "New" to create the register and "Ok" to add it to the diagram. Connect the "reset" output of the register block to the "Reset" input of the counter.



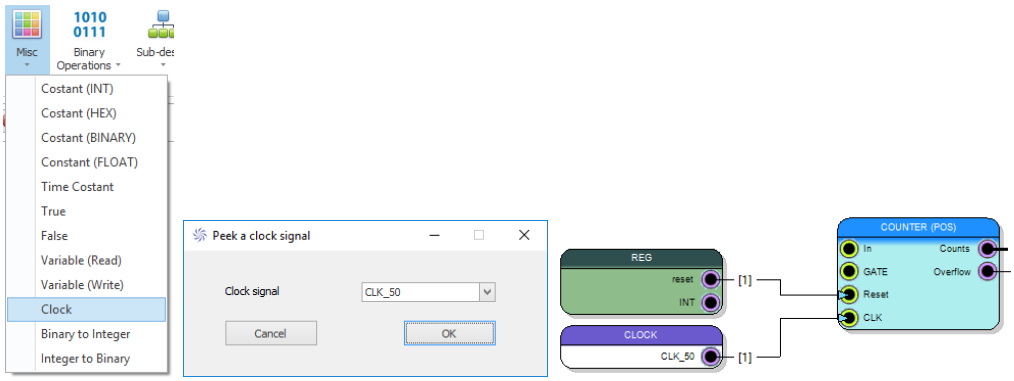
6. Add the *counts* and *overflow* register. Click Register à *Register (Write)* and in the following window create the *counts* register. Set “Mode”=Bit Vector and "Size"=32. Do the same to create the *overflow* register with “Mode”=Bit Vector and "Size"=1. Select them from the list and press “Ok” to add them to the diagram. Connect the “counts” and “overflow” pins of the registers to the correspondent outputs of the counter block.



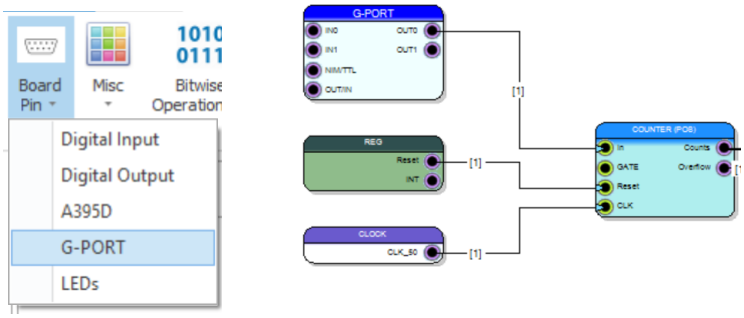
7. In order to allow writing in the *counts* and *overflow* register, click "Misc" à"*True*" and connect the block to the "WR" input of the two registers.



8. Add a clock: click on *Misc* à *Clock* and in the following window select CLK_50. Then connect the clock to the CLK input of the counter block.



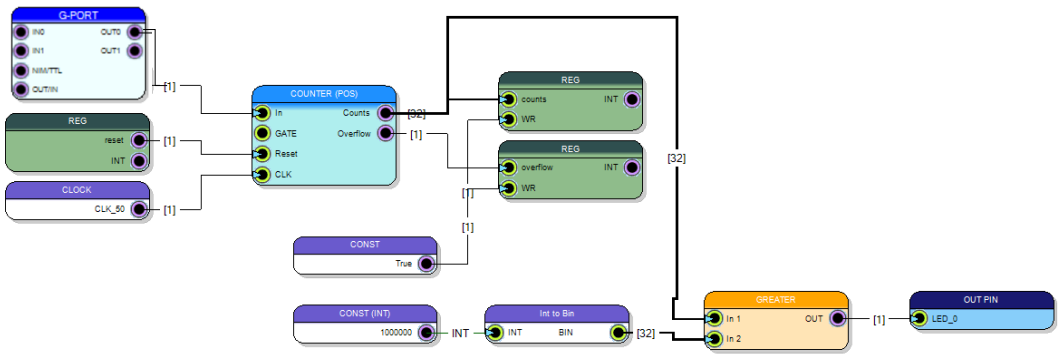
9. Add the Input. Click Board Pin à to add the G0 port of the board to the diagram. Connect the "OUT0" of the block to the "In" input of the counter block.



10. Create the chain to implement a comparator which turn on LED 0 of the board if the counter exceeds the value of 10^6. Create the following blocks:

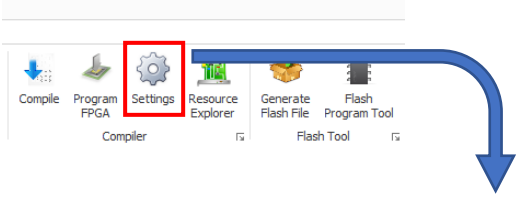
- Click on *Misc* à *Constant (INT)* and in the following window set it to 1000000.
- Click on *Misc* à *Integer to Binary.*
- Click on *Comparator à Greater.*
- Click on *Board Pin* à *Digital Output* and in the following window select LED_0.

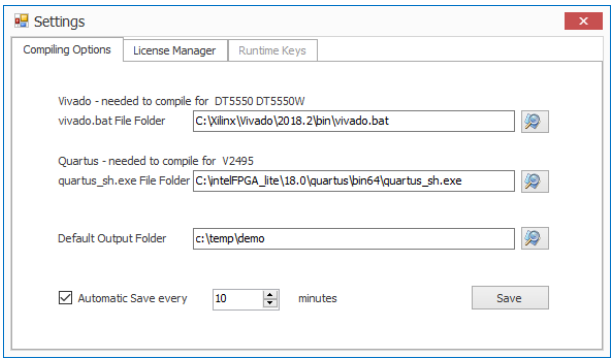
Then connect the created blocks as shown in the following picture:



11. The firmware diagram is now complete, and it should look as the one displayed in the previous picture.

12. Check the Sci-Compiler Settings. In order to perform the following steps, you need **Altera Quartus** to be installed on your PC and Sci-Compiler must be set to link at the **quartus_sh.exe** executable, as shown in the picture below.



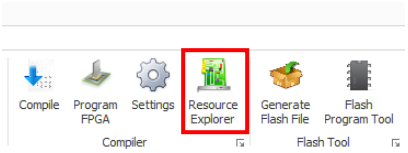


13. Compile the firmware. Select the "Home" toolbar and press the ***Compile*** button. The Sci-Compiler executes the firmware compilation by running Quartus and redirecting the output in the "Compiler Output" tab. At the end of the process, the text "Successful Compilation!" is shown in the output.

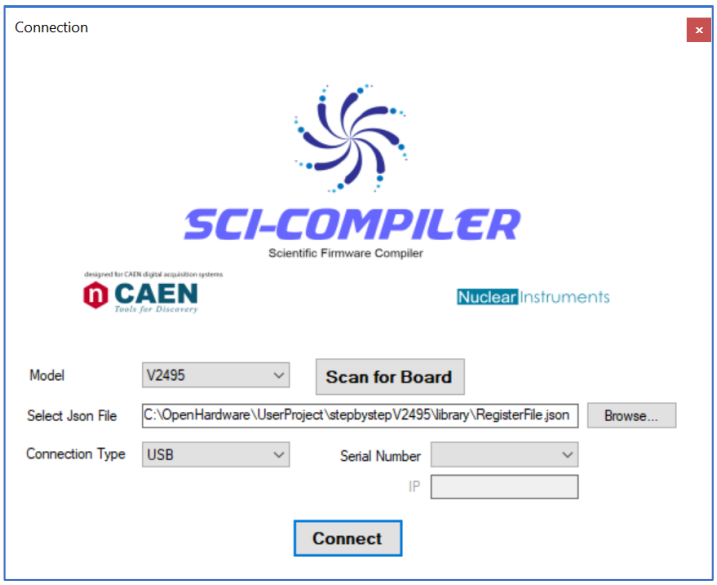
14. Program the FPGA. For this step, the V2495 board should be powered on and connected to the computer through one of the supported programmer cables (USB Altera Blaster). Then press the ***Program FPGA*** button to automatically connect to the USB Blaster and download the generated firmware to the FPGA. The text "Target device programmed successfully" that appears in the "Compiler Output" states the end of the process.

Note: When using the “Program FPGA” button and the programmer cable, the firmware is written in the volatile FPGA memory: a power cycle of the board will cause the firmware loss, and the firmware stored in the flash FPGA memory is loaded again. In order to upgrade the firmware flashing permanently the FPGA, you can use CAENUpgrader and load the flash firmware file (.rpd for V2495) generated by Sci-Compiler in the project *output* folder. Refer to **[RD3]** for more details.

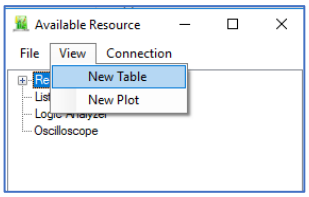
15. Test the firmware. The generated firmware can be tested by using the **Resource Explorer** tool. Connect the V2495 to the PC via mini-USB port and click the *Resource Explorer* button.



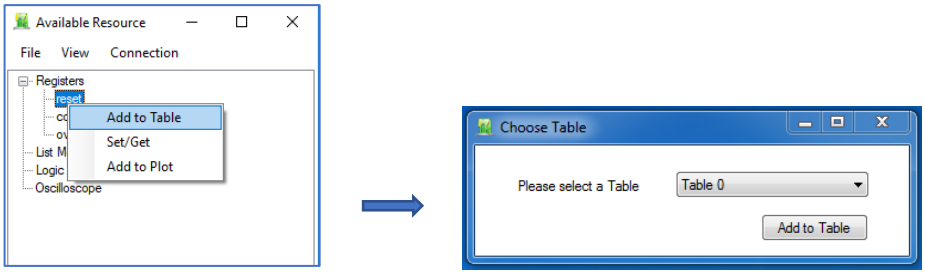
In the "Connection" window select the V2495 board, the desired "Connection Type" (USB) and the correct "Serial Number". The "Select Json File" will be automatically filled with path of the .json file created during the firmware compilation process. By clicking the "Connect" button the connection with the board will be established.



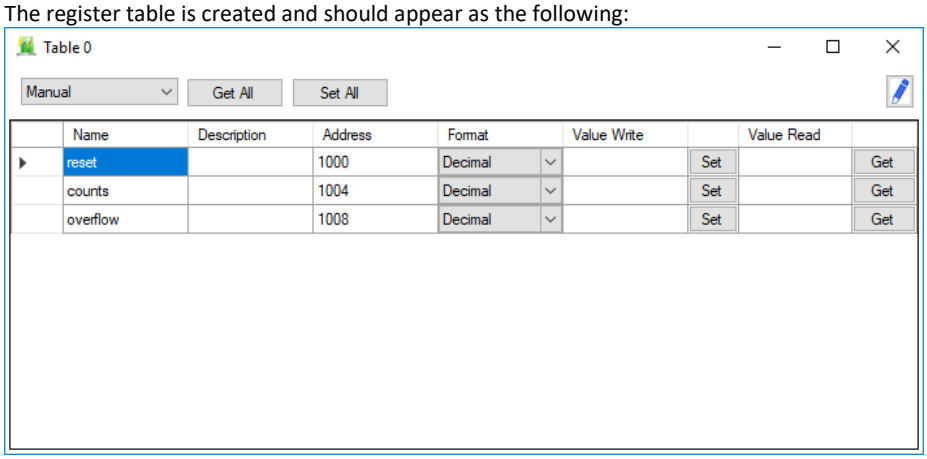
In the following window the resources available on the FPGA (Registers, List Modules, Logic Analyzers and Oscilloscopes) will be shown. Click the "View" menu and create a table to control the registers by clicking the "New Table" button.



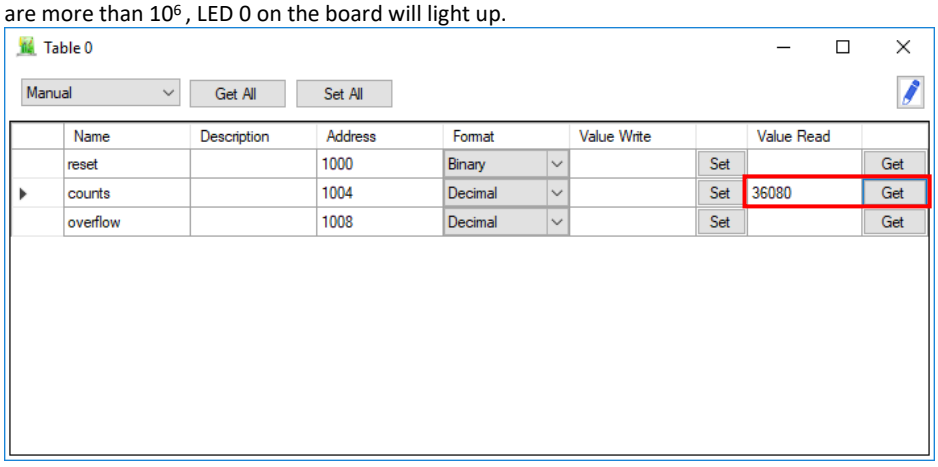
Then right -click on each register and click the "Add to Table" button: select the "Table 0" in the “Choose Table” window and click the "Add to Table" button.



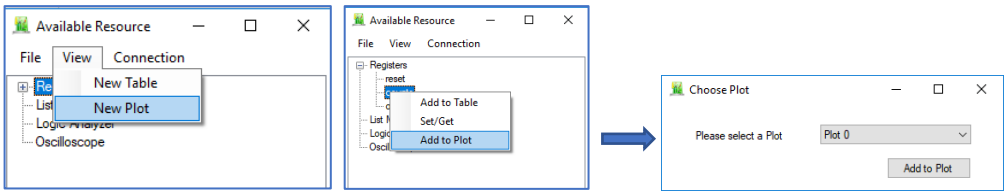
The register table is created and should appear as the following:



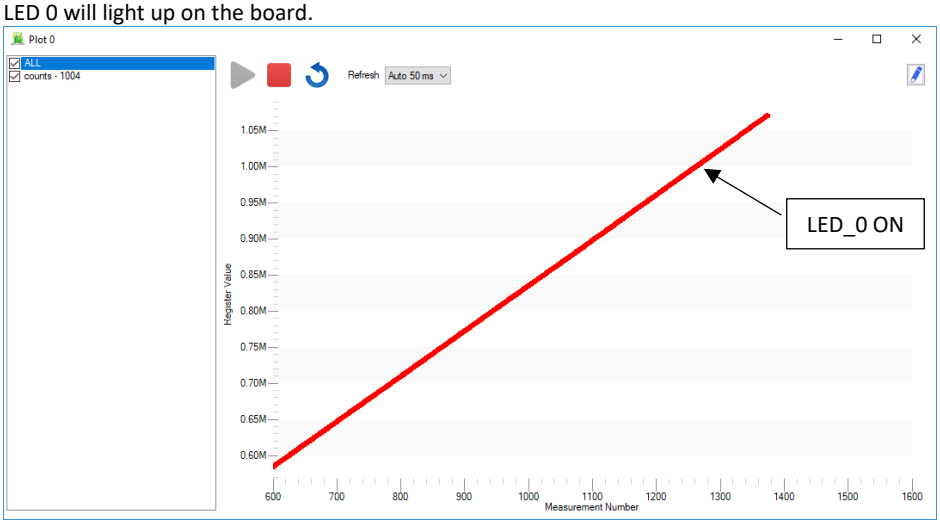
While feeding a NIM input at the G0 port of the V2495, you can press the “Get” button on the counts register line and see if the counts are rising up accordingly to you input signal. You can control that when the counts are more than 10^6, LED 0 on the board will light up.



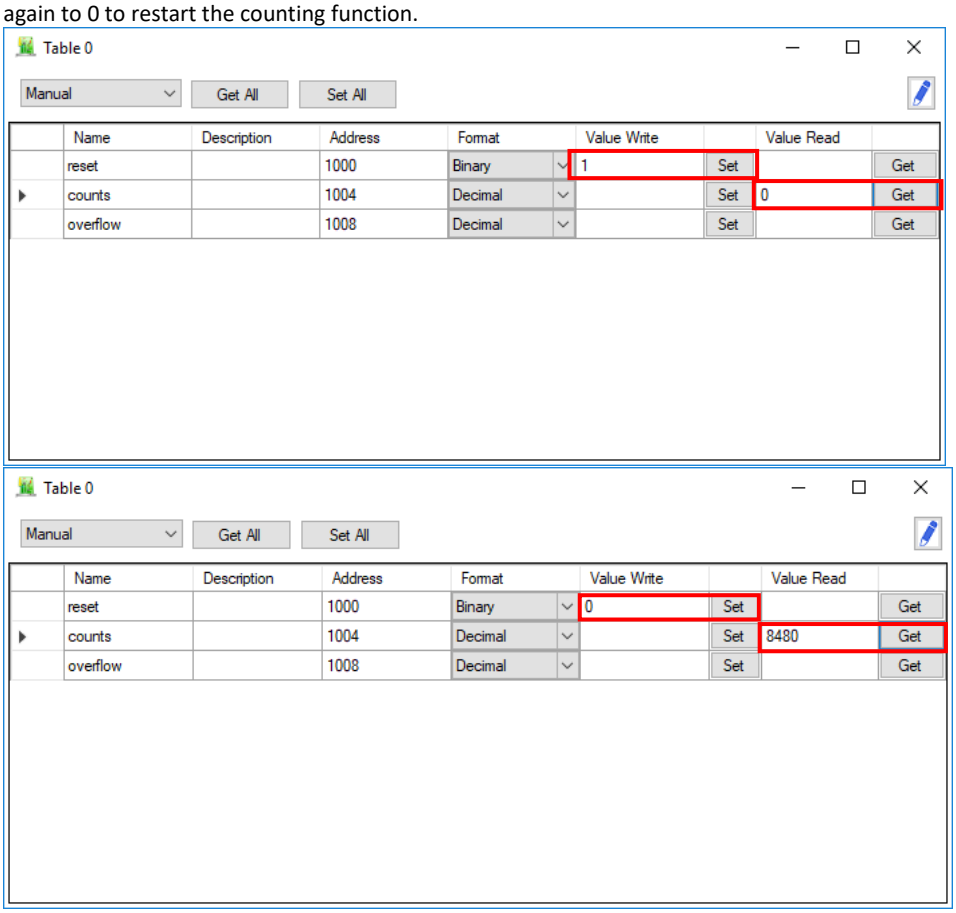
You can also monitor the “counts” register value on a plot. Click the "View" menu and create a “New Plot”. Then right-click on “counts” register and “Add to Plot”: select “Plot 0” in the following window and click the “Add to Plot” button.



Press the Start button and you will see the counts register value plotted in time. When reaching 1M value, the LED 0 will light up on the board.



You can use the "Set" function on the reset register: set it to 1 (Binary format) to reset the counter and then again to 0 to restart the counting function.



11 Appendix: on-fly FPGA programming

Sci-Compiler offers the possibility to program on-fly the FPGA of some compatible boards, allowing to test the firmware with no need to permanently flash the FPGA.

Note: on-fly programming works for DT5550, DT5550W, x5560, V2495, DT5495 only when **local compilation** is used

The DT5550/DT5550W/x5560 must be connected to the computer through its specific JTAG port (use the latter for a fast-volatile firmware upgrade). The V2495/DT5495 can be programmed trough the supported programmer cable USB-Blaster, described in **[RD9]**.

After compilation, the *Program FPGA* button in the "Home" toolbar can be clicked to start the FPGA programming process. The "Compiler Output" automatically displays the redirected Vivado/Quartus output, the progress bar on the right corner of the status bar starts to indicate the progression of the programming process, while the status changes from IDLE to PROGRAMMING.

For the DT5550x series boards and x5560, Sci-Compiler generates a .tcl file in the "HDL" folder of the current project, to execute from the shell the Vivado software. In this file it is specified the .bit file that must be used to program the FPGA, which is automatically taken from the "output" folder of the project currently opened in Sci-Compiler. First, Vivado opens the available programmer cables, connects to them and opens the correspondent FPGA. Then Vivado programs the FPGA with the bitstream file and when the process is terminated, if no error occurred, the "Target device programmed successfully!" message is displayed. Otherwise, the process is aborted, and an error message is reported in red to notify to the user the issue that occurred. In both cases, at the end of the process, the status bar returns to "IDLE" and the progression bar is reset to be ready for a new process.

For the x495 series boards, Sci-Compiler automatically uses the .sof file in the "output" folder as firmware file of the currently open project. First, Quartus is executed to list the available USB-Blaster cables, which will be shown in the "Compiler Output". Then, if a USB-Blaster cable has been identified, automatically the firmware is downloaded to the correspondent FPGA. The "Target device programmed successfully!" message is displayed at the end of the process if no error occurred. In case of error, the process is aborted, and a message is reported in red to indicate the issue to the user. In both cases, at the end of the process, the status bar returns to "IDLE" and the progression bar is reset to be ready for a new process.

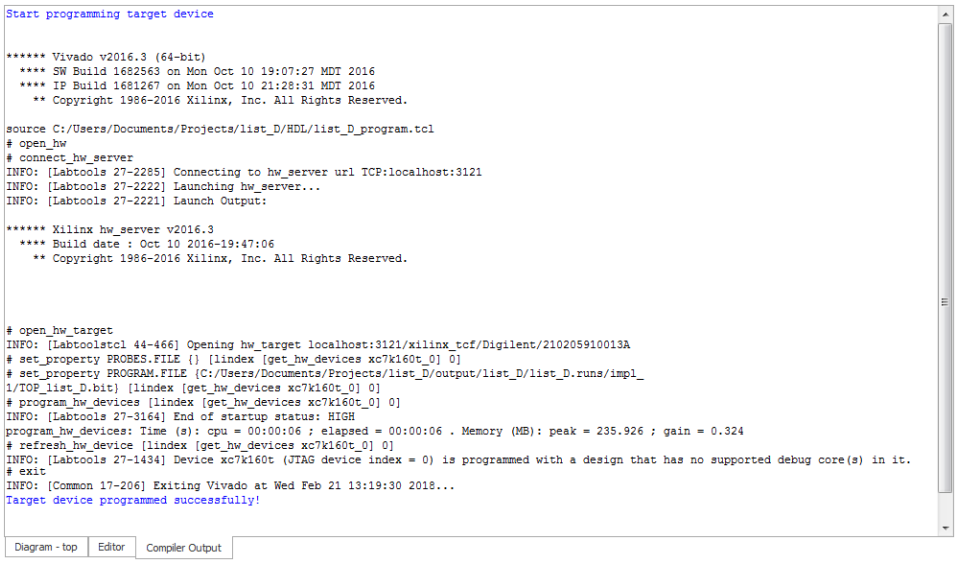


Figure 11.1: the “Compiler Output” after successful FPGA programming for a DT5550.

12 Technical Support

To contact CAEN specialists for requests on the software, hardware, and board return and repair, it is necessary a MyCAEN+ account on www.caen.it:

https://www.caen.it/support-services/getting-started-with-mycaen-portal/

All the instructions for use the Support platform are in the document:

A paper copy of the document is delivered with CAEN boards.

The document is downloadable for free in PDF digital format at:

https://www.caen.it/wp-content/uploads/2022/11/Safety_information_Product_support_W.pdf

Sci-Compiler Quick Start
Graphical Programming Language for CAEN Open FPGA Boards

CAEN GmbH
Eckehardweg 10
42653 - Solingen
Germany
Phone +49 212 254 40 77
Fax +49 212 254 40 79
info@caen-de.com
www.caen-de.com

CAEN Technologies, Inc.
1 Edgewater Street - Suite 101
Staten Island, NY 10305
USA
Phone: +1 (718) 981-0401
Fax: +1 (718) 556-9185
info@caentechnologies.com
www.caentechnologies.com

CAEN S.p.A.
Via Vetraia 11
55049 - Viareggio
Italy
Phone +39 0584 388 398
Fax +39 0584 388 959
info@caen.it
www.caen.it

CAENspa INDIA Private Limited
B205, BLDG42, B Wing,
Azad Nagar Sangam CHS,
Mhada Layout, Azad Nagar, Andheri (W)
Mumbai, Mumbai City,
Maharashtra, India, 400053
info@caen-india.in
www.caen-india.in

UM6520 - Sci-Compiler Quick Start rev. 7 - March 14th, 2025 00000-22-SCICOMP-MUTX