Aurora link with Plexim via SFP

PN111 | Posted on January 22, 2026 | Updated on February 2, 2026

François LEDENT
Development Engineer
imperix

Table of Contents

An introduction to Aurora communication with third-party devices is available in Aurora link with third-parties via SFP. This page extends that overview by presenting a practical example of SFP communication with Plexim simulators, specifically the RT-Box 1, RT-Box 2 and RT-Box 3.

The proposed example relies on the user application and bitstream generation scripts introduced previously, and includes a fully functional PLECS model for the RT-Box simulator.

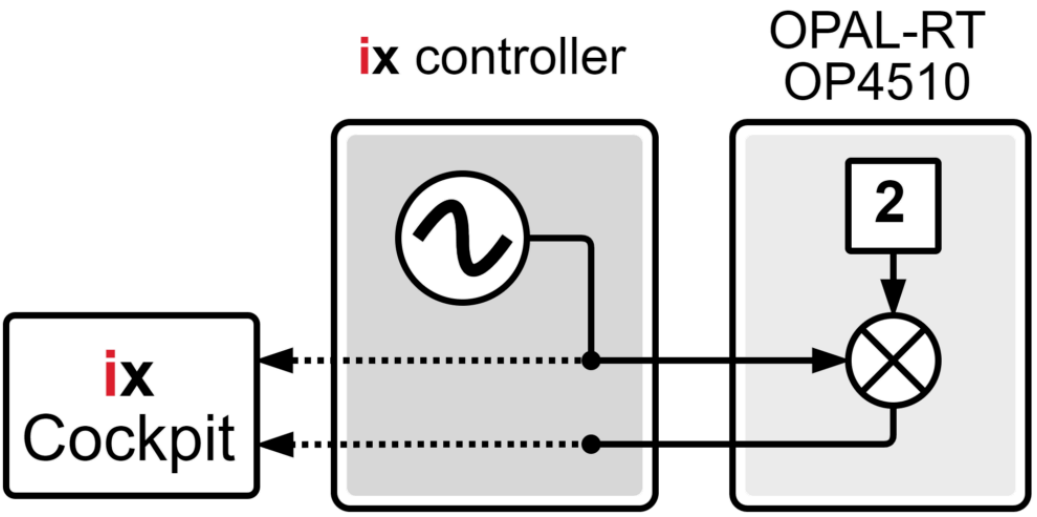
A simple loopback configuration is used to illustrate the setup: a three-phase sine wave is transmitted from the imperix controller, multiplied by two within the RT-Box simulator, and then sent back to the controller.

A page introducing the SFP communication with OPAL-RT devices through a similar loopback example is available at Aurora link with OPAL-RT via SFP.

Case study

This case study demonstrates a straightforward signal processing loop:

- The controller generates a three-phase sine wave which is transmitted to the OP4510 via the Aurora protocol.
- Upon receipt, the OP4510 applies a gain of 2 to the three signals – doubling the amplitude of the sine wave – and returns them to the controller.
- Finally, the original transmitted values and the received return signals are compared in real-time in Cockpit, showing the proper operation of the system.



Required software

- **Vivado Design Suite** (version **2022.1** is recommended)
  The Xilinx installation page details the installation procedure.
- **FPGA sandbox template 3.10** or later.
  Available on the FPGA download page.
- **C++ or ACG SDK version 2024.3** or later.
  Available on the SDK download page.

This project has been tested with a **Plexim RT-Box 1** and **PLECS 4.5.9**.

Downloads

As explained in setup overview, the SFP communication with any third-party device requires the three following software parts:

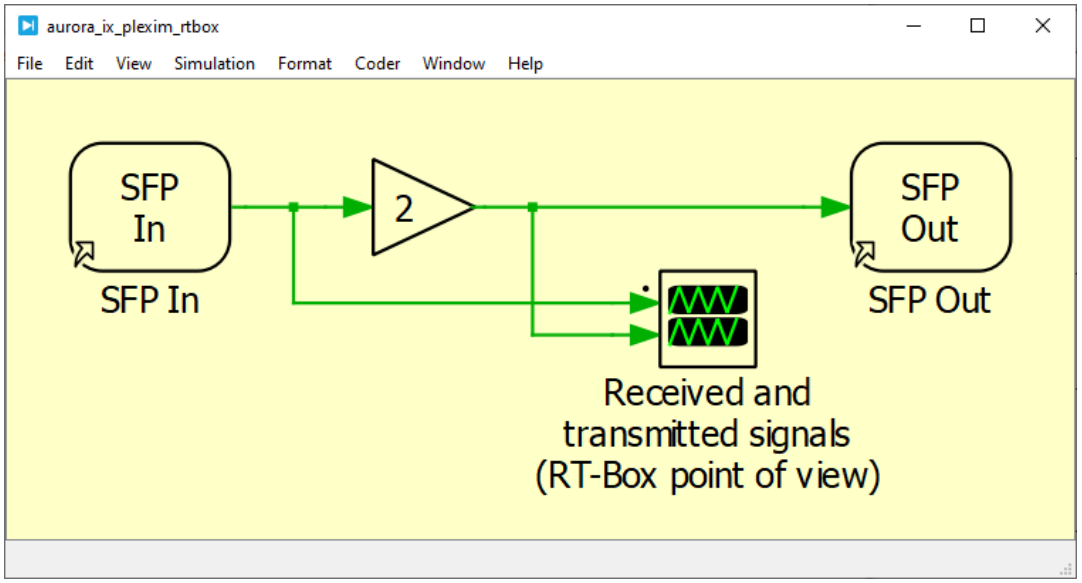
- The **user application**, running in the imperix controller's CPU, provided as a Simulink or PLECS script.
- The **FPGA bitstream**, running in the imperix controller's FPGA, provided as generation scripts. The scripts must be launched with Vivado to create a ready-to-use project.
- The **RT-Box application**, provided as a simple PLECS model.

User application	FPGA bitstream	RT-Box application
aurora_ix_template.slx aurora_ix_template.plecs	aurora_ix_plexim_gen_scripts.zip	**aurora_ix_plexim_rtbox.plecs**

RT-Box application

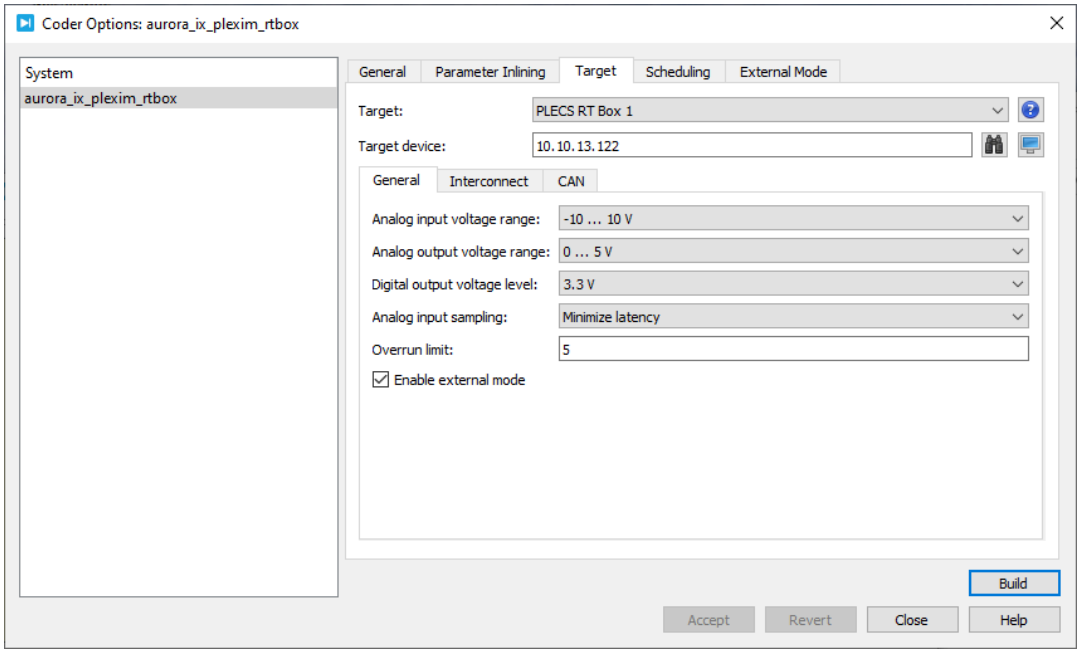
The application running in the RT-Box simply reads the values from the imperix controller, multiplies them by a gain of 2, and sends them back to the imperix controller. A scope is added to enable the real-time control of the setup via the External Mode of the RT-Box.

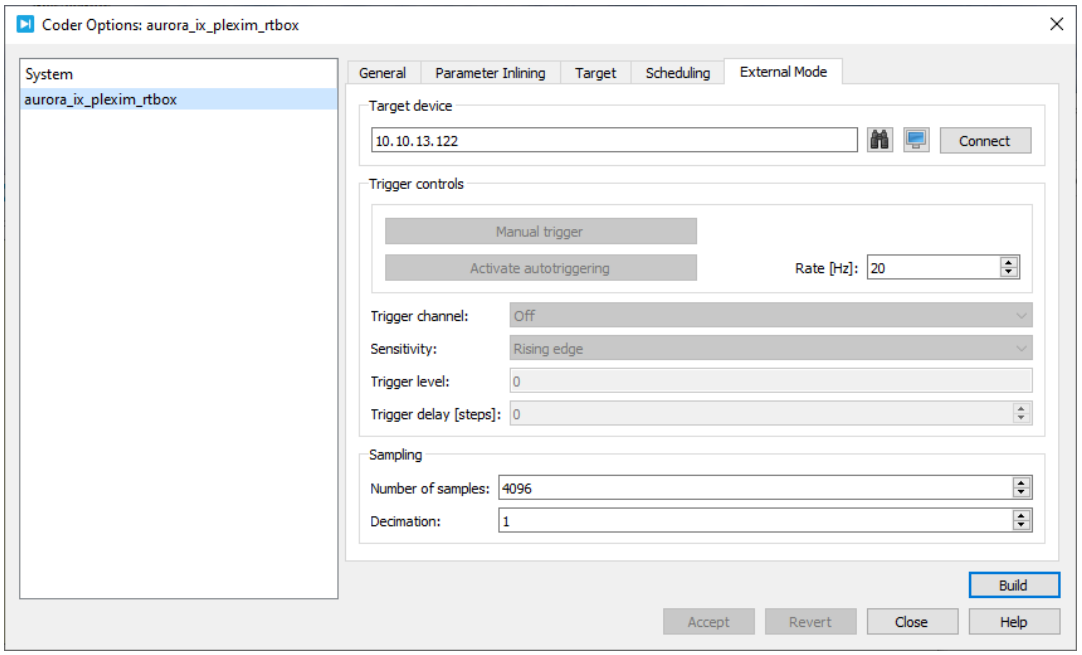
The SFP In block is configured to read 3 signals from the **SFP A** port of the RT-Box. The SFP Out block is also configured on **SFP A**.



To build and load the model on the RT-Box, press Ctrl+Alt+B to open the **Coder Options menu**. In the **Target tab**, configure the target type and IP, and enable the External Mode via the checkbox. Then, navigate to the **External Mode tab** and press Build.

To launch the acquisition using the External Mode of the RT-Box, wait for the code to be built and loaded, then press **Connect** and **Activate autotriggering**.



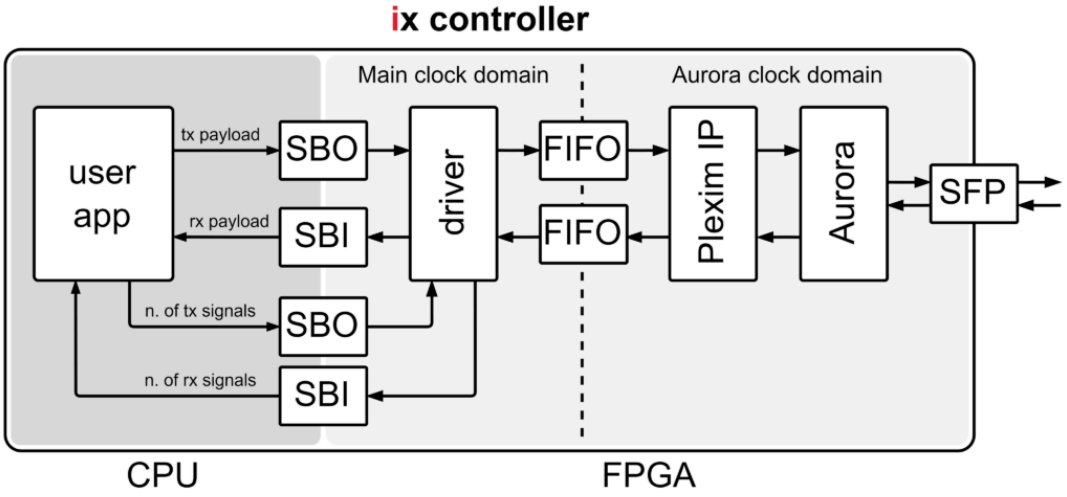


The code should now be running in the RT-Box and the scope acquisition started (showing only zeroes if the imperix controller is not yet configured).

Communication chain

Overview

The overview of the communication chain is presented in the setup overview. However, for Plexim, the logic is slightly different: a proprietary IP is added in the design, between the FIFOs and the Aurora IP. This IP implements an additional layer of encapsulation to comply with the frame structure expected by the RT-Box.



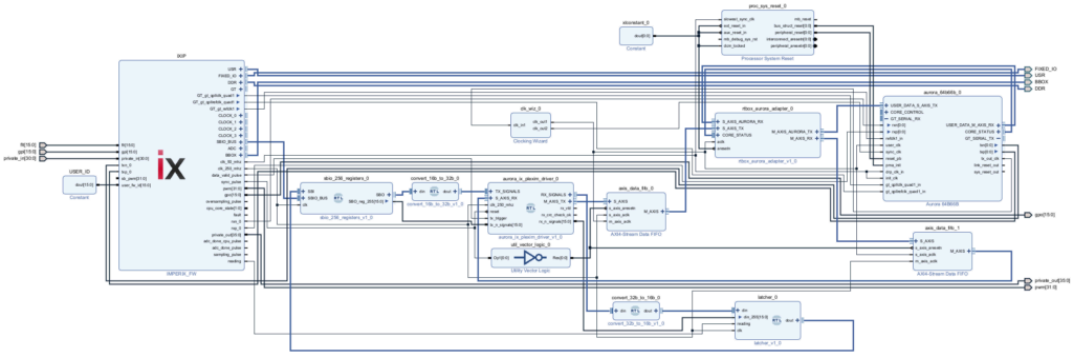
The IP provided by Plexim is named rtbox_aurora_adapter in Vivado.

The main clock domain always runs at 250 MHz, while the frequency of the Aurora clock domain varies with the configuration of the Aurora IP. In this example, the frequency is 97.656 MHz with the configuration presented in the Aurora parameters section.

Vivado project

The Vivado project is provided in the form of generation scripts. As explained in the Generate the bitstream section, the scripts automatically create and open the project illustrated below. The bitstream can be directly generated by simply pressing **Generate Bitstream** in the left navigation bar in the Vivado environment.

As provided, the driver supports the exchange of up to **32 signals** in each direction and the Aurora communication is linked to the **SFP 0 (UP)** port of the imperix controller.



Once generated, the bitstream can be loaded onto the imperix controller using Cockpit. A reboot is required for the bitstream change to take effect.

To increase the number of exchanged signals or change the SFP port used for the Aurora communication, please refer to How to exchange more signals and How to assign a different SFP port.

Modules/IPs description

The Vivado project contains the following VHDL modules and IPs.

Module name	Type	Description
sbio_256_registers	VHDL module	Instantiates and provides access to SBIO bus registers in the FPGA. See Exchanging data between the CPU and the FPGA for more details.
convert_16b_to_32b	VHDL module	Converts the 16-bit words of the SBIO bus back into the 32-bit words of the payload.
sfp_aurora_rtbox_driver	VHDL module	Custom driver provided by imperix to communicate with the RT-Box from Plexim ; mainly acts are a parallel-to-serial transmitter and serial-to-parallel receiver.
convert_32b_to_16b	VHDL module	Converts the 32-bit words received from the RT-Box through the driver into 16-bit words compatible with the SBIO bus.
latcher	VHDL module	Ensures data coherency by preventing the update of the SBI registers while the CPU is reading.
AXI4-Stream Data FIFO	Vivado IP (Xilinx)	Handles the clock domain crossing between the main 250 MHz domain of the imperix firmware and the Aurora clock domain ; buffers the frame in the transmission direction.
rtbox_aurora_adapter	Vivado IP (Plexim GmbH)	Slightly truncates the transmitted and received frames to comply with the RT-Box requirements ; filters out the frames with an invalid CRC ; handles the synchronization mechanism when used (here, not used).
Processor System Reset	Vivado IP (Xilinx)	Handles reset signals to properly initialize the Aurora IP.
Aurora 64B66B	Vivado IP (Xilinx)	Handles the Aurora communication and interfaces with the underlying hardware logic.
Clocking Wizard	Vivado IP (Xilinx)	Handles the clock signals, providing the Aurora domain clock and adding proper buffers.
Utility Vector Logic	Vivado IP (Xilinx)	Converts the active-high reset signal from the `sync_pulse` into an active-low reset signal for the FIFOs.

Constant	Vivado IP (Xilinx)	Provides a constant high signal to annihilate unused active-low reset signals.

Aurora parameters

To establish communication with Plexim devices, the Aurora IP must be configured with the parameters listed below. Any settings not specified here should remain at their default values.

Protocol	Aurora 64B66B	**Flow Control**	None
Line Rate (Gbps)	6.25	**Little Endian Support**	No
Dataflow Mode	Duplex	**CRC**	Yes
Interface	Framing	**DRP Mode**	Disabled

Plexim-specific notes

- Received frames with an invalid checksum are dropped by the Plexim IP. This behavior can be changed manually by double-clicking the rtbox_aurora_adapter Plexim IP in Vivado and unchecking the **Drop packages with invalid checksum** checkbox.
- An additional rx_CRC_check_ok signal is available at the output of the driver. This output is updated simultaneously with rx_vld and indicates if the frame has a valid CRC. If invalid frames are dropped by the Plexim IP (see previous item), the CRC_check_ok is expected to be always high.

Experimental validation

Physical setup

The physical setup is straightforward:

1. Connect both devices to the network, so that they can be configured and monitored from the PC.
2. Connect the imperix controller to the RT-Box with an SFP cable. As provided, this example considers the port **SFP 0 (UP)** on the controller and **SFP A** on the RT-Box.
3. Turn on the two devices.

Software-side setup

To experimentally validate the system:

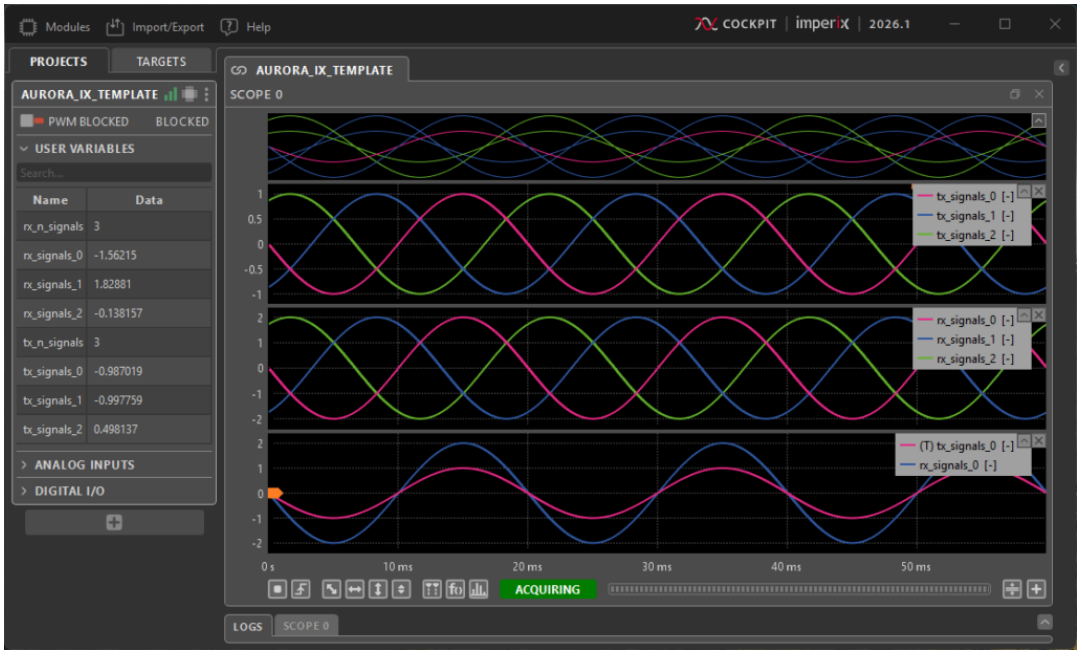
1. Download the three software parts available in the downloads section.
2. Build and load the RT-Box application on the RT-Box.
3. Generate the bitstream for the imperix controller.
4. Load the bitstream on the imperix controller via Cockpit.
5. Build the user application template and launch it on the imperix controller via Cockpit.
6. Use Cockpit to monitor the exchanged signals.

The whole system should now be running.

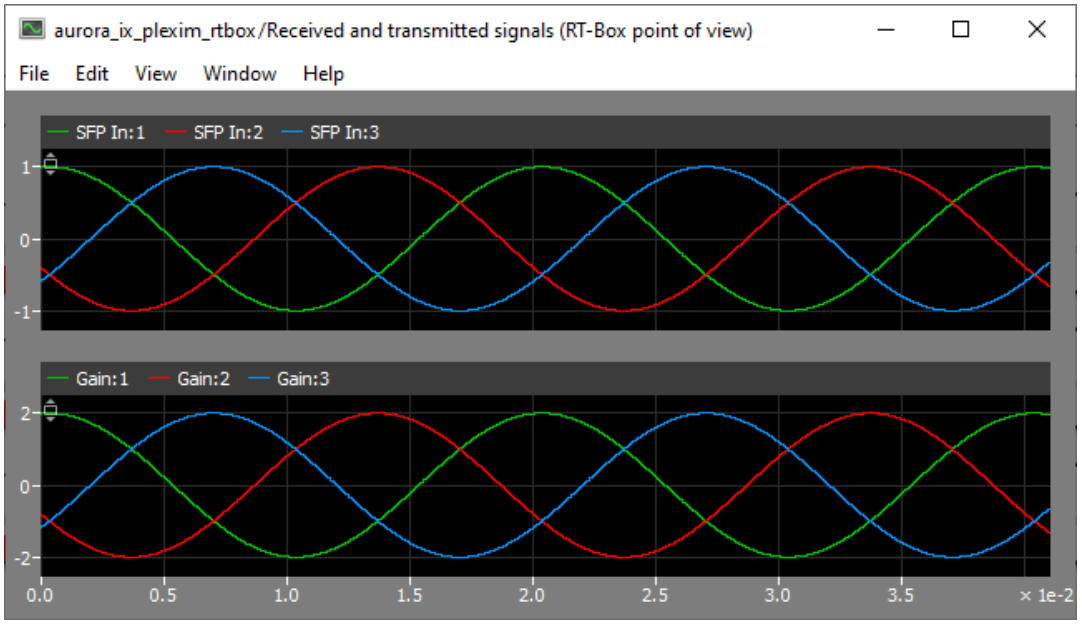
Real-time monitoring

Connect to the imperix controller with Cockpit. Add a scope in the project (from the **Modules tab** in the top bar) and drag-and-drop the variables of interest.

The exchanged signals can now be monitored in real-time in Cockpit. As expected, the amplitude of the transmitted signals is multiplied by two in the RT-Box.



The exchanged signals can also be monitored from the RT-Box, using the External Mode, as described in the RT-Box application section.


aurora_ix_plexim_rtbox/Received and transmitted signals (RT-Box point of view)
File
Edit
View
Window
Help
SFP In:1
SFP In:2
SFP In:3
1
0
-1
Gain:1
Gain:2
Gain:3
2
0
-2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
× 1e-2