

Accessing the USR pins in the FPGA sandbox

PN179 | Posted on April 16, 2026 | Updated on April 17, 2026



François LEDENT

Development Engineer

imperix • [in](#)

Table of Contents

- [Physical location of the USR pins](#)
- [Using the USR pins](#)
- [To go further](#)

Most imperix controllers feature 36 bidirectional 3.3V I/O lines, commonly known as USR pins. Driven directly from the FPGA, these custom-application I/Os are ideal for various uses, including communicating with digital encoders, interfacing with additional ADCs, and connecting to other peripheral devices.

Note on FPGA development for imperix controllers

Customizing the FPGA firmware involves instantiating the [imperix firmware IP](#) within AMD/Xilinx Vivado to edit the surrounding programmable logic, known as the sandbox. For step-by-step instructions on creating the required FPGA sandbox template, refer to the [getting started](#) guide.

Physical location of the USR pins

The following table details the physical locations of USR pins on imperix controllers.

imperix controller	Location of the USR pins	Related datasheet/page
B-Box 4	VHDCI connector B (*)	B-Box 4 datasheet
B-Box RCP 3.0	VHDCI connector B (*)	B-Box RCP datasheet

B-Box Micro	VHDCI connector (*)	B-Box Micro datasheet
B-Board PRO on its Eval-Board	36-pin 2.54 mm header	Carrier Board description page
B-Board PRO	JX2 connector	B-Board PRO datasheet
Programmable inverter (TPI8032)	<i>Not available.</i>	

(*) Accessible via a breakout board such as the [VHDCI breakout board](#).

Using the USR pins

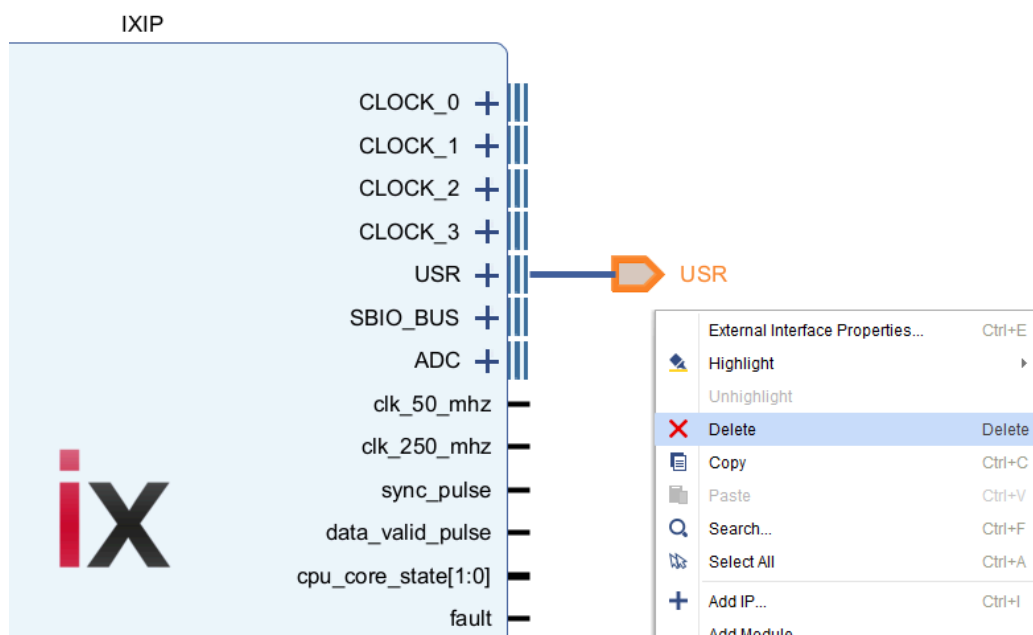
There are two specific cases where the USR pins are reserved by the imperix system and remain unavailable for custom applications:

- The target is a [TPI8032](#).
- A [motor interface](#) is connected to the VHDCI connector.

If neither of these conditions applies, the default USR interface port can be safely deleted to free up the pins.

Delete the existing USR interface port

- Right-click on the existing USR interface.
- Click **Delete**.



Delete the existing USR interface port

Create new interface ports

- Right-click, then **Create Port...**
- Give it a **name**, select a **direction**, and click **OK**.



Create new interface ports

Edit the constraints

The constraints\sandbox_pins.xdc must be edited accordingly. As shown below, we recommend commenting (#) the unused pins to avoid generated unnecessary warning the Vivado. For more information on constraints in Xilinx FPGA please refer to the [using constraints in Vivado Design Suite](#) user guide.

```
L:/constraints/sandbox_pins.xdc
1  set_property BITSTREAM.CONFIG.UNUSEDPIN PULLNONE [current_design]
2
3  ##### =====
4  ##### USR
5  ##### =====
6  set_property -dict {PACKAGE_PIN AE10 IOSTANDARD LVCMOS33} [get_ports {usr_0}]
7  set_property -dict {PACKAGE_PIN AF10 IOSTANDARD LVCMOS33} [get_ports {usr_1}]
8  set_property -dict {PACKAGE_PIN AE12 IOSTANDARD LVCMOS33} [get_ports {usr_2}]
9  #set_property -dict {PACKAGE_PIN AF12 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[3]}]
10 #set_property -dict {PACKAGE_PIN AE13 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[4]}]
11 #set_property -dict {PACKAGE_PIN AF13 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[5]}]
12 #set_property -dict {PACKAGE_PIN AF14 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[6]}]
13 #set_property -dict {PACKAGE_PIN AD14 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[7]}]
14 #set_property -dict {PACKAGE_PIN AD13 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[8]}]
15 #set_property -dict {PACKAGE_PIN AC14 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[9]}]
16 #set_property -dict {PACKAGE_PIN Y16 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[10]}]
17 #set_property -dict {PACKAGE_PIN AD15 IOSTANDARD LVCMOS33} [get_ports {USR_tri_io[11]}]
```

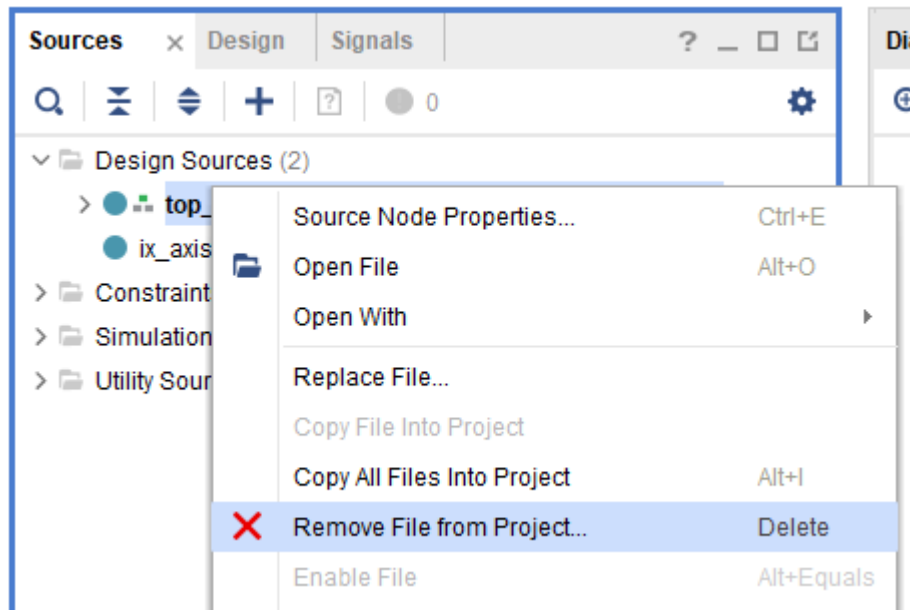
Edit the constraints

Update the block design wrapper

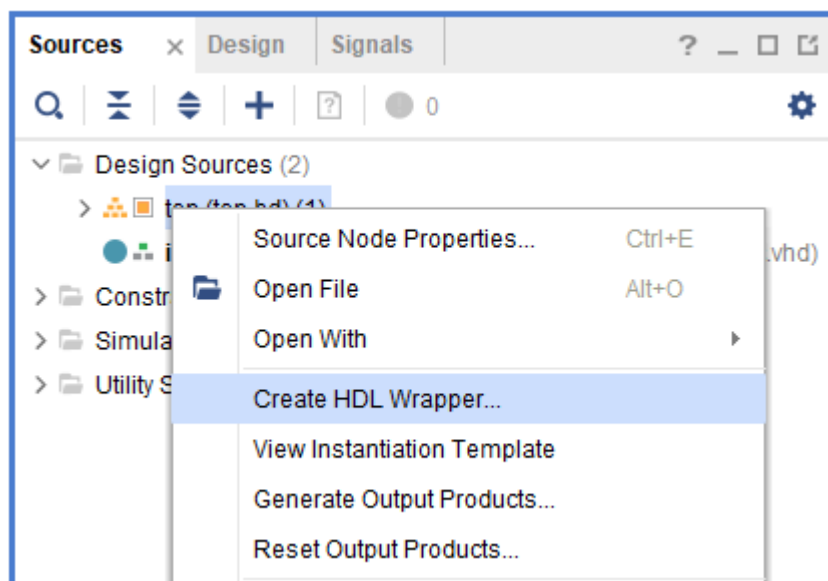
Whenever top level ports are modified, it is required to updated the block design wrapper to reflect the changes. The recommended procedure is the following:

- Remove the existing wrapper by right-clicking on **top_wrapper** and select **Remove File from Project**.
- Check the **Also delete the project file from disk** option, then **OK**.

- In the **Design sources** of the **Sources** panel on the left, right-click on **top (top.bd)** and select **Create HDL wrapper**.
- Check **Copy generated wrapper to allow user edits**. Press **OK**.



Delete the outdated HDL wrapper



Re-create an up-to-date HDL wrapper

To go further

The following application notes illustrate practical uses of the 3.3V USR pins for communicating with external peripherals:

- [TN130](#): SPI communication with an **LTC2314** ADC chip
- [TN149](#): decoding Delta-Sigma modulation from an **AMC1035** ADC