Getting started with the TPI 8032

PN190 | Posted on July 18, 2023 | Updated on June 23, 2025

Shu WANG
Development Engineer
imperix • in

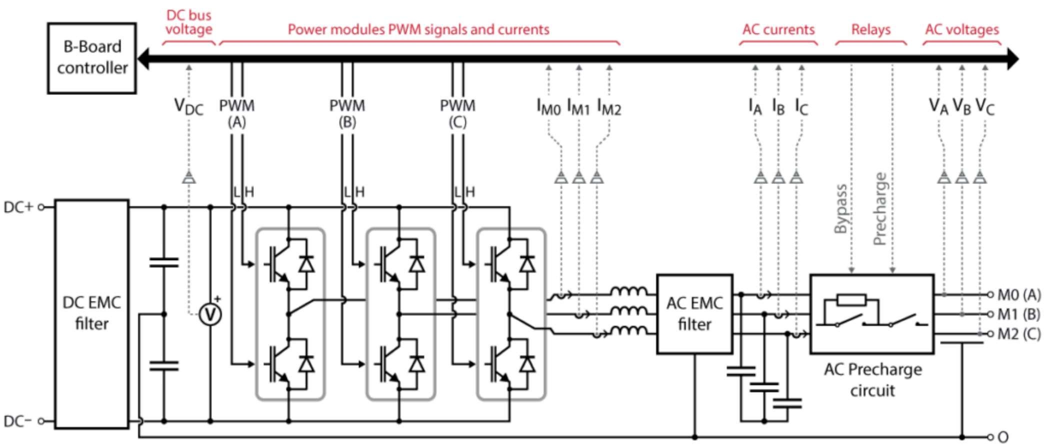
Table of Contents

This page explains how to get started with the TPI 8032 all-in-one programmable inverter. It provides an overview of the hardware architecture and detailed instructions to program the device.

An open-loop voltage-source inverter example with passive loads is provided as a demo example for the TPI 8032, with a step-by-step operation procedure for users to become familiar with the product.

What is the TPI 8032?

The TPI 8032 is an all-in-one product that integrates all the necessary components for running a two-level three-phase inverter, covering from the digital controller to the power stage. The typical applications include grid-tied inverters, back-to-back converters, etc. Thanks to the imperix RealSync proprietary technology, multiple TPIs can work together or with other imperix controllers in various setups, covering almost any converter topology.



System overview of the TPI 8032

The power stage of the TPI 8032 is a three-phase inverter made of three half-bridge power modules based on Silicon Carbide (SiC) MOSFETs with one inductor at each phase output. Two EMC filters are placed on both sides of the inverter to reduce the electromagnetic interferences (EMI) generated by the switching devices (refer to the datasheet for detailed information on the EMC filters).

The control stage of the TPI 8032 is built with a B-Board PRO and a carrier board that collects the sensor inputs and distributes the PWM signals to the power modules.

The TPI 8032 has numerous embedded current/voltage sensors for various uses. The currents flowing through the power modules $I_{m,0,1,2}$ and the DC bus voltage V_{dc} are measured on the modules, with over-current and over-voltage protection that protects the device against possible damage. Besides, it also has three-phase current and voltage sensors after the AC EMC filter providing AC-side current $I_{a,b,c}$ and voltage $V_{a,b,c}$ measurements.

In the case of grid-connected operation, the DC bus voltage must be higher than the rectified AC voltage. For this purpose, TPI8032 has an integrated AC precharge circuit that enables automated and secured precharge of the DC bus from the grid.

The figure below shows the front and back panels of the TPI 8032. The internal power circuit is accessible from the back panel's DC/AC power connector. Besides this, the TPI 8032 also has various external ports for control and communication.

In addition to all the internal embedded sensors, the TPI 8032 also has 4 analog inputs for external sensors accessible through the RJ45 connectors on the back panel. These inputs are compatible with the imperix DIN 800V voltage sensor and DIN 50A current sensor.

As for communication, the TPI has SFP interconnect ports that enable ultra-fast communication between imperix controllers with the imperix RealSync proprietary technology. It also supports other widely-used communication protocols, such as CAN and UART, through the two field bus connectors. Please refer to the datasheet for detailed information on all the port functions and parameters.

FRONT PANEL

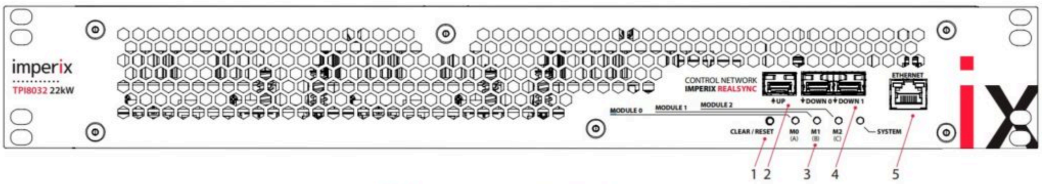


Fig. 1. Front panel view of the TPI8032

1) Clear/reset push button
2) SFP interconnect – UP link
3) System and power modules status LEDs
4) SFP interconnect – DOWN links
5) Gigabit Ethernet port (RJ45)

BACK PANEL

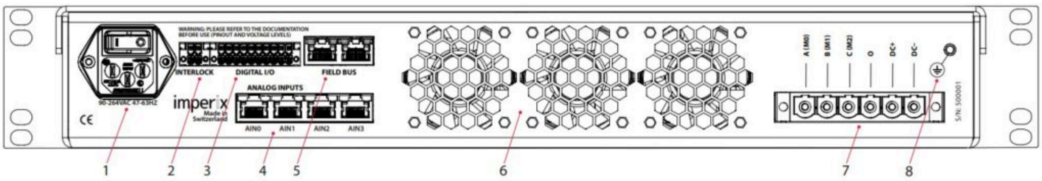


Fig. 2. Back panel view of the TPI8032.

1) AC mains switch and socket (IEC C14, 90-264V, 47-63 Hz)
2) Electrical interlock connector (IN/OUT)
3) GPI/GPO port
4) Analog inputs (RJ45, ±5V)
5) Field bus ports (RJ45)
6) Fan outlets
7) DC/AC power connector
8) Earth (ground) terminal

Front and back views of the TPI 8032

How to program the TPI 8032?

The TPI 8032 can be programmed either in C++ or with Automated Code Generation (ACG) tools from Simulink and PLECS. The programming procedure is identical to the B-Box RCP and the B-Board PRO.

This section will focus on the unique features of programming the TPI and briefly introduce the features common to other imperix controllers. If you are unfamiliar with programming the imperix controllers, please refer to the Programming and operating imperix controllers.

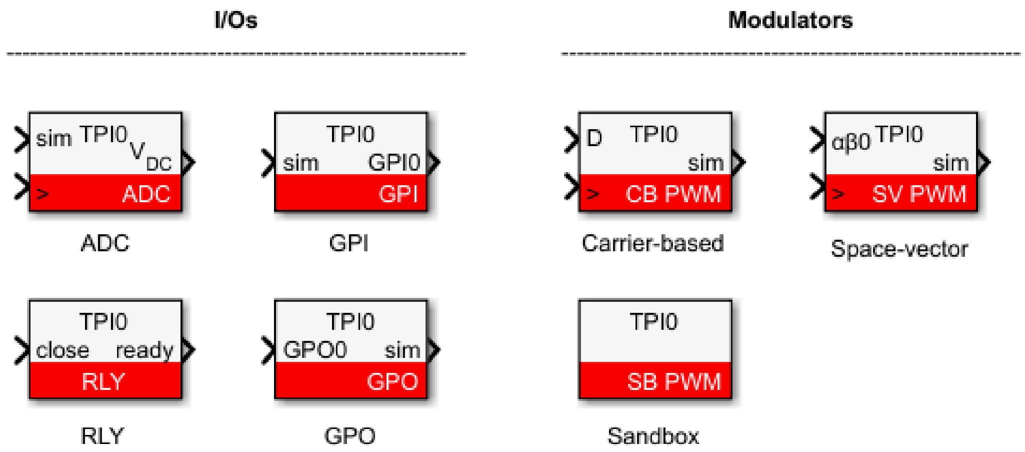
The TPI shows better ease of use regarding wiring and programming than the combination of B-Box RCP and power modules. The hardware circuits have been wired inside the chassis and are easily accessible through the dedicated helper blocks in Simulink and PLECS, which can be found in the sublibrary named *TPI* of the imperix blockset.

ACG SDK 2024.2 or a later version is required to use the helper blocks for the TPI. The latest version of ACG SDK can be downloaded at imperix.com/downloads/.

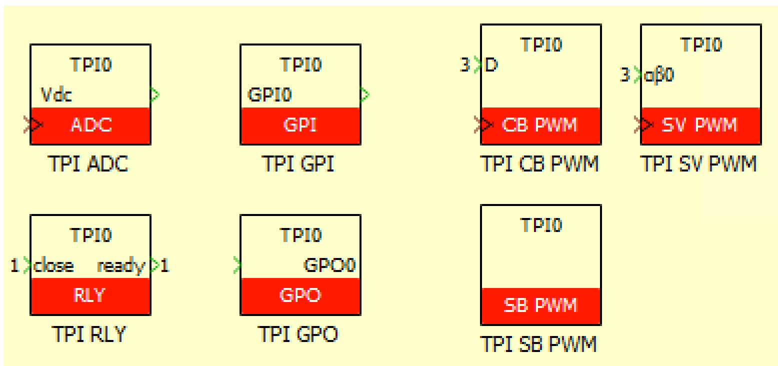
The helper blocks are used similarly to any other imperix blocks but hide the hardware layer and provide only information relevant to the control, such as the dedicated ADC and PWM channels, fixed sensor sensitivities, sensor offsets in

Volt/Amp, etc. More details can be found on the following software documentation pages:

- TPI ADC
- TPI RLY
- TPI GPI
- TPI GPO
- TPI CB-PWM
- TPI SV-PWM



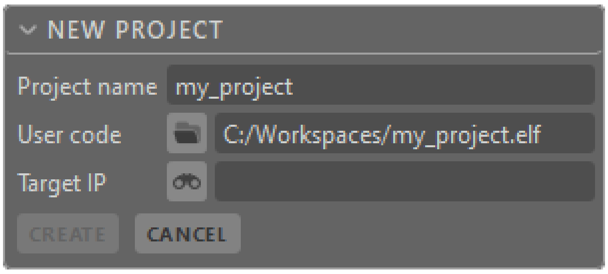
Helper blocks for the TPI in Simulink



Helper blocks for the TPI in PLECS

It's important to mention that the helper blocks are not mandatory for programming the TPI. The hardware resources on the TPI are still accessible using the generic imperix blocks such as ADC and CB-PWM as long as the configurations are made correctly as mentioned in the datasheet. However, it's strongly recommended to use the helper blocks since they are much simpler to use and less prone to configuration mistakes. Besides, the TPI RLY block is necessary for the AC precharge circuit because there is no functionally equivalent block in the generic imperix blockset.

The C++ code or model must first be built before it can be deployed onto the controller. Once the user program is built, regardless of the workflow, Cockpit will automatically launch and create a new project with a pre-filled project name and executable file path. To load the user code (.elf file) to the target and start the code execution, enter the IP address of the TPI and click *Create*.



Cockpit creates a new project using the user code built from Simulink, PLECS, or imperix IDE.

Cockpit is a powerful monitoring software designed for imperix controllers. If you are unfamiliar with Cockpit, please refer to the Cockpit user manual.

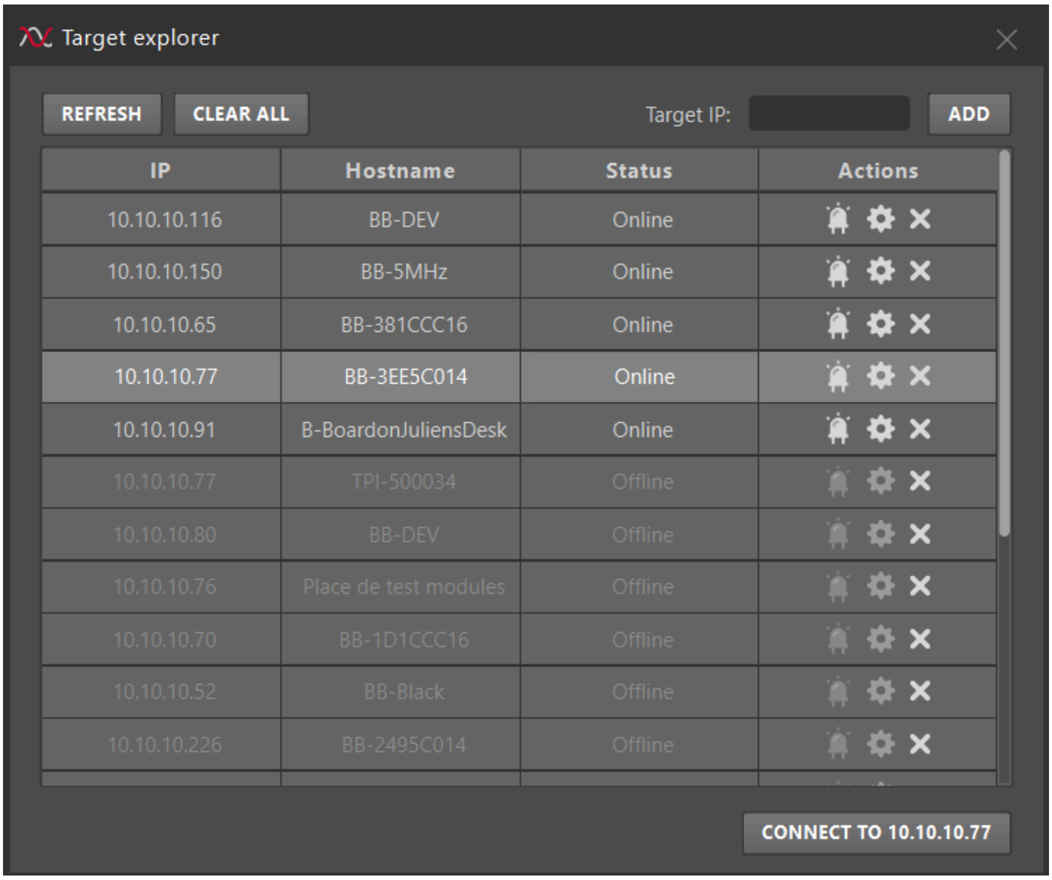
The ethernet connection to the host PC and IP address configuration can be done similarly to the other imperix controllers using the **default static IP address** 192.168.222.22. However, it might be tricky to identify which device is the target to be programmed, especially when multiple controller devices are in the local network. With the help of the *LED blinking* function of Cockpit, it's easy to identify the correct target. The detailed procedure is listed in the pane below.

Identify the correct target with the LED blicking function

1. Make sure the TPI device to be programmed is powered ON.
2. In Cockpit, open the *Target explorer* by clicking on the button next to *Target IP*.

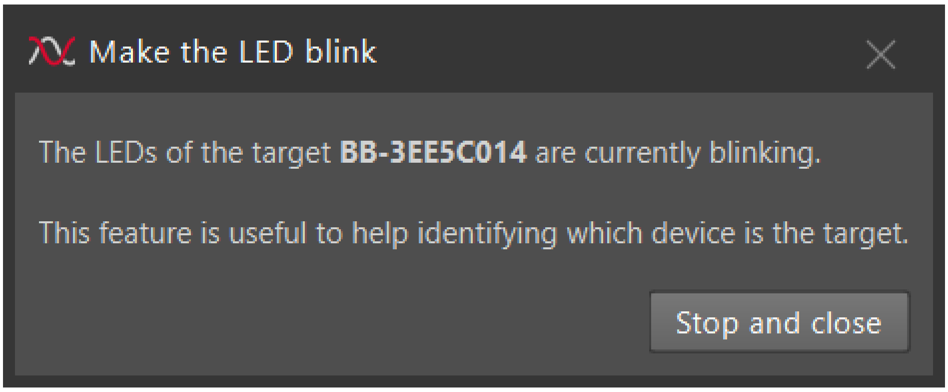
New project window in Cockpit

3. The *Target explorer* window displays all the imperix controller devices in the local network. If the TPI device is not listed, click on the *REFRESH* button to search again.



The TPI device is listed in the Target explorer

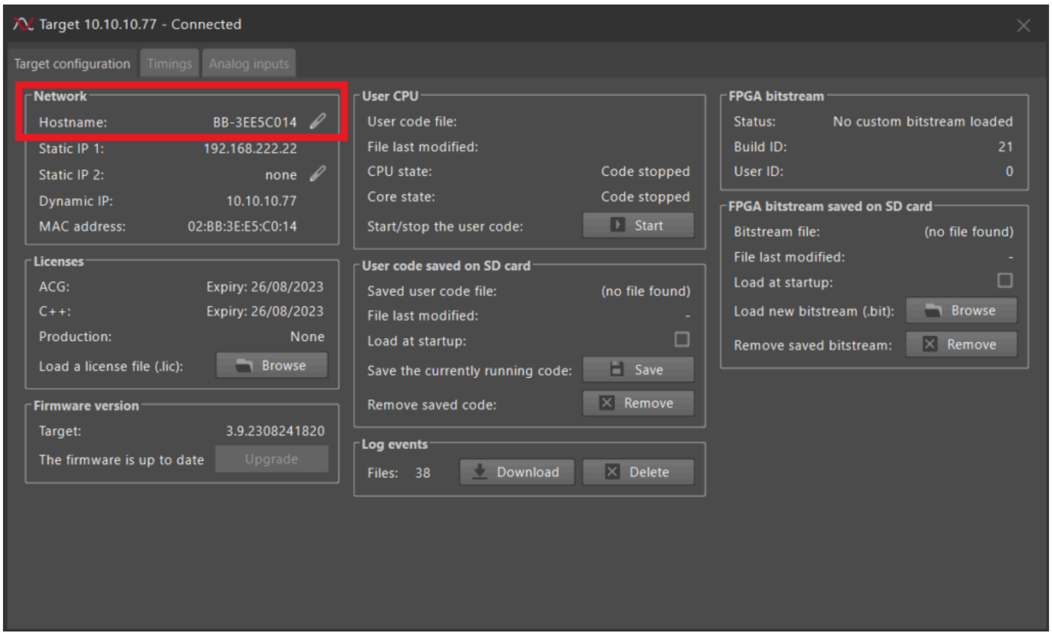
4. To make sure the selected device is the correct target, click on the LED button (the first button on the left in the *Actions* column) to make the LED of the target blink. A window will pop up in Cockpit, and the SYSTEM LED on the front panel will blink green.



Pop-up window in the Cockpit

SYSTEM LED blink green on the front panel

5. Once the target is identified, click on *Stop and close* to stop the LED blinking.
6. For ease of the next use, the hostname of the TPI can be renamed in the *Target config* window (clicking the middle button in the *Actions* column). Note that the TPI has to be rebooted for the new hostname to take effect.



Target config window in Cockpit

Color codes of the status LEDs

The front LEDs also indicate the status of the TPI while operating.

Color	Blinking	M0 (A) LED	M1(B) LED	M2 (C) LED	M0-2 LEDs together	SYSTEM LED
None	*None*					*No code running*
Red	*None*					*Core state FAULT*
	Slow (2Hz)	*Over-current*	*Over-current*	*Over-current*	*DC bus overvoltage*	
	Fast (10 Hz)	*Desaturation*	*Desaturation*	*Desaturation*	*Interlock*	
Orange	*None*					*Core state BLOCKED*
	Slow (2Hz)	*Over-temperature or fan error (module)*	*Over-temperature or fan error (module)*	*Over-temperature or fan error (module)*	*Over-temperature or fan error (enclosure)*	
	Intermediate (4Hz)					*Synchronizing*
	Fast (10 Hz)	*Bad power supply*	*Bad power supply*	*Bad power supply*		
Green	*None*	*No fault*	*No fault*	*No fault*		*Core state OPERATING*
	Slow (2Hz)					*Device ID*

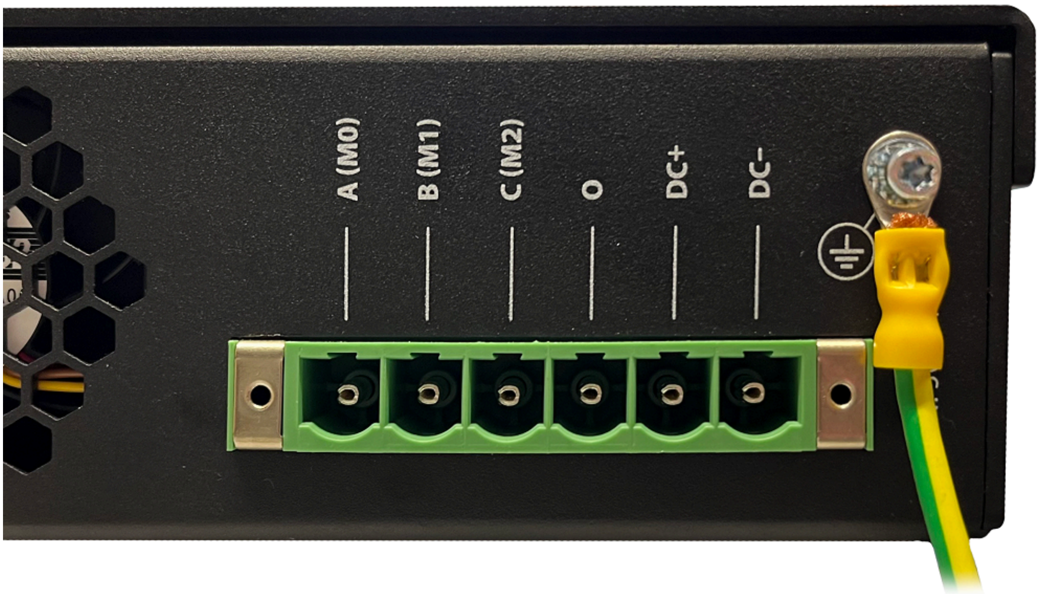
Table 12. Color codes of the status LEDs.

Instructions for safe use

Before starting any experiment, it is essential to take all necessary precautions to operate the system safely. Please observe the safety measures detailed in the panes below.

Protective Earth (PE) connection

The PE terminal must be connected to the earth to ensure the electrical safety of the system and optimal EMC performance. Additionally, the earthing cable should be as short as possible and connected to the common ground with other devices in the setup (if any) to avoid ground loops.



PE connection on the back panel

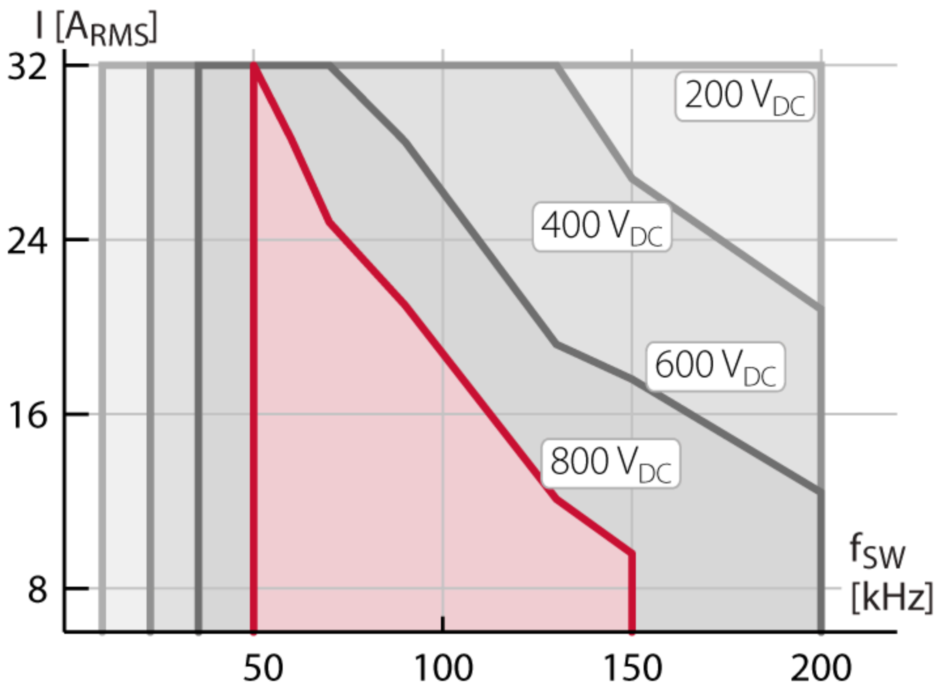
Safe Operating Area (SOA)

The TPI 8032 has a built-in protection circuit against over-current, over-voltage, and over-temperature on the power modules. The protection thresholds are dynamically configured based on the switching frequency and DC bus voltage, considering the derating of the converter and EMC filters.

In general, there are mainly two factors that cause the derating:

- With the switching frequency or the DC voltage increasing, the average RMS current that the power modules can handle has to decrease.
- With the switching frequency dropping below 50kHz, the maximum DC voltage has to decrease to avoid saturating the common mode inductors on the EMC filters.

The following figure shows the Safe Operating Area (SOA) of the TPI, considering the current derating of the power modules and the voltage derating of the EMC filters.



Safe Operating Area

The B-Board controller enforces the SOA according to the converter's switching frequency and the measured DC bus voltage. Once the SOA is exceeded, the TPI will enter the *FAULT* state and stop all the PWM outputs.

The switching frequency is measured at the FPGA level by counting the time between each two **rising edges** of the PWM signals. Depending on the shape of the PWM carrier, the measured switching frequency might be slightly different from the carrier frequency.
In addition, a common reason that may trigger the SOA frequency protection is a saturated PWM modulator, which can significantly increase the time between two PWM rising edges.

Demo example for the TPI 8032

This page provides code examples and step-by-step instructions for both Simulink and PLECS. If you are not familiar with the code generation feature of Simulink or PLECS, please refer to the Getting started with ACG SDK on Simulink or Getting started with ACG SDK on PLECS.

ACG SDK 2024.2 or a later version are required to run the demo example. To update the ACG SDK and Cockpit, please go to imperix.com/downloads/.

The required equipment for running the first test is listed below:

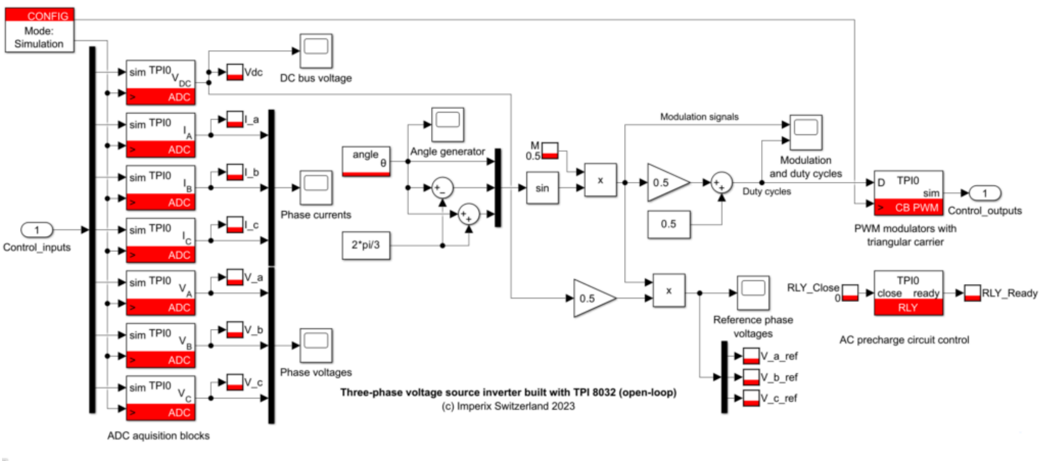
- 1x TPI 8032 three-phase inverter
- ACG SDK toolbox for automated generation of the controller code from Simulink or PLECS
- 1x DC power supply (50V to 800V)
- 3x power resistors (5Ω to 100Ω). The current rating depends on the DC voltage.
- All the necessary cables

An emergency stop button can be connected to the TPI 8032 through its INTERLOCK input. This button is optional and not included in the product. If not used, it must be replaced by the included dummy plug to avoid reporting a fault (please refer to the datasheet for more details).

The Simulink and PLECS model provided below implements a three-phase voltage source inverter with passive loads. The phase voltage is controlled in an open-loop manner with a tunable modulation index M.

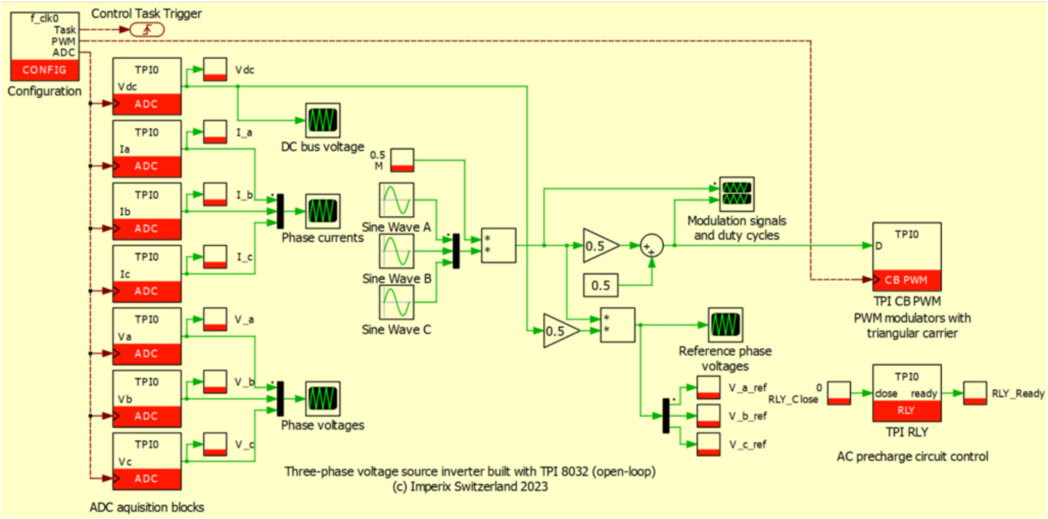
Simulink model

Download **TPI_demo_example_simulink**



PLECS model

Download **TPI_demo_example_plecs**



Passive components sizing

The TPI 8032 has a 0.95mH inductor at each phase-leg output. Considering the operating conditions of the inverter, a DC bus voltage of 200 V and 3x 8.5Ω resistive loads guarantee a maximum output current of 8.3 A with a modulation index of 1. This is confirmed by the formula for the load current from the AN002 :

$$I_{RMS}(M=1) = \frac{\sqrt{2}}{4}\frac{V_{dc}}{\sqrt{R^2+(2\pi f L)^2}} = 8.3\,\mathrm{A}$$

	Chosen values	Suggested values
DC Bus	200V	50-800V
Inductors	0.95mH	Fixed value
Resistors	8.5Ω	5-100 Ω

To ensure safe operation during this demo test, we suggest selecting V_{dc} and R such that I_{RMS} is always smaller than the current ratings of the TPI and the resistors even with modulation index $M = 1$.

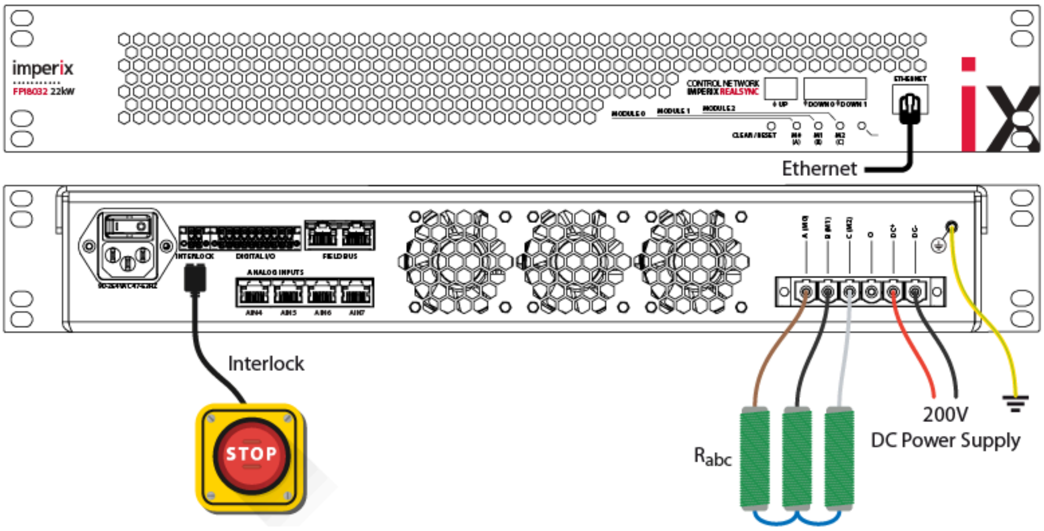
Wiring of the TPI 8032

Starting from the front panel of the TPI8032, connect the ethernet port to the computer's local network or directly to the PC. Then, on the back panel of the TPI8032,

1. Connect the Protective Earth (PE) terminal to the ground.
2. Connect the *DC-* and *DC+* terminals to the DC power supply. Double-check that the DC polarity is correct.

3. Connect each AC terminal A, B, and *C* to one of the resistors, and then connect three resistors in a star configuration.

The figure below shows the complete wiring of the TPI 8032.



Wiring of the TPI 8032 for the demo example with passive loads

Building the model with ACG tools

… in Simulink

1. Open the Simulink model. Open the *Controller* subsystem and set the mode to *Automated Code Generation* in the *CONFIG* block.
2. Build the model (Ctrl + B). It will automatically launch *Cockpit*.

… in PLECS

1. Open the PLECS model. Check the *CLOCK_0* frequency is set to *f_clk0* in the *CONFIG* block. The definition of the variable *f_clk0* can be found in *Simulation -> Simulation parameters (Ctrl + E) -> Initialization*.
2. Open the *Coder Options* menu by selecting *Coder -> Coder options… (Ctrl + Alt + B)*. In the *General* tab, set the *Discretization step size* to *1/f_clk0*. This ensures that the main interrupt rate is set to *1/f_clk0*.
3. Click on the *Build* button to build the model. It will automatically launch *Cockpit*.

Setting up the workspace in Cockpit

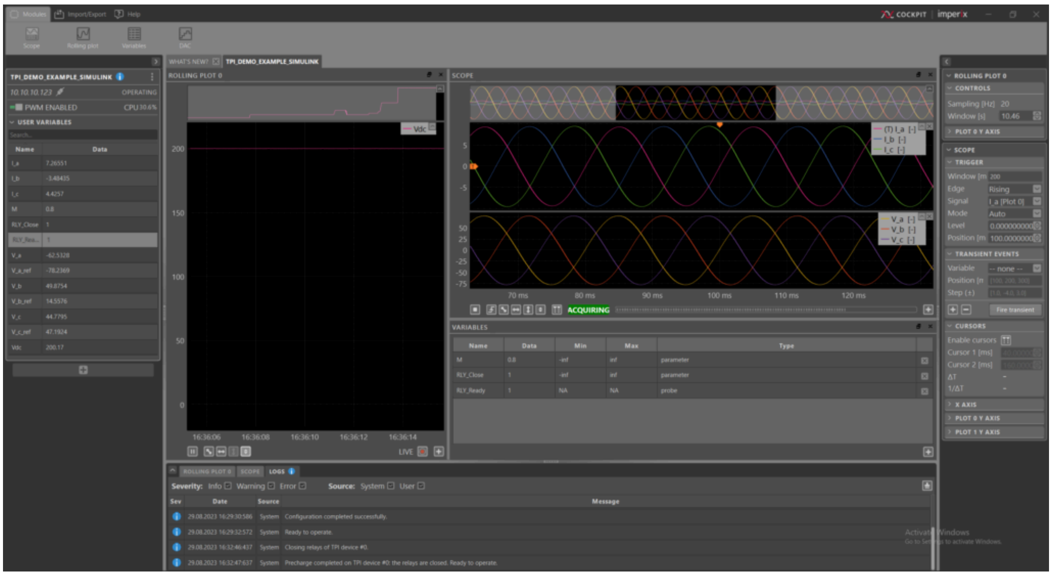
1. Set the *Target IP* in Cockpit and click on *Create* to generate a new project. Use the *LED blinking* function to identify the target device if needed.

2. Add a new **rolling plot** module and drag-&-drop the *Vdc* variable to monitor the DC bus voltage. The rolling plot allows for monitoring the DC bus voltage over a long period.
3. Add a new **scope** module and drag-&-drop the *I_a, I_b, and I_c* variables. The scope can display every sample made available to the control.
4. Add a new subplot to the **scope** module by clicking the '**+**' at the bottom right and drag-&-drop the *V_a, V_b, and V_c* variables.
5. Add a new **variables** module and drag-&-drop the *RLY_Close, RLY_Ready, M* variables. This way, the main parameters are easy to find.

Step-by-step test procedure

1. Ensure the relays are open by checking that the *RLY_Close* variable is set to '0'.
2. Turn on the laboratory DC source and gradually increase the DC bus voltage from 0 to 200V. Check with Cockpit that *Vdc* matches the voltage of the source.
3. Close the relays by setting the *RLY_Close* variable to '1'. Check that the *RLY_Close* goes to '1', which indicates the relays are closed and ready to operate.
4. Check that *M* = 0.5. Enable the PWM outputs from Cockpit and check the peak value of *V_a, V_b, V_c* by $V_{a,b,c,peak} = \frac{M}{2}V_{dc}$. Check the peak value of *I_a, I_b, I_c* by $I_{a,b,c,peak} = \frac{V_{a,b,c,peak}}{R}$.
5. Set *M* to 0.8 and check that *V_a, V_b, V_c,* and *I_a, I_b, I_c* follow the new reference.
6. At the end of the experiment, disable the PWM outputs first, and then open the relays by setting the *RLY_Close* variable to '0'.
7. Reduce the output of the DC source to 0 V. Double-check in Cockpit that the DC bus is fully discharged.

The screenshot below shows how the workspace could look in the end, while running the test procedure.



Running the open-loop test using Cockpit

To go further…

The TPI 8032 is particularly suited for AC microgrid applications. The integrated AC precharge circuit simplifies the grid connection procedure. Active Front End is a basic example showing the principle of grid-connected operations and the precautions of the precharge circuit. **It is recommended to read this page first before running the TPI with the grid.**

Furthermore, the TPI 8032 can be easily programmed to fit any purpose. More examples of TPI in microgrid applications are available in the knowledge base, such as:

- Grid-Following Inverter
- Grid-Forming Inverter