Programming and operating imperix controllers

PN138 | Posted on September 20, 2022 | Updated on July 24, 2025

Julien ORSINGER
Power Applications Specialist
imperix

Table of Contents

This page helps new users get started with imperix power electronic controllers. In particular, it explains how to deploy a user code onto these controllers, and how they can be operated and monitored during run-time, using the imperix Cockpit software. Most of this content also applies to the TPI8032 programmable inverter.

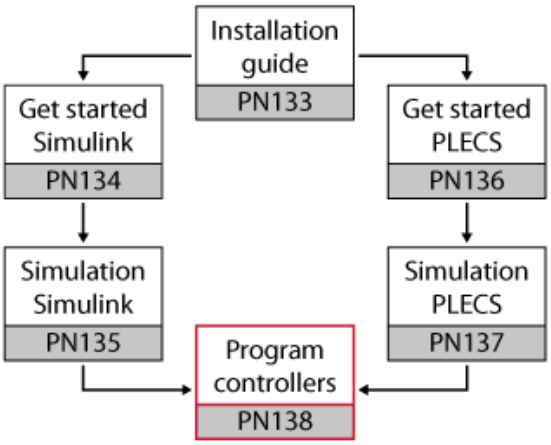
For a complete introduction to imperix software development kits, it is recommended to read the following articles beforehand:

ACG SDK:
Installation guide for the ACG SDK

- **Simulink:**
  Getting started with Simulink
  Simulation essentials with Simulink

- **PLECS:**
  Getting started with PLECS
  Simulation essentials with PLECS

CPP SDK:
Installation and utilization of CPP SDK



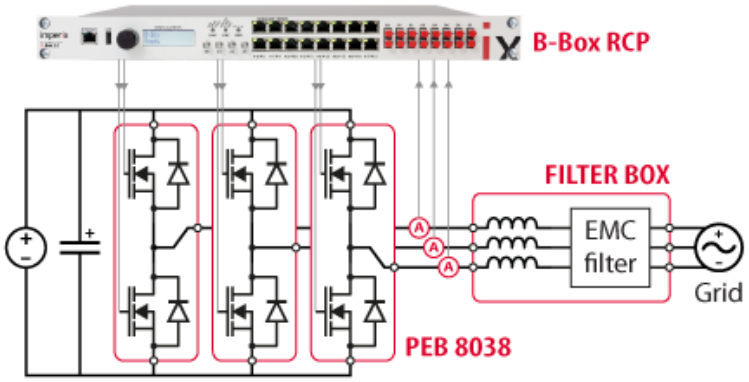
Getting started with imperix ACG SDK on Simulink [Part 3]

Which programmable controller should I use?

The B-Box RCP and B-Box Micro are two fully programmable digital controllers, specifically designed to facilitate the experimental validation of power converter control techniques in a laboratory environment. The former offers superior I/O capabilities, while the latter is rather dedicated to teaching applications.

The B-Board PRO is a small control board, which is designed to be directly integrated within custom-designed power converters. It happens to be also used inside the B-Box RCP and Micro, as well as inside the programmable inverter.

For all controllers, the typical use cases include grid-tied inverters, motor drive inverters, multi-level converters, or DC/DC converters. Combined with imperix power modules, an experimental setup of almost any converter topology can be built easily and rapidly.

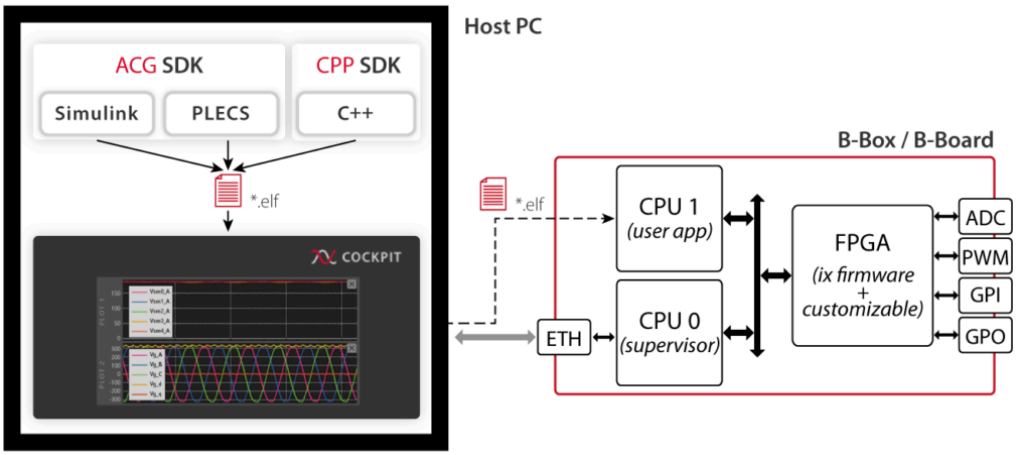


Typical use case of the B-Box RCP
Control of a grid-tied voltage source inverter

All imperix controllers can be programmed the same way, from a PC by writing C code (CPP SDK) or by drawing a block diagram in Simulink or PLECS (ACG SDK). The code is deployed onto the controller with the Cockpit software, using an Ethernet link between the PC and the controller. Cockpit also provides run-time interaction and monitoring from the host PC.

B-Box RCP | B-Board PRO | B-Box Micro

Imperix controllers are built on AMX/Xilinx Zynq 7030 SoC, featuring a dual-core ARM processor (CPU) mated with a Kintex-based programmable logic (FPGA). The control task (user code) runs on the CPU, whereas the low-level and time-critical tasks (such as PWM modulators or safety logic) run in the FPGA.



Main specifications

	B-Box RCP *(datasheet)*	**B-Box Micro** *(datasheet)*	**B-Board PRO** *(datasheet)*
Processor (AMD/Xilinx Zynq)	2x ARM 1GHz	2x ARM 1GHz	2x ARM 1GHz

	B-Box RCP *(datasheet)*	**B-Box Micro** *(datasheet)*	**B-Board PRO** *(datasheet)*
FPGA (AMD/Xilinx Zynq)	Kintex 7 125K	Kintex 7 125K	Kintex 7 125K
Analog inputs	16x 16bits @500ksps	8x 16bits @2Msps	8x 16bits @2Msps
PWM outputs	16x optical 32x electrical (3.3V)	8x optical	16x electrical (1.8V) 16x electrical (3.3V)
Communication	1x Ethernet 1 Gbps 3x SFP+ 5Gbps (RealSync) 1x CAN (RJ45)	1x Ethernet 1Gbps	1x Ethernet 1Gbps (carrier board) 3x SFP+ 5Gbps 1x CAN (carrier board)
C++ programming	Yes	Yes	Yes
Simulink/PLECS programming	Yes	Yes	Yes
Flexible analog front-end	Yes	No	No
Hardware-level protections	Yes	No	No

How to program the chosen controller?

Once selected, programming the controller requires software tools contained in the CPP SDK or ACG SDK packages. The following pages explain how to install and use the SDKs to develop power converter control algorithms. Many code examples are also available from this knowledge base.

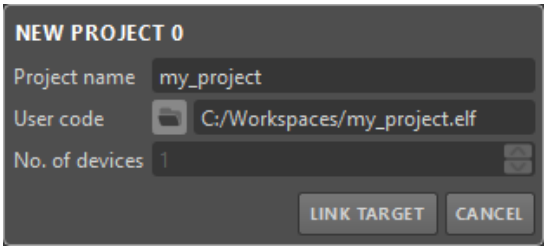
- ACG SDK (programming from Simulink or PLECS): Installation guide for imperix ACG SDK
- CPP SDK (programming with C code): Installation and use of the CPP SDK

Licensing policy
ACG SDK and CPP SDK contain a paid license that gives unlimited access to the controllers. All software licenses are target-locked (i.e. usable on multiple computers) and lifetime (no renewal fees, free software updates). In multi-controller configurations, only the master unit must be licensed (i.e. the one running the user-defined CPU program). For more info, please visit https://imperix.com/software/.

Once developed, the code or model must first be built before it can be deployed onto the controller. This is done using the following procedure:

Simulink	PLECS	C++
Make sure that “Automated Code Generation” is selected in the Configuration block of the model, and click the “Build” button (Ctrl+B).	Open the “Coder > Coder option” window (Ctrl+alt+B), make sure that the correct subsystem is highlighted in the left column, and click the “Build” button.	In imperix IDE, make sure the current project is selected in the project explorer and click on the “run” icon.

Regardless of the workflow used, the build procedure is done so that everything takes place automatically. This comprises code generation, compilation, and upload of the user code onto the controller. At the end of a successful build, Cockpit automatically launches and creates a new project with a pre-filled project name and path of the executable file. To load the user code (.elf file) to the target and start the code execution, click on “Link target” and choose your device using the Cockpit left bar.

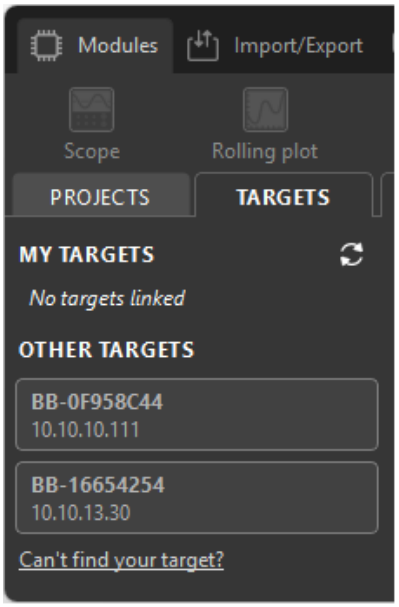


Cockpit creates a new project using the user code built from Simulink, PLECS, or imperix IDE.

Upon clicking “Link target” Cockpit displays a list of potential targets to upload the user code to

In subsequent launches, Cockpit will restore the project configuration, automatically connect to the target, and start the code. If another project is built from a different path – i.e. the user code (.elf) path differs – a new project will be automatically created on Cockpit. If Cockpit is not yet running, it will be automatically started.
For more info, please refer to the Cockpit online documentation.

How to connect the controller to the host computer?



The communication between the controller (referred to as the “target”) and Cockpit on the host computer is established via the Ethernet port, which is located on:

- the front of the B-Box RCP
- the front of the TPI
- the back of the B-Box Micro
- on the B-Board PRO carrier board

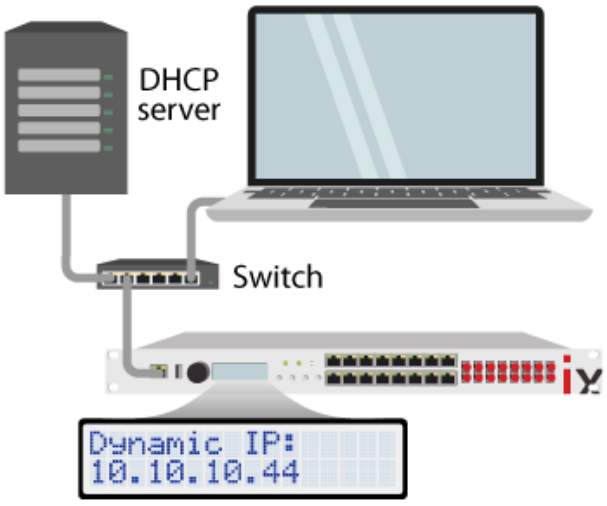
The host computer connects to the target using an IP address (e.g. `10.10.10.111`). Once the connection is established, the target will appear in the Cockpit target list, as shown on the left.

The target can either be connected to an existing Ethernet local network, or directly to the Ethernet port of a computer. These two scenarios are presented below in more detail.

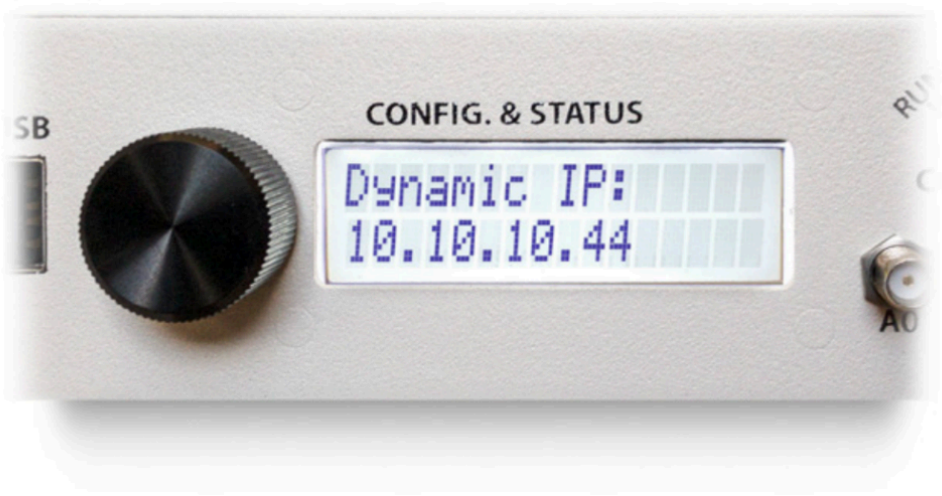
When multiple devices are interconnected in a master-slaves configuration, using SFP optical fiber, only the top master Ethernet port needs to be connected.

The target is connected to an existing Ethernet local network

If the target is connected to an existing network, it will usually receive a dynamic IP address automatically from the DHCP server. Most of the time, Cockpit should be able to automatically find the target on the network and display it in the target list. If the target is not found, its IP address can be manually added by clicking the *Can’t find your target?* link.



On a B-Box RCP, this address can be found by simply turning the push button of the front panel at the end of the startup sequence. With a TPI, B-Box Micro, or B-Board PRO, the dynamic IP address can be found using a device browser (e.g. Bonjour browser) and searching for a device with a hostname starting with “BB-” or “TPI-“.



Display of the IP address on the B-Box screen

If the B-Box shows an IP in the **`169.254.x.x`** range, it means it did not receive any IP address from the DHCP server. This is a self-assigned address, automatically allocated when no DHCP response is received.

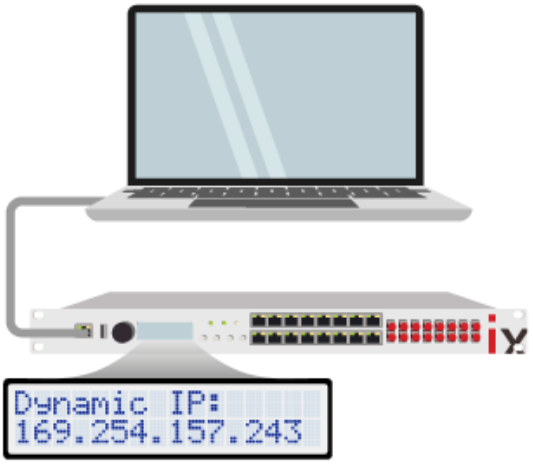
Note that communication between the controller and the host PC uses the following TCP ports: 4840, 59000 through 59010. These ports must not be blocked by your firewall.

The target is directly connected to the host computer

This section applies only to targets running firmware 2025.2 and later (available since ACG SDK 2025.2 beta)
For older versions, it is required to use a *static IP address*. For detailed instructions, please refer to the Advanced Network Configuration page.

Starting with firmware version 2025.2, if no DHCP server is detected, the target automatically assigns itself an IP address in the `169.254.x.x` range. When connected to a PC, the target should appear in Cockpit automatically.

Automatic IP assignment works only when a single Ethernet port is used on the computer. If multiple targets must be connected directly to the same computer, the use of a network switch is strongly recommended. Otherwise, static IP configuration is required. For detailed instructions, refer to the *Advanced Network Configurations* page.



For automatic IP addressing to function correctly, the host PC's Ethernet interface must be configured to obtain an IP address automatically. This is the default setting on Windows, so no changes are necessary unless it has been manually modified.

If needed, this setting can be updated following these steps:

Windows 10

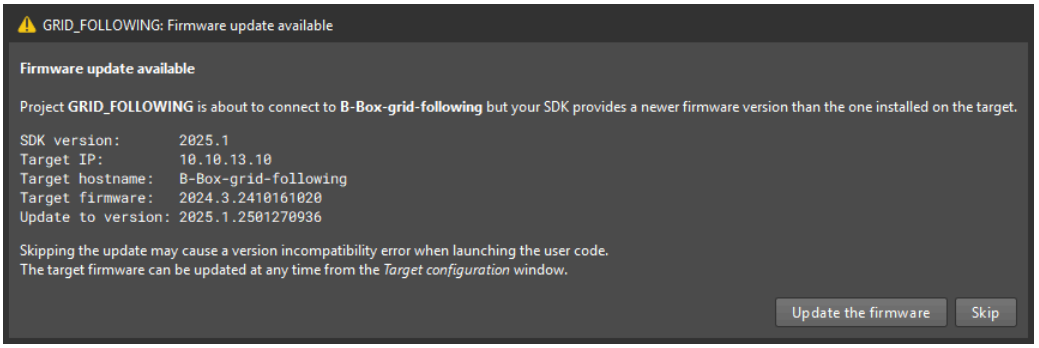
- Navigate to **Control Panel** > **Network and Internet** > **Network and Sharing Center** > from the left pane, **Change adapter settings**.
- Right-click on the Ethernet adapter that is connected to the target and select **Properties**.
- Highlight Internet **Protocol Version 4 (TCP/IPv4)** and click **Properties**.
- Select **Obtain an IP address automatically**

Windows 11

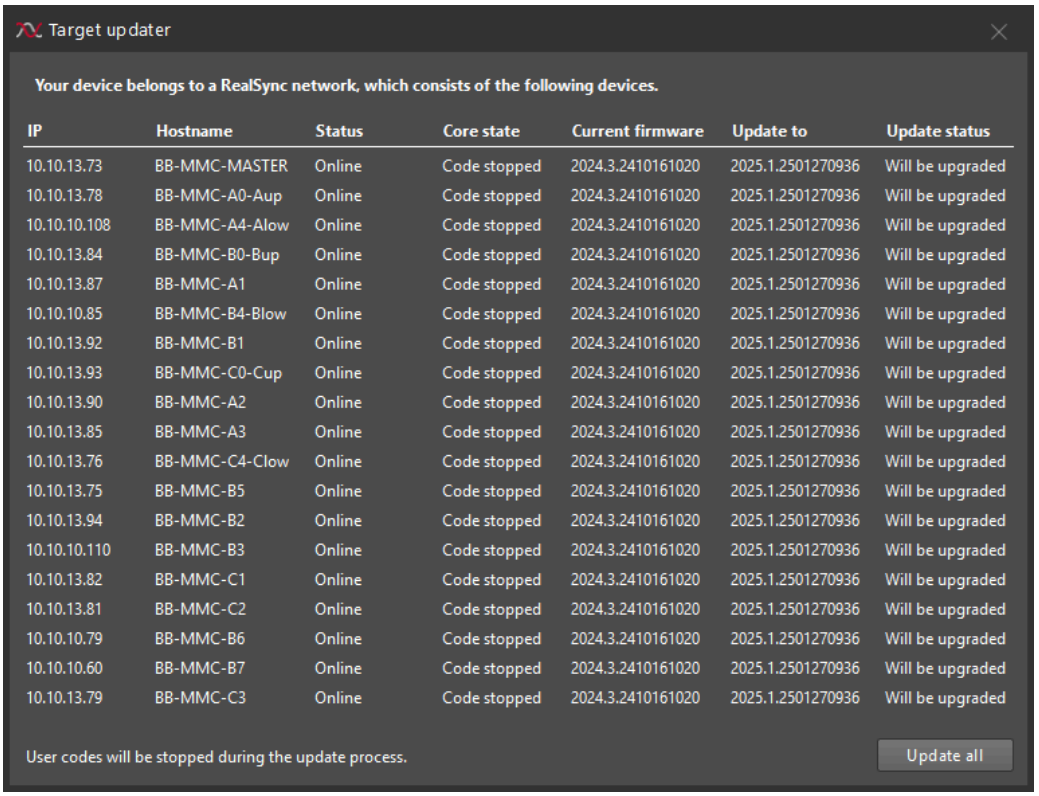
- Open the **Settings** app (`Windows + I`).
- Go to **Network & internet**.
- Select **Ethernet**.
- Select the Ethernet interface connect to the target.
- Make sure the **IP assignment** is set to **Automatic (DHCP)**.

How to update the controller firmware?

After an SDK update, if the controller firmware requires an update, Cockpit will display the following notification window.



Clicking *Update the firmware* opens the Target update window shown below. If multiple devices are interconnected via SFP fibers, they can all be updated with a single click.



Batch updates require *Ethernet over RealSync*, a feature introduced in SDK 2024.3. For controllers running older firmware, each device must be updated individually following these steps:

1. Connect an Ethernet cable to the controller to update.
2. From Cockpit, navigate to the target perspective.
3. Select the target and click on the *Upgrade* button.
4. Repeat the process for the other targets.
5. Power cycle all the targets after completing the updates.

How to operate the controller?

Hardware configuration of the analog inputs of the B-Box RCP

Deploying the user code onto the CPU of the B-Box is not the only action required to operate the controller; the B-Box also requires hardware configurations of its analog front end. The configuration

parameters are not contained within the user CPU code and need to be set from the front panel (LCD screen) of the B-Box RCP. The hardware parameters that need to be configured are the following:

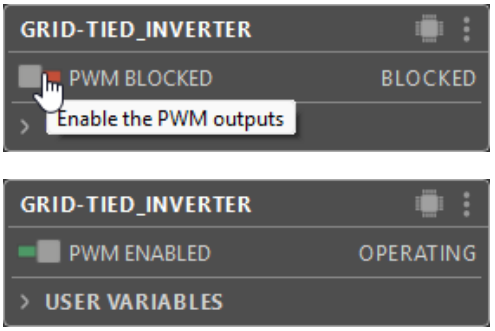
- Impedance of the analog input
- Gain of the analog input
- Cutoff frequency of the input filter (optional)
- Thresholds of the overvalue detection mechanism

To learn more about how to set up those parameters, please refer to:

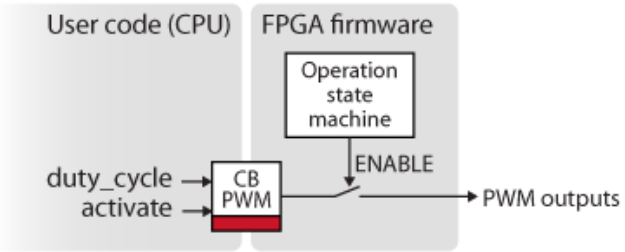
- Analog front-end configuration on B-Box RCP.
- Analog inputs configuration on B-Box Micro.

Enabling/disabling PWM signals

Once the code has been deployed to the target, the code is automatically started, meaning that the CPU interrupt starts running. However, the PWM signals (i.e. gating signals) are not physically produced until they are deliberately enabled by the operator. Enabling and disabling the gating signals is done from Cockpit by clicking the dedicated button at the top of the project. This can be seen as a start/stop switch to control when the converter is actually running (i.e. converting power).



The enabling/disabling process is completely independent of the running control algorithm and acts as a gating switch on all PWM signals. If anything goes wrong during operation of the converter (see `FAULT` state below), the PWM outputs are immediately and automatically disabled to stop the converter operation.



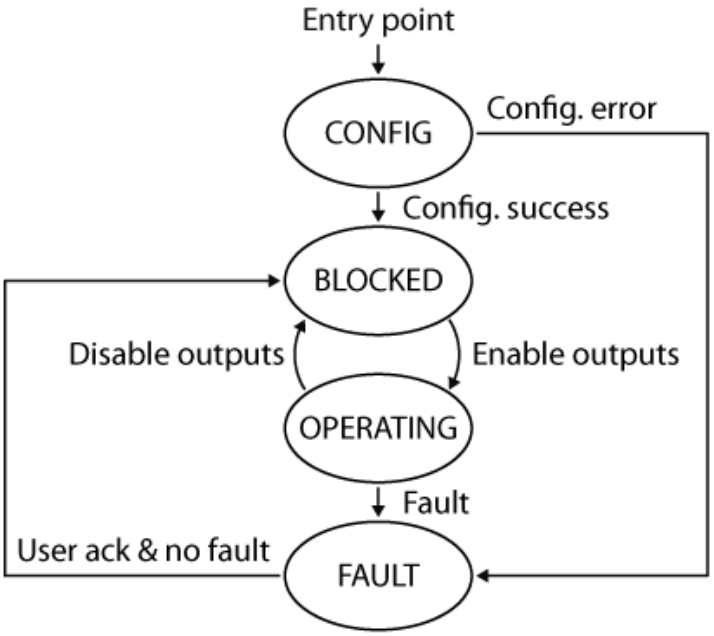
PWM outputs enabling/disabling mechanism

The enabling/disabling mechanism applies **globally** to all the PWM outputs and shall not be confused with the activation/deactivation of a PWM modulator, which serves to **selectively** configure which channel/lane should produce PWM signals once enabled. The activation/deactivation is handled by the user code, using the dedicated input of the PWM modulator block (see PWM – Pulse Width Modulators).

The operating states of the controller

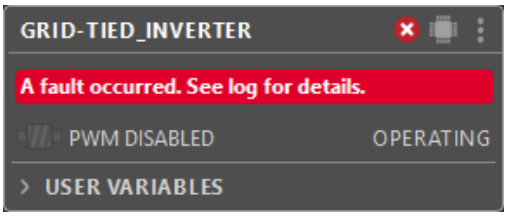
Imperix controllers have four possible states and conform to the state diagram below:

- `CONFIG`: the system is checking that the peripherals (ADC, PWM,…) are correctly configured.
- `BLOCKED`: the system is configured correctly and PWM outputs are blocked (ready to be enabled).
- `OPERATING`: The PWM outputs are enabled, and the system is operating without error.
- `FAULT`: An error occurred and the system waits for its acknowledgment. The PWMs are blocked.



Operation states of BBOS

The controller goes into `OPERATING` state when the PWM outputs are enabled, and goes back to `BLOCKED` state when disabled. In case of error (e.g. due to an overcurrent detection in the power setup), the controller immediately goes into `FAULT` state, which immediately disables the PWM outputs. The source of the fault is described in Cockpit, either in the red banner located in the “Project pane“ (see image below) or in the “Log module”. The return to the `OPERATING` mode is only allowed once the fault is cleared and acknowledged by the user.



The most common types of faults

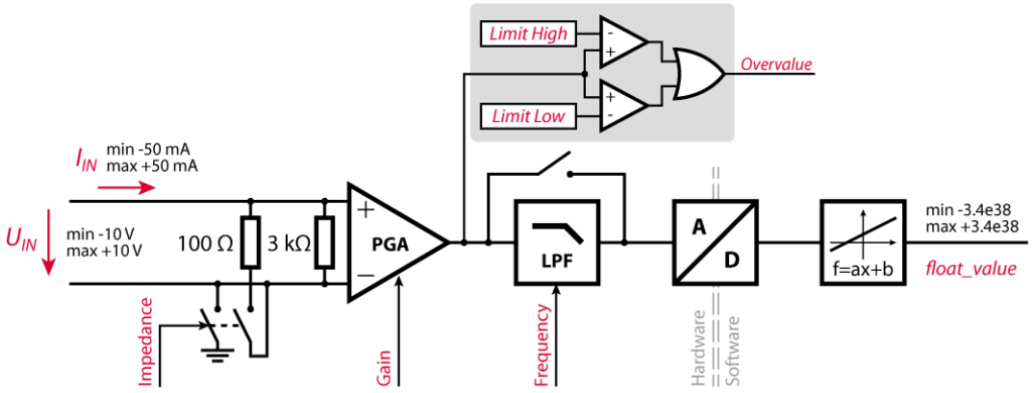
Controllers can reach the `FAULT` state for various reasons. The most common ones are listed in the table below. A fault can be recoverable or unrecoverable, depending on whether clearing it requires re-starting the code (possibly re-compile) or not. Obviously, any type of fault immediately switches off the PWM outputs to stop the operation of the converter.

Fault type	Reason(s)	Recoverable
Configuration	Misconfiguration in the user code, such as: – Physical resource used multiple times (e.g. analog input) – Excessive interrupt frequency considering the code complexity	No
Code crash	Can be CPU or FPGA watchdog expired, C exception, etc…	No
Overvalue detected	The configured threshold on one of the analog inputs of the B-Box is exceeded. *(See section below)*	Yes – requires user ack
Interlock	A fault is triggered by an external device connected to the interlock connector (electrical or optical). *(See B-Box datasheet)*	Yes
Fault line	One of the fault inputs (FLT) is pulled high. *(See B-Box datasheet)*	Yes

Most common types of faults encountered on a B-Box or B-Board

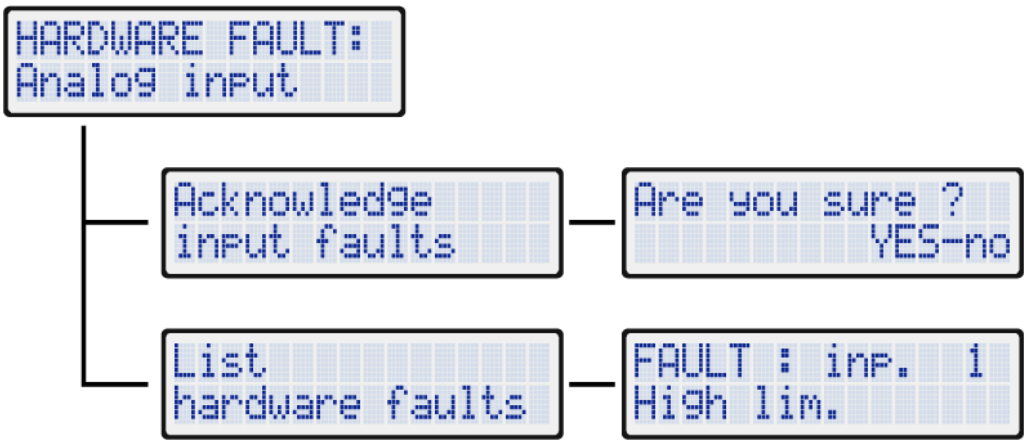
Overvalue detection on the B-Box RCP

As detailed in analog front-end configuration, the analog inputs of the B-Box RCP embed analog comparators that trigger a fault in case the measured value exceeds one of the user-defined limits. This mechanism is particularly useful to ensure that the power converter stays within safe operating conditions (for itself and the surrounding equipment). Furthermore, as the overvalue detection mechanism is implemented 100% with hardware, it is entirely independent from the software and therefore guarantees maximum reliability.



Hardware-based overvalue detection in the B-Box RCP

If an overvalue is detected, the PWM outputs are immediately disabled, the screen of B-Box displays "`HARDWARE FAULT: Analog input`", and the orange LED of the corresponding analog input lights up. The user can use the button on the front panel to navigate through the menus of the screen to get further information about the fault and acknowledge it:



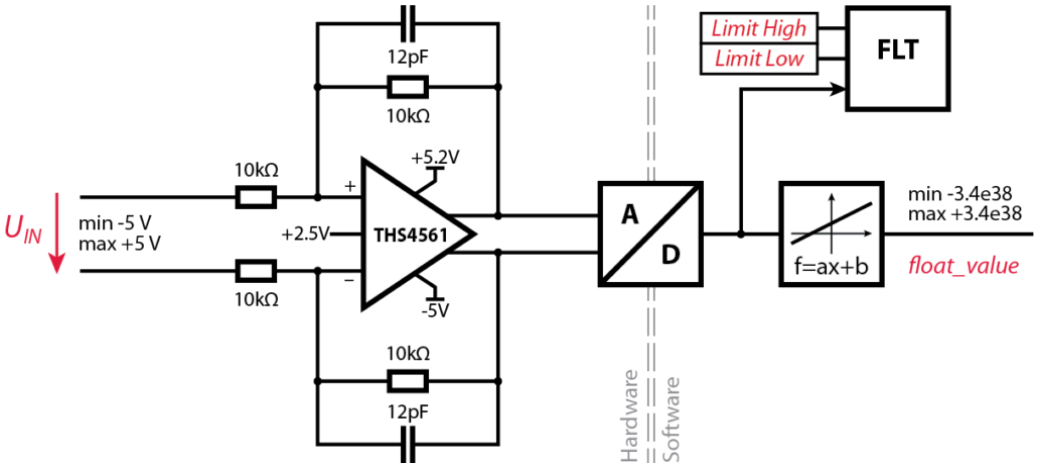
Messages displayed on the B-Box screen in case of overvalue detection

After hardware faults are acknowledged, the B-Box returns to BLOCKED state.

Overvalue detection on the B-Box Micro

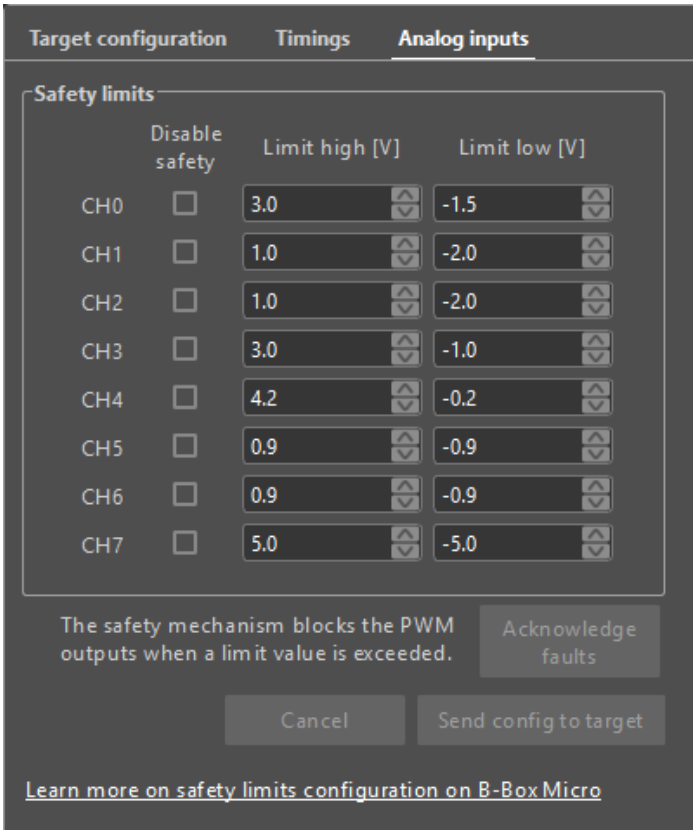
As detailed in analog inputs configuration, the analog inputs of the B-Box Micro are more rudimentary than on the B-Box RCP. Notably, the programmable impedance, programmable gain and programmable filter are not available.

The overvalue detection mechanism also differs, as it is not implemented in hardware, but rather in the FPGA firmware. This offers an intermediate level of reliability, as independence with the user software is guaranteed, but protections remain however dependent from the proper operation of the FPGA.



Software-based overvalue detection in the B-Box Micro

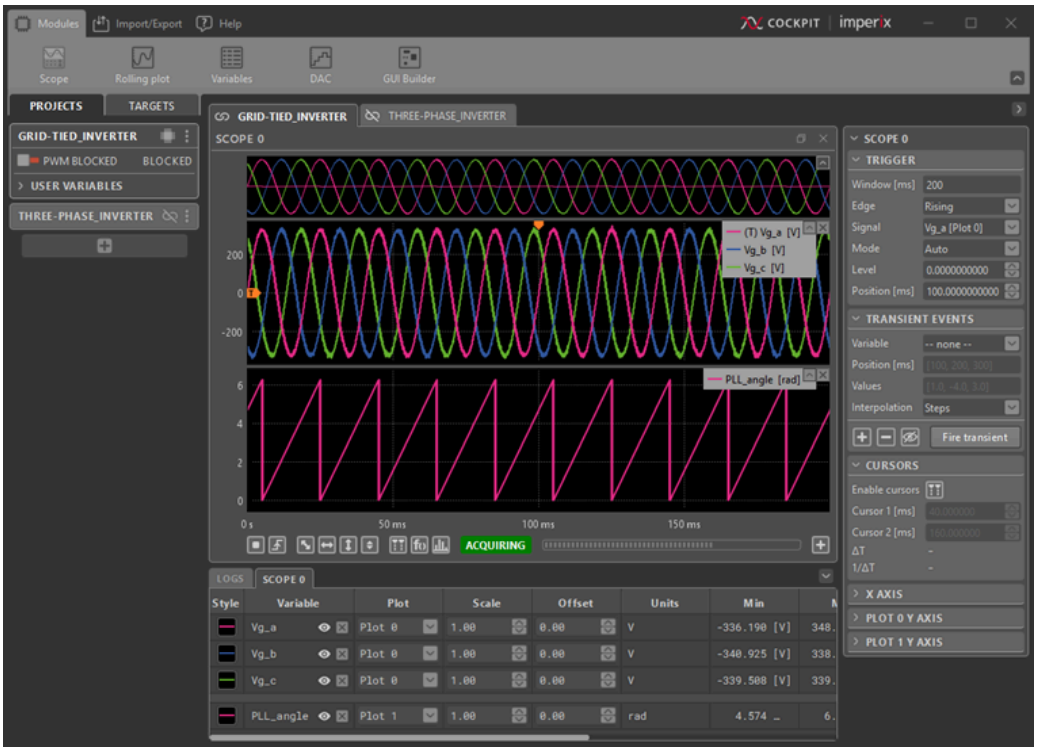
In the B-Box Micro, the safety limits can be directly configured from within Cockpit. For that, switching to the *Targets* perspective and selecting the corresponding B-Box Micro will display the *Target configuration* menu in the Cockpit central view. This view offers an additional tab called Analog inputs (see image below).



Target configuration tab in Cockpit.

Run-time monitoring and parameter tuning in Cockpit

Each Cockpit project contains a view in which the user can drag and drop modules to monitor and alter user-defined variables. Any signal in the user code connected to a Probe block can be monitored in real time using the *Scope* or *Rolling plot* modules. Additionally, the Tunable parameters defined in the user code can be modified at run time from within the *Variables* module.



The Cockpit user manual provides more details on the capabilities of the Cockpit software.

Where to go from here?

Learn more about Cockpit

- Cockpit user manual

Build power converter setups using imperix power modules

- How to build a buck converter
- How to build a 3-phase inverter

Get inspired by numerous code examples to control

- Grid-tied inverters
- Motor drives
- Multi-level converters

Learn how to program the FPGA of the controller to develop high-performance control loops

- Getting started with FPGA development