Hardware protections on imperix controllers

PN263 | Posted on July 7, 2026 | Updated on July 7, 2026

Shu WANG
Development Engineer
imperix · in

Nicolas CHERIX
Head of Engineering
imperix · in

Table of Contents

Imperix controllers possess protection mechanisms that operate fully independently from all software, i.e., independently from the *user app CPU*, the *supervisor CPU*, and the FPGA. This guarantees that the protections can always be trusted, even if the software or the custom FPGA logic isn't completely reliable (yet). A side benefit, the software-independent nature of the protections avoids adding any computational burden to the *user app CPU*.

A typical example of a protection trip is the occurrence of an over-value (e.g., an over-current) on an analog input channel. Another example is the failure to respect the control processing timings (loss of real-time). In case of such faults, all PWM outputs are instantly blocked, thereby preventing damage to the power stage. Successively, the controller is switched to the FAULT state, preventing outputs from being re-enabled before the root cause is clearly identified.

This page provides details specific to the implementation of hardware protections on imperix controllers, as well as regarding their management at the software level.

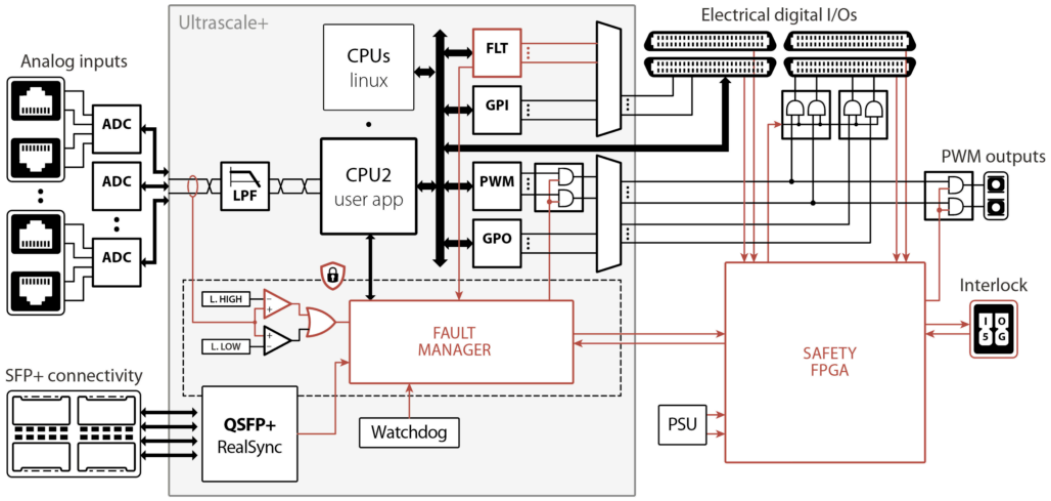
More general information on the operating principles of imperix controllers is given in PN261.

Before conducting laboratory experiments, it is recommended to fully understand how the protection mechanisms operate. Apart from reading this page, useful practical guidelines are also provided in:
(PN257) Over-current and over-voltage protection
(TN181) Safety recommendations for working in the lab

Protection system architecture

At the hardware level, all fault signals are routed to a central fault manager inside the FPGA, which oversees the system's operating state and controls the global activation of PWM outputs. The fault manager is also coupled with external logic, which is implemented on a dedicated FPGA in the B-Box 4, and discrete electronics on other devices.

Below is an overview of how protections are implemented on the B-Box 4.

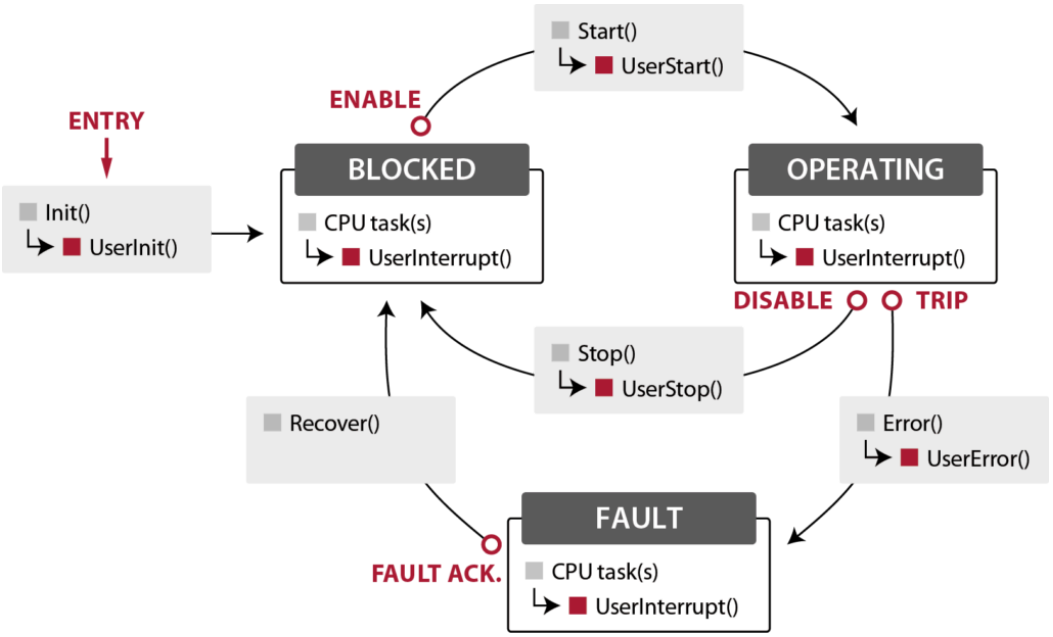


Implementation of the protections inside the B-Box 4

At the software level, the protection system authorizes, respectively prevents, the generation of PWM gating signals depending on the so-called *core* state of the controller. Three distinct states are possible in relation to the operating state of the power stage:

- **BLOCKED**: The controller is operating normally but all PWM outputs are inhibited. The CPU control task is hence running (including all the contained control algorithms and communication functions), but the power stage remains blocked.
- **OPERATING**: The controller is operating normally and all PWM outputs are normally produced. The power stage is actively driven.

- **FAULT**: An error occurred and the system is waiting for its acknowledgment. All PWM outputs are blocked similarly to the BLOCKED state.

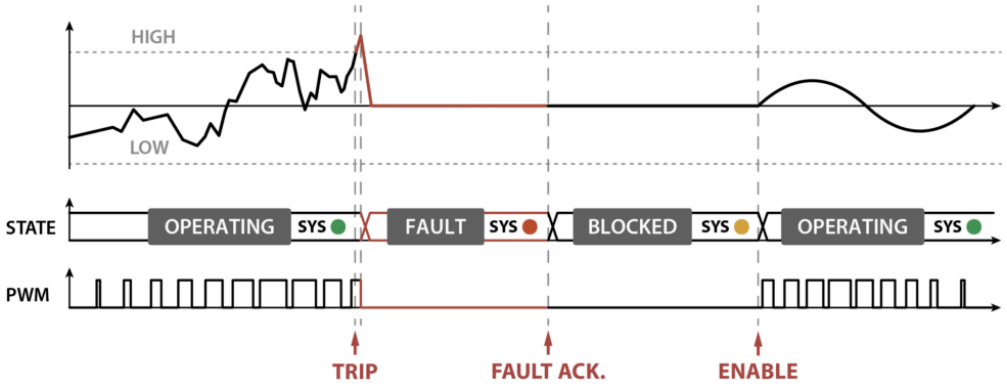


More information on the core state machine as well as the related transitions is given in PN261.

Protection tripping

Operating principles

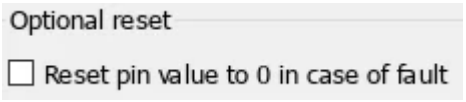
The illustration below shows the protection trip and acknowledgement sequence using the analog over-value protection as an example. In case of a fault, the following sequence takes place:



1. Fault is detected. The controller reacts almost instantaneously. An analog over-value fault is typically detected within 0.8 µs for the B-Box 4 (reaction times for other faults vary based on different controllers and fault sources; see the datasheet for more details). During this ultra-short interval, the fault current remains strictly

bounded. The primary delay factors during this phase are the bandwidth of the external sensor and the analog input filtering.

2. Actual trip. All PWM outputs are instantly blocked, allowing the fault current to decay naturally based on the power stage inductance. GPOs remain unaffected unless explicitly configured otherwise within the software (via a checkbox

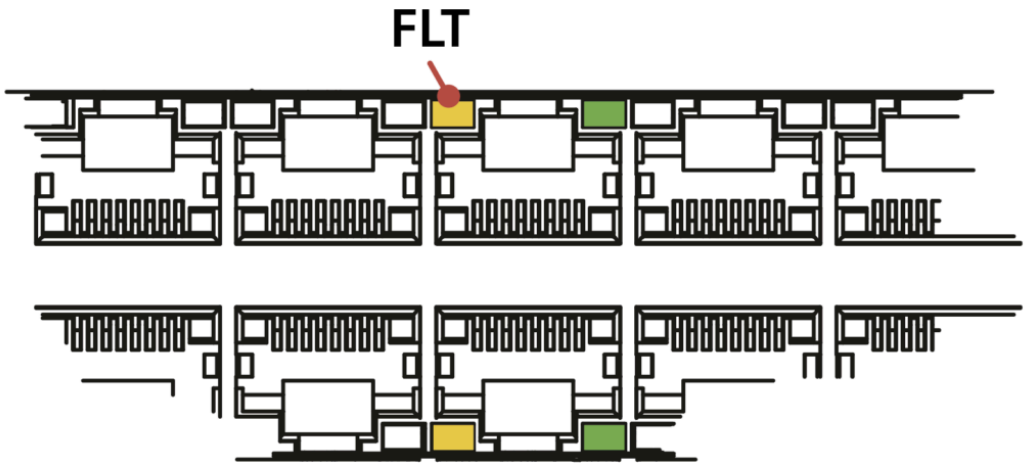


)(this highlights the critical functional distinction between GPOs and PWMs).

The controller is safely blocked at this stage, but the power stage itself is not automatically de-energized.

Concurrently, at the software level:

- The *core* state is switched to FAULT.
- A log message is displayed in Cockpit.
- The system (SYS) or CORE LED is switched to red.
- If the fault was triggered by an analog input over-value, the orange LED of the faulty channel is lit.
- For CPP SDK users, the routine UserError() is called, which offers the opportunity to implement specific actions to be executed at this point.



Orange FLT LED on analog input ports

As long as the fault condition is present (e.g., the voltage or the temperature is too high), the fault cannot be acknowledged, and the operation cannot be resumed.

3. Fault condition disappearance. Three sequences are possible upon clearance of the fault:

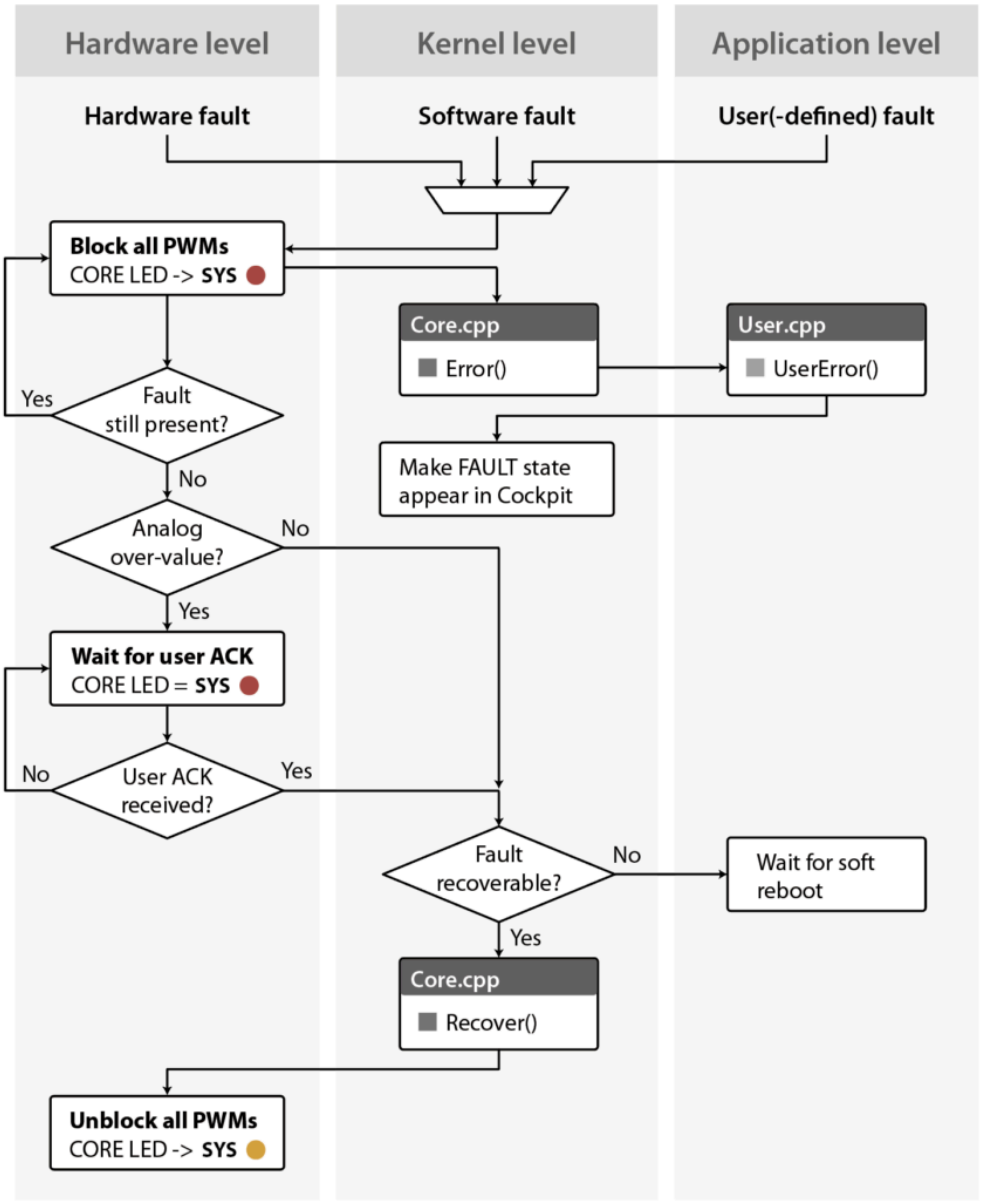
- Most faults are automatically acknowledged (e.g., interlock line). Operation can be resumed immediately.

- Analog over-values must be manually acknowledged. Operation can only be resumed afterward.
- Software faults cannot be immediately recovered from. The user code must be restarted.

4. User acknowledgement. The fault should be acknowledged only after the users have clearly understood the root cause of the fault. Additionally, at the software level:

- The *core* state is switched to BLOCKED, meaning all PWM outputs remain blocked, but they can be re-enabled on demand.
- The system (SYS) or CORE LED is switched to orange.
- Any orange LED on analog inputs is turned off.

The whole sequence is summarized below:



Fault source types

A protection trip can be triggered by any of three possible source types:

1. **Hardware fault**: Triggered at the hardware level, such as an analog measurement (voltage, current) exceeding its configured threshold. External fault flags (digital I/Os), such as on the interlock connector or VHDCI connectors, also induce so-called hardware faults.
2. **Software fault**: Triggered at the kernel level to prevent unsafe CPU operation. The possible causes are either a software crash or an excessive execution time (overrun).
3. **User fault**: Triggered intentionally by returning a fault flag from the user-level routines. By doing so, the user decides *on purpose* that safety mechanisms must be activated.

The following table lists the common fault sources:

Cause	Available on	Type	Clear	Reason
Analog input over-value	B-Box only	Hardware	Manual ack.	Analog input exceeds the programmed safety limit
VHDCI fault lines	All except TPI	Hardware	Automatic	Active-low fault input from VHDCI ports
General FLT inputs	All except B-Box Micro, TPI	Hardware	Automatic	Active-high fault input from FLT ports
Interlock	All	Hardware	Automatic	Fault input from Interlock
RealSync	All except B-Box Micro	Hardware	Automatic	Fault propagated from other controllers inside the RealSync network
SOA	TPI only	Hardware	Automatic	Exceeding the TPI's Safe Operating Area (SOA)

Cause	Available on	Type	Clear	Reason
Watchdog	All	Software	Unrecoverable	CPU software crash
CPU overrun	All	Software	Unrecoverable	Unable to finish the CPU task within a control period
User-defined faults	All	User	Automatic	User-declared fault

Protection trip sources

1. Hardware faults

Hardware faults can originate from five distinct sources:

a. Analog over-value protection (B-Box only)

This critical safety feature protects the system by continuously comparing incoming analog signals against user-defined safety thresholds. If a threshold is violated, a trip signal is sent to the fault manager.

The reaction times and configuration methods for these analog thresholds differ among controllers, as summarized in the table below:

	B-Box 4	B-Box 3 (RCP)	B-Box micro	B-Board	TPI8032
Reaction time	0.8 or 1.6 µs	1.6 µs	4.0µs	N/A	4.0 µs
Configuration	On front panel or in Cockpit	On front panel	In Cockpit	N/A	Auto-computed Safe Operating Area (SOA)
User ack.	On front panel or	On front	In Cockpit	N/A	Automatic

	B-Box 4	B-Box 3 (RCP)	B-Box micro	B-Board	TPI8032
	in Cockpit	panel			

See also:

- PN257 provides a detailed comparison of over-value protections among imperix controllers with detailed configuration guidelines.
- PN260 details the FPGA-level implementation of the analog input stage inside the B-Box 4.

b. VHDCI fault lines

Each VHDCI connector A, B, C, and D has one general fault input that is active-low and internally pulled up to 3.3 V via a 10 kΩ resistor. These inputs can be left floating when unused.

c. General FLT inputs

These specialized active-high fault inputs are available on the VHDCI connectors A and B. Thanks to the optical expansion boards, these inputs are also compatible with optical feedback signals. The B-Box 4 provides 24 FLT inputs, the B-Box 3 (RCP) and B-Board PRO 16 FLT inputs. Other devices do not possess such inputs.

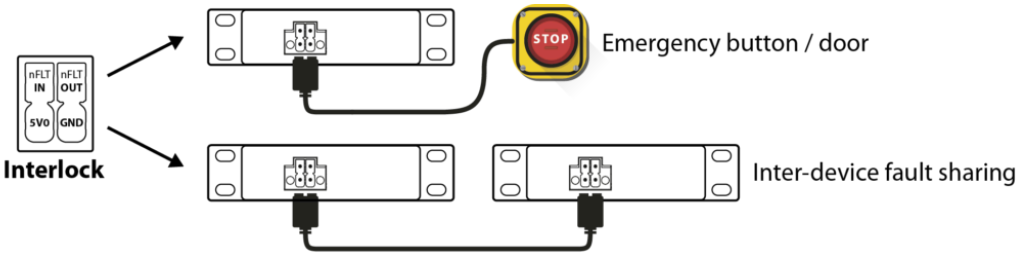
FLT inputs are disregarded by default. They can be enabled by instantiating the FLT block or calling the corresponding C/C++ configuration routine.

d. Interlock

All imperix controllers, except the B-Board PRO, feature an interlock input/output connector for emergency fault inputs or inter-device fault sharing. The interlock input is active-low and internally pulled down by default. Therefore, it must be shorted to the 5V supply or disabled when unused.

To enable/disable the interlock from the B-Box front panel:

1. Push the rotary button once to show the front panel menu.
2. Rotate the button to select interlock, and push to enter the submenu.
3. Rotate to switch between enabled and disabled, then push to save the configuration.



Hardware-related information about the interlock is given in the respective device datasheets.

The reaction time from the interlock fault input to the blocking of PWM signals is generally a few milliseconds, which is very slow compared to the other protections. For fault sharing between imperix controllers, it is recommended to use the imperix RealSync protocol instead, which offers a much lower response time.

e. Fault sharing over RealSync (SFP communication)

In multi-controller setups connected via the imperix RealSync network, any fault is instantly propagated across the entire network with an ultra-low latency of 170 ns per hop (to the adjacent device).

Furthermore, because the RealSync operates over the Xilinx Aurora 8B/10B protocol, it benefits from its intrinsic robust data integrity mechanisms. On imperix controllers, any communication issue immediately generates a protection trip:

- **Cyclic Redundancy Check (CRC):** triggers a fault if any data frame is corrupted.
- **Invalid code & disparity error:** triggers a fault if the received 10-bit code doesn't have the correct disparity or is invalid (not in the 8b/10b table).
- **Link loss:** triggers a fault if the communication link encounters major disruptions, such as loss of transceiver lock or a broken physical link.

More information on operating principles and fault propagation mechanism of RealSync is given on PN264.

2. Software faults

To guarantee the safety and determinism of the hard real-time environment, the controller inherently monitors its own computational and communication integrity:

- **Watchdog**: triggers if the user code crashes or the FPGA fails to receive data from the CPU at the end of the control period.
- **CPU overrun**: triggers if the *user app CPU* fails to finish executing its interrupt within one interrupt period.

A software fault can cause unpredictable errors if the operation continues; these are considered **unrecoverable**. Therefore, the user code must be relaunched (at least).

3. User-defined faults

Additional faults can be declared by the user code using the User fault block or the corresponding C/C++ routine. For instance, in the Grid-Following Inverters (GFLIs) example, PWMs are shut down when the grid voltage drops below a certain threshold or when the grid synchronization is lost.

To go further

- Device-specific practical guidance for configuring the analog protection thresholds is given in PN257. The page also explains how these protections should be coordinated with other safety-related mechanisms (on-module protections, circuit breakers, etc.).
- PN261 develops further on the operating principles of imperix controllers in general, also addressing the devices' architecture and their specificities.