

# Analog front-end configuration on B-Box RCP

PN105 | Posted on March 29, 2021 | Updated on May 7, 2025



**Nicolas CHERIX**  
Head of Engineering  
imperix • in



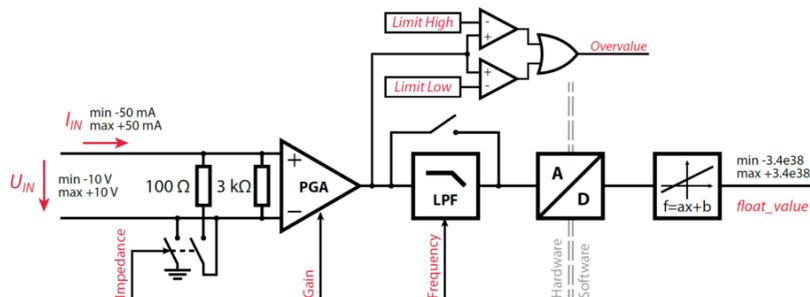
**Benedikt BORTER**  
Hardware Development Engineer  
imperix • in

## Table of Contents

- [Front-end configuration \(hardware\)](#)
  - [Configuring the hardware protection limits](#)
  - [Storing and restoring a configuration](#)
  - [Reading and writing configuration files](#)
- [Software configuration](#)
  - [Simulink blockset](#)
  - [PLECS blockset](#)
  - [C/C++ configuration](#)

This page covers the configuration of the analog front-end of the B-Box RCP. The B-Box RCP possesses 16 analog inputs with strictly identical channels. The equivalent schematic of the complete data acquisition chain is depicted below. Each channel consists of two parts:

- A **hardware** part, which contains a configurable input impedance, a programmable gain amplifier (PGA), a low-pass filter (LPF), and the analog-to-digital converter (ADC).
- A **software** part, which is notably responsible for transferring all data to the CPU memory in the shortest possible time as well as transforming the converted value into a meaningful quantity (i.e. considering the sensitivity of the whole chain, from the sensor to the converted value).



The B-Box RCP offers a fully-configurable front-end, which is not available on the B-Board PRO. Nevertheless, on the latter, all characteristics are designed (or even compensated) such that both analog input chains are 100% compatible. More information is available in the [datasheet](#) as well as in [PN201](#).

## Front-end configuration (hardware)



The LCD screen and rotary-push button allow to read and write all configuration parameters of the analog front-end. To access the related menu:

1. Push once, select ANALOG INPUTS and confirm (push again).
2. Select the desired input channel and confirm.

Once inside the configuration menu of each channel, the following options are available:

Menu	Available values	Comment
Input impedance	Low, High	Low = 100Ω / High = 3kΩ
Gain	1x, 2x, 4x, 8x	This is equivalent to setting the input scale to ±10V, ±5.0V, ±2.5V and ±1.25V.
Filter	0.5kHz, 1kHz, 1.6kHz, 2.5kHz, 4kHz, 6.4kHz, 8kHz, 10kHz, 16kHz, 20kHz, 32kHz, 40kHz	More information on the corresponding group delay can be found in the B-Box RCP <a href="#">datasheet</a> .
Limit high	-10V to +10V	Increments of 0.1V. Precision is approx ±0.2V.
Limit low	-10V to +10V	Increments of 0.1V. Precision is approx ±0.2V.
Disable safety	Yes / No	Can be used to disregard the protection thresholds. In this case, the green LED indicator switches off on the corresponding input channel.

The low-impedance input configuration is often useful when interfacing the B-Box RCP with LEM sensors (or similar), which present a current that is proportional to the sensed quantity (instead of a voltage). This approach generally offers superior immunity to inductive-type EMI noise.

## Configuring the hardware protection limits

The analog frontend of the B-Box RCP provides a hardware protection mechanism that blocks its operation if one of the analog inputs exceeds a configured limit value. This protection is particularly valuable during the development phase of control algorithms and, if configured properly, it ensures that the ratings of the controlled power converter are never exceeded. Furthermore, this mechanism is totally independent from the user code running on the device.

As seen on the schematic at the top of this page, the amplified analog input is compared to the programmed *Limit High* and *Limit Low* values. Thus, the limit values have to be expressed in volts and taking the the programmable gain value into account.

### Numerical example

We consider a sinusoidal current measured by a [DIN50A](#) sensor, that should not exceed 20 A (RMS value). The sensor sensitivity is 99.0 mV/A and a gain of x2 is configured on that analog input channel. The *Limit High* and *Limit Low* values are computed as follows:

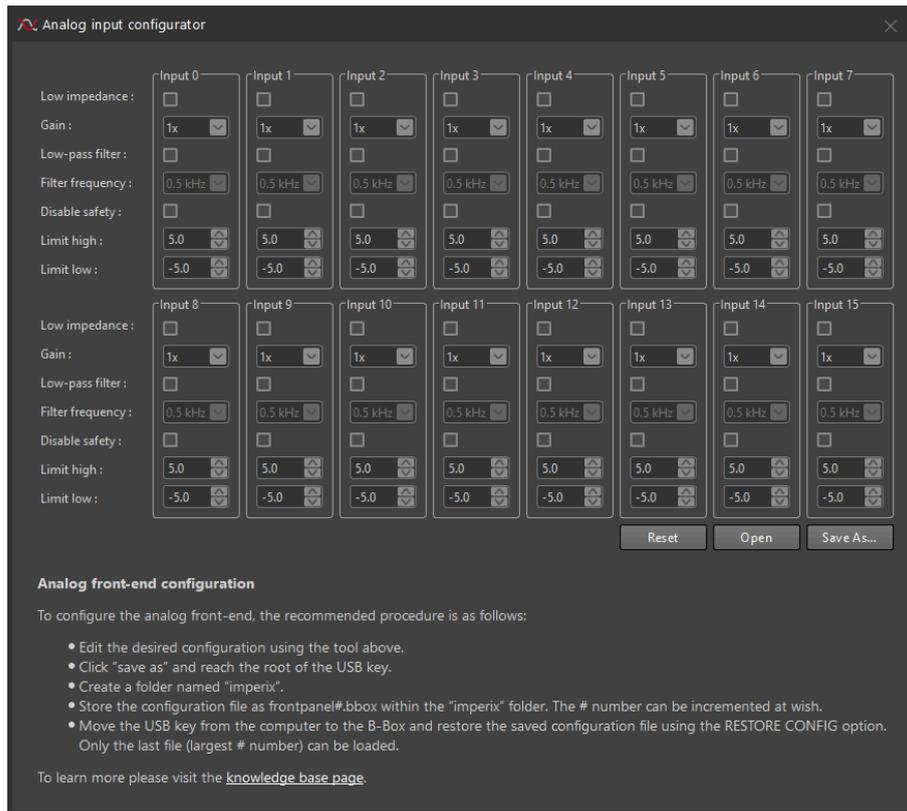
- Maximum acceptable instantaneous current:  $\pm 20 \text{ A} \times \sqrt{2} = \pm 28.28 \text{ A}$
- Corresponding sensor output:  $\pm 28.28 \text{ A} \times 99 \text{ mV/A} = \pm 2.8 \text{ V}$
- *Limit High* and *Limit Low* values:  $\pm 2.8 \text{ V} \times 2 = \pm 5.6 \text{ V}$

## Storing and restoring a configuration

The configuration of the front panel can be saved on a USB key, using the BACKUP CONFIG menu of the front panel. The configuration is automatically (and always!) stored in a file named "frontpanel0.bbox" on the USB key. In this way, the configuration can be saved for different applications.

The configuration can be restored by using the RESTORE CONFIG menu. The B-Box will search for files named "frontpanel#.bbox" in the imperix folder of the USB key, # being a number. The configuration file with the highest number is restored.

# Reading and writing configuration files



The configuration file can be generated using Cockpit. To do so:

1. On a computer, edit the desired configuration using Cockpit.
2. Click "save as" and reach the root of the USB key.
3. Create a folder named "imperix".
4. Store the configuration file as frontpanel#.bbox within the "imperix" folder. The # number can be incremented at wish.
5. Move the USB key from the computer to the B-Box and restore the saved configuration file using the RESTORE CONFIG option. Only the last file (largest # number) can be loaded.

The front-end is physically independent of the main processing circuit *by design*. This guarantees superior reliability of the safety settings, which cannot be altered by software. For this reason, the configuration *cannot* be automatically loaded from Cockpit; using a USB key is necessary.

## Software configuration

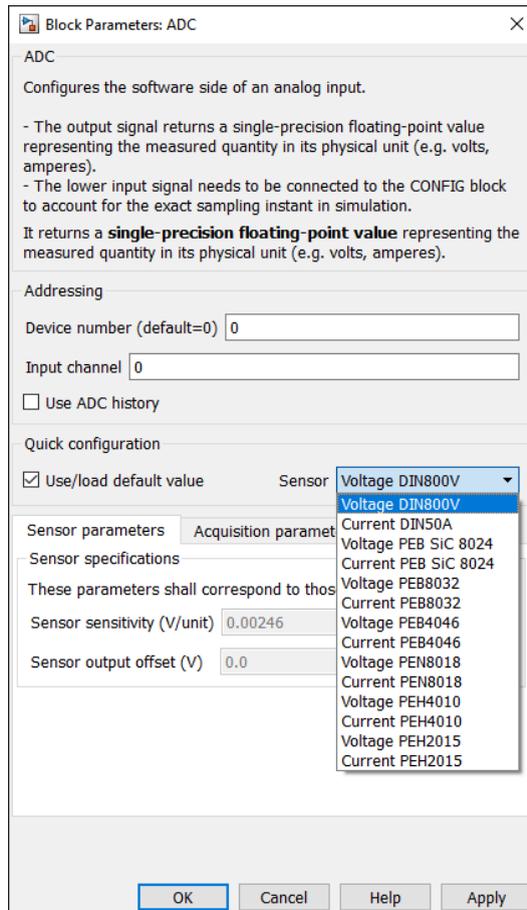
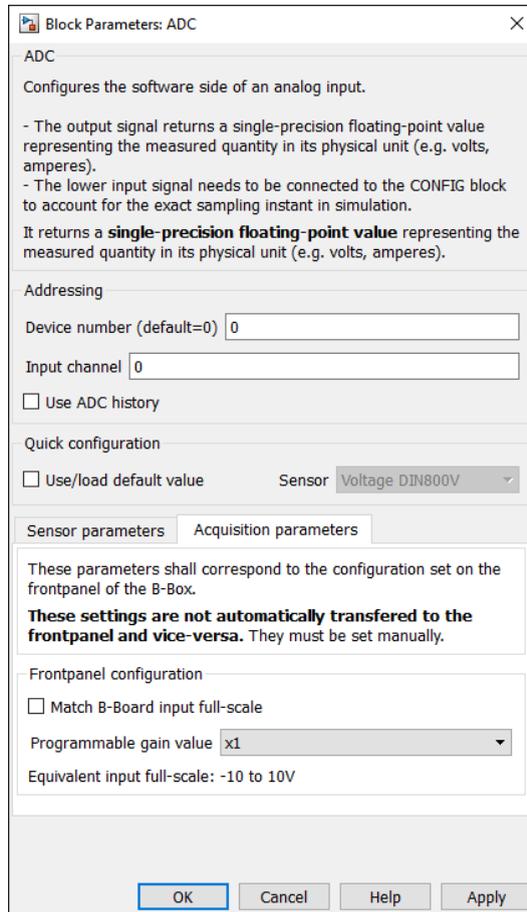
In addition to (and independently of) the hardware configuration of the analog front-end, the B-Box RCP must have its software configured such that the corresponding parameters are appropriately accounted for.

For the same reason as above, the front-end configuration cannot be automatically reported into the user-level software (C++, Simulink, or PLECS). All parameters must, therefore, be manually adjusted.

## Simulink blockset

The dedicated Simulink block is described in [ADC – Analog data acquisition](#).

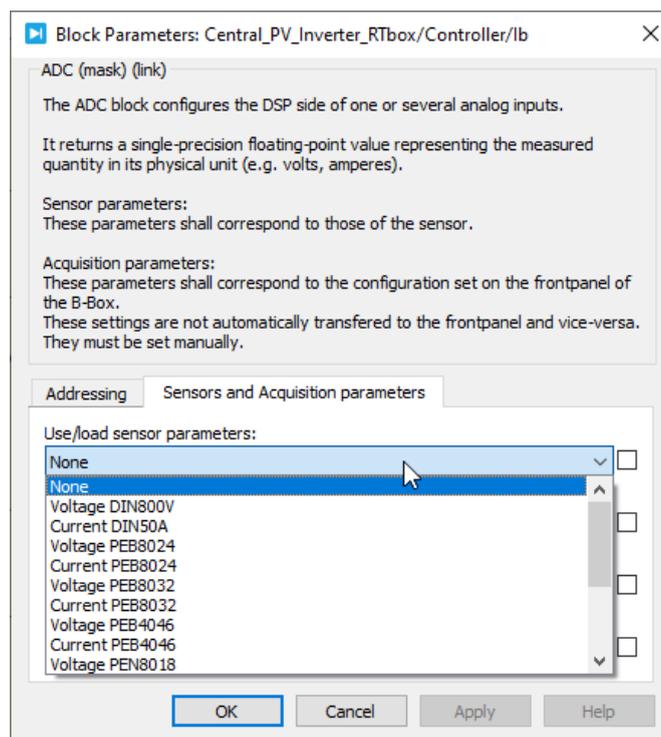
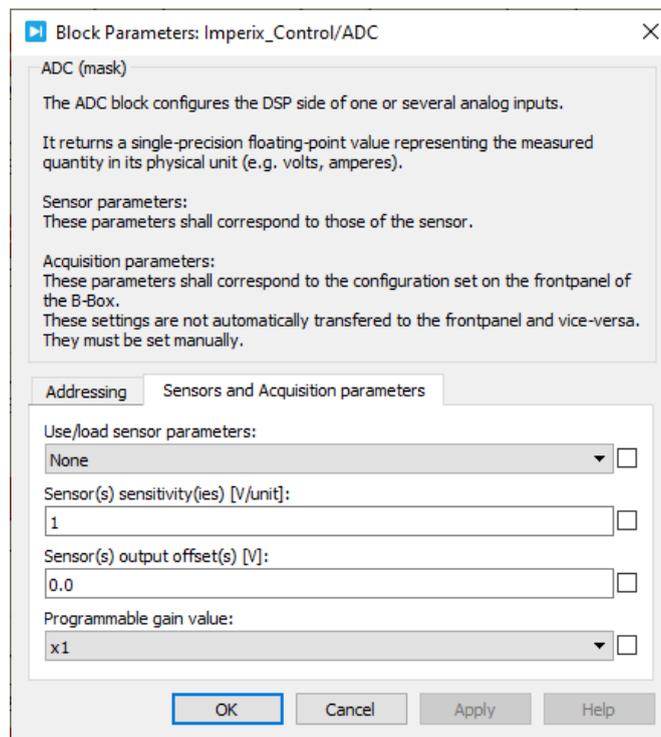
The ADC blocks allow reporting the selected analog input gain (second tab, Acquisition parameters), as well as to indicate the sensors parameters (first tab). When imperix sensors are used, the associated parameters can be automatically loaded from a drop-down list.



## PLECS blockset

The dedicated PLECS block is described in [ADC – Analog data acquisition](#).

The same configuration parameters are accessible from the PLECS blockset, as shown below.



## C/C++ configuration

The dedicated C/C++ routines are described in [ADC – Analog data acquisition](#).

Each ADC channel must be configured during the initialization phase using `Adc_ConfigureInput()`:

```
void Adc_ConfigureInput(uint channel, float gain, float offset, uint bboxid=0);Code language: C++ (cpp)
```

- `channel` is the analog input channel number.
- `gain` and `offset` must be configured considering that the returned value during operation is computed as  $y = ax + b$ , where  $a$  is the gain and  $b$  the offset.
- `bboxid` is the ID of the addressed B-Box (optional parameter, useful in a multi-B-Box configuration only).

### Numerical example:

This example considers the current sensor of a [PEB8024](#) module. Its sensitivity is  $S = 50.0 \text{ [mV/A]}$ . As recommended in the [datasheet](#), the chosen front-end gain is selected as  $G = 2$ . Considering that the ADC offers 16 bits over the  $\pm 10\text{V}$  input range, this results in a total sensitivity of  $\alpha = S \cdot G \cdot 32768/10 = 327.68 \text{ [bit/A]}$ .

In this example, gain must therefore be equal to  $a = 1/\alpha = 3.052 \text{ [mA/bit]}$ . The offset value can be adjusted empirically to cancel the measured value when no current is flowing through the sensor (static offset).

Subsequently, within each interrupt, the latest value can be retrieved using `Adc_GetValue()`:

```
float Adc_GetValue(uint channel, uint bboxid=0);Code language: C++ (cpp)
```

- `channel` is the analog input channel number.
- `bboxid` is the ID of the addressed B-Box (optional parameter, useful in multi-B-Box configuration only).