SBO - Sandbox output towards FPGA

SD023 | Posted on April 2, 2021 | Updated on June 30, 2025

Benoît STEINMANN
Software Team Leader
imperix

Table of Contents

The Sandbox Output towards FPGA (SBO) block writes the value of the SBO registers in the FPGA. It is used to transfer data from the CPU to the user-made code within the FPGA. To transfer data from user-made code within the FPGA to the CPU, the SBI block should be used.

An SBO register can be configured as:

- a *configuration register*: the value is written only once, at the code launch
- a *real-time register*: the value can change anytime during the control

In Simulink and PLECS, *configuration register* values are defined from the block mask and *real-time registers* from the block input signal.

Information on FPGA edition is available on:

- Editing the FPGA firmware (sandbox) (PN116)

Usage examples of the SBI block are available on:

- FPGA-based hysteresis current control (TN120)
- FPGA-based SPI communication IP for A/D converter (TN130)
- FPGA-based direct torque control using Vivado HLS (TN133)

Simulink block

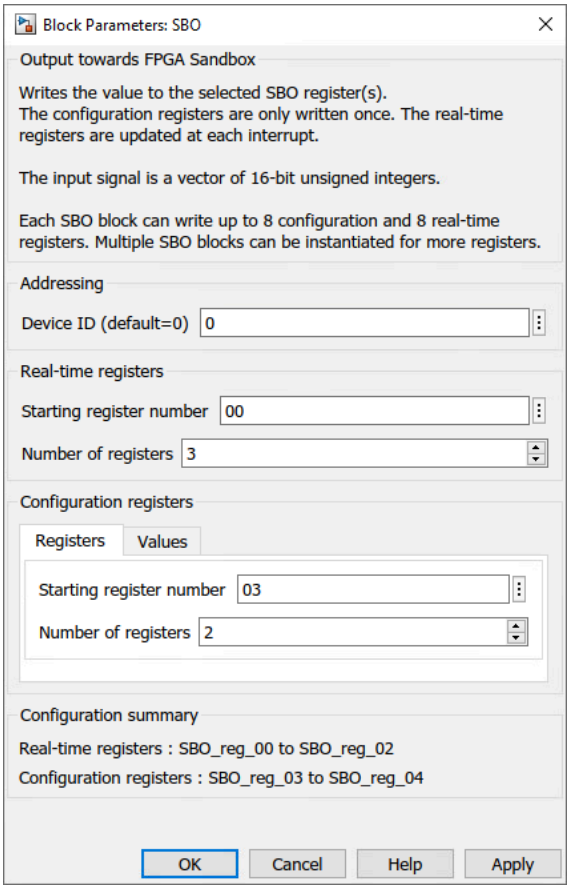
Signal specification

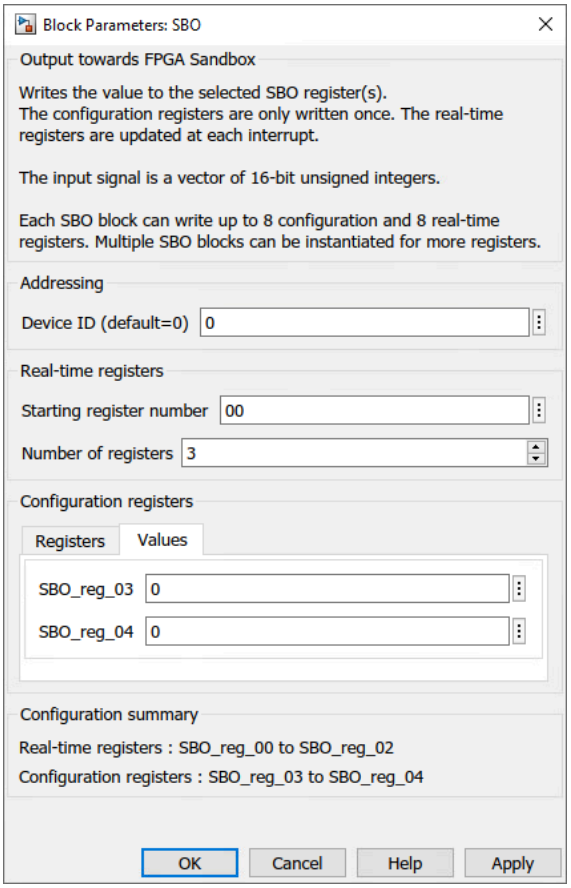
The input expects a vector of 16-bit unsigned integer values to write to the SBO registers.

Up to 8 *real-time registers* and 8 *configuration registers* can be written from a single SBO block. Multiple SBO blocks can be used to write to more registers.

Parameters

- `Device ID` selects which B-Box/B-Board to address when used in a multi-device configuration.
- Real-time registers: `Starting register number` and `Number of registers` define the range of registers to write to.
- Configuration registers: `Starting register number` and `Number of registers` define the range of registers to write to. Their values can be set from the `Values` tab.

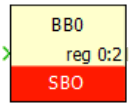




PLECS block

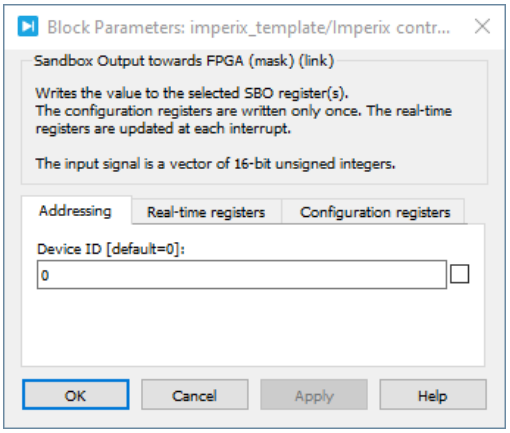
Signal specification

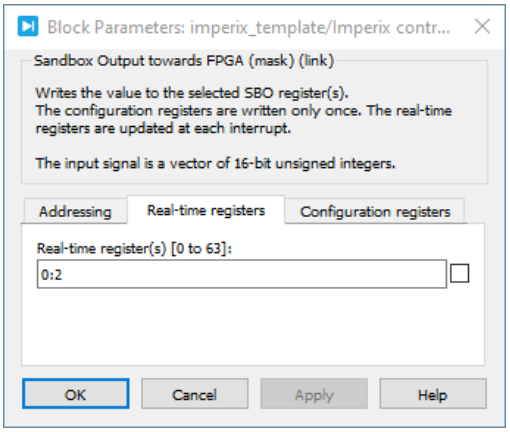
The input expects a vector of 16-bit unsigned integer values to write to the SBO registers

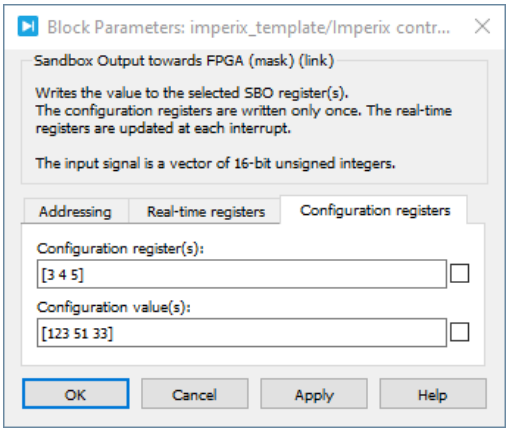


Parameters

- `Device ID` selects which B-Box/B-Board to address when used in a multi-device configuration.
- `Real-time register(s)`(vectorizable) defines the real-time registers to write to using the input signal
- `Configuration register(s)`(vectorizable) defines the configuration registers to write to and `Configuration values(s)` (vectorizable) sets their constant values.







C++ functions

`Sbo_WriteDirectly` **— Write during init phase**

```
void Sbo_WriteDirectly(unsigned int address, uint16_t data, unsigned int device=0);Code language: C++ (cpp)
```

Writes a constant value to an SBO register.

It can only be called in `UserInit()`.

Parameters

- `address`: address of the targeted register (0 to 63)
- `data`: value to write
- `device`: the id of the addressed device (optional, used in multi-device configuration only)

`Sbo_Write` **— Write during run-time**

```
void Sbo_Write(unsigned int address, uint16_t data, unsigned int device=0);Code language: C++ (cpp)
```

Updates the value of an SBO register configured as real-time. It has to be called in the control interrupt.

For this function to work the addressed register must be set as *real-time* using `Sbo_ConfigureAsRealTime()`.

Parameters

- `address`: address of the targeted register (0 to 63)
- `data`: value to write
- `device`: the id of the addressed device (optional, used in multi-device configuration only)

`Sbo_ConfigureAsRealTime` **— Configure as writable during run-time**

```
void Sbo_ConfigureAsRealTime(unsigned int address, unsigned int device=0);Code language: C++ (cpp)
```

Updates the value of an SBO register configured as real-time. It has to be called in the control interrupt.

Tags an SBO register as *real-time*, meaning that its value can be updated from the interrupt routine using `Sbo_Write()` and is transferred to the FPGA at the end of the interrupt routine execution.

It has to be called in `UserInit()`.

Parameters

- `address`: address of the targeted register (0 to 63)
- `device`: the id of the addressed device (optional, used in multi-device configuration only)