Multi-master feature for distributed networked control systems

PN158 | Posted on March 29, 2021 | Updated on May 7, 2025

Benoît STEINMANN
Software Team Leader
imperix • in

Table of Contents

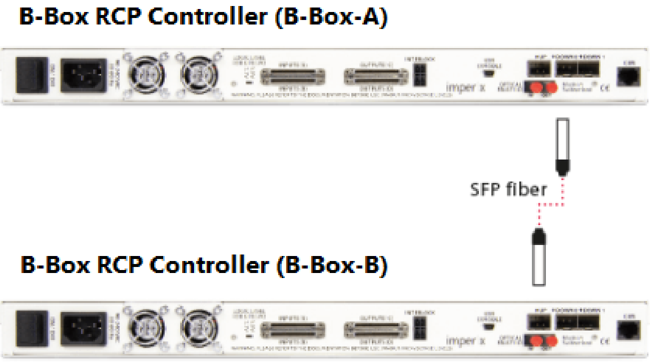
This note focuses on the multi-master feature which allows executing control codes on multiple imperix power converter controllers interconnected using optical fiber (SFP). The imperix in-house protocol RealSync allows for low latency communication (sub-µs) and high data exchange rate (hundreds of kHz) all while keeping synchronized sampling instants and PWM signals (±2ns).

The Multi-converter system for micro-grid (AN008) example illustrates how the multi-master features can be used to implement coordinates control between TPI8032 programmable inverters. Another use case of multi-master communication is presented Coordinated control of a back-to-back inverter (TN155).

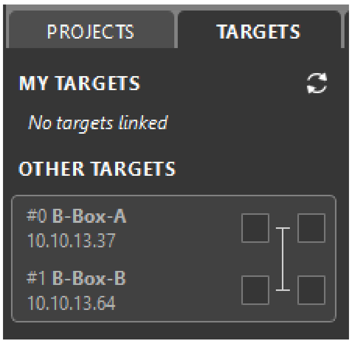
Imperix distributed converter control solutions

The process presented below is based on **SDK version 2025.1**, available on the download page.

The imperix proprietary RealSync communication protocol allows connecting multiple devices using SFP fibers, as shown below with two B-Boxes RCP. The same applies for the B-Board PRO, and the Programmable Inverter.


B-Box RCP Controller (B-Box-A)
SFP fiber
B-Box RCP Controller (B-Box-B)

Two B-Boxes RCP connected using an SFP fiber



B-Boxes are displayed as targets in Cockpit (SDK 2025.1)

Thanks to the Ethernet over RealSync feature, only an Ethernet cable needs to be connected to the top device. However, connecting additional Ethernet cables can help increase bandwidth, especially when a large amount of data is being logged in Cockpit.

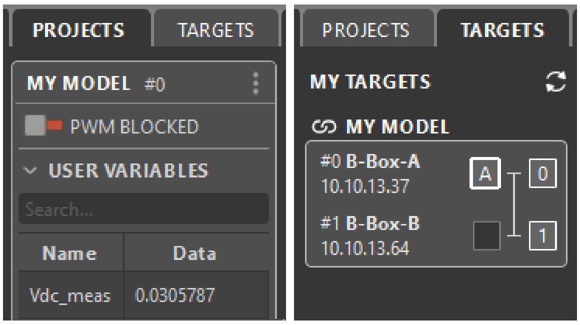
When imperix devices are interconnected:

- All CLOCKs are natively synchronized and by extension the analog inputs sampling, control tasks execution, and the PWM outputs are all synchronized.
- In the event of a fault anywhere one the network, the PWM outputs of all devices are immediately disabled with node-to-node propagation delay of less than 180ns.

A multi-device system can operate in different modes:

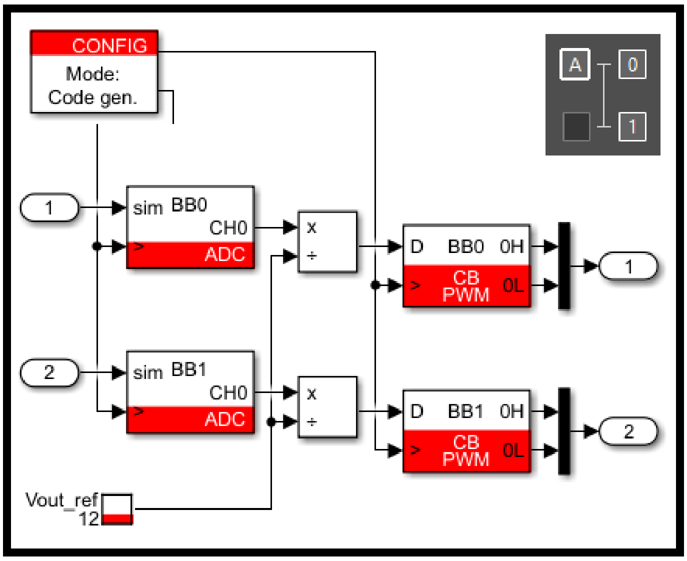
- In **master-slave** mode (aka I/O extension mode) the control algorithm runs on the master unit's CPU, while the slave devices operate as I/O extension units, with only their FPGAs active. Slaves are addressed transparently from the peripheral's block in Simulink/PLECS/C++ code, using the **device ID** parameter (indicated as BB0, BB1, etc. on the blocks).
- In **multi-master** mode, each master device has its CPU running. Since each CPU code requires its own Simulink/PLECS/C++ model, communication between devices must be handled explicitly using communication blocks. The CAN or ETH blocks can be used to exchange data, but using the SFP blocks offers superior performance (sub-microsecond latency and exchange rates in the hundreds of kHz). Detailed information about the SFP blocks can be found in a later chapter.
- In **multi-master with slaves** (or combined mode) both modes are mixed, allowing for the implementation of a wide range of distributed converter control techniques.

Illustration of a **master-slave** scenario



Name	Data
Vdc_meas	0.0305787

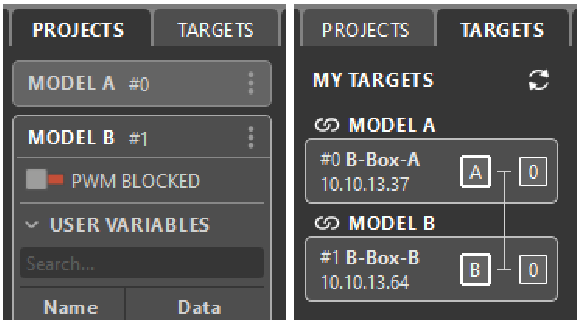
One project (one model) linked to two devices



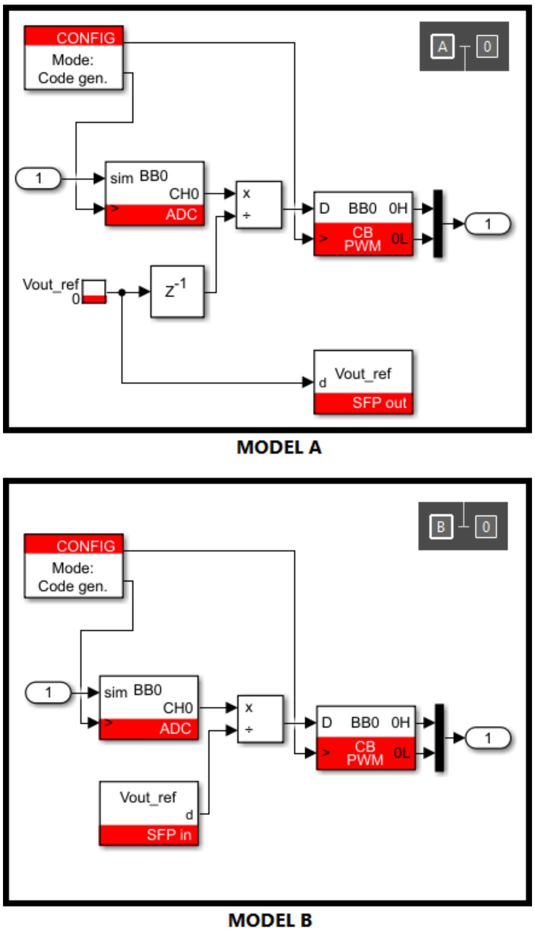
MY MODEL

Simulink model using I/Os (ADC and PWM) of B-Box A (device ID 0) and B-Box B (device ID 1)

Illustration of a **master-master** scenario



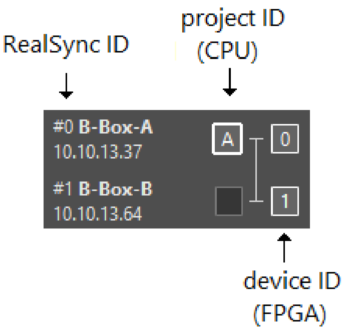
Two projects (two models), linked to B-Box A and B-Box B



Two Simulink models exchanging data using SFP blocks.

A device can only be associated with a single master. This means that, in the example above, the code running on B-Box A cannot directly access the I/O of B-Box B. To exchange data between B-Box A and B-Box B, the SFP communication blocks must be used as explained later.

The meaning of the different numbers and letters is explained below.



Two B-Boxes in a RealSync network

- The ***device ID*** (0, 1, …) is the number used to address the FPGA
- The ***project ID*** (A, B, …) shows where the generated code is running
- The ***RealSync ID*** (#0, #1, …) is the number uniquely identifying a device in a RealSync network

Setting up a multi-master system

To set up multi-master operation, as shown in the video below, the following steps must be followed:

1. Create one model for each master, using Simulink, PLECS or the C++ SDK.
2. Build the models. After the first successful build a project is automatically created in Cockpit.

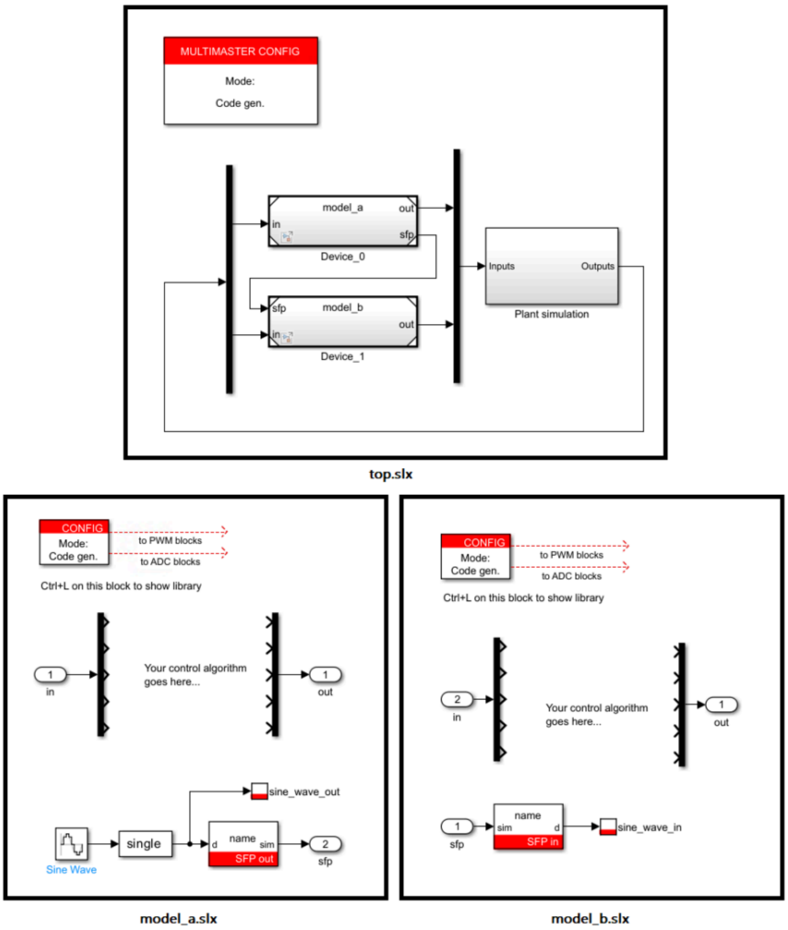
3. Once the projects are created, proceed with linking each project to its respective device as illustrated on the screenshots below. Once a device is linked to a project, it becomes a master, identified by a letter (e.g., A, B, etc.).

When a model is rebuilt, all the user codes are automatically stopped then restarted by Cockpit.
During the linking process, a configuration error may occur, indicating that not all devices are linked. Once all devices are properly linked, the codes will restart, and the error should disappear.

Multi-master using Simulink

The Simulink template provided below simplifies the development of distributed control using the multi-master feature. This template aims to facilitate distributed control development. It uses Simulink Model references to support both simulation and code generation in the same development environment.

Download **multimaster_template.zip**



top.slx

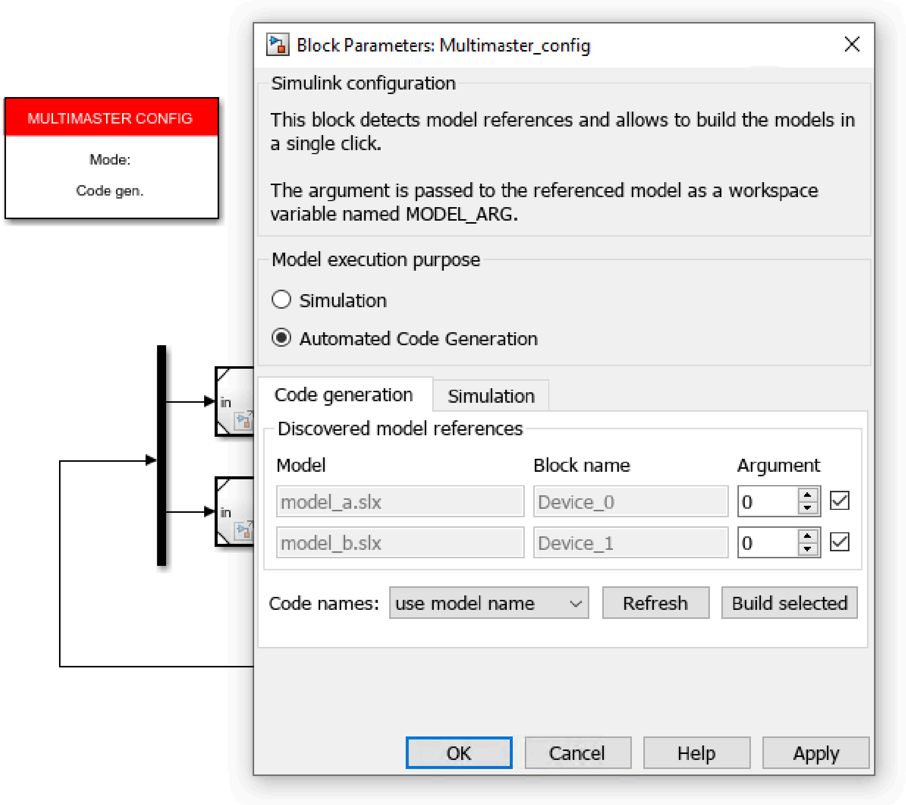
model_a.slx

model_b.slx

Simulink multi-master template

The image below shows the *Multimaster config* block provided in the template.

- The user can add additional model references and even reference the same model multiple times. Clicking the *Refresh* button will update the list of model references.
- If a model is referenced multiple times, the option *use block name* must be selected to avoid naming conflicts.
- Changing the model execution purpose in the multi-master config block will apply the change to all referenced models.
- Clicking *Build selected* will generate and build the codes for all the models that are checked in the list.



Block Parameters: Multimaster_config

Simulink configuration

This block detects model references and allows to build the models in a single click.

The argument is passed to the referenced model as a workspace variable named MODEL_ARG.

Model execution purpose

○ Simulation

● Automated Code Generation

Code generation | Simulation

Discovered model references

Model	Block name	Argument	
model_a.slx	Device_0	0	☑
model_b.slx	Device_1	0	☑

Code names: use model name

Simulink multi-master config block

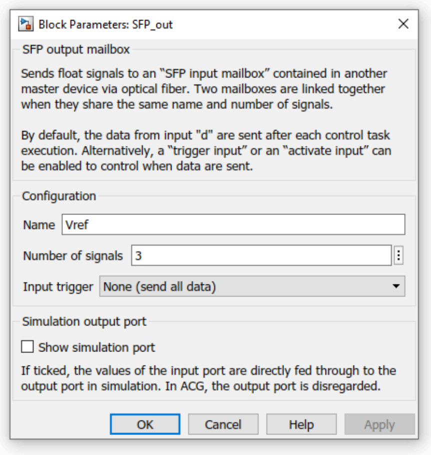
When using referenced models, launching the code generation using Simulink build button (or Ctrl+B) is no longer supported. The build must be launched from the *multimaster config* block instead.

Exchanging data between the models

The SFP blocks shown below allow exchanging data between models via the optical fiber links. These SFP blocks work in pairs: an *SFP output mailbox* must be placed in one model, and an *SFP input mailbox* in another. The two mailboxes are automatically linked together when they share the same name and number of signals.

The SFP mailboxes are not broadcasting. They operate in a one-to-one configuration, each output mailbox can send data only to a single designated input mailbox.

By default, the *SFP output mailbox* sends data at each control period. However, the user can enable a *trigger input* to control precisely when data is sent. On the *SFP input mailbox* side, a *data valid output* port indicates when new data has been received.



Block Parameters: SFP_out

SFP output mailbox

Sends float signals to an "SFP input mailbox" contained in another master device via optical fiber. Two mailboxes are linked together when they share the same name and number of signals.

By default, the data from input "d" are sent after each control task execution. Alternatively, a "trigger input" or an "activate input" can be enabled to control when data are sent.

Configuration

Name: Vref

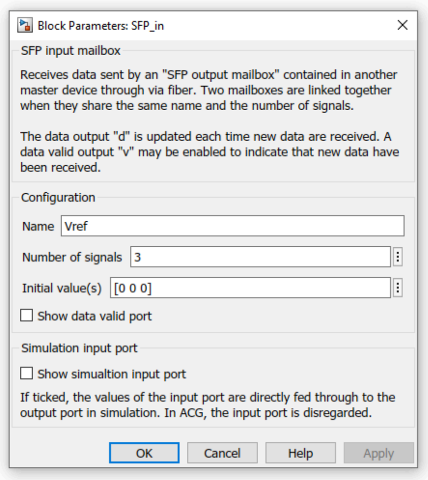
Number of signals: 3

Input trigger: None (send all data)

Simulation output port

□ Show simulation port

If ticked, the values of the input port are directly fed through to the output port in simulation. In ACG, the output port is disregarded.



Block Parameters: SFP_in

SFP input mailbox

Receives data sent by an "SFP output mailbox" contained in another master device through via fiber. Two mailboxes are linked together when they share the same name and the number of signals.

The data output "d" is updated each time new data are received. A data valid output "v" may be enabled to indicate that new data have been received.

Configuration

Name: Vref

Number of signals: 3

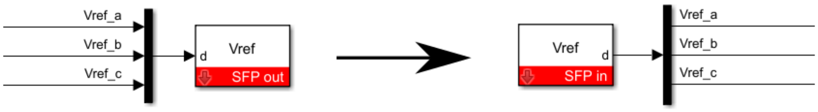
Initial value(s): [0 0 0]

□ Show data valid port

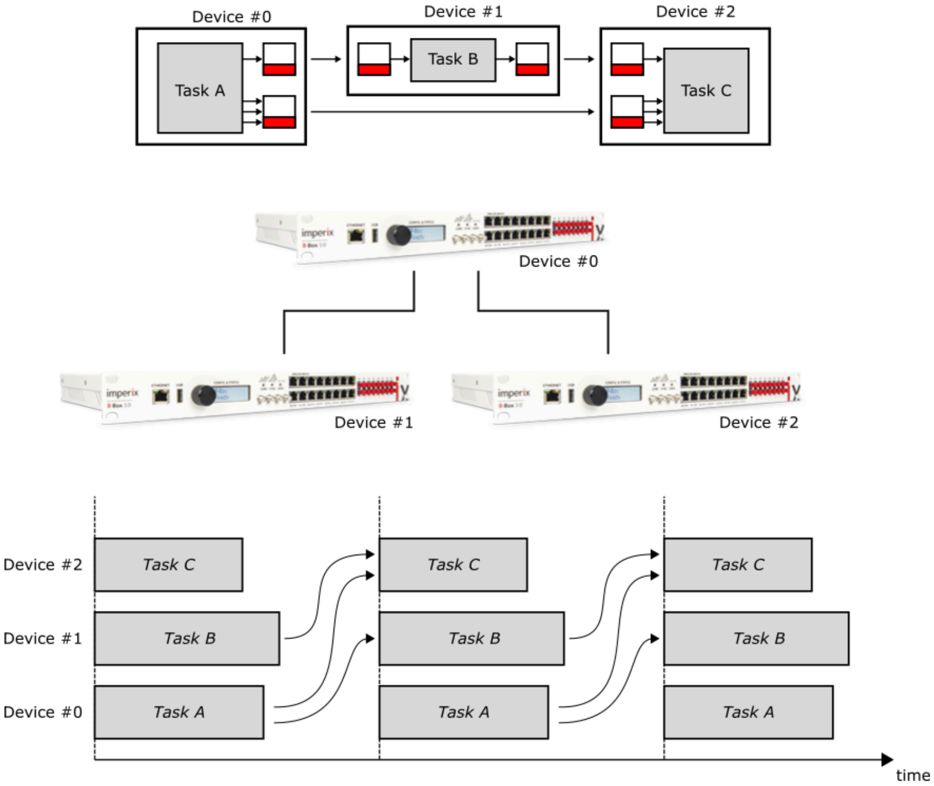
Simulation input port

□ Show simulaltion input port

If ticked, the values of the input port are directly fed through to the output port in simulation. In ACG, the input port is disregarded.



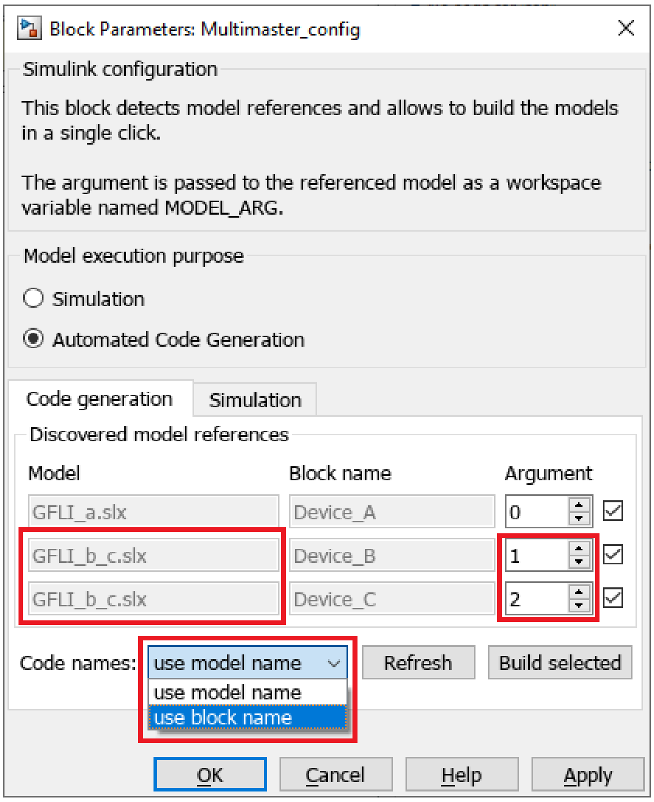
The image below illustrates an example where three control models are running on three B-Boxes and exchanging data using SFP blocks. The control models operate at the same control frequency and share the same sampling phase. Consequently, the sampling occurs at the same time on all devices and the three control tasks start simultaneously. Thanks to the low communication latency (~200 ns per hop), the transferred data arrives before the next task execution, resulting in only one control period delay. Note that the two mailboxes can be located on any device within the network, the devices do not need to be adjacent.



Example of distributed control using 3 B-Boxes

Referencing the same model multiple times

In some applications it might be useful to execute the same model on multiple devices. As illustrated below, it is possible to reference the same Simulink model multiple times. In that case, selecting the *use block name* option is necessary to prevent code name conflicts. The `MODEL_ARG` workspace variable can be used when models need to behave differently. It can also help preventing SFP mailboxes naming conflicts.



Block Parameters: Multimaster_config

Simulink configuration

This block detects model references and allows to build the models in a single click.

The argument is passed to the referenced model as a workspace variable named MODEL_ARG.

Model execution purpose

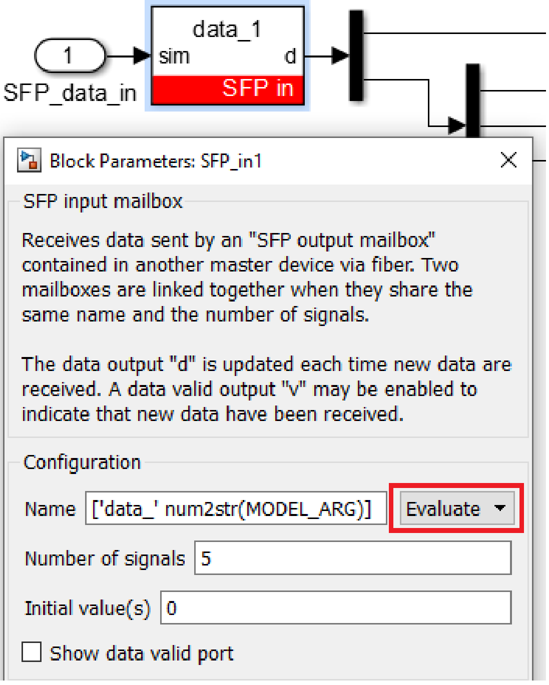
○ Simulation

● Automated Code Generation

Discovered model references

Model	Block name	Argument	
GFLI_a.slx	Device_A	0	☑
GFLI_b_c.slx	Device_B	1	☑
GFLI_b_c.slx	Device_C	2	☑

Code names: use model name



Block Parameters: SFP_in1

SFP input mailbox

Receives data sent by an "SFP output mailbox" contained in another master device via fiber. Two mailboxes are linked together when they share the same name and the number of signals.

The data output "d" is updated each time new data are received. A data valid output "v" may be enabled to indicate that new data have been received.

Configuration

Name ['data_' num2str(MODEL_ARG)] Evaluate

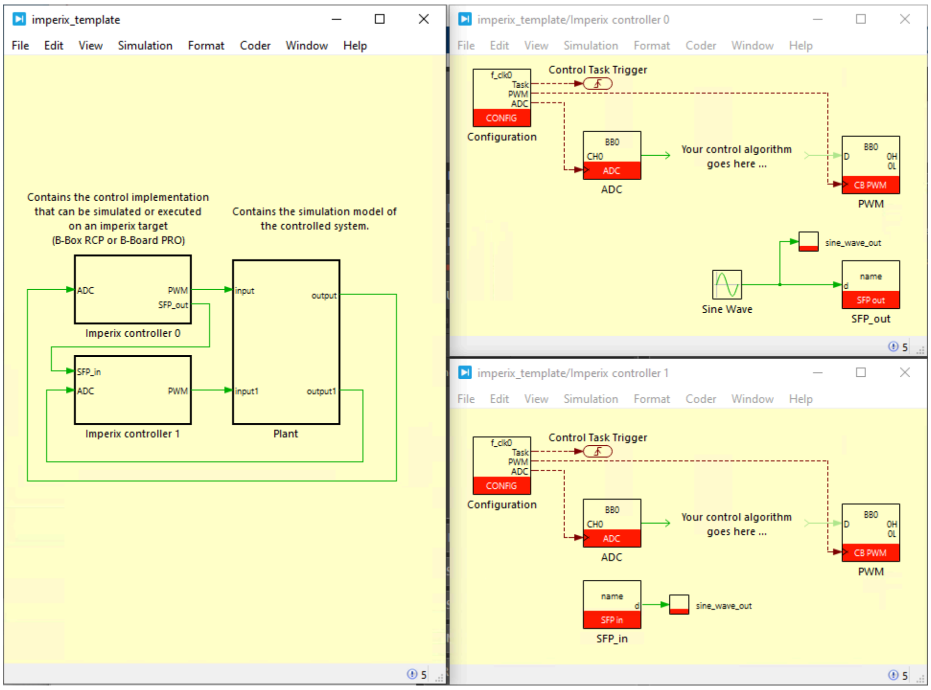
Number of signals 5

Initial value(s) 0

□ Show data valid port

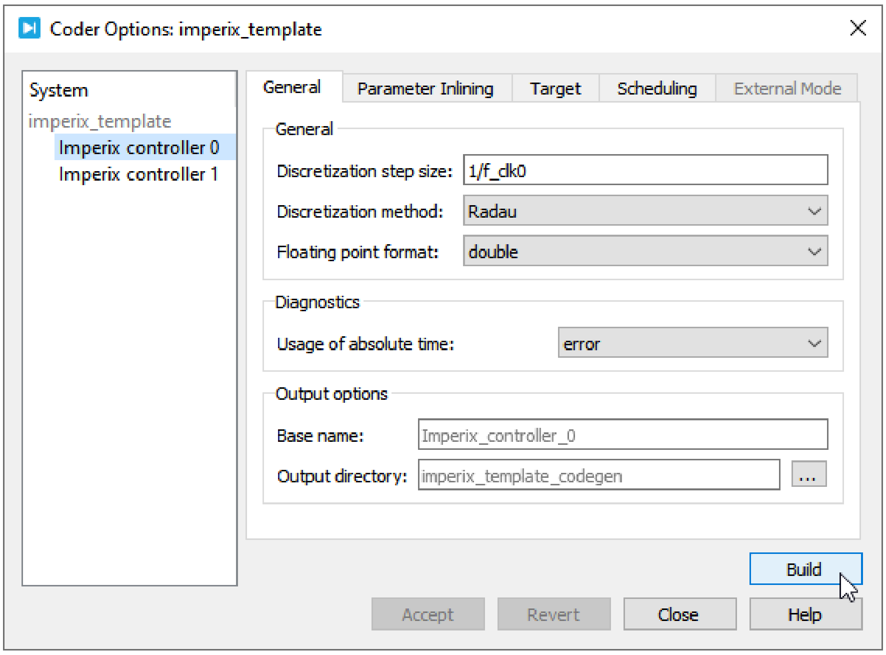
Multi-master using PLECS

Multi-master development using PLECS is more straightforward. The user can start from the provided template and duplicate the *Imperix controller* subsystem.


imperix_template
File Edit View Simulation Format Coder Window Help
Contains the control implementation that can be simulated or executed on an imperix target (B-Box RCP or B-Board PRO)
Contains the simulation model of the controlled system.
ADC
PWM
SFP_out
Imperix controller 0
SFP_in
ADC
PWM
Imperix controller 1
input
output
input1
output1
Plant
imperix_template/Imperix controller 0
Control Task Trigger
CONFIG
Configuration
CH0
ADC
ADC
Your control algorithm goes here ...
CB PWM
PWM
sine_wave_out
Sine Wave
SFP out
SFP_out
imperix_template/Imperix controller 1
Control Task Trigger
CONFIG
Configuration
ADC
Your control algorithm goes here ...
PWM
SFP in
SFP_in
sine_wave_out

PLECS model with 2 imperix controller devices

The *Imperix controller* subsystems are listed on the *Coder Options* page. ([Ctrl+Alt+B] or click on *Coder > Coder Options*). The user must select the subsystem for which they wish to generate CPU code and then click *Build.*



General

Discretization step size: 1/f_clk0

Discretization method: Radau

Floating point format: double

Diagnostics

Usage of absolute time: error

Output options

Base name: Imperix_controller_0

Output directory: imperix_template_codegen

PLECS coder options window with 2 imperix controllers