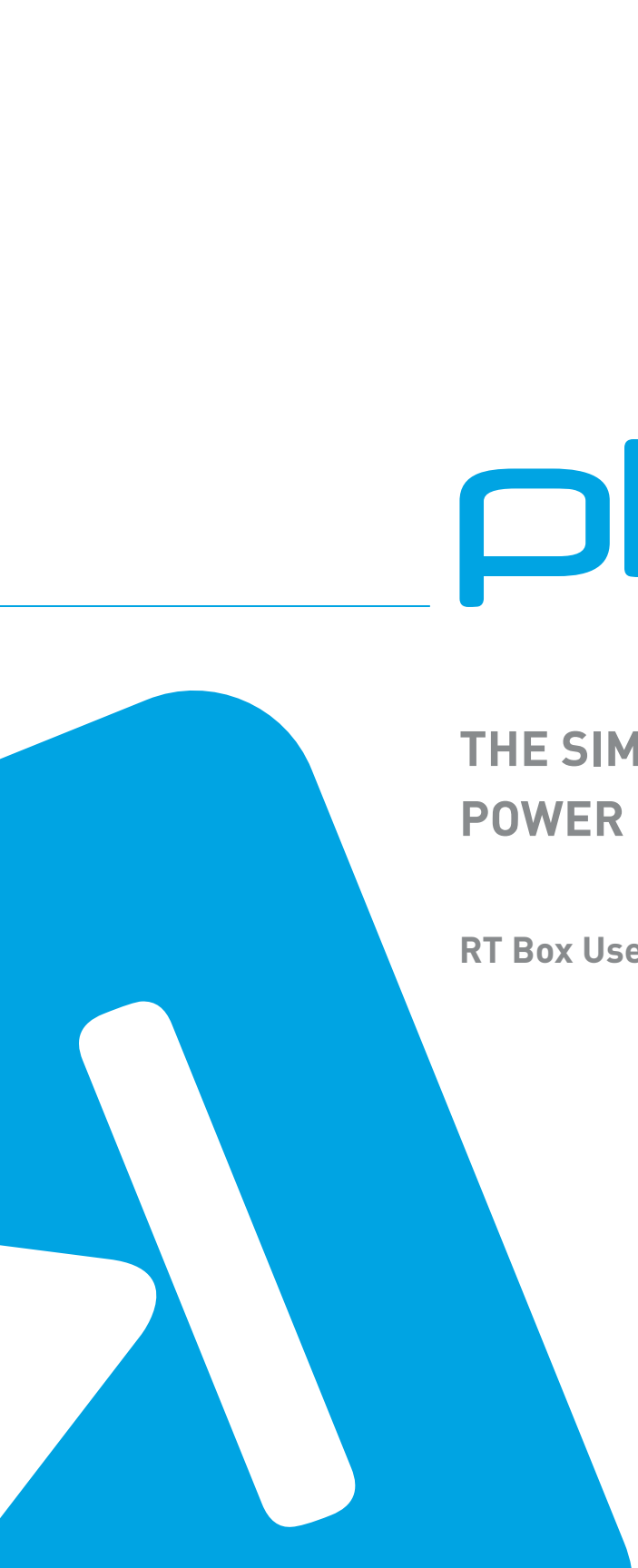


THE SIMULATION PLATFORM FOR POWER ELECTRONIC SYSTEMS

RT Box User Manual April 2018

How to Contact Plexim:

☎	+41 44 533 51 00	Phone
	+41 44 533 51 01	Fax
✉	Plexim GmbH Technoparkstrasse 1 8005 Zurich Switzerland	Mail
@	`info@plexim.com`	Email
	`http://www.plexim.com`	Web

RT Box User Manual

Contents

1

Quick Start

Requirements

In order to operate your RT Box you need

- a host computer (with Microsoft Windows, Mac OS X or Linux),
- PLECS Blockset or Standalone 4.0 or newer and
- PLECS Coder.

If you have not done so yet, please download and install the latest PLECS release on your host computer.

Install the Target Support Package

Download the appropriate ZIP archive from the web page `www.plexim.com/download/rt_box`, extract it and move the folder `PLECS_RT_Box` e.g. to `HOME/Documents/PLECS/CoderTargets`. In PLECS, choose the entry **Preferences...** from the **File** menu (**PLECS** menu on Mac OS X) to open the PLECS Preferences dialog.

On the **Coder** tab click on the **Change** button and select the folder `HOME/Documents/PLECS/CoderTargets`. The RT Box Target should now be listed under **Installed targets**.

Set up Zeroconf

The RT Box uses the *Zero Configuration Networking* (Zeroconf) protocol to facilitate network communication without the aid of additional network servers. To use Zeroconf, the corresponding software components are required on your

host computer (which runs PLECS). It is also possible to use the RT Box without Zeroconf, but this requires additional steps for assigning an IP address, see “Network Configuration” (on page 7).

On Mac OS X, Zeroconf is integrated into the operating system.

On Linux, the avahi daemon and the libdnssd compatibility library must be installed. On Debian/Ubuntu Linux, the corresponding packages are named `avahi-daemon` and `libavahi-compat-libdnssd1`.

On Windows, Zeroconf is available through Apple's Bonjour drivers, which may already be installed on your computer. Alternatively, you can install the appropriate `mDNSInstaller` package that is bundled with the Target Support Package. You find the 32-bit and 64-bit installer packages in the folder `PLECS_RT_Box/bin/win`.

Connect the RT Box

Use an Ethernet cable to connect the RT Box to your local network or – if your host computer supports Zeroconf – directly to an Ethernet port on your host computer.

Turn on the RT Box by flicking the power switch on the rear panel. The green **Power** LED at the front panel will turn on. When the RT Box has acquired a dynamic IP address from a DHCP server on the network, it will turn on the green **Ready** LED to indicate that it is ready to communicate.

Note The **Ready** LED will not turn on if you have connected the box directly with your host computer so that it uses a self-assigned IP address.

To test the network connection, open a web browser on the host computer and enter the address `http://rtbox-`*`0123456789ab`*`.local`, where *0123456789ab* is the MAC address of the RT Box (without colons), which you find on a sticker below the SD card slot on the back.

Upload a Model

In PLECS, choose the entry **Demo Models** from the **Window** menu to open the PLECS Demo Model Library. Open the demo model **Basic Topologies / Boost converter** and save it e.g. as `HOME/Documents/PLECS/Boost-RT.plecs`.

In order to generate code for a complete model, you need to use a fixed-step solver. To do so, choose the entry **Simulation parameters...** from the **Simulation** menu. Choose the **Fixed-step** solver and set the **Fixed step size** parameter to e.g. `1e-5`.

To open the Coder Options dialog, choose the entry **Coder options...** from the **Coder** menu and select the **Target** tab.

Switch the **Target** selector from **Generic** to **PLECS RT Box 1**. If this option is not available, the RT Box Support Package has not been installed correctly (see "Install the Target Support Package" on page 3).

In the **Target device** field, enter the address of your RT Box. If your host computer supports Zeroconf, a button marked **...** appears next to this field. A click on this button opens a browser that shows the RT Boxes that are visible in the network.

Click on the **Build** button at the bottom of the Coder Options dialog to start the build process. This will generate the C code for the model, compile it and upload it to the RT Box. After the successful upload, the blue **Running** LED on the front panel of the RT Box will turn on to indicate that the real-time application is running.

Start the External Mode

Select the **External Mode** tab of the Coder Options dialog and click on the **Connect** button. Set the **Number of samples** parameter to e.g. `200` and click on the **Start autotriggering** button. PLECS will now capture the real-time signals of the Ammeter and Voltmeter in the model running on the RT Box and display them on the Scope in your PLECS model.

You can synchronize the data capture to a specific trigger event. To do so, set the **Trigger channel** selector from **Off** to the desired signal. The Scope will now show a small square indicating the trigger level and delay. If the level or delay are outside the current axes limits, a small triangle will be shown instead. Drag the trigger icon to change the trigger level; drag it with the left mouse-button pressed to change the trigger delay. Both parameters can also be set in the External Mode dialog.

Note While a trigger channel is active, the Scope signals are only updated when a trigger event is detected.

While the PLECS model is connected to the RT Box, the model is locked against modifications. To disconnect from the RT Box, click on the **Disconnect** button or close the Coder Options dialog.

Shutdown

To turn off the RT Box, simply flip the power switch on the rear panel.

Next Steps

To connect the real-time application with real-world I/O signals, use the blocks from the **PLECS RT Box** library in the PLECS Library Browser. For instance, to use a digital input channel as the gate signal for the FET, replace the Pulse Generator with a Digital In block (see page 41). To feed the current and/or voltage measurement back to an external controller via an analog output channel, use an Analog Out block (see page 39) and connect it to the desired meter.

2

Setting up the RT Box

Network Configuration

Communication with the RT Box is done via Ethernet. The box is identified either by its IP address or by a unique hostname. The RT Box supports *Zero Configuration Networking* (Zeroconf) to allow identifying the box by its hostname without any additional network servers (see “Set up Zeroconf” on page 3).

Three networking scenarios are supported by the RT Box:

- Dynamically assigned IP addresses: The RT Box is connected to a larger network and receives an IP address from an external DHCP server. Name resolution is done by an external DNS server or optionally by using Zeroconf.
- Static IP address: A fixed IP address is assigned to the RT Box. The box can be accessed by its IP address. Name resolution is done by an external DNS server or optionally by using Zeroconf.
- Self-assigned IP address: The RT Box is connected directly to a PC with no additional network infrastructure. Both the RT Box and the PC use automatically assigned IP addresses. Address assigning and name resolution is done by Zeroconf, the installation of a Zeroconf driver is mandatory.

Setting the hostname

Since the IP address of the RT Box is typically not known it is more convenient to access the box by its hostname. By default the hostname is set to the MAC address (without colons) with the prefix `rtbox-`, for example `rtbox-20b0f70361c2`. The MAC address is printed on a sticker below the SD card slot on the backside of the RT Box. A custom hostname can be set by creating

a text file named `hostname` in the directory `/config/etc` on the SD card of the RT Box. The file should contain a single line with the custom hostname.

Using dynamically assigned IP addresses (DHCP)

By default the RT Box is configured to use a dynamically assigned IP address from a DHCP server. No additional setup is necessary for this scenario.

After the box has been switched on the green *Ready* LED will turn on when the box has acquired an IP address from the DHCP server and is ready for communication.

If the box was previously configured with a static IP address it can be reconfigured to use dynamic address by removing the file `/config/etc/network/interfaces` from the SD card.

Using static IP addresses

Create a text file with the following structure with your specific addresses (the parameters):

```
auto lo
iface lo inet loopback

auto eth0
# IPv4 addres: all parameters except for 'address' and 'netmask' are optional
iface eth0 inet static
        address 192.168.0.100
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway  192.168.0.1
```

The file must be named `interfaces` and be placed in the directory `/config/etc/network` on the SD card of the RT Box.

Firmware Update

The firmware of the RT Box is stored on a SD Card which is accessible on the back side of the box. The version of the installed firmware is displayed in the *Info* tab in the RT Box web interface.

Follow these steps for updating the firmware:

1. Make sure that the latest version of the target support package is installed on your PC. The target support package is available from the Plexim website.
2. Turn off the RT Box. Remove the SD Card from the slot on the back of the box by pushing it in.
3. Put the card into an SD card reader connected to your PC.
4. Locate the new firmware: Open the directory containing the target support package, then open the folder **PLECS_RT_Box\firmware**. It contains a sub-directory *versionnumber*, e.g. 1.0.0. The firmware files are the two files **image.ub** and **BOOT.BIN** in this directory.
5. Copy the two files **image.ub** and **BOOT.BIN** to the root directory of the SD card.
6. Eject the SD card and remove it from the card reader.
7. Place the SD card back into the slot on the back of the RT Box.
8. Turn on the box and reload the web interface of the RT Box in your web browser. The new firmware version should be displayed in the **Info** tab of the RT Box Web Interface.

3

RT Box Architecture

Front Panel

The front panel of the RT Box comprises 4 connectors for analog and digital inputs and outputs (I/Os). The I/Os need to be connected by the user to the device-under-test (DUT). If the RT Box is employed for HIL simulations, the DUT is the real control hardware and the RT Box emulates the plant. If the RT Box is used for rapid control prototyping, the DUT is the real plant controlled by the RT Box.

All I/O connectors are 37 pin normal-density D subminiature (D-Sub) connectors. Male connectors are used for inputs while female ones are used for outputs.

In addition, the front panel contains 4 status LEDs.

Analog Inputs

The RT Box provides 16 analog input channels. The input voltage range can be set for all channels together to either $\pm 10\,\mathrm{V}$ or $\pm 5\,\mathrm{V}$. Each channel can be individually configured for differential or single-ended measurement.

The analog-to-digital converters (ADCs) inside the RT Box are specified for a maximum sampling rate of $2\,\mathrm{MSPS}$ with no-cycle latency. As the firmware of the RT Box currently limits the cycle time to a minimum of $1\,\mu\mathrm{s}$, a sampling rate of up to $1\,\mathrm{MSPS}$ can be realized.

The analog inputs play an important role when the RT Box is used for rapid control prototyping. To allow instantaneous sampling of the input signals, the analog inputs do not have internal anti-aliasing filters. If the bandwidth of the input signal shall be limited, such filters must be added externally.

Differential measurement

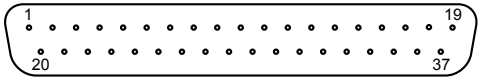
The analog inputs are equipped for fully differential measurement. The measured voltage is the difference between the positive and the negative input. The full-scale differential input range is $\pm 10\,\mathrm{V}$ or $\pm 5\,\mathrm{V}$, depending on the configuration.

A ground connection between the DUT and the RT Box is required even in differential mode, because the inputs cannot float freely with respect to GND. For linear operation, the input voltages referenced to GND shall not exceed $\pm 10.8\,\mathrm{V}$. As a consequence, the acceptable common mode voltage range is $\pm 5.8\,\mathrm{V}$ (for $\pm 10\,\mathrm{V}$ operation) and $\pm 8.3\,\mathrm{V}$ (for $\pm 5\,\mathrm{V}$ operation).

Single-ended measurement

To configure an input channel for single-ended measurement, only the positive input is used for signal measurement while the corresponding negative input is clamped to GND. The full-scale input voltage range is either $\pm 10\,\mathrm{V}$ or $\pm 5\,\mathrm{V}$, depending on the configuration.

Analog Input Connector



37 pin male D-Sub (front view)

Pinout of analog input connector

Pin	Description	Pin	Description
1	Analog input 0 positive	20	Analog input 0 negative
2	Analog input 1 positive	21	Analog input 1 negative
3	Analog input 2 positive	22	Analog input 2 negative
4	Analog input 3 positive	23	Analog input 3 negative
5	Analog input 4 positive	24	Analog input 4 negative
6	Analog input 5 positive	25	Analog input 5 negative

Pinout of analog input connector (contd.)

Pin	Description	Pin	Description
7	Analog input 6 positive	26	Analog input 6 negative
8	Analog input 7 positive	27	Analog input 7 negative
9	Analog input 8 positive	28	Analog input 8 negative
10	Analog input 9 positive	29	Analog input 9 negative
11	Analog input 10 positive	30	Analog input 10 negative
12	Analog input 11 positive	31	Analog input 11 negative
13	Analog input 12 positive	32	Analog input 12 negative
14	Analog input 13 positive	33	Analog input 13 negative
15	Analog input 14 positive	34	Analog input 14 negative
16	Analog input 15 positive	35	Analog input 15 negative
17	n/c	36	n/c
18	n/c	37	GND
19	n/c	Shield	PE

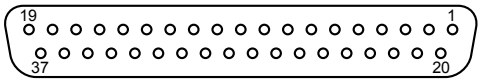
Analog Outputs

The 16 analog output channels of the RT Box are heavily used for HIL simulations. The analog outputs provide the voltage signals from sensors inside the simulated plant and need to be connected to the analog inputs of the controller.

The full-scale voltage range of the outputs can be set to $\pm 10\,\mathrm{V}$, $0 \ldots 10\,\mathrm{V}$, $\pm 5\,\mathrm{V}$ and $0 \ldots 5\,\mathrm{V}$.

For testing purposes, the analog outputs can be connected with the analog inputs of the same or a different RT Box using a 37 pin D-Sub extension cable.

Analog Output Connector



37 pin female D-Sub (front view)

Pinout of analog output connector

Pin	Description	Pin	Description
1	Analog output 0	20	GND
2	Analog output 1	21	GND
3	Analog output 2	22	GND
4	Analog output 3	23	GND
5	Analog output 4	24	GND
6	Analog output 5	25	GND
7	Analog output 6	26	GND
8	Analog output 7	27	GND
9	Analog output 8	28	GND
10	Analog output 9	29	GND
11	Analog output 10	30	GND
12	Analog output 11	31	GND
13	Analog output 12	32	GND
14	Analog output 13	33	GND
15	Analog output 14	34	GND
16	Analog output 15	35	GND
17	n/c	36	n/c
18	n/c	37	GND

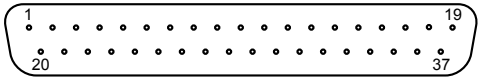
Pinout of analog output connector (contd.)

Pin	Description	Pin	Description
19	n/c	Shield	PE

Digital Inputs

Active bus hold circuitry holds unused or floating data inputs at a valid logic level. If a specific logic level is required for high-impedance inputs, pull-down or pull-up resistors must be connected externally.

Digital Input Connector



37 pin male D-Sub (front view)

Pinout of digital input connector

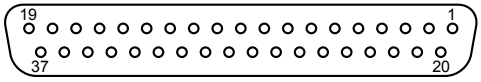
Pin	Description	Pin	Description
1	Digital input 0	20	Digital input 1
2	Digital input 2	21	Digital input 3
3	Digital input 4	22	Digital input 5
4	Digital input 6	23	Digital input 7
5	GND	24	Digital input 8
6	Digital input 9	25	Digital input 10
7	Digital input 11	26	Digital input 12
8	Digital input 13	27	Digital input 14
9	Digital input 15	28	GND

Pinout of digital input connector (contd.)

Pin	Description	Pin	Description
10	Digital input 16	29	Digital input 17
11	Digital input 18	30	Digital input 19
12	Digital input 20	31	Digital input 21
13	Digital input 22	32	Digital input 23
14	GND	33	Digital input 24
15	Digital input 25	34	Digital input 26
16	Digital input 27	35	Digital input 28
17	Digital input 29	36	Digital input 30
18	Digital input 31	37	GND
19	5 V, 1.5 A max.	Shield	PE

Digital Outputs

Digital Output Connector



37 pin female D-Sub (front view)

Pinout of digital output connector

Pin	Description	Pin	Description
1	Digital output 0	20	Digital output 1
2	Digital output 2	21	Digital output 3
3	Digital output 4	22	Digital output 5

Pinout of digital output connector (contd.)

Pin	Description	Pin	Description
4	Digital output 6	23	Digital output 7
5	GND	24	Digital output 8
6	Digital output 9	25	Digital output 10
7	Digital output 11	26	Digital output 12
8	Digital output 13	27	Digital output 14
9	Digital output 15	28	GND
10	Digital output 16	29	Digital output 17
11	Digital output 18	30	Digital output 19
12	Digital output 20	31	Digital output 21
13	Digital output 22	32	Digital output 23
14	GND	33	Digital output 24
15	Digital output 25	34	Digital output 26
16	Digital output 27	35	Digital output 28
17	Digital output 29	36	Digital output 30
18	Digital output 31	37	GND
19	n/c	Shield	PE

Grounding Concept

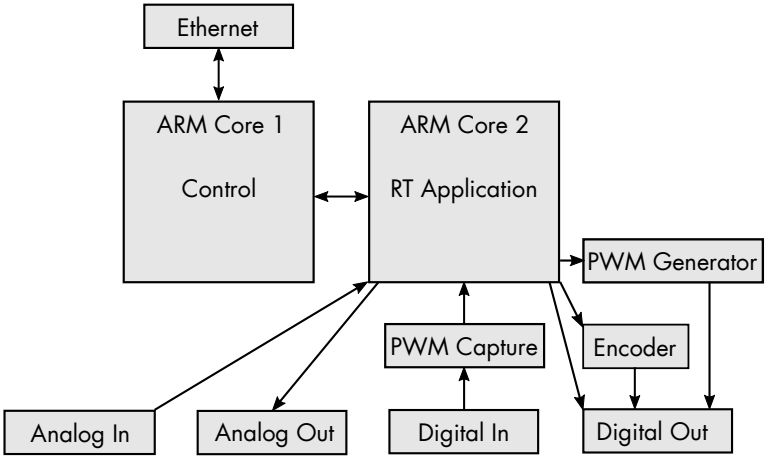
Inside the RT Box, signal Ground (GND) and Protective Earth (PE) are not connected direcly. Instead, GND and PE are loosly coupled by means of a high impedance RC network, consisting of a $1\,\mathrm{M}\Omega$ resistor connected in parallel with a $1\,\mu\mathrm{F}$ capacitor. This configuration avoids ground loops that might otherwise occur if the device-under-test (DUT) used single-ended analog measurement and provided a low-impedance connection between GND and PE.

As the communication lines of both the Ethernet port and the SFP+ modules are AC coupled, multiple RT Boxes can be connected with each other and to a host computer without creating ground loops. At the RT Box, the shields of those lines are connected to PE. The service port is also referenced to PE and isolated from GND of the RT Box.

At the rear panel, only the SD card slot and the USB A receptable are referenced to GND of the RT Box. Mass storage devices such as SD cards and USB memory sticks usually do not have a PE connection, so they can be used at these ports without creating ground loops. However, care should be taken when using externally powered USB devices with a PE connection, such as printers.

Principle of Operation

The basic building blocks of the RT Box are shown in the figure below.



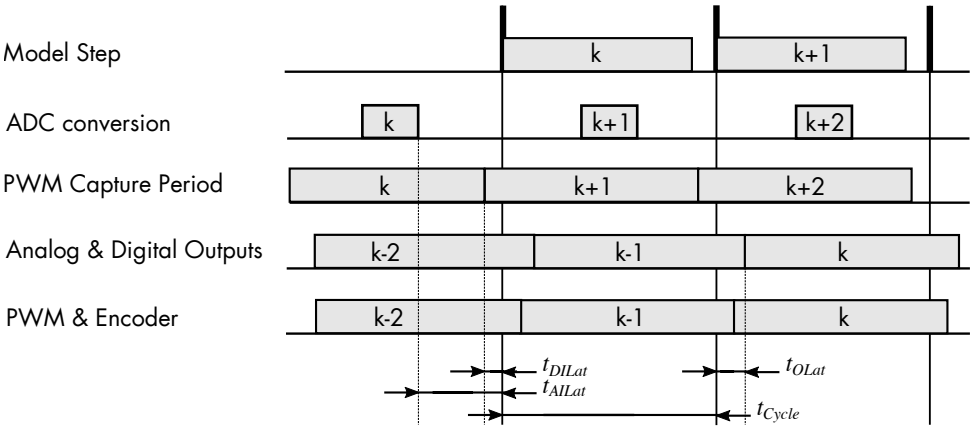
RT Box Overview

The RT Box uses two ARM cores. The first core is responsible for communication, both directly with PLECS or through the web interface of the RT Box.

The second ARM core is dedicated to the real-time application. It has direct access to the different peripheral blocks. Digital Out signals can be set either directly or through the PWM Generator or Encoder unit. Digital Inputs can be sampled by the PWM capture block which allows for high accuracy duty cycle measurement. The Digital Inputs can also be read directly.

Execution of the Real-time Application

After the real-time application has been initialized, its step function is called in fixed periodic intervals, t_{Cycle}. The cycle time must be chosen large enough to ensure that the step function is always executed completely before a new cycle is started.



Input / output timing

4

Specifications

Overview

Processor	Xilinx Zynq Z-7030	1 GHz
Analog inputs	Channels	16, simultaneous sampling
	Resolution	16 bit
	Voltage ranges	-10 ... 10 V -5 ... 5 V
	Input type	Differential
	Sample rate	2 Msps, no cycle latency
	Input impedance	1 MΩ, 24 pF
	Connector	D-Sub 37 pin male
Analog outputs	Channels	16, simultaneous sampling
	Resolution	16 bit
	Voltage ranges	-10 ... 10 V 0 ... 10 V -5 ... 5 V 0 ... 5 V
	Sample rate	2 Msps, no cycle latency
	Output impedance	0 Ω

	Output current	≤ 5 mA
	Connector	D-Sub 37 pin female
Digital inputs	Channels	32
	Logic levels	3.3 V (5 V tolerant)
	High-level input	min. 2 V
	Low-level input	max. 0.8 V
	Connector	D-Sub 37 pin male
Digital outputs	Channels	32
	Logic levels	3.3 V 5 V
	Output impedance	220Ω
	Output current	≤ 10 mA
	Connector	D-Sub 37 pin female
I/O Protection	Short-circuit	Permanent
	Overvoltage	-24 ... 24 V
Connectivity	Ethernet	RJ-45, Gigabit
	High speed interconnect	4 x SFP+ 6.25 Gbps per lane
Power supply	USB device Host PC Firmware Internal	USB 2.0 high speed, type A USB 2.0, type B SD card 100 ... 240 Vac 50 ... 60 Hz 50 VA
Size	D x W x H	310 x 250 x 100 mm

Table 4.1: Technical Specifications of RT Box 1

Mechanical Dimensions

Front Panel Connectors

The four 37 pin D-Sub connectors on the front panel are displaced 19.05 mm (0.75 inch) vertically and 83.82 mm (3.3 inch) horizontally.

Ground Impedance

Inside the RT Box, GND and PE are coupled via a $1\,\mathrm{M}\Omega$ resistor in parallel to a $1\,\mu\mathrm{F}$ capacitor.

RT Box Coder Options

The **Target** page contains code generation options which are specific to the PLECS RT Box.

General

Analog input voltage range The allowed voltage range for analog input signals. The specified range is discretized with a resolution of 16 bit. For best results the smallest matching range should be selected.

Analog output voltage range The voltage range for analog output signals. The specified range is discretized with a resolution of 16 bit. For best results the smallest matching range should be selected.

Digital output voltage level The voltage level for all digital outputs of the RT Box. Possible values are 3.3 V and 5 V.

Enable external mode This setting adds code to use the external mode on the RT Box. Code size and memory consumption are slightly increased when the external mode is enabled.

Overrun limit The maximum number of consecutive overruns that may occur before the simulation is aborted. Increasing this parameter allows the simulation to continue in certain conditions where the calculation time exceeds the model step size for a short period of time, e.g. when sending the first trigger command in external mode.

Interconnect

These options allow to synchronize multiple connected RT Boxes. Please also read section “Time and Startup Synchronization” (on page 29).

Master for startup/clock Specifies on which SFP port the RT Box that is acting as a master is connected. For the top level master or for standalone boxes this option must be set to `Self`.

Synchronize startup with SFP x Enables startup synchronization on the given SFP port. Boxes acting as a master must enable startup synchronization on all ports to which a slave is connected. Boxes acting as a slave must enable startup synchronization on the port to which the master is connected.

Use clock from master Enables simulation time step synchronization. The SFP port to which the RT Box acting as master is connected must be specified in option **Master for startup/clock**. No settings are necessary on the RT Box acting as master.

6

Distributed Realtime Simulation

A simulation model may be split up and run on multiple hardware nodes in parallel. The advantages of this approach are:

- Increased number of I/Os
- Shorter simulation time steps

However, running different parts of a model on different hardware nodes also poses some difficulties:

- The nodes may need to exchange information in every simulation step.
- The I/Os of the different nodes must be synchronized to guarantee consistent electrical states.
- The simulation startup must be synchronized to guarantee consistent model states.

This chapter describes how distributed realtime simulation can be faciliated using multiple RT Boxes with minimal additional modifications to the simulation model.

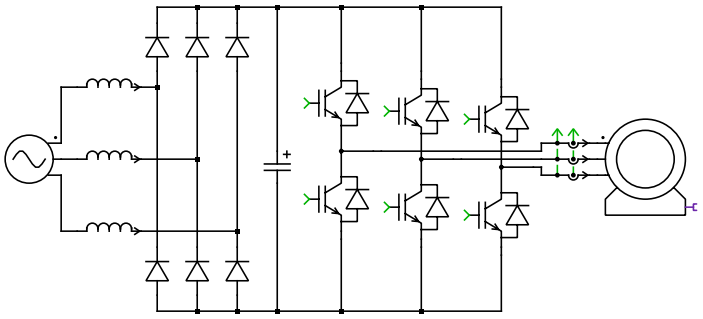
Model Setup for Distributed Realtime Simulation

Identifying Suitable Subsystems

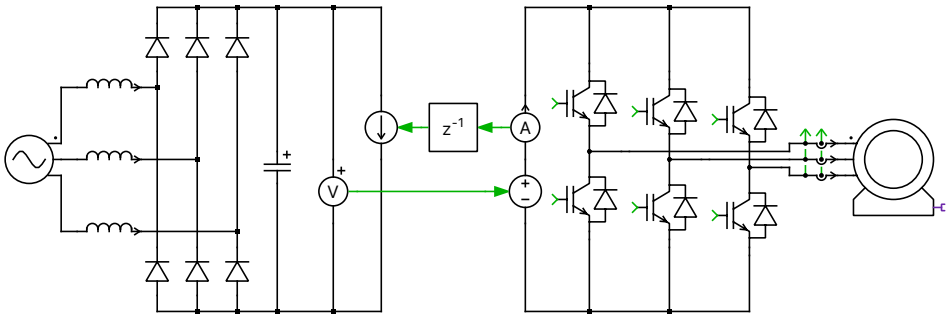
As a first step the simulation model needs to be divided into suitable subsystems for parallel simulation. Obvious candidates for independent subsystems are isolated electrical circuits.

Another common solution is to separate circuit models at a capacitor or inductor that can be replaced by a current source in one part of the model and

a voltage source in the other part. Both sources are controlled by respective measurements in the other part of the model. This approach can often be applied to DC-link capacitors: Note that in an offline simulation a Delay block



Model with DC-link before split-up



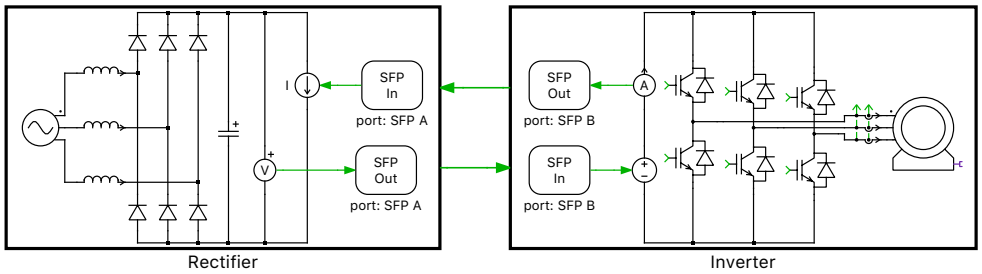
Model with DC-link after split-up

is necessary in at least one direction of communication to prevent algebraic loops.

Exchanging signals between two models

For physical distribution of the model onto different RT Boxes the code for each part of the model must be generated separately. Therefore we need to create a subsystem for each RT Box and place the corresponding part of the model inside it.

The signals between the subsystems are now replaced by SFP blocks. The SFP ports establish a high-speed bidirectional serial communication link between two RT Boxes with a bandwith of 6.25 Mbps. To send signals from a model over a SFP port use the SFP Out block (see page 55). To receive signals from an SFP port and use it in a model use the SFP In block (see page 54). The SFP communication introduces a latency of two simulation timesteps



Model with communication over SFP

(i.e. model B processes the information from model A two timesteps after it was processed by Model A). For offline simulations, these delays are modelled internally in the SFP blocks, so no external delay block is necessary anymore.

In this example, SFP port A is used on the first box running the rectifier and SFP port B is used on the second box running the inverter. This assumes that a SFP cable connects port A on the first box with port B on the second box.

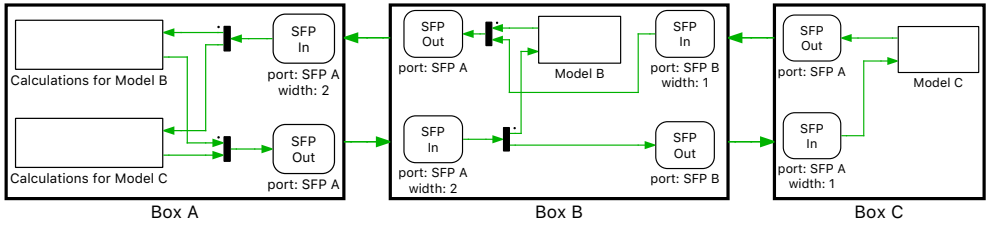
Signal relaying

Signals can be exchanged between any two boxes that have a direct physical SFP connection. To exchange signals between boxes that are connected only via other boxes, signal relaying must be manually provided on the boxes in between:

In the case shown above the latency for a signal from Model A to Model C (and vice versa) is four simulation steps.

Time and Startup Synchronization

In addition to simulation data exchange, the SFP connection allows synchronizing the simulation time steps and the simulation startup between multiple



Signal relaying

RT Boxes. Synchronization requires assigning “master” and “slave” roles to SFP ports. For time step synchronization the clock signal from the master is distributed to a number of slaves. All boxes with time step synchronization enabled must use the same simulation step time.

For startup synchronization the master first waits for all registered slaves to become available (and optionally synchronized with the master clock). Then the master sends a start signal to all slaves simultaneously and also starts its own simulation model.

The assignment of master and slave roles is done in the **Interconnect** settings of the RT Box specific coder options. Please see section “RT Box Coder Options” (on page 25) for further details.

Note that one RT Box can connect to a master on one SFP port and act as a master for further RT Boxes connected to the other SFP ports. This results in a tree-like structure where the clock signal is distributed from the tree root to the subsequent levels. However, you should consider that clock jitter increases with each level that is added to the tree. Configuration settings for four RT-Boxes arranged in a tree with 3 levels are shown in the figure below.

Synchronized versus Unsynchronized Simulation

As stated above, the latency of a signal sent from one box to another box is two simulation steps when the two boxes use synchronized simulation time steps. This is the best solution for distributed simulations where the calculation time for each simulation step is roughly the same on all boxes.

Another common scenario is to use additional RT Boxes for I/O extension only. To minimize I/O latency it may be advisable to not use simulation time step synchronization. The RT Boxes running as I/O extender should run very simple models that only connect incoming SFP blocks to perpheral blocks and vice

Startup and time synchronization in a 3 level tree

versa. These models can typically be run with a step size of just $1\,\mu$s. In this case, the SFP latency is reduced to the step size of the fast I/O extenders plus

the time needed for SFP data transmission (typically well below $1\,\mu$s).

Another scenario where time step synchronization is not recommended is when splitting the model into a fast part (e.g. for the electrical system) and a slow part (e.g. the mechanical system). Both parts can then run on two RT Boxes with different time steps.

7

Scripting

For automated test environments the RT Box can be controlled via external scripts using an XML-RPC interface. XML-RPC is a lightweight protocol for executing functions on a remote machine. The RT Box acts as an XML-RPC server, which processes requests sent from scripts running on another computer. Many scripting languages support XML-RPC out of the box, for example Python or Ruby. For other scripting language, extensions for XML-RPC support are available.

In the following examples, Python 2.x syntax and script excerpts are used.

Establishing an XML-RPC Connection to the RT Box

The RT Box listens to XML-RPC connections on TCP port 9998. The following Python code initiates an XML-RPC connection to the RT Box:

```
import xmlrpclib
import socket
ip = socket.gethostbyname("examplebox.local")
server=xmlrpclib.Server("http://" + ip + ":9998/RPC2")
```

Note that the URL must end with "/RPC2", which is an XML-RPC convention.

Overview of XML-RPC Commands

Commands for PLECS start with `rtbox` followed by a dot. In Python they are invoked on the server object, for example

```
server.rtbox.start()
```

Loading, starting and stopping a real-time simulation

The command

```
rtbox.load(binaryObject)
```

loads a new executable (ELF file) on the RT Box. The parameter *binaryObject* must be a base64 encoded XML-RPC binary object. In Python, this object can be created using the `Binary` object from `xmlrpclib`. The following code shows how to read an ELF executable file and load it on the RT Box using Python:

```
with open("TestModel_codegen/TestModel.elf", "rb") as f:
   server.rtbox.load(xmlrpclib.Binary(f.read()))
```

The command

```
rtbox.start()
```

starts the execution of the real-time simulation.

The command

```
rtbox.stop()
```

stops the execution of the real-time simulation.

Sending data to the real-time simulation

The command

```
rtbox.setProgrammableValue('componentPath', valueList)
```

sets the output of the Programmable Value block (see page 44) indicated by *componentPath*. The parameter *componentPath* is the path of the block relative to the code generation subsystem.The parameter *valueList* must be an XML-RPC array where the number of elements corresponds to the width of the output signal. If the Programmable Value Block has an output signal with a width of 1 a scalar value may be used as parameter. The example code below changes an output of width 2 to the values 5 and 7:

```
rtbox.setProgrammableValue('Value1', [5.0, 7.0])
```

Another example shows setting an output of width 1 to the value 7:

```
rtbox.setProgrammableValue('Value1',[7])
```

or

```
rtbox.setProgrammableValue('TestModel/Value1',7)
```

Getting data from the real-time simulation

The command

```
rtbox.getCaptureData('componentPath')
```

returns the last filled data buffer from the Data Capture block (see page 40) indicated by *componentPath*. The parameter *componentPath* is the path of the block relative to the code generation subsystem.The returned value is a struct with the following fields:

data
The data as a two-dimensional array. The inner dimension corresponds to the width of the input signal, the outer dimension corresponds to the number of samples.

triggerCount
Specifies how many times the sample buffer has been filled.

sampleTime
Specifies the time between two samples, in seconds.

The command

```
rtbox.getCaptureTriggerCount('componentPath')
```

returns how many times the sample buffer of the Data Capture block (see page 40) indicated by *componentPath* has been filled. The parameter *componentPath* is the path of the block relative to the code generation subsystem.

The command

```
rtbox.getDataCaptureBlocks()
```

returns a list of all Data Capture blocks in the current model, with path.

The command

```
rtbox.getProgrammableValueBlocks()
```

returns a list of all Programmable Value blocks in the current model, with path.

Example Script

This Python script loads a new executable on the RT Box and starts it. It sets a value and reads some values back from the model after the corresponding data block has captured sufficient data. The Data Capture block “Capture1” and the Programmable Value block “Value1” must be placed in the root schematic of the model. Finally, the real-time simulation is stopped.

```
import xmlrpclib
import socket
import time
import random

ip = socket.gethostbyname("examplebox.local")
server=xmlrpclib.Server("http://" + ip + ":9998/RPC2")
with open("TestModel_codegen/TestModel.elf", "rb") as f:
   print("Uploading executable")
   server.rtbox.load(xmlrpclib.Binary(f.read()))
f.closed
print("Starting executable")
server.rtbox.start()
print("Simulation running")
val = random.random()
print("Setting value %.7f" % val)
server.rtbox.setProgrammableValue('Value1', [val])
while server.rtbox.getCaptureTriggerCount('Capture1') == 0:
   print("Waiting for data")
   time.sleep(1)

data = server.rtbox.getCaptureData('Capture1')
print("Got value %.7f" % data['data'][0][0])
print("Stopping executable")
server.rtbox.stop()
```

8

RT Box Target Support Library Component Reference

This chapter lists the contents of the RT Box Target Support library in alphabetical order.

Analog In

Purpose

Output the measured voltage at an analog input channel.

Library

PLECS RT Box

Description

The output signal is scalable and can be used with an offset. The output signal is calculated as input*Scale+Offset.

Parameters

Analog input channel(s)
Index of the analog input channel. For vectorized input signals a vector of input channel indices must be specified.

Scale
A scale factor for the input signal.

Offset
An offset for the scaled input signal.

Analog Out

Purpose

Set the output voltage of an analog output channel.

Library

PLECS RT Box

Description

The output voltage is scalable and can be used with an offset. The output voltage is calculated as input*Scale+Offset. Further, an output voltage limitation can be set.

Parameters

Analog output channel(s)
Index of the analog output channel. For vectorized output signals a vector of output channel indices must be specified.

Scale
A scale factor for the output signal.

Offset
An offset for the scaled output signal.

Minimum output voltage
The lowest value that the output voltage can reach.

Maximum output voltage
The highest value that the output voltage can reach.

Data Capture

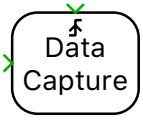
Purpose

Capture data to be transferred via XML-RPC.

Library

PLECS RT Box

Description



The Data Capture block captures simulation data from real-time simulations that are controlled by an external script. The captured data can be queried via the XML-RPC interface of the RT Box.

All captured data is written into an internal buffer first. The size of the buffer is controlled by the parameter "Number of samples". Once the buffer is full, the data can be read via XML-RPC. See "Getting data from the real-time simulation" on page 35 for a description of XML-RPC commands to access the Data Capture block.

The start of data capturing can be controlled with an optional trigger signal.

Parameters

Number of samples
The number of samples that are captured before data is accessible via XML-RPC.

Trigger type
Specifies the edge of the trigger signal that starts data capturing. If set to "continuous" the trigger input is ignored.

Trigger level
Defines the value that the trigger signal must reach to become active.

Digital In

Purpose

Read a digital input.

Library

PLECS RT Box

Description

The output signal is 1 if the input voltage is higher than 2 Volts and 0 if it is lower than 0.8 Volts. For other input voltages the output signal is undefined.

Parameters

Digital input channel(s)

Index of the digital input channel. For vectorized input signals a vector of input channel indices must be specified.

Digital Out

Purpose

Set a digital output.

Library

PLECS RT Box

Description

The output is set low if the input signal is zero, it is set high for all other values. During an offline simulation the block behaves like a simple feedthrough.

Parameters

Digital output channel(s)

Index of the digital output channel. For vectorized output signals a vector of output channel indices must be specified.

Incremental Encoder

Purpose

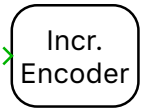
Generate quadrature encoder signals on 2 or 3 digital outputs.

Library

PLECS RT Box

Description

The Incremental Encoder generates quadrature encoder signals [A, B] or [A, B, I] on selected digital outputs of the RT Box. During an offline simulation the pulses are available on the outputs.



Parameters

Encoder module
Specifies used encoder module.

Digital output channels (A, B,) or (A, B, I)
Indices of the digital output channels for [A, B] or [A, B, I].

Line pairs
Defines the amount of pairs of black and white lines and therefore the encoder resolution. For every line pair, 4 state transitions at A, B per revolution can be seen. The index pulse at I is set once per revolution when the angle crosses 0.

Coding in forward direction
Defines the A,B state sequence in forward direction.

Inputs

[ω, ϕ, $\dot{\omega}$]
The rotational speed, angle and speed derivative of the encoder shaft. The latter is optional and can be omitted.

Programmable Value

Purpose

Output values received via the XML-RPC interface.

Library

PLECS RT Box

Description

The output of the Programmable Value block can be set via the XML-RPC interface. See “Sending data to the real-time simulation” on page 34 for a description of XML-RPC commands to access the Programmable Value block.

Parameters

Width
The width of the outgoing signal.

Initial value
The output value of the block before data has been received via XML-RPC.

PWM Capture

Purpose

Averages a digital input over the period of a model step.

Library

PLECS RT Box

Description

The PWM Capture output represents the percentage of time during which a digital input signal was active over the last model step period. The PWM capture active polarity can be set channel wise. During an offline simulation the percentage time in the active state within the last interval is calculated.

Parameters

Digital input channel(s)
Index of the digital input channel. For vectorized input signals a vector of input channel indices must be specified.

Channel(s) active polarity
Active level of a digital input. For vectorized input signals either a scalar or a vector with the length of the input channels must be specified.

Averaging interval (offline only)
Defines the averaging interval in seconds for offline simulations.

PWM Out

Purpose

Generate a PWM signal on a digital output.

Library

PLECS RT Box

Description

The PWM Out generates a configurable PWM signal on a digital output of the RT Box. The modulation index for each channel must be provided via the input signal, which is a vectorized signal if the block uses multiple channels. During an offline simulation it behaves as a normal PWM generation block.

Parameters

Digital output channel(s)
Index of the digital output channel. For vectorized output signals a vector of output channel indices must be specified.

Carrier type
Selects the carrier waveform, either sawtooth or symmetrical.

Carrier frequency
The frequency of the carrier in hertz (Hz).

Carrier phase shift
The phase shift of the carrier signal, in p.u. of the carrier period. The parameter can be either a scalar or a vector with the same width as the number of output channels.

Carrier limits
The range of the carrier signal. The default is `[-1 1]`.

Turn-on delay
Turn-on delay of the PWM output in seconds.

Polarity
The polarity of the PWM output determines whether the PWM signal is in on-state (value 1) or off-state (value 0) when the modulation index exceeds the carrier. The parameter can be either a scalar or a vector with the same width as the number of output channels.

Update
If “On carrier minimum”, “On carrier maximum” or “On carrier minimum/maximum” is selected the modulation index will be sampled when the carrier reaches its minimum or maximum values. If “Immediately” is selected the modulation index will be updated after each simulation step of the RT Box.

Synchronization with model step

If “Enabled” is selected and the discretization step size is an integer multiple of the PWM period, the PWM generation will be synchronized with the simulation steps of the RT Box.

PWM Out (Variable)

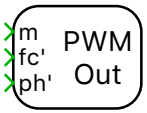
Purpose

Generate PWM signals with variable frequency and variable phase-shift.

Library

PLECS RT Box

Description



The PWM Out (Variable) block generates PWM signals on one or multiple digital output channels of the RT Box. The modulation index for each channel must be provided via the input signal m, which is a vectorized signal if the block uses multiple channels.

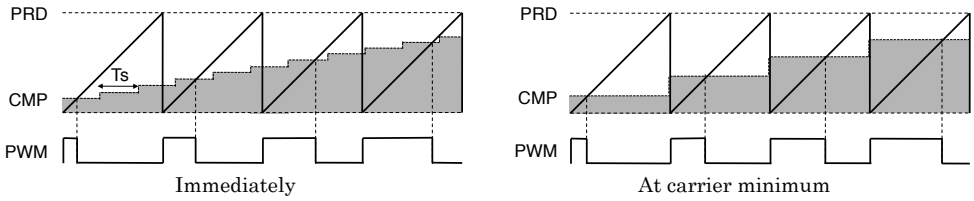
The carrier frequency f_c is common to all channels of the same block. It can be controlled during the simulation using the scalar input signal f'_c. The resulting carrier frequency f_c is calculated as the product of the nominal carrier frequency specified in the block parameters and the input signal f'_c. For a constant carrier frequency, a constant value 1 must be fed into the block. If the PWM generation is synchronized to the simulation steps of the RT Box, the carrier frequency is rounded to the nearest integer multiple of the simulation frequency.

The phase shift between the carriers of the individual PWM channels can be controlled with the vectorized input signal ph'. Each element of ph' specifies the phase delay of the PWM carrier in the corresponding channel. The delay is given in p.u. of the carrier period and must lie between 0 and 1.

To understand the PWM generation for variable frequency and variable phase-shift, an overview of the internal implementation is given:

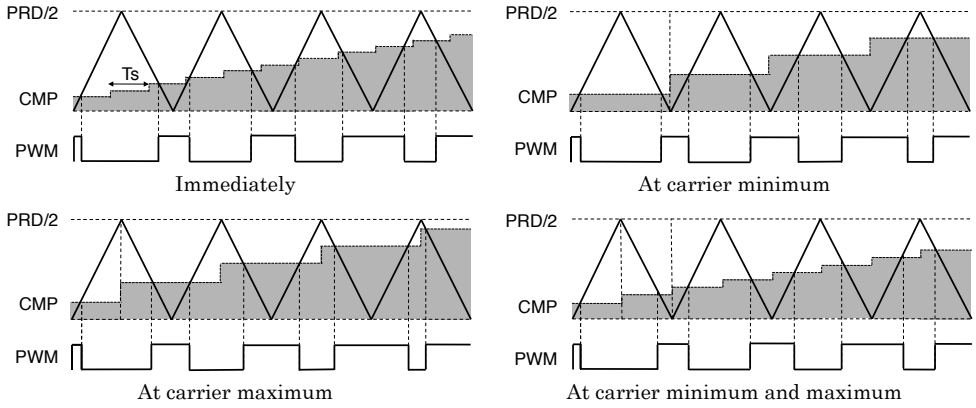
Each PWM channel has its own a ramp generator for a carrier signal. In case of a sawtooth carrier, the carrier signal ramps up with a constant slope from its initial value 0 to reach the value PRD at the end of the PWM period. When it reaches PRD, the carrier is instantaneously reset to 0. The period PRD is computed as the inverse of the current carrier frequency f_c. In case of a symmetrical carrier, the rising ramp reaches its maximum value PRD/2 after half the PWM period and the carrier then ramps down to reach 0 at the end of the PWM period.

The modulation index for each channel is mapped from its original input range defined by the carrier limits to a compare value CMP. CMP has a minimum value of 0 and a maximum value of PRD for sawtooth carriers or PRD/2 for symmetrical carriers. If the parameter **Polarity** of a channel is set to 0 its PWM output becomes high whenever CMP is greater than the carrier signal.



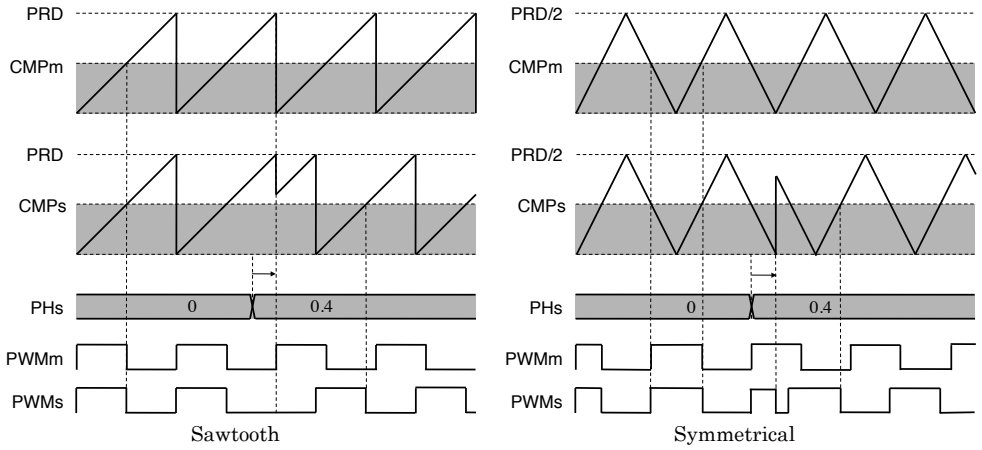
Update schemes for modulation index with sawtooth carrier

The user can choose how often the CMP value is updated. If the parameter **Update** is set to “Immediately”, CMP will be updated asynchronously whenever a new modulation index has been computed in a model step. This can be helpful if the simulation step size is much shorter than the PWM period. For updating CMP synchronously after every complete PWM period, the parameter **Update** must be set to “At carrier minimum”. In case of a symmetrical carrier, there are two additional options available for synchronously updating CMP: If the parameter **Update** is set to “At carrier maximum”, CMP will be refreshed in the middle of the PWM period at the tips of the carrier. If set to “At carrier minimum and maximum” the update will be performed twice in every PWM period.



Update schemes for modulation index with symmetrical carrier

If a PWM Out (Variable) block contains multiple output channels, the first channel automatically becomes a master while the other channels are configured as slaves. Every time the ramp generator of the master becomes 0, the

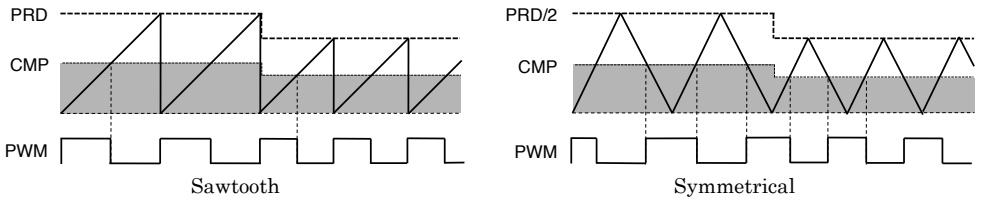


Phase-shift variation with different carrier types

master transmits a synchronization signal to the slaves of the same block. When this happens, the ramp generators of the slaves are set to their initial values computed from the input signal ph' to achieve the desired phase shift.

The first element of the input signal ph' corresponds to the phase delay of the master channel. The phase delays between multiple PWM Out (Variable) blocks are only defined if the blocks have the same **Nominal carrier frequency**, share a constant frequency input signal f_c' and are synchronized with the simulation steps of the RT Box.

With the synchronization signal from the master, the PWM period is updated as well. A new value of PRD is computed from the carrier frequency and applied to all channels. A change in PRD will change the compare value CMP in order to obtain the desired modulation index.



Carrier frequency variation with different carrier types

Parameters

Digital output channel(s)
Index of the digital output channel. For vectorized output signals a vector of output channel indices must be specified.

Carrier type
Selects the carrier waveform, either sawtooth or symmetrical.

Nominal carrier frequency
The nominal frequency of the carrier in hertz (Hz). The actual PWM frequency is equal to the nominal carrier frequency multiplied by the input signal f_c'.

Carrier limits
The range of the carrier signal. The default is `[-1 1]`.

Turn-on delay
Turn-on delay of the PWM output in seconds.

Polarity
The polarity of the PWM output determines whether the PWM signal is in on-state (value 1) or off-state (value 0) when the modulation index exceeds the carrier. The parameter can be either a scalar or a vector with the same width as the number of output channels.

Update
If “On carrier minimum”, “On carrier maximum” or “On carrier minimum/maximum” is selected the modulation index will be sampled when the carrier reaches its minimum or maximum values. If “Immediately” is selected the modulation index will be updated after each simulation step of the RT Box.

Synchronization with model step
If “Enabled” is selected and the discretization step size is an integer multiple of the PWM period, the PWM generation will be synchronized with the simulation steps of the RT Box.

Quadrature Encoder Counter

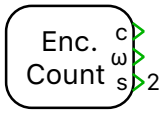
Purpose

Counts edges generated by a quadrature encoder.

Library

PLECS RT Box

Description



Counter1

The Quadrature Encoder Counter counts edges which were generated from a quadrature encoder. The A and B outputs of the encoder are connected to digital inputs on the RT-Box. Optionally, an index signal I and differential input signals $\bar{A}$, $\bar{B}$ and $\bar{I}$ can be connected.

The block outputs the current counter value (c), the current velocity in rad/s (ω) and status information (s). The status information output provides a direction signal and a signal indicating whether an index pulse was received during the last simulation step.

The counter counts up or down depending on the sequence of input pulses. The parameter **Direction (rising edge)** specifies the sequence of rising edges which results in incrementing counter values.

Once the counter reaches the value specified in parameter **Maximum counter value** it is reset to zero on the next detected edge in positive direction. Vice versa, the counter is set to **Maximum counter value** when it is zero and detects an edge in negative direction. If connected and configured, the counter is also reset when a positive edge of the index input is detected.

Parameters

Counter module
Specifies which of the two internal counter modules should be used.

Maximum counter value
The counter is reset to zero when it has reached the **Maximum counter value** and detects an input edge in positive direction. The counter is set to the Maximum counter value when it is zero and detects an input edge in negative direction.

For correct velocity output, the **Maximum counter value** must match the number of line pairs of the encoder multiplied by the number of counted edges per line pair (see parameter **Counter mode** below) minus 1. If, for example, the encoder has 1024 line pairs and all edges of A and B are counted, the resulting value is 4095.

Counter mode
Selects which edges of the input signal are counted.

Direction (rising edge)
Selects the sequence of input pulses for incrementing the counter.

Counter reset method
Selects whether the counter should be reset by a positive pulse on the index input or on overflow only.

Input type
Selects whether the input signals are differential or single-ended.

Index input
If the index input is enabled, the counter is reset when an index pulse is detected.

Digital input for signal [A /A]
Indices of the digital inputs for signal A. If the signal is differential the value must be a vector with two elements where the second element specifies the input for the differential signal $\bar{A}$. For single-ended inputs the value must be a scalar or a vector with one or two elements. The second element is ignored in the latter case.

Digital input for signal [B /B]
Indices of the digital inputs for signal B. If the signal is differential the value must be a vector with two elements where the second element specifies the input for the differential signal $\bar{B}$. For single-ended inputs the value must be a scalar or a vector with one or two elements. The second element is ignored in the latter case.

Digital input for index signal [I /I]
Indices of the digital inputs for signal I. If the signal is differential the value must be a vector with two elements where the second element specifies the input for the differential signal $\bar{I}$. For single-ended inputs the value must be a scalar or a vector with one or two elements. The second element is ignored in the latter case. The parameter value is ignored if the index input is disabled.

SFP In

Purpose

Receives signals from another RT Box

Library

PLECS RT Box

Description

For distributed models, the SFP In block receives signals from an SFP Out block (see page 55) on a remote RT Box. In each simulation step the block provides the last values received by SFP in the previous simulation step. The parameter **Initial value** specifies which signal values to output when no valid values have been received by SFP yet.

See “Distributed Realtime Simulation” on page 27 for a detailed discussion of SFP communication.

Parameters

SFP port
The name of the SFP port which is connected to the remote RT Box.

Width
The width of the incoming signal. This parameter must match the width of the signal that is connected to the corresponding SFP Out block on the remote RT Box.

Initial value
The output value of the block before a valid signal has been received from the remote RT Box.

Sample time (offline only)
For offline simulation the incoming signal is delayed by one sample to match the realtime behavior.

SFP Out

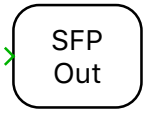
Purpose

Sends signals to another RT Box

Library

PLECS RT Box

Description



For distributed models, the SFP Out block sends the signals at its input terminal to an SFP In block (see page 54) on a remote RT Box. The data transfer takes place in the subsequent simulation step. The number of signals to send is calculated from the width of the incoming signal.

Parameters

SFP port

The name of the SFP port which is connected to the remote RT Box.

Sample time (offline only)

For offline simulation the outgoing signal is delayed by one sample to match the realtime behavior.

SPI Master

Purpose

Implement SPI communication via digital outputs/inputs.

SPI allows synchronous communication with peripheral devices. The SPI Master block provides a clock signal which generates a configurable number of clock pulses during each simulation step. Data is read and written on the edges of the clock signal. For high-speed communication, the clock output signal can be looped back to an input to provide a clock input signal which is synchronized to the input data.

The block can output data on up to four data lines in parallel (using a common clock).

Library

PLECS RT Box

Description

SPI allows synchronous communication with peripheral devices. The SPI Master block provides a clock signal which generates a configurable number of clock pulses during each simulation step. Data is read and written on the edges of the clock signal. The chip select signal is held low during the data transfer. For high-speed communication, the clock output signal can be looped back to an input to provide a skew-matched clock input which is synchronized to the input data.

The block can output data on up to four data lines in parallel (using a common clock and chip select signal).

An output value of 1 at the **valid** port indicates that the appropriate number of clock edges has been detected on the feedback clock input and valid data is present on the other output ports. If the number of detected clock edges does not match the output value at the **valid** port is set to 0 and the other output ports are not updated, i.e. they retain their values from the previous simulation step. If the skew-matched clock input is disabled, the output of the **valid** port is always 1.

Parameters

Setup

SPI module
Specifies which SPI module to use.

SPI clock frequency [Hz]
Clock frequency used for data transmission [20e3 ... 40e6]. Each clock phase is rounded to the nearest integer multiple of 4ns.

Delay first clock pulse after CS active [ticks]
Number of SPI clock periods between CS edge and first clock edge.

Hold CS active after last clock pulse [ticks]
Number of SPI clock periods between last clock edge and CS edge.

Delay CS after simulation step
Defines the begin of the SPI transmission within a simulation step. If "Minimum" is selected the transmission starts immediately when data is available to minimize the output data latency. For "Maximum" the transmission delay is computed to minimize the input data latency. "Specify delay time" is used to adjust the delay manually.

CS delay time [s]
Time for manual delay.

Mode [CPOL, CPHA]
SPI mode as a combination of clock polarity (CPOL) and clock phase (CPHA). CPOL controls whether the clock signal is high (1) or low (0) when idle. CPHA controls whether data is shifted in and out on the rising or falling edge of the clock signal. The following table summarizes the combination of clock polarity and phase:

SPI Modes

Mode	CPOL	CPHA	Output Edge	Data Capture
SPI_MODE0	0	0	Falling	Rising
SPI_MODE1	0	1	Rising	Falling
SPI_MODE2	1	0	Rising	Falling
SPI_MODE3	1	1	Falling	Rising

Bits per word
Length of a single data word during transmission. Depending on this setting the input signals are interpreted as either uint8 or uint16.

Bit order
Defines if LSB or the MSB is transmitted first.

Skew-matched clock input
Enables the clock input for clock feedback. Typically required if the signal delay from the master output to its input exceeds half a SPI clock period.

If it exceeds a full period the parameter “Hold CS active after last clock pulse [ticks]” should also be increased.

Number of parallel data channels
Specifies how many of the four available data channels are actually being used.

Words per transmission (up to 127)
Configures number of transmitted data words in each simulation step.

Input

Digital input channel for skew-matched SPI clock
Clock input for clock feedback.

Digital input channel(s) for SPI data
Data input channel/s as a scalar or vector. The width of this parameter corresponds to “Number of parallel data channels”. “-1” denotes an unused output.

Output

Digital output channel for SPI clock
Clock output channel. “-1” denotes an unused output.

Digital output channel for CS
Chip select output channel. “-1” denotes an unused output.

Digital output channel(s) for SPI data
Data output channel/s as a scalar or vector. The width of this parameter corresponds to “Number of parallel data channels”. “-1” denotes an unused output.

plexim

electrical engineering software

Plexim GmbH info@plexim.com www.plexim.com