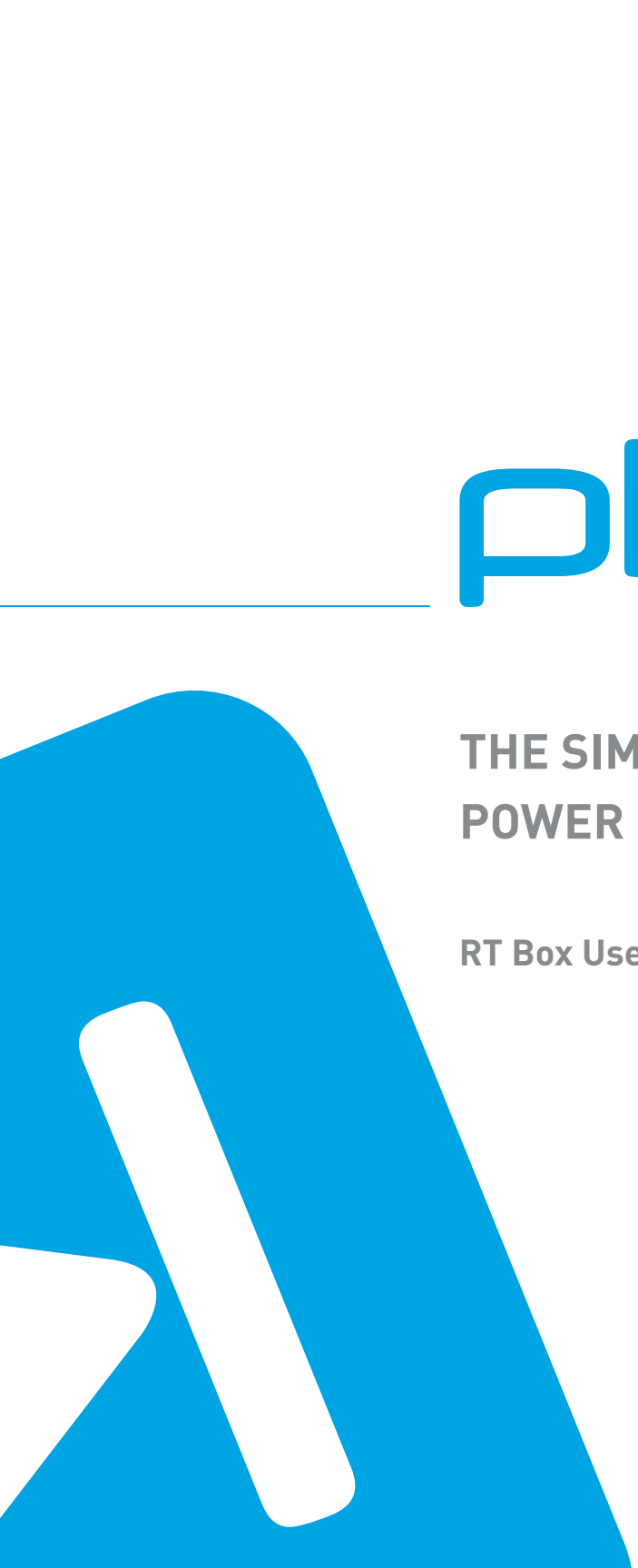


THE SIMULATION PLATFORM FOR POWER ELECTRONIC SYSTEMS

RT Box User Manual Version 4.1

How to Contact Plexim:

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

RT Box User Manual

© 2016–2026 by Plexim GmbH

The product described in this manual is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from Plexim GmbH.

PLECS and Nanostep are a registered trademark of Plexim GmbH. MATLAB, Simulink and Simulink Coder are registered trademarks of The MathWorks, Inc. Bonjour is a registered trademark of Apple, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.

Contents

Contents **iii**

What's New in Target Support Package Version 4.1 **1**

Major New Features 1

Enhanced Library Components 1

1 Quick Start **3**

Requirements 3

Install the Target Support Package 3

Set up Zeroconf 3

Connect the RT Box 4

Upload a Model 5

Start the External Mode 5

Shutdown 6

Next Steps 6

2 Setting up the RT Box **7**

Network Configuration 7

Setting the hostname 7

Using dynamically assigned IP addresses (DHCP) 8

Using static IP addresses 8

Network ports used by the RT Box 9

Firmware Update 9
Automatic Firmware Update 10
Manual Firmware Update 11
Automatic Start of an Executable 11

3 RT Box Architecture 13
Front Panel 13
Analog Inputs 13
Analog Outputs 16
Digital Inputs 17
Digital Outputs 19
Resolver 20
Nanostep Out 21
Rear Panel 24
CAN 24
UART 24
Displayport 25
Industrial Ethernet 26
Grounding Concept 26
Principle of Operation 27
Execution of the Real-time Application 27
Unsafe Math Operations 27

4 Specifications 29
Overview 29
Mechanical Dimensions 32
RT Box CE 32
RT Box 1 33
RT Box 2 33
RT Box 3 34
RT Box 4 35

5 RT Box Coder Options **37**
General 37
Interconnect 39
CAN 39

6 Running Simulations on the FPGA **41**
Nanostep solver 41
FlexArray solver 42
Enabling the FlexArray solver 42
Prerequisites 42
Minimizing the FlexArray solver step size 43

7 Distributed Realtime Simulation **45**
Model Setup for Distributed Realtime Simulation 45
Identifying Suitable Subsystems 45
Exchanging signals between two models 46
Signal relaying 47
Time and Startup Synchronization 47
Synchronized versus Unsynchronized Simulation 48

8 Scripting **51**
Establishing an XML/JSON-RPC Connection to the RT Box 51
Overview of XML-RPC Commands 52
Loading, starting and stopping a real-time simulation 52
Sending data to the real-time simulation 53
Getting data from the real-time simulation 54
Example Script 55

9 RT Box Target Support Library Component Reference **57**
Analog In 58
Analog Out 60
CAN Receive 61

CAN Transmit 63
Data Capture 65
Digital In 66
Digital Out 67
Digital Out Override 68
EnDat Sensor 69
Incremental Encoder 72
Nanostep Out 74
Powerstage Protection Unit 76
Programmable Value 78
PWM Capture 79
PWM Out 80
PWM Out (Variable) 83
Quadrature Encoder Counter 91
Resolver In 93
Resolver Out 95
SFP In 97
SFP Out 98
Sigma-Delta Modulator 99
SPI Controller 101
SPI Peripheral 104
To File 107
UDP Receive 109
UDP Send 110
Wall Clock 111

What's New in Target Support Package Version 4.1

Major New Features

- CAN FD is now supported on compatible external USB devices. See "CAN" (p. 39) for details.
- The accuracy of the FlexArray solver has been increased, allowing more models to be simulated on the FPGA at small step sizes.
- The XCP parameters now include a 'Decimation' parameter. See "Enable XCP" (p. 38) for details.

Enhanced Library Components

- An Analog In block (p. 58) that is triggered by a PWM block is now also triggered if a Powerstage Protection Unit block (p. 76) disables the output of the PWM block.

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 4.5 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

The following steps will download and install the newest version of the RT Box target support package.

1. Open the PLECS Preferences dialog by choosing **Preferences...** from the **File** menu (or the **PLECS** menu on macOS) and switch to the **Coder** page.
2. Check that the **Target Support Packages Path** has been configured.
3. In the table **Installed Targets** click on the cloud icon next to the entry **PLECS RT Box (Preview)** and then on **Install** (see Fig. 1.1 on p. 4).

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

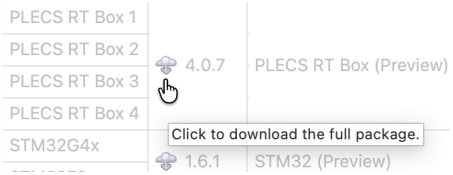


Fig. 1.1: Installing the RT Box Target Support Package

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” (p. 7).

On macOS, 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 the complete model, open the Coder Options dialog by choosing the entry **Coder options...** from the **Coder** menu. In the **General** tab set the **Discretization step size** parameter to e.g. `1e-5`.

Now select the **Target** tab and switch the **Target** selector from **Generic** to either `PLECS RT Box 1`, `PLECS RT Box 2`, `PLECS RT Box 3` or `PLECS RT Box 4` depending on which target the model should later run. If this option is not available, the RT Box Support Package has not been installed correctly (see "Install the Target Support Package" (p. 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 **Activate 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 (p. 66). To feed the current and/or voltage measurement back to an external controller via an analog output channel, use an Analog Out block (p. 60) 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” (p. 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

Follow these steps for configuring the RT Box using static IP addresses:

1. Choose a suitable private IP subnet. For example, enter `192.168.0.3` for the RT Box and choose `192.168.0.2` for the host computer.
2. 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 192.168.0.3
    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.

3. Plug the ethernet cable of the RT Box to either an ethernet port or an ethernet adapter of the host computer and boot the RT Box.
4. For Windows:
   - Open the “Network and Sharing Center” in the Windows Control Panel and click on the appropriate connection.
   - In the “Local Area Connection x Status” window, click on **Properties**.

- In the “Local Area Connection x Properties” window, double-click on **Internet Protocol Version 4 (TCP/IPv4)**.
- In the “Internet Protocol Version 4 (TCP/IPv4) Properties” window, configure the static IP address of the host computer. For example, enter `192.168.0.2` as the “IP address” and `255.255.255.0` as the “Subnet mask”.

5 For Mac:

- “Open Network Preferences” and select the appropriate interface with a new “Service Name”.
- “Manually” configure the IPv4 of the host computer. For example, enter `192.168.0.2` as the “IP address” and `255.255.255.0` as the “Subnet mask”.

6 The RT Box should now be listed when clicking the Browse RT Boxes button. The RT Box can be targeted by entering the chosen static IP address of the RT Box, for example, enter `192.168.0.3` or the hostname `rtbox-xyz.local` as the *Target Device* in the Coder Options dialog.

Network ports used by the RT Box

If the RT Box is to be operated behind a firewall the following TCP ports need to opened to be able access the RT Box through the firewall:

Table 2.1: Network ports used by the RT Box

TCP Port	Purpose
80	Web interface
9998	RPC
9999	External mode

Additionally all UDP ports used in UDP Send or UDP Receive blocks may have to be forwarded.

Firmware Update

Users can update the RT Box firmware either automatically or manually.

Automatic Firmware Update

The automatic firmware update is done via PLECS.

1. Make sure that the latest version of the target support package is installed on your PC. Meanwhile the firmware on the SD card inserted in your RT Box is still at a lower version.
2. Turn on the RT Box and make sure the ethernet cable connection between the RT Box and your PC is working.
3. Create a new PLECS model.
4. In PLECS Coder Options window, in the **Target** field of the **Target** tab, choose your RT Box type.
5. Next in the **Target device** field, click the binoculars icon to browse available RT Boxes. Select your box and you will see with an exclamation mark inside a yellow triangle, if there is an unmatch between patch versions (e.g. version *3.1.3* and *3.1.2*). In case of an unmatch between major versions (e.g. version *3.1.3* and *3.0.6*), you will see the exclamation mark inside a red circle, instead of a yellow triangle.
6. Click on the exclamation mark. Click Yes to update automatically. See the screenshot below as an example.

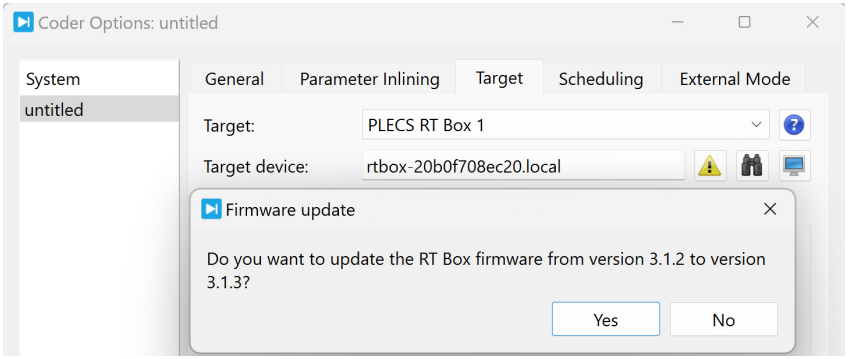


Fig. 2.1: Confirmation dialog when updating firmware

7. Wait for a while when the firmware on the box is automatically updated and then the box is rebooting.
8. Once the box is rebooted, you can close this window. The automatic update is

finished. The exclamation mark no long appears since the two versions are matching now.

Manual 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[1].
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. 1.2.0, with another sub-directory **rtbox1** (for RT Box 1 and CE) and **rtbox2** (for RT Box 2, 3 and 4). The firmware consists of three files: **image.ub**, **BOOT.BIN** and **uboot.env**.
5. Copy the files **image.ub**, **BOOT.BIN** and **uboot.env** 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.

Automatic Start of an Executable

It is possible to automatically run an executable directly after the startup of the RT Box. This option is useful if an RT Box should always run the same model or if no connection to a host PC is available.

Follow these steps to configure the box for this purpose:

[1] https://www.plexim.com

1. Leave the **target device** field of the **Target** tab in the PLECS Coder options empty and generate code.
2. Open the output directory that is specified in the **General** tab of the PLECS Coder options.
3. Rename the file *`modelname.elf`* to `autostart.elf` and copy it to the root directory of the SD card which also contains the RT Box firmware.
4. Reboot the RT Box with the same SD card. The model will start after the RT Box has booted up.

3

RT Box Architecture

This chapter describes the hardware architecture of the RT Box 1, 2, 3 and 4. Most of the following properties are common to all versions of the RT Box. Differences between versions are specifically mentioned in the following sections.

Front Panel

The front panel of the RT Box contains connectors for the analog and digital inputs and outputs (I/Os). For RT Box 1, 2 and 4 there are four of these connectors whereas the RT Box 3 has eight. 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 of the RT Box 1 contains 4 LEDs to indicate the status of the box. For the RT Box 2, 3 and 4 the status of the box is shown in an interactive display.

Analog Inputs

The RT Box 1, 2 and 4 provide 16 analog input channels the RT Box 3 provides 32 thereof. 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 for the RT Box 1 and

a maximum sampling rate of 5 MSPS with 1 cycle latency for the RT Box 2, 3 and 4.

As the firmware of the RT Box currently limits the cycle time to a minimum of 1 μs, a sampling rate of up to 1 MSPS can be realized when the ADC measurements are synchronized to the simulation step (default). Higher sampling rates are possible by configuring channel specific software or hardware triggers, which enables continuous ADC sampling. Linear interpolation of ADC values in hardware is then used for even finer resolution, as the analog inputs are still transferred once per simulation step only.

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

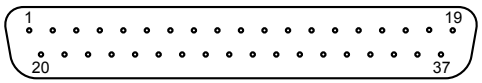
The analog inputs are equipped for fully differential measurement. The measured voltage is the difference between the positive and the negative input. The full-scale differential input range is ± 10 V or ± 5 V, depending on the configuration.

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 ± 10.8 V. As a consequence, the acceptable common mode voltage range is ± 5.8 V (for ± 10 V operation) and ± 8.3 V (for ± 5 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 ± 10 V or ± 5 V, depending on the configuration.

Analog Input Connector



37 pin male D-Sub (front view)

Table 3.1: Pinout of the analog input connector. *The value in brackets (x) specifies the channel number for the second analog input connector of the RT Box 3.

Pin	Description*	Pin	Description
1	Analog input 0 (16) positive	20	Analog input 0 (16) negative
2	Analog input 1 (17) positive	21	Analog input 1 (17) negative
3	Analog input 2 (18) positive	22	Analog input 2 (18) negative
4	Analog input 3 (19) positive	23	Analog input 3 (19) negative
5	Analog input 4 (20) positive	24	Analog input 4 (20) negative
6	Analog input 5 (21) positive	25	Analog input 5 (21) negative
7	Analog input 6 (22) positive	26	Analog input 6 (22) negative
8	Analog input 7 (23) positive	27	Analog input 7 (23) negative
9	Analog input 8 (24) positive	28	Analog input 8 (24) negative
10	Analog input 9 (25) positive	29	Analog input 9 (25) negative
11	Analog input 10 (26) positive	30	Analog input 10 (26) negative
12	Analog input 11 (27) positive	31	Analog input 11 (27) negative
13	Analog input 12 (28) positive	32	Analog input 12 (28) negative
14	Analog input 13 (29) positive	33	Analog input 13 (29) negative
15	Analog input 14 (30) positive	34	Analog input 14 (30) negative
16	Analog input 15 (31) positive	35	Analog input 15 (31) negative
17	n/c	36	n/c
18	n/c	37	GND
19	n/c	Shield	PE

> **Note**
>
> For the RT Box CE, analog input positive-negative channel pair from number 8 to 15 are not connected (n/c).

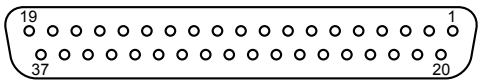
Analog Outputs

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)

Table 3.2: Pinout of the analog output connector. *The value in brackets (x) specifies the channel number for the second analog output connector of the RT Box 3.

Pin	Description*	Pin	Description
1	Analog output 0 (16)	20	GND
2	Analog output 1 (17)	21	GND
3	Analog output 2 (18)	22	GND
4	Analog output 3 (19)	23	GND
5	Analog output 4 (20)	24	GND
6	Analog output 5 (21)	25	GND

continues on next page

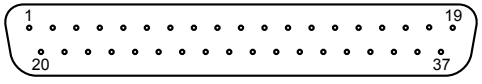
Table 3.2 – continued from previous page

Pin	Description*	Pin	Description
7	Analog output 6 (22)	26	GND
8	Analog output 7 (23)	27	GND
9	Analog output 8 (24)	28	GND
10	Analog output 9 (25)	29	GND
11	Analog output 10 (26)	30	GND
12	Analog output 11 (27)	31	GND
13	Analog output 12 (28)	32	GND
14	Analog output 13 (29)	33	GND
15	Analog output 14 (30)	34	GND
16	Analog output 15 (31)	35	GND
17	n/c	36	n/c
18	n/c	37	GND
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



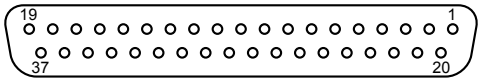
37 pin male D-Sub (front view)

Table 3.3: Pinout of the digital input connector. *The value in brackets (x) specifies the channel number for the second digital input connector of the RT Box 3.

Pin	Description*	Pin	Description
1	Digital input 0 (32)	20	Digital input 1 (33)
2	Digital input 2 (34)	21	Digital input 3 (35)
3	Digital input 4 (36)	22	Digital input 5 (37)
4	Digital input 6 (38)	23	Digital input 7 (39)
5	GND	24	Digital input 8 (40)
6	Digital input 9 (41)	25	Digital input 10 (42)
7	Digital input 11 (43)	26	Digital input 12 (44)
8	Digital input 13 (45)	27	Digital input 14 (46)
9	Digital input 15 (47)	28	GND
10	Digital input 16 (48)	29	Digital input 17 (49)
11	Digital input 18 (50)	30	Digital input 19 (51)
12	Digital input 20 (52)	31	Digital input 21 (53)
13	Digital input 22 (54)	32	Digital input 23 (55)
14	GND	33	Digital input 24 (56)
15	Digital input 25 (57)	34	Digital input 26 (58)
16	Digital input 27 (59)	35	Digital input 28 (60)
17	Digital input 29 (61)	36	Digital input 30 (62)
18	Digital input 31 (63)	37	GND
19	5 V, 1.5 A max.	Shield	PE

Digital Outputs

Digital Output Connector



37 pin female D-Sub (front view)

Table 3.4: Pinout of the digital output connector. *The value in brackets (x) specifies the channel number for the second digital output connector of the RT Box 3.

Pin	Description*	Pin	Description
1	Digital output 0 (32)	20	Digital output 1 (33)
2	Digital output 2 (34)	21	Digital output 3 (35)
3	Digital output 4 (36)	22	Digital output 5 (37)
4	Digital output 6 (38)	23	Digital output 7 (39)
5	GND	24	Digital output 8 (40)
6	Digital output 9 (41)	25	Digital output 10 (42)
7	Digital output 11 (43)	26	Digital output 12 (44)
8	Digital output 13 (45)	27	Digital output 14 (46)
9	Digital output 15 (47)	28	GND
10	Digital output 16 (48)	29	Digital output 17 (49)
11	Digital output 18 (50)	30	Digital output 19 (51)
12	Digital output 20 (52)	31	Digital output 21 (53)
13	Digital output 22 (54)	32	Digital output 23 (55)
14	GND	33	Digital output 24 (56)
15	Digital output 25 (57)	34	Digital output 26 (58)

continues on next page

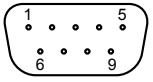
Table 3.4 – continued from previous page

Pin	Description*	Pin	Description
16	Digital output 27 (59)	35	Digital output 28 (60)
17	Digital output 29 (61)	36	Digital output 30 (62)
18	Digital output 31 (63)	37	GND
19	n/c	Shield	PE

Resolver

The RT Box 2, 3 and 4 provide a resolver interface that can be used for Hardware-in-the-loop (HIL) and Rapid Control Prototyping (RCP) applications.

Resolver In Connector (RCP)



9 pin male D-Sub (front view)

Table 3.5: Pinout of Resolver In connector

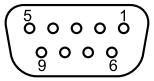
Pin	Description
1	GND
2	+COSIN
3	+SININ
4	+EXCOUT
5	GND
6	-COSIN
7	-SININ

continues on next page

Table 3.5 – continued from previous page

Pin	Description
8	GND
9	-EXCOUT

Resolver Out Connector (HIL)



9 pin female D-Sub (front view)

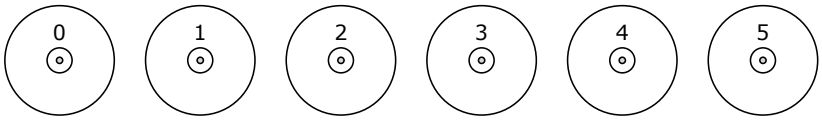
Table 3.6: Pinout of Resolver Out connector

Pin	Description
1	GND
2	+COSOUT
3	+SINOUT
4	+EXCIN
5	GND
6	-COSOUT
7	-SINOUT
8	GND
9	-EXCIN

Nanostep Out

The Nanostep Out is available on RT Box 4 and provides a fast analog output from a Nanostep signal source.

Nanostep Out BNC Sockets

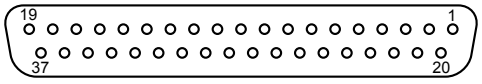


6 BNC sockets (front view)

Table 3.7: Pinout of Nanostep Out BNC sockets. (Outer conductor is connected to GND and not to PE.)

Inner conductor	Outer conductor
Nanostep Out 0	GND
Nanostep Out 1	GND
Nanostep Out 2	GND
Nanostep Out 3	GND
Nanostep Out 4	GND
Nanostep Out 5	GND

Nanostep Out D-Sub Connector



37 pin female D-Sub (front view)

Table 3.8: Pinout of Nanostep Out D-Sub connector. (Depending on the voltage range setting, the D-Sub has a voltage reference of either 0V or 2V on pin 16.)

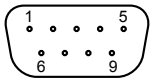
Pin	Description	Pin	Description
1	Nanostep Out 0	20	GND
2	Nanostep Out 1	21	GND
3	Nanostep Out 2	22	GND
4	Nanostep Out 3	23	GND
5	Nanostep Out 4	24	GND
6	Nanostep Out 5	25	GND
7	n/c	26	GND
8	n/c	27	GND
9	n/c	28	GND
10	n/c	29	GND
11	n/c	30	GND
12	n/c	31	GND
13	n/c	32	GND
14	n/c	33	GND
15	n/c	34	GND
16	D-Sub ref. (0 / 2V)	35	GND
17	n/c	36	n/c
18	n/c	37	GND
19	n/c	Shield	PE

Rear Panel

CAN

The CAN connector is available on RT Box 2, 3, 4 and hardware revision 1.2 of the RT Box 1.

CAN Connector



9 pin male D-Sub (front view)

Table 3.9: Pinout of CAN connector

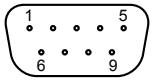
Pin	Description
1	n/c
2	CAN_L channel 1
3	GND
4	CAN_L channel 2
5	n/c
6	n/c
7	CAN_H channel 1
8	CAN_H channel 2
9	n/c

The CAN connector is electrically isolated from the rest of the RT Box.

UART

The UART Interface is only available on the RT Box 2, 3 and 4.

RS-232/422/485 Connector



9 pin male D-Sub (front view)

Table 3.10: Pinout of RS-232/422/485 connector

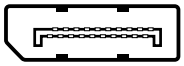
Pin	RS-232	RS-422	RS-485
1	RXD channel 2	Rx+ channel 2	n/c
2	RXD channel 1	Rx+ channel 1	n/c
3	TXD channel 1	Tx+ channel 1	TRx+ channel 1
4	TXD channel 2	Tx+ channel 2	TRx+ channel 2
5	GND	GND	GND
6	RTS channel 2	Tx- channel 2	TRx- channel 2
7	RTS channel 1	Tx- channel 1	TRx- channel 1
8	CTS channel 1	Rx- channel 1	n/c
9	CTS channel 2	Rx- channel 2	n/c

The RS-232/422/485 connector is electrically isolated from the rest of the RT Box.

Displayport

A displayport connector is only available on the RT Box 2, 3 and 4. There is currently no software implementation in the RT Box target support package for this interface.

Displayport Connector

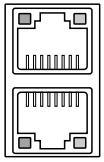


Displayport connector (front view)

Industrial Ethernet

Two Industrial Ethernet ports are available on the RT Box 2, 3 and 4. The EtherCAT fieldbus interface is available via this port on RT Box 2 and RT Box 3 with hardware revision 1.2 and newer, and also on the RT Box 4.

Industrial Ethernet Connector



Industrial Ethernet connector (front view)

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, a $1\,\mu\mathrm{F}$ capacitor and a 5V zener diode (SMAJ5.0CA), all connected in parallel. 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.

The CAN connector (only available on certain versions of the RT Box) and the RS-232/422/485 connector (only available on RT Box 2, 3 and 4) are electrically isolated from the rest of the RT Box.

Principle of Operation

The basic building blocks of the RT Box are shown in Fig. 3.1.

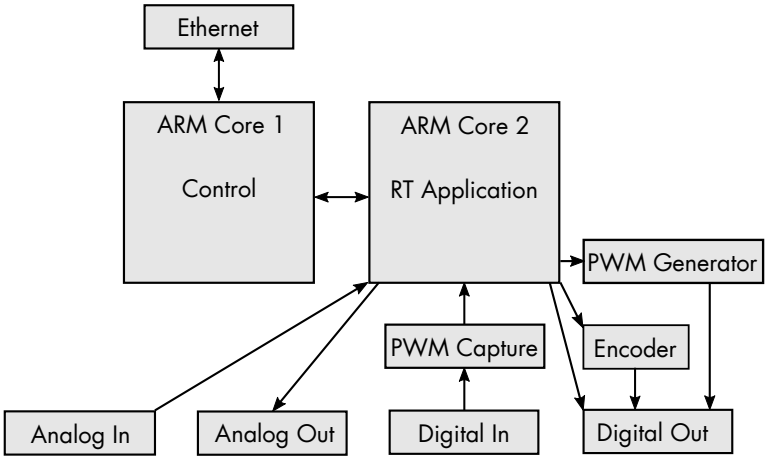


Fig. 3.1: RT Box Overview

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.

Unsafe Math Operations

The C standard specifies that the result of the divide operators "/" or "%" is undefined if the second operand is zero. PLECS does not add extra checks when

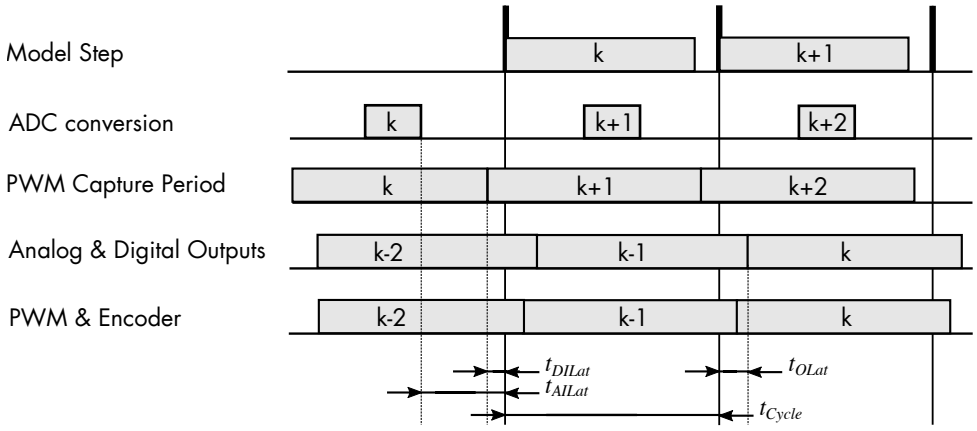


Fig. 3.2: Input / output timing

generating code which uses a divide operator. For this reason it is the responsibility of the model creator to ensure that a division-by-zero condition does not occur.

In most cases, a floating point division-by-zero results in a floating point value of *NaN* (Not a Number) or $\pm\infty$, whereas an integer division-by-zero results in immediate program termination. However, this behavior should not be relied upon.

The same applies to other math operations that are undefined, e.g. $tan(\pi/2)$, $asin(2)$ etc.

4

Specifications

Overview

Table 4.1: Technical Specifications of RT Box CE, 1, 2, 3 and 4

		RT Box CE	RT Box 1	RT Box 2	RT Box 3	RT Box 4
Processor	Simulation Cores	1 × 1 GHz		3 × 1.5 GHz		
	Xilinx Zynq	Z-7030		ZU9EG		
Analog inputs	Channels	8	16		32	16
	Resolution	16 bit simultaneous sampling				
	Voltage ranges	-10 … 10 V -5 … 5 V				
	Input type	Differential				
	Max. sample rate	2 Msps		5 Msps	2.5 Msps	5 Msps
	Input impedance	1 MΩ, 24 pF				
	Connector	D-Sub 37 pin male				

continues on next page

Table 4.1 – continued from previous page

		RT Box CE	RT Box 1	RT Box 2	RT Box 3	RT Box 4
Nanostep out	Channels	-				6
	Voltage ranges	-				-2 … 2 V 0 … 4 V
	Update rate	-				125 Msps
	Output impedance	-				50 Ω 100 Ω
	Connector	-				BNC D-Sub 37
Analog outputs	Channels	16			32	16
	Resolution	16 bit simultaneous update				
	Voltage ranges	-10 … 10 V 0 … 10 V -5 … 5 V 0 … 5 V				
	Max. update rate	2 Msps		5 Msps		
	Output impedance	0 Ω				
	Max. output current	10 mA				
	Connector	D-Sub 37 pin female				

continues on next page

Table 4.1 – continued from previous page

		RT Box CE	RT Box 1	RT Box 2	RT Box 3	RT Box 4
Digital inputs	Channels	32			64	32
	Logic levels	3.3 V (5 V tolerant)				
	High-level input	min. 2 V				
	Low-level input	max. 0.8 V				
	Configurable input impedance	high or 10 kΩ against 3.3 V or GND				
	Connector	D-Sub 37 pin male				
Digital outputs	Channels	32			64	32
	Logic levels	3.3 V and 5 V				
	Output impedance	220 Ω				
	Output current	≤ 10 mA				
	Connector	D-Sub 37 pin female				
Resolver	Input/Output	-/-		1/1	2/2	1/1
	Connector	-/-		D-Sub 9 pin male/fem.		
I/O Protection	Short-circuit	Permanent				
	Overvoltage	-24 … 24 V				
						-5 … 5 V (Nanostep)

continues on next page

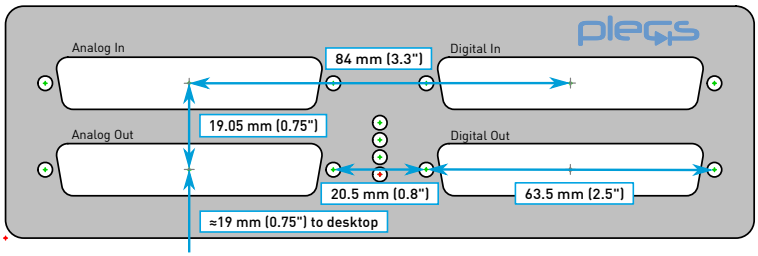
Table 4.1 – continued from previous page

		RT Box CE	RT Box 1	RT Box 2	RT Box 3	RT Box 4
Connectivity	Gigabit Ethernet	1		2		
	SFP+ (6.25 Gbps /lane)	-	4	8		
	Ind. Ethernet	-		2 (EtherCAT)		
	CAN bus	2				
	RS 232/422/485	-		2		
	USB device	USB 2.0 High-Speed, type A		USB 3.0 SuperSpeed, type A		
	Display Port	-		1		
Storage	Internal SSD	-		480 GB		
	Firmware	SD card				
Power supply	Internal	100 … 240 Vac 50 … 60 Hz				
		30 VA	50 VA	65 VA	100 VA	
Size	Depth × Width (mm × mm)	225 × 165	310 × 250			
	Height (mm)	55	100		145	
	Weight (kg)	1.3	3.1	3.2	3.9	

Mechanical Dimensions

RT Box CE

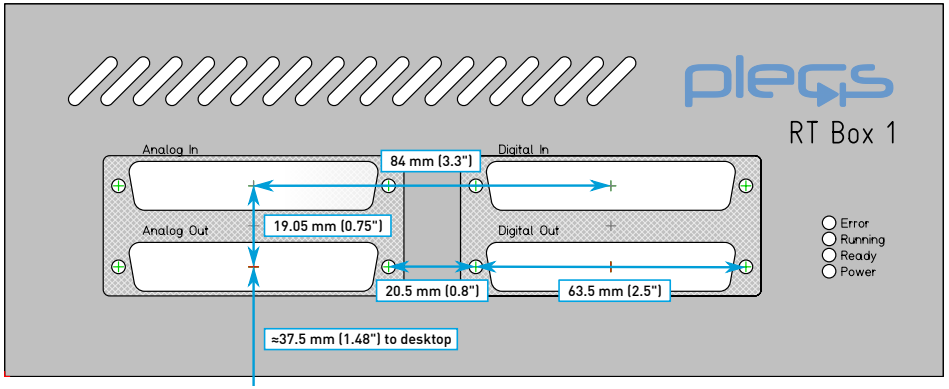
The four 37 pin D-Sub connectors on the front panel are displaced 19.05 mm (0.75 inch) vertically and 84 mm (3.3 inch) horizontally.



Mechanical dimensions of the front panel of the RT Box CE

RT Box 1

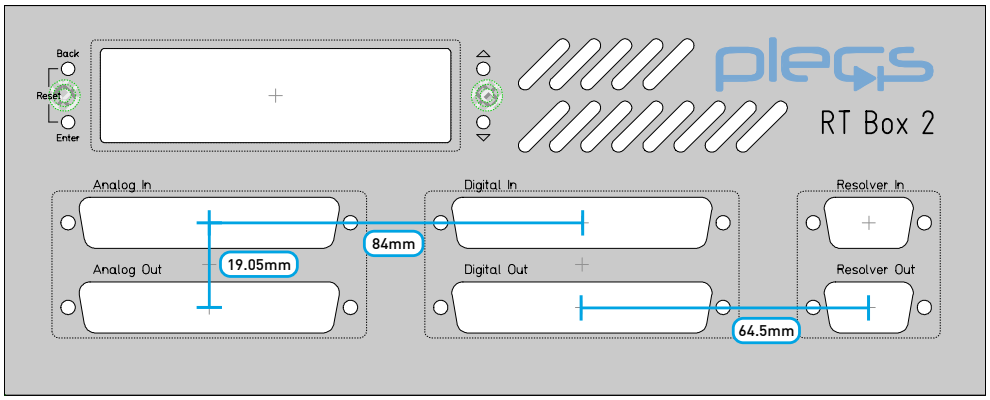
The four 37 pin D-Sub connectors on the front panel are displaced 19.05 mm (0.75 inch) vertically and 84 mm (3.3 inch) horizontally.



Mechanical dimensions of the front panel of the RT Box 1

RT Box 2

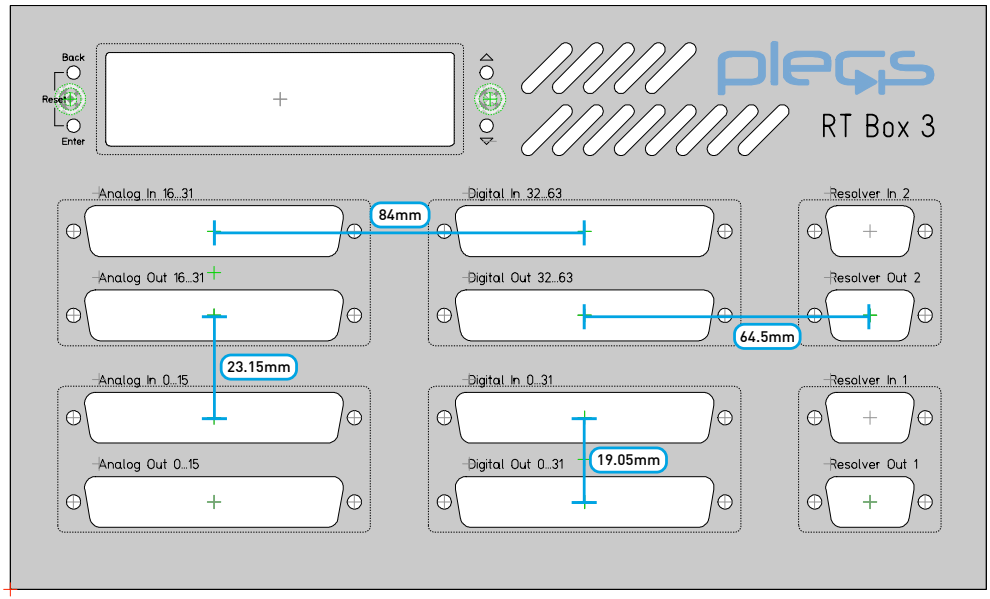
The four 37 pin D-Sub connectors on the front panel for digital and analog in-/out-puts are displaced 84 mm (3.3 inch) horizontally and 19.05 mm (0.75 inch) vertically. The resolver connectors have the same vertical displacement as the analog and digital in-/outputs and a horizontal distance of 64.5 mm (2.54 inch) to the digital in- and outputs.



Mechanical dimensions of the front panel of the RT Box 2

RT Box 3

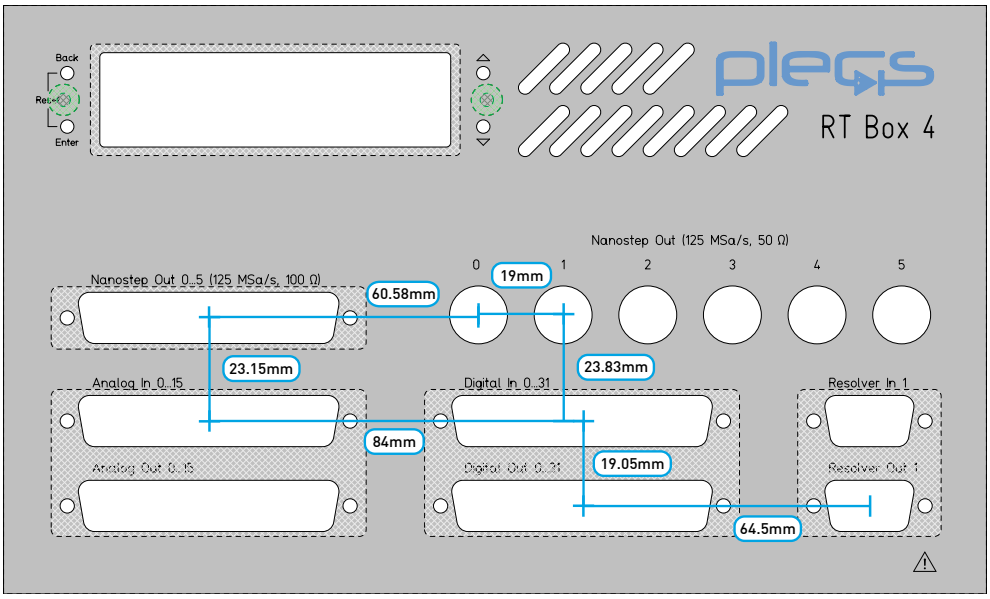
The eight 37 pin D-Sub connectors on the front panel for digital and analog in-/outputs are displaced 84 mm (3.3 inch) horizontally and 19.05 mm (0.75 inch) / 23.15 mm (0.91 inch) vertically. The resolver connectors have the same vertical displacements as the analog and digital in-/outputs and a horizontal distance of 64.5 mm (2.54 inch) to the digital in- and outputs.



Mechanical dimensions of the front panel of the RT Box 3

RT Box 4

The four 37 pin D-Sub connectors on the front panel for digital and analog in-/outputs are displaced 84 mm (3.3 inch) horizontally and 19.05 mm (0.75 inch) vertically. The resolver connectors have the same vertical displacements as the analog and digital in-/outputs and a horizontal distance of 64.5 mm (2.54 inch) to the digital in- and outputs. The Nanostep output D-Sub connector has a vertical distance of 23.15 mm (0.91 inch) to the analog inputs. The Nanostep output BNC connector channel 0 has a horizontal distance of 60.58 mm (2.385 inch) to the Nanostep output D-Sub connector. The horizontal distance between each Nanostep output BNC connector is 19 mm (0.75 inch). The vertical distance between the Nanostep output BNC connectors and the digital inputs is 23.8 mm (0.94 inch).



Mechanical dimensions of the front panel of the RT Box 4

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.

Nanostep D-Sub output range
(RT Box 4 only) This option specifies whether the Nanstep outputs on the D-Sub connector are **unipolar** or **bipolar**. The voltage range for the **bipolar** setting is -2 V to 2 V if the outputs are unterminated, -1 V to 1 V if correctly terminated to ground with 100 Ω. For the **unipolar** setting, the voltage output range is 0 V to 4 V if the outputs are unterminated, 0 V to 2 V if correctly terminated to ground with 100 Ω.

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

Analog input sampling / Sampling delay
The delay (in seconds) between the start of the simulation step and the sampling of the analog inputs for the next simulation step. If a sampling delay of 0 is specified, sampling for the next simulation step is done at the beginning of the current simulation step (corresponding to the maximum

latency). To sample the analog inputs as late as possible, select **Minimize latency**. Minimum latency is usually desirable unless sampling must be synchronized to, for example, the peaks of the PWM carrier. Alternatively, the Analog In block allows to configure individual software or hardware triggers.

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.

Simulation mode

When set to **HIL**, the default value, the RT Box behaves as discussed in chapter "Principle of Operation" (p. 27): The outputs of the RT Box are updated at the start of the next simulation step. In rapid control prototyping applications, this can lead to undesirable latency. When switching to **RCP**, the outputs of the RT Box are updated immediately after the calculation for the simulation step is completed (at the expense of less time available for the calculation). **RCP** mode cannot be used when multitasking is enabled.

> **Note**
>
> The **Simulation mode** parameter is deprecated. The RCP mode will be removed in a future version of the TSP.

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.

Enable XCP

XCP is the Universal Measurement and Calibration Protocol published by the Association for Standardisation of Automation and Measuring Systems (www.asam.net[2]). RT Box 2, 3 and 4 support the XCP protocol version 1.5 over a UDP network connection. The RT Box acts as an XCP slave device. It can be accessed on UDP port 5555.

When enabled, all signals to scopes, XY plots and display blocks are available as DAQ measurements. All tuneable parameters can be modified using STIM

[2] https://www.asam.net

packets. An A2L file is generated in the output directory of the code generation process for import into an XCP master (e.g. CANape from Vector Informatik GmbH).

XCP slave identity
This options specifies how the RT Box is identified in the generated A2L file. When set to **Use target device**, the name of the box is taken from the Target device input field. When set to **Specify host name** or **Specify IP address**, the host name or IP address of the RT Box can be explicitly specified. For use with CANape it seems to be necessary to specify the IP address explicitly.

XCP decimation
By default, XCP measurements are sent at every CPU base step. If a large number of scopes are used in the codegen subsystem, this can significantly impact CPU utilization. To reduce CPU load, a decimation factor larger than 1 can be used. In this case, the XCP measurement values are sent at lower resolution every n-th CPU base step.

Interconnect

These options allow to synchronize multiple connected RT Boxes. Please also read section “Time and Startup Synchronization” (p. 47).

Primary box for startup/clock
Specifies on which SFP port the primary RT Box is connected. For the top level primary box or for standalone boxes this option must be set to `Self`.

Synchronize startup with SFP x
Enables startup synchronization on the given SFP port. Primary boxes must enable startup synchronization on all ports to which a secondary box is connected. Secondary boxes must enable startup synchronization on the port to which the primary box is connected.

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

CAN

These options allow to set up the CAN interfaces of the RT Box.

Enable CAN*x*

Enables or disables a CAN interface. If a CAN interface is enabled and configured to use digital IO pins, these pins cannot be used for other purposes (e.g. PWM generation or PWM capturing). CAN1 and CAN2 refer to the internal CAN interfaces of the RT Box. Additionally, a PCAN-USB Pro or a PCAN-USB FD adapter from Peak Systems can be connected via USB. These adapters offer CAN FD connections.

CAN*x* bit rate

The bit rate that is used on the connected CAN bus. All devices on a CAN bus must be configured to use the same bit rate.

Bit rate switching

For PCAN devices connected over USB, bit rate switching for CAN FD can be enabled. Additionally, bit rate switching must be enabled for each individual CAN message type in the CAN Transmit block (p. 63)).

CAN*x* data bit rate

If bit rate switching for CAN FD devices is enabled, this parameter specifies the bit rate that is used during the data phase.

TX/RX pin

Specifies which IOs to use for CAN communication. Each CAN channel requires one input and one output pin. Select 'Internal CAN interface' for the in-built CAN interface on the back of some RT Box models (see section "CAN" (p. 24)). This option is only available with RT Box 1. RT Box 2, 3 and 4 always use the internal CAN interface.

CAN*x* error recovery delay [ms]

If too many errors occur on the CAN bus, the CAN bus switches to a *bus-off* state. If this parameter is set to a non-zero value, automatic recovery is activated. The value specifies the time in milliseconds after which the recovery should be started. A value of 0 disables automatic recovery.

6

Running Simulations on the FPGA

Depending on the RT Box model, two distinct methods are available for executing an electrical model or the time-critical parts of it on the FPGA.

The **Nanostep** solver enables simulations with ultra-small simulation step sizes for certain parts of an electrical model.

The **FlexArray** solver allows to simulate complete electrical systems on the FPGA, delivering high update rates and low latency. Both solvers run independently and can even be combined on RT Box 2, 3 and 4.

Nanostep solver

The Nanostep solver uses ultra-small simulation step sizes for controlling common power architectures, supporting MHz-level switching frequencies. Each simulation step includes sampling, calculation, and diode switch state determination. On RT Box 2, 3 and 4 the simulation time step is just 4 ns, on RT Box 1 and CE it is 7.5 ns.

With the Nanostep solver, only the time-critical part of a circuit is calculated with nanostep resolution. The remaining part of the system is calculated either on the CPU or the FlexArray solver.

The power modules supported by the Nanostep solver can be found in the Nanostep category of the PLECS components library. At least one Nanostep power module must be present in the model to use the Nanostep solver. To run the power module on a Nanostep solver, the **Configuration** parameter of the power module must be set to “Nanostep”.

The top line of the display of an RT Box 2, 3 or 4 indicates an active Nanostep solver by displaying the step size of the Nanostep solver before the FlexArray /

CPU update rate, e.g. "4 ns/250 ns". An active Nanostep solver is also indicated in the web interface of the RT Box.

The number of Nanostep power modules that can be simulated within one model is limited. Each Nanostep power module is associated with a weight (a simple dual active bridge, for example, has weight of 1, a full-bridge resonant converter a weight of 3 - see the documentation of the respective power modules). RT Box 2, 3 and 4 allow the simulation of Nanostep power modules with a total weight sum of 6, for RT Box CE and 1 the total weight sum is limited to 3.

FlexArray solver

RT Box 2, 3 and 4 allow to run simulations on the FPGA using the FlexArray solver. Depending on the complexity of the model, this allows step sizes below 100 ns.

Only the electrical domain of a PLECS model can be run on the FlexArray solver. This usually comprises PWM Capture (p. 79) blocks, an electrical circuit with power modules, and meters that are connected to Analog Out (p. 60) blocks. The control part of the model will always be simulated on the CPU.

When a simulation is running using the FlexArray solver, the analog outputs of the RT Box are updated after each FlexArray solver step, but no faster than every 200 ns.

Enabling the FlexArray solver

Whether an electrical circuit is simulated on the CPU or the FlexArray solver is controlled by the **Electrical Model Settings** block. Place the block on the schematic and connect it to the electrical circuit that should be simulated on the FlexArray solver. The **Target** parameter controls whether the code for this circuit is generated for the CPU or the FlexArray solver.

Prerequisites

Code generation for the FlexArray solver is only supported for circuits based on Power Modules. Additionally, the following discrete Power Semiconductors and Switches are supported:

- Diode
- Thyristor
- Triac

- Switch
- Double Switch
- Triple Switch
- Breaker

Power Modules must be connected directly to PWM capture blocks.

The FlexArray solver uses a non-ideal switch model. Once a circuit is configured for FlexArray, all switches use the non-ideal switch model automatically, regardless of their **Switch model** parameter.

To facilitate updates of the analog outputs with the full rate of the FlexArray solver step sizes, Meters (e.g. Voltmeters, Ammeters) must be connected directly to AnalogOut blocks. Exemptions are additional gain or offset blocks in the signal path.

Minimizing the FlexArray solver step size

The calculation time of one simulation step of the FlexArray solver depends on the complexity of the model. It does not vary from one step to the next.

The FlexArray solver step size is automatically calculated as the minimum integer fraction of the CPU discretization time that is larger than the required calculation time. The calculated step size is shown in the web interface and the upper right corner of the front display of the RT Box.

For reference, the smallest possible FlexArray solver step size for the given model is displayed in the diagnostics messages in the web interface of the RT Box. This value may be used to choose a suitable CPU discretization time (as an integer multiple of the smallest possible FPGA simulation time) to achieve the smallest possible time step.

7

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 volt-

age 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:

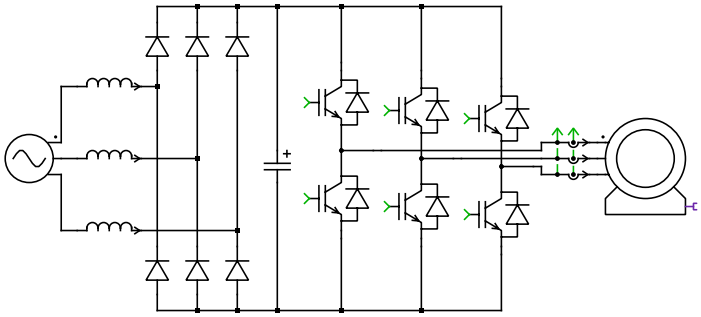


Fig. 7.1: Model with DC-link before split-up

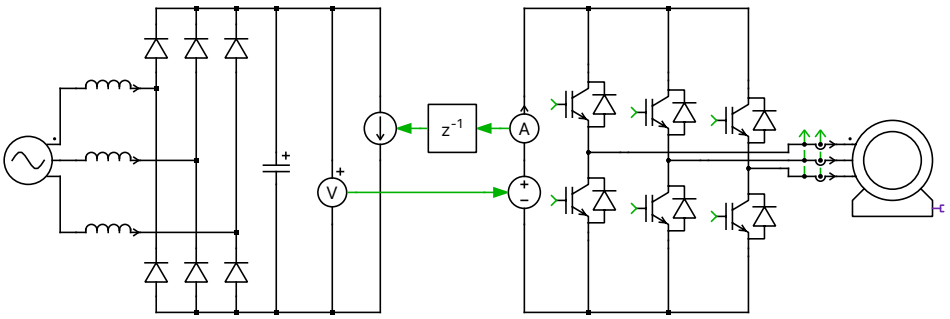


Fig. 7.2: Model with DC-link after split-up

Note that in an offline simulation a Delay block 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 Gbps. To send signals from a model over a SFP port use the SFP Out (p. 98). To receive signals from an SFP port and use it in a model use the SFP In (p. 97).

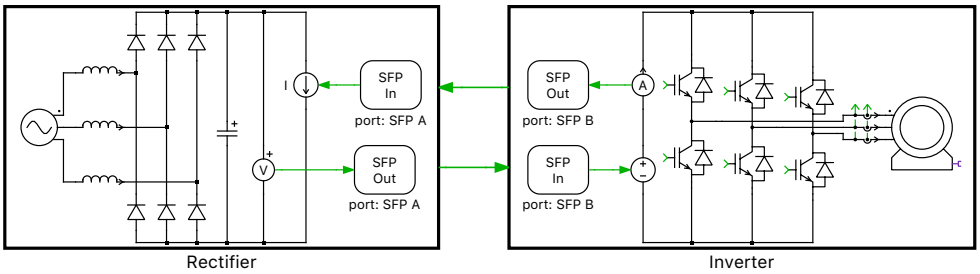


Fig. 7.3: Model with communication over SFP

The SFP communication introduces a latency of two simulation timesteps (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 RT Boxes. Synchronization requires assigning “primary” and “secondary” roles to SFP ports. For time step synchronization the clock signal from the primary box

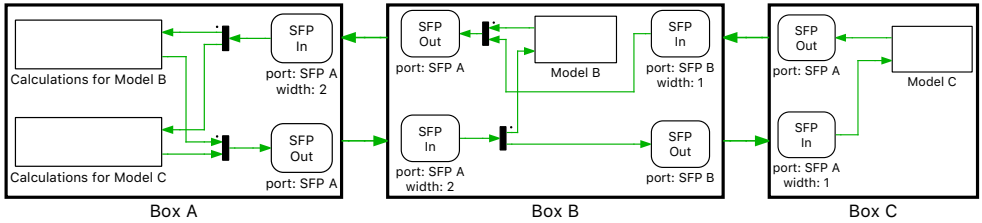


Fig. 7.4: Signal relaying

is distributed to a number of secondary boxes. All boxes with time step synchronization enabled must use the same simulation step time.

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

The assignment of primary and secondary roles is done in the **Interconnect** settings of the RT Box specific coder options. Please see section “Interconnect” (p. 39) for further details.

Note that one RT Box can connect to a primary box on one SFP port and act as a primary box 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 peripheral blocks and vice 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.

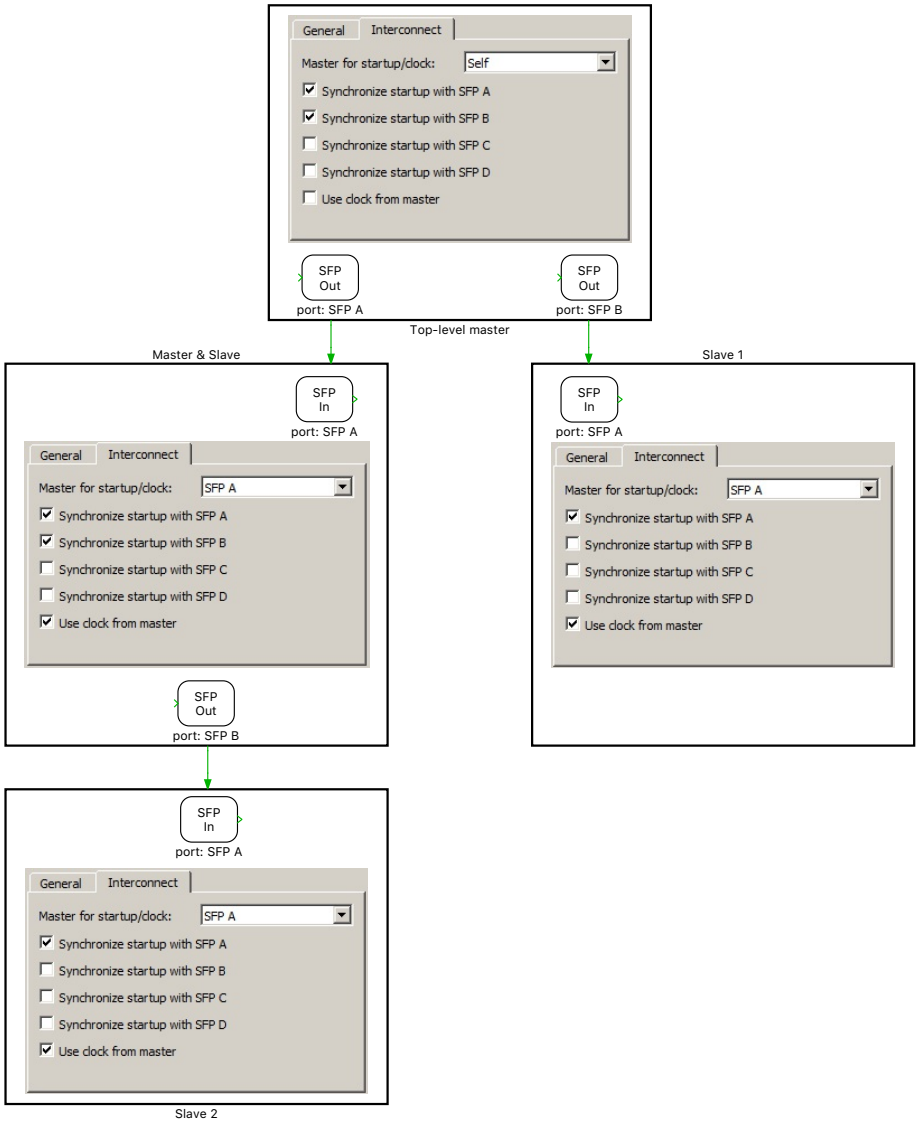


Fig. 7.5: Startup and time synchronization in a 3 level tree

8

Scripting

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

For Python, extensions for XML/JSON-RPC support are available. XML-RPC is an RPC protocol that uses XML to encode its calls. JSON-RPC is an RPC protocol that uses JSON-encoded messages. They both use HTTP as a transport mechanism.

In the following examples, Python 3.x syntax and script excerpts are used.

Establishing an XML/JSON-RPC Connection to the RT Box

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

```
import socket
ip = socket.gethostbyname("examplebox.local")
```

- For XML format:

```
import xmlrpc.client
server = xmlrpc.client.ServerProxy("http://" + ip + ":9998/RPC2")
```

- For JSON format:

```
import jsonrpc_requests
server = jsonrpc_requests.Server("http://" + ip + ":9998/RPC2")
```

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

Overview of XML-RPC Commands

Commands for RT Box 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 RPC binary object. For XML format in Python, this object can be directly created using the `Binary` object from `xmlrpc.client`. The following code shows how to read an ELF executable file and load it on the RT Box using Python.

- For XML format:

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

- For JSON format:

```
import base64  # deserialise MAT-file contents for JSON-RPC
with open("TestModel_codegen/TestModel.elf", "rb") as f:
    server.rtbox.load(base64.b64encode(f.read()).decode())
```

The command

```
rtbox.start()
```

starts the execution of the real-time simulation.

The command

```
rtbox.stop()
```

stops the execution of the real-time simulation.

The command

```
rtbox.reboot('reboot')
```

stops the execution of the real-time simulation and reboots the RT Box.

The command

```
rtbox.querySimulation()
```

returns a struct with (at least) the following fields:

modelName
The name of the model or subsystem that is currenty executed.

sampleTime
The base discretization step size of the model.

status
The current execution state of the simulation. Possible values are “stopped”, “running” and “error”.

The command

```
rtbox.getApplicationLog()
```

returns the log messages from the running simulation.

Sending data to the real-time simulation

The command

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

sets the output of the Programmable Value (p. 78) 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 (p. 65) 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 (p. 65) 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 socket
import xmlrpc.client
import random
import time

ip = socket.gethostbyname("examplebox.local")
server = xmlrpc.client.ServerProxy("http://" + ip + ":9998/RPC2")

with open("TestModel_codegen/TestModel.elf", "rb") as f:
    print("Uploading executable")
    server.rtbox.load(xmlrpc.client.Binary(f.read()))
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()
```

9

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.

Description

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



Parameters

General

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.

Trigger

Trigger source
If “Global setting” is selected, the Analog In measurements happen simultanously, once per simulation step, as configured in the *Analog input sampling* of the Coder Options Dialog (see section General (p. 37)). Otherwise, continuous ADC sampling is activated and Analog In measurements can be triggered individually. If “Specify delay within simulation step” option is selected, the “Trigger delay(s)” field allows to set a custom delay. If “Use PWM trigger port” is selected, a trigger block input is enabled that is intended for connection to an ADC trigger output of a PWM Out block.

Trigger delay(s)
Trigger delay for the measurement in seconds. Only available when Trigger source is configured as “Specify delay within simulation step”. The parameter can be either a scalar or a vector with the same width as the number of input channels.

Initial ADC voltage(s)

Initial ADC voltage in Volts. Only available when Trigger source is configured as “Use PWM trigger port”. This value specifies a virtual input voltage that is used at the start of the simulation until the first trigger signal is received from the PWM block. The parameter can be either a scalar or a vector with the same width as the number of input channels.

Analog Out

Purpose

Set the output voltage of an analog output channel.

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.

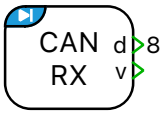
Maximum output voltage
The highest value that the output voltage can reach.

CAN Receive

Purpose

Receives CAN messages.

Description



The block initiates the reception of CAN messages with the given ID on the given CAN interface. On reception of a CAN message the data is made available on the block output **d** as a vectorized signal consisting of 8 bytes. The output **v** is **1** in each simulation step where new data is received, **0** otherwise.

The data at block output **d** always has a width of 8 bytes regardless of the length of the received CAN message.

Parameters

CAN interface
The CAN interface to use. The selected CAN interface must be configured in the RT Box Coder Options Dialog (see section CAN (p. 39)).

CAN ID source
Selects whether the CAN ID is specified as a parameter or is supplied as an input signal.

CAN ID
The CAN identifier for which this block receives CAN messages. The CAN ID can be supplied as either a 11 bit value (vor CAN 2.0A) or 29 bit value (for CAN 2.0B).

Frame format
Specifies the frame format that is used when filtering for matching CAN messages. Possible values are:

- **Base** for CAN 2.0A messages with an 11 bit identifier
- **Extended** for CAN 2.0B messages with an 29 bit identifier
- **Auto** uses the **Base** format if the specified **CAN ID** is smaller than 2047. Otherwise, the **Extended** format is used.

Frame length
The length of the expected CAN frame, in bytes. For CAN 2.0, the standard length of 8 bytes should be used. If the CAN implementation supports CAN FD, up to 64 bytes can be used.

Offline simulation
Enables or disables data injection in an offline simulation. If set to **enabled**, terminals are added to the subsystem for which code is generated. In an offline simulation the data which is to be received by the CAN Receive block can be injected into these terminals. The block must be placed in the root schematic of the subsystem for which code is generated. If set

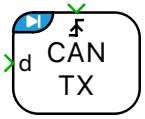
to **disabled**, no such terminals will be added and the block can be used at any subsystem level.

CAN Transmit

Purpose

Transmits CAN messages.

Description



The CAN Transmit block sends out data on a CAN bus. The data to send must be provided on the block input **d** as a vectorized signal with data type **uint8**. The length of the transmitted CAN message is determined by the width of the input signal (1 to 8 bytes).

Messages are either sent regularly with a fixed sample time or on demand when the trigger input changes. When configured for triggered execution, messages are sent when the trigger signal changes in the manner specified by the **Trigger type** parameter:

rising
Data is sent when the trigger signal changes from 0 to a non-zero value.

falling
Data is sent when the trigger signal changes from a non-zero value to 0.

either
Data is sent when the trigger signal changes from 0 to a non-zero value or vice versa.

Parameters

CAN interface
The CAN interface to use. The selected CAN interface must be configured in the RT Box Coder Options Dialog (see section CAN (p. 39)).

CAN ID source
Selects whether the CAN ID is specified as a parameter or is supplied as an input signal.

CAN ID
The CAN identifier that is used for CAN messages sent by this block. The CAN ID can be supplied as either an 11 bit value (for CAN 2.0A compliance) or 29 bit value (for CAN 2.0B compliance).

Bit rate switching
If the CAN interface supports CAN FD, bit rate switching for higher data rates can be enabled. Bit rate switching must also be enabled in the RT Box Coder Options Dialog (see section CAN (p. 39)).

Execution
Selects between **regular** and **triggered** execution.

Sample time
The time period for regular packet transmission (for regular execution only).

Trigger type
The direction of the edges of the trigger signal upon which the data is sent, as described above (for triggered execution only).

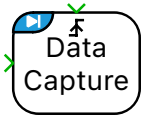
Offline simulation
Enables or disables data inspection in an offline simulation. If set to **enabled**, terminals are added to the subsystem for which code is generated. In an offline simulation the data sent to the CAN Transmit block is available at these terminals. The block must be placed in the root schematic of the subsystem for which code is generated. If set to **disabled**, no such terminals will be added and the block can be used at any subsystem level.

Data Capture

Purpose

Capture data to be transferred via XML-RPC.

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” (p. 54) 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. With the setting **continuous**, the data acquisition is always active. When set to **once at startup**, the data buffer is filled once when the simulation is startet. In both cases, 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.

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.

Input characteristic

Specifies whether an internal pull-up (10 kΩ against 3.3 Volt) or pull-down resistor (10 kΩ against GND) is connected to the digital input.

Digital Out

Purpose

Set a digital output.

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)

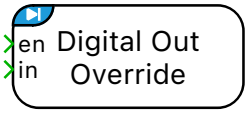
Index of the digital output channel. For vectorized output signals a vector of output channel indices must be specified.

Digital Out Override

Purpose

Override a digital output set by another block.

Description



The Digital Out Override block is intended to be used in parallel with another block that controls a digital output channel. If 'en' is low, the other block has full control of the output channel. If 'en' is high, then the output channel is set to the value of 'in'.

The Digital Out Override block cannot be used to override PWM outputs that are protected by a Powerstage Protection Unit.

Parameters

Digital output channel(s)

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

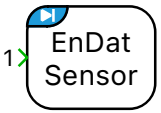
EnDat Sensor

Purpose

Provide an EnDat sensor implementation to transmit position information.

Description

The EnDat sensor takes a rotational angle θ as input.



The EnDat sensor is available on RT Box 2 and RT Box 3 with hardware revision 1.2 and later, and also on the RT Box 4. It supports a subset of the EnDat 2.1 command set, in particular:

Table 9.1: Supported mode commands

Mode command	M2	M1	M0	(M2)	(M1)	(M0)
Encoder send position values	0	0	0	1	1	1
Selection of memory area	0	0	1	1	1	0
Encoder send parameter (for selected parameters, see below)	1	0	0	0	1	1
Encoder receive parameter	0	1	1	1	0	0
Encoder receive reset	1	0	1	0	1	0

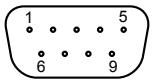
The following parameters can be queried using a combination of mode commands “Selection of memory area” and “Encoder send parameter”:

Table 9.2: Supported parameters

Parameter	Parameter no.	MRS code
EnDat version	8	161
Resolution	13	161
EnDat command set	42	165

In each of the EnDat OEM parameter areas (MRS codes 169, 171, 173 and 175) a range of 256 words may be written (using the “Encoder receive parameter” command) and read back (using the “Encoder send parameter” command).

An EnDat Controller must be connected to the RS-232/422/485 port on the back of the RT box. The connector layout for the EnDat sensor is shown below:



9 pin male D-Sub (front view)

Table 9.3: Pinout of the RS-232/422/485 connector for EnDat usage

Pin	EnDat
1	n/c
2	n/c
3	CLK+
4	DATA+
5	GND
6	DATA-
7	CLK-
8	n/c
9	n/c

The EnDat connector is electrically isolated from the rest of the RT Box. The GND line of the EnDat connector needs to be connected to the controller board.

Parameters

General

Resolution (bits)

Specifies the resolution of the EnDat position signal, in bits. The maximum resolution is 31 bits.

Advanced

Maximum clock pulse width (s)

Specifies the maximum duration of a positive or negative clock pulse. If a transfer is active and the clock line remains stable for longer than the specified time, the internal state machine of the EnDat receiver is reset

and the start of a new transfer is expected. The allowed range for this parameter is $10\mu s$ to $650\mu s$, the default is $100\mu s$.

Allow dynamic (dis-)connect
If enabled, an additional input port named **en** is shown. This port can be used to simulate a connection failure (see below).

Additional initialization code
Allows to add C code that is executed during the initialization of the EnDat block.

This parameter can be used to initialize the OEM parameter area using the command `plxWriteEndatMemory(uint8_t aMrsCode, uint8_t aAddress, uint16_t aValue)`, where

- `aMrsCode` is the MRS code of the OEM area (169, 171, 173 or 175),
- `aAddress` is the address within the OEM area (0-255) and
- `aValue` is the value to write to the address (0-65535).

Inputs

θ

The rotational angle or rotor position, in rad.

en

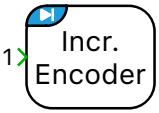
If the parameter **Allow dynamic (dis-)connect** is set to enabled, a boolean signal 0 on this port deactivates all EnDat communication, i.e. the RT Box no longer responds to EnDat commands. A boolean signal 1 on this port activates the EnDat operations.

Incremental Encoder

Purpose

Generate quadrature encoder signals on digital outputs.

Description



The Incremental Encoder takes a rotational angle θ as input and generates quadrature encoder signals with an optional index pulse on selected digital outputs of the RT Box. During an offline simulation the pulses are available on the outputs.

The RT Box contains two independent Incremental Encoder units.

Parameters

General

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.

Output when $\theta = 0$
Specifies the value of signals A and B when the input angle θ is zero and an index pulse is generated.

Offline implementation
With the setting **Continuous** θ and ω are made available in for offline simulation. With the setting **Quadrature Encoded** the quadrature signals A, B and I are made available as a vectorized signal.

Output

Digital encoder output [A] or [A, /A]
Indices of the digital outputs for signal A. If the signal is differential the value must be a vector with two elements where the second element specifies the output for the differential signal $\bar{A}$. For single-ended outputs the value must be a scalar or a vector with one element.

Digital encoder output [B] or [B, /B]
Indices of the digital outputs for signal B. If the signal is differential the value must be a vector with two elements where the second element specifies the output for the differential signal $\bar{B}$. For single-ended outputs the value must be a scalar or a vector with one element.

Digital index output [I] or [I, /I]

Indices of the digital outputs for index signal I. If the signal is differential the value must be a vector with two elements where the second element specifies the output for the differential signal $\bar{I}$. For single-ended outputs the value must be a scalar or a vector with one element. If the index signal output is not needed the value must be an empty vector.

Inputs

θ

The rotational angle or rotor position, in rad.

Nanostep Out

Purpose

Set a fast analog output from a Nanostep signal source on a RT Box 4.

Description



The output voltage at the unterminated BNC terminal is calculated as input*scale+offset, and is limited to the range -2V to 2V.

The output voltage range at the D-Sub terminal is configured in the Coder Target settings (see section “General” (p. 37)) to either unipolar or bipolar operation. In the bipolar configuration, the output voltage range is identical to the BNC terminal. In the unipolar configuration, the unterminated output voltage range is shifted by 2V resulting in output voltages from 0V to 4V.

For best signal quality and transmission, the output terminals must be terminated to ground with external resistors ($50\,\Omega$ for the BNC outputs, $100\,\Omega$ for the D-Sub outputs). This halves the output voltages.

For convenience, the offline simulation model includes configurable termination resistors, individually for both BNC and D-Sub terminals.

Parameters

General

Nanostep output channel(s)
Index of the Nanostep output channel. For vectorized output signals a vector of output channel indices must be specified. Valid values range from 0 to 5 to address the six Nanostep outputs of the RT Box 4.

Scale
A scale factor for the unterminated output signal. Proper termination halves the output signal.

Offset
An offset for the unterminated scaled output signal. Proper termination halves the output signal.

Offline only

Port visibility
Select which terminals are added to the subsystem for which code is generated. Enabled terminals are then available in an offline simulation and the block must be placed in the root schematic of the subsystem for which code is generated. If not enabled, no such terminals will be added and the block can be used at any subsystem level.

BNC termination

If “Terminated” is selected, an additional parameter for the “BNC termination resistance” is shown.

BNC termination resistance (Ohms)

Enter the termination resistance(s) in Ohms for the BNC terminals in the offline simulation. Only available when BNC termination is configured as “Terminated”.

D-Sub termination

If “Terminated” is selected, an additional parameter for the “D-Sub termination resistance” is shown.

D-Sub termination resistance (Ohms)

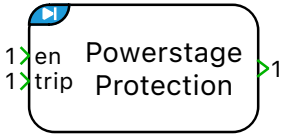
Enter the termination resistance(s) in Ohms for the D-Sub terminals in the offline simulation. Only available when D-Sub termination is configured as “Terminated”.

Powerstage Protection Unit

Purpose

Provide safety features for PWM outputs.

Description



The Powerstage Protection Unit enables PWM outputs to be switched to a safe state. The safe state is active while a logic high level is detected on one of the fault detection inputs (digital inputs or **trip** block input). The Powerstage Protection Unit also enters safe state at the start of the simulation if the **Reset behavior** parameter is set to **Rising edge at 'en' block input**, otherwise the initial state depends on the state of the fault detection inputs.

It is possible to configure whether normal operation should be resumed either on the rising edge of a model signal or automatically when the fault detection inputs become 0. The output of the Powerstage Protection Unit is 0 when the unit is in safe state, otherwise 1. The safe state of the PWM signals is configured in the PWM block.

An unconfigured Powerstage Protection Unit (for which there is no block in the simulation model) is automatically set to normal operating mode when the simulation is started. Since no inputs are configured, it never enters the safe state.

Parameters

Protection unit
The Powerstage Protection Unit to configure.

Tripping via block input
If set to **enabled**, safe state is reached when a non-zero signal is detected on the **trip** block input. The block input is hidden when this parameter is set to **disabled**.

Tripping via digital input(s)
Configures the digital inputs which are monitored. The Powerstage Protection Unit switches to safe state while a high input is detected on one of the digital inputs. The parameter can be either a scalar or a vector with up to three input signal indices. If left empty or an empty vector is specified, no digital input is monitored.

Digital input characteristic
Specifies whether an internal pull-up or pull-down resistor is connected to the digital inputs.

Reset behavior

Specifies when to resume normal operation. If set to **Rising edge at 'en' block input** the Powerstage Protection Unit waits for a rising edge on the **en** block input after all fault detection inputs are 0 before resuming normal operation. If set to **Automatic**, normal operation resumes when all fault detection inputs are 0, ignoring the **en** model input.

Inputs

en

A rising edge on the **en** block input resumes normal operation if all fault detection inputs (digital inputs and **trip** block input) are 0. This input is hidden when the parameter **Reset behavior** is set to **Automatic**.

trip

Any value other than 0 on the **trip** block input switches the Powerstage Protection Unit into safe state. This input is hidden when the parameter **Trip via block input** is set to **disabled**.

Programmable Value

Purpose

Output values received via the XML-RPC interface.

Description



The output of the Programmable Value block can be set via the XML-RPC interface. See “Sending data to the real-time simulation” (p. 53) for a description of XML-RPC commands to access the Programmable Value block.

An output value of 1 at the **v** port indicates that new data has been received via XML-RPC.

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.

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 simulation step is calculated.

The RT Box FPGA sample time of the incoming PWM is:

- 3.75 ns (clock frequency of 266 MHz) for RT Box 1 or CE;
- 3.33 ns (clock frequency of 300 MHz) for RT Box 2, 3 or 4.

Note that for step size larger than 15.3 μs on RT Box 1 or CE, and 13.6 μs on RT Box 2, 3 or 4, the following resolution will be used: 7.5 ns for RT Box 1 or CE, 6.66 ns for RT Box 2, 3 or 4.

Next for each multiple of these times, the sample rate changes again. This means, e.g. for RT Box 2, 3 or 4, for step size larger than 27.2 μs, 10 ns resolution is used; for step size larger than 40.8 μs, 13.3 ns resolution is used; etc.

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.

Offline behavior
Configures the averaging interval for the offline simulation to match the connected power module configuration. Possible values are **Sub-cycle average** for power modules that use sub-cycle averaging or **Nanostep** for power modules that employ a Nanostep solver.

Input characteristic
Specifies whether an internal pull-up or pull-down resistor is connected to the digital input.

PWM Out

Purpose

Generate a PWM signal on a digital output.

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

General

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). Note that the duty cycle resolution of the PWM generator is directly linked to the selected carrier frequency. Lower carrier frequencies result in reduced resolution. For details, see “Selected nominal carrier frequency” and “Duty cycle resolution” columns in tables Tab. 9.4 on p. 84 and Tab. 9.5 on p. 85 in the PWM Out (Variable) (p. 83) component.

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. If the block is placed inside of a task frame, the step size of the task frame will be used for synchronization.

Protection

Safe state

Specifies the state to which the output signal is set when the Powerstage Protection Unit is activated. A value of 0 indicates a low signal value (0 Volts), a value of 1 indicates a high signal value (3.3V or 5V, depending on the output voltage level). The parameter can be either a scalar or a vector with the same width as the number of output channels. The Safe state is also applied before the real-time simulation is started and after it is stopped (manually or due to a runtime error).

Powerstage protection unit

Specifies the Powerstage Protection Unit that is associated with this PWM block. If the Powerstage Protection feature is not needed, the option can be set to “None”. Otherwise, up to four Powerstage Protection Unit can be configured per model. If a Powerstage Protection Unit is selected that is not configured in the model, PLECS shows a warning when running an offline simulation or generating code. See Powerstage Protection Unit (p. 76) for details.

Advanced

Active polarity for turn-on delay

Specifies whether the turn-on delay is applied during the rising edge (1, default) or the falling edge (0) of the PWM signal. The polarity should be changed to 0 only for switches that are turned on when the output voltage is 0 Volts. This parameter has no influence on the polarity nor the safe state of the output signal. The parameter can be either a scalar or a vector with the same width as the number of output channels.

ADC trigger output

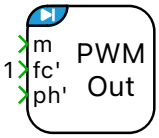
Allows to generate an ADC trigger output, either “On carrier minimum”, “On carrier maximum” or “On carrier minimum/maximum”. This block output is hidden when “Disabled” is selected. Otherwise, the ADC output is intended to get connected to a trigger input of an Analog In block.

PWM Out (Variable)

Purpose

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

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.

By enabling the "Variable turn-on delay mode" (in the advanced parameters section), the input signal ph' gets replaced with input signal d', that controls the turn-on delay (in seconds) for the individual PWM channels.

PWM generation limits

In order to understand the operational limits of the PWM generation an overview is given as follows:

The PWM generation ratings (i.e. maximum/minimum operating frequencies, maximum/minimum duty cycle resolution, etc.) depend on the selected nominal carrier frequency specified in the block parameters.

The PRD value of each PWM carrier signal is a 16-bit number. This sets a limit on the maximum PWM carrier period (minimum PWM carrier frequency) to a length of 2^{16} FPGA clock periods. Nevertheless, the maximum PWM carrier period can be increased, by increasing the FPGA clock period, with the use of a prescaler. The prescaler is selected as the minimum integer so that the nominal carrier frequency specified in the block parameters can be realised.

The minimum achievable PWM period (maximum PWM frequency) is set to be 20 FPGA clock periods. The maximum and minimum PWM carrier frequencies can be calculated as follows:

$$f_{\min} = \frac{1}{T_{\max}}, \quad T_{\max} = \frac{2^{\text{prescaler}}}{f_{\text{FPGA}}} \cdot 2^{16}.$$

$$f_{\max} = \frac{1}{T_{\min}}, \quad T_{\min} = \frac{2^{\text{prescaler}}}{f_{\text{FPGA}}} \cdot 20.$$

The duty-cycle, frequency, and phaseshift resolution will depend on the PWM carrier frequency being employed. The lowest resolution is obtained when operating at the highest PWM frequency. In case of a sawtooth carrier the lowest resolution is $1/20 = 5\%$, while the highest is $2^{-16} = 0.00153\%$. In case of a symmetrical carrier, the resolution is half of the sawtooth carrier resolution.

When frequency modulation is employed, the following table shows the maximum and minimum operating frequencies depending on the selected nominal carrier frequency for each of the RT Box models.

Table 9.4: RT Box 2/3/4 with 150 MHz FPGA clock frequency

Selected nominal carrier frequency	Minimum operating frequency	Maximum operating frequency	Duty cycle resolution
$2.288\,\text{kHz} < f_c < 7.500\,\text{MHz}$	2.288 kHz	7.500 MHz	6.667 ns (prescaler = 0)
$1.144\,\text{kHz} < f_c < 2.288\,\text{kHz}$	1.144 kHz	3.750 MHz	13.33 ns (prescaler = 1)
$572.2\,\text{Hz} < f_c < 1.144\,\text{kHz}$	572.2 Hz	1.875 MHz	26.67 ns (prescaler = 2)
$286.1\,\text{Hz} < f_c < 572.2\,\text{Hz}$	286.1 Hz	937.5 kHz	53.33 ns (prescaler = 3)
$143\,\text{Hz} < f_c < 286.1\,\text{Hz}$	143 Hz	468.7 kHz	106.7 ns (prescaler = 4)
$71.52\,\text{Hz} < f_c < 143\,\text{Hz}$	71.52 Hz	234.4 kHz	213.3 ns (prescaler = 5)
$35.7\,\text{Hz} < f_c < 71.52\,\text{Hz}$	35.7 Hz	117.26 kHz	426.7 ns (prescaler = 6)

Table 9.5: RT Box CE/1 with 133 MHz FPGA clock frequency

Selected nominal carrier frequency	Minimum operating frequency	Maximum operating frequency	Duty cycle resolution
2.035 kHz $< f_c <$ 6.667 MHz	2.035 kHz	6.667 MHz	7.5 ns (prescaler = 0)
1.017 kHz $< f_c <$ 2.035 kHz	1.017 kHz	3.333 MHz	15 ns (prescaler = 1)
508.6 Hz $< f_c <$ 1.017 kHz	508.6 Hz	1.667 MHz	30 ns (prescaler = 2)
254.3 Hz $< f_c <$ 508.6 Hz	254.3 Hz	833.3 kHz	60 ns (prescaler = 3)
127.2 Hz $< f_c <$ 254.3 Hz	127.2 Hz	416.6 kHz	120 ns (prescaler = 4)
63.6 Hz $< f_c <$ 127.2. Hz	63.6 Hz	208.3 kHz	240 ns (prescaler = 5)
31.8 Hz $< f_c <$ 63.6 Hz	31.8 Hz	104.2 kHz	480 ns (prescaler = 6)

The tables show the frequency values for prescaler values from 0 to 6. Since the prescaler is a 4-bit number, and its maximum value is 15, the minimum achievable operating frequency would be 0.0621 Hz for RT Box 1 and CE, and 0.07 Hz for RT Box 2, 3 and 4.

Note

Keep in mind that the prescaler is calculated once during the compilation process, and it is not modified during runtime. Therefore, in case of variable frequency operation, the nominal frequency should be selected as the lowest frequency required during operation.

PWM implementation details

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.

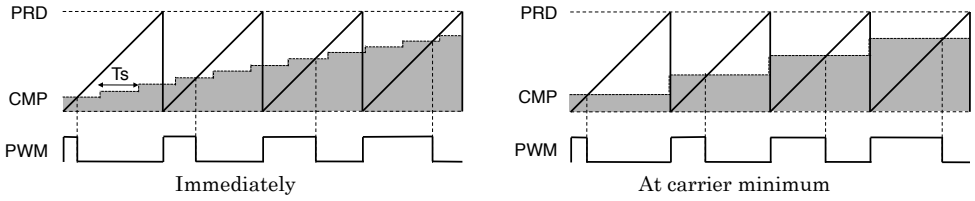


Fig. 9.1: Update schemes for modulation index with sawtooth carrier

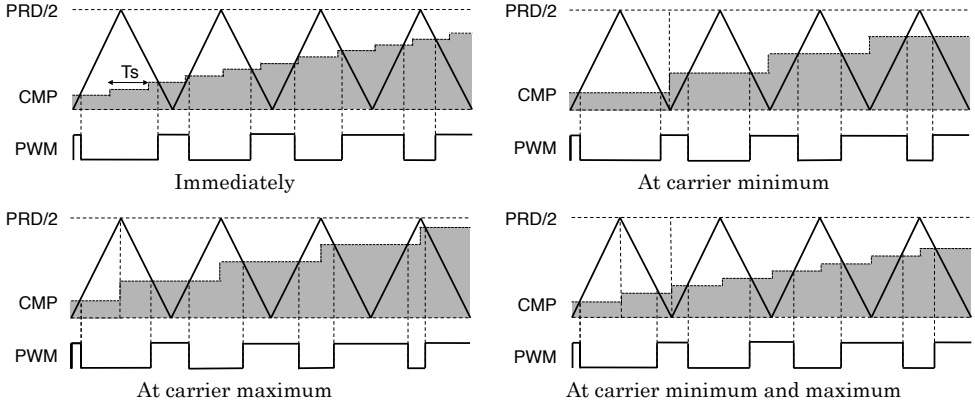


Fig. 9.2: Update schemes for modulation index with symmetrical carrier

The user can choose how often the CMP value is updated. If the parameter **Up-**

date 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.

If a PWM Out (Variable) block contains multiple output channels, the first channel automatically becomes the primary channel while the other channels are configured as secondary. Every time the ramp generator of the primary channel becomes 0, the primary channel transmits a synchronization signal to the secondary channels of the same block. When this happens, the ramp generators of the secondary channels are set to their initial values computed from the input signal ph' to achieve the desired phase shift.

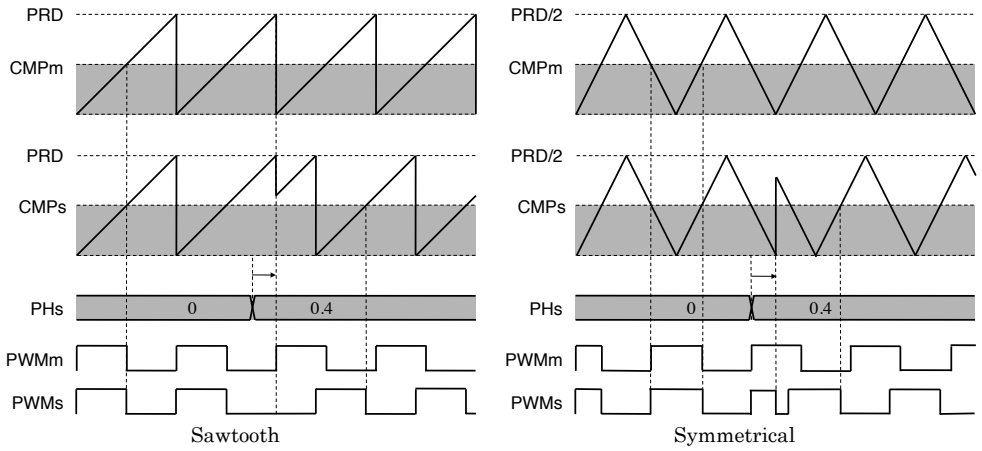


Fig. 9.3: Phase-shift variation with different carrier types

Note

The values at input signal ph' for *all the secondary channels* (i.e. all elements other than the first) represent their individual phase delay in p.u. *in relation to the primary channel*.
In addition, if required, the first element of the input signal ph' can be used to generate a phase delay for the *primary channel carrier in relation to the CPU simulation step* if the parameter **Synchronization with model step** is set to **Enabled**.
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 model step** of the RT Box.

With the synchronization signal from the primary channel, 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.

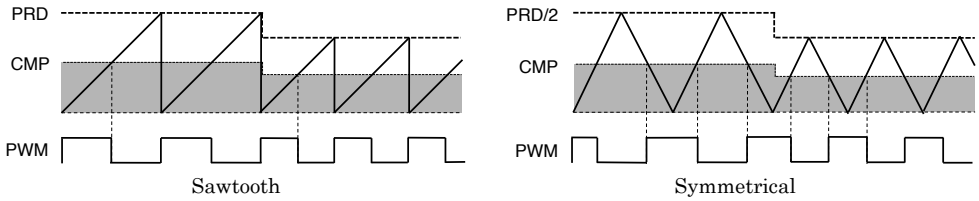


Fig. 9.4: Carrier frequency variation with different carrier types

Parameters

General

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 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. This field is only available when "Variable turn-on delay mode" is enabled.

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. This field is not available when "Variable turn-on delay mode" is enabled.

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. If the block is placed inside of a task frame, the step size of the task frame will be used for synchronization.

Protection

Safe state
Specifies the state to which the output signal is set when the Powerstage Protection Unit is activated. A value of 0 indicates a low signal value (0 Volts), a value of 1 indicates a high signal value (3.3V or 5V, depending on the output voltage level). The parameter can be either a scalar or a vector with the same width as the number of output channels. The Safe state is also applied before the real-time simulation is started and after it is stopped (manually or due to a runtime error).

Powerstage protection unit
Specifies the Powerstage Protection Unit that is associated with this PWM

block. If the Powerstage Protection feature is not needed, the option can be set to “None”. Otherwise, up to four Powerstage Protection Unit can be configured per model. If a Powerstage Protection Unit is selected that is not configured in the model, PLECS shows a warning when running an offline simulation or generating code. See Powerstage Protection Unit (p. 76) for details.

Advanced

Active polarity for turn-on delay

Specifies whether the turn-on delay is applied during the rising edge (1, default) or the falling edge (0) of the PWM signal. The polarity should be changed to 0 only for switches that are turned on when the output voltage is 0 Volts. This parameter has no influence on the polarity nor the safe state of the output signal.

Variable turn-on delay mode

When “Enabled” is selected, the input signal ph' gets replaced with input signal d', that controls the turn-on delay (in seconds) for the individual PWM channels. Additionally, the field “Minimal turn-on delay” allows to specify the lower limit, for safety reasons. The upper limit of the turn-on delay is about 0.4ms. Furthermore, a field to specify constant carrier phase shift per channel is available in the general parameter section.

Minimal turn-on delay

Minimal turn-on delay of the PWM output in seconds. This field is only available when “Variable turn-on delay” is enabled.

ADC trigger output

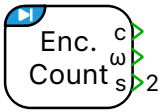
Allows to generate an ADC trigger output, either “On carrier minimum”, “On carrier maximum” or “On carrier minimum/maximum”. This block output is hidden when “Disabled” is selected. Otherwise, the ADC output is intended to get connected to a trigger input of an Analog In block.

Quadrature Encoder Counter

Purpose

Counts edges generated by a quadrature encoder.

Description



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.

The RT Box contains two independent Quadrature Encoder Counter units.

Parameters

General

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.

Counter reset condition
Selects the expected values of A, B and I to detect a counter reset.

Offline implementation
With the setting **Continuous** the inputs θ and ω are made available for offline simulation. With the setting **Quadrature Encoded** the block uses A, B and I as a vectorized input signal.

Input

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(s) for signal [A] or [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(s) for signal [B] or [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(s) for index signal [I] or [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.

Resolver In

Purpose

Generates an exciter voltage and calculates the rotation angle or speed based on sine- and cosine-modulated feedback signals.

Description



The *Resolver In* input of the RT Box 2, 3 and 4 is intended to be connected to a physical two-pole transmitter resolver. The input connects to an AD2S1210 resolver-to-digital converter from Analog Devices, Inc. The Resolver In connector on the RT Box provides a differential, sinusoidal exciter voltage with variable frequency and amplitude on the +EXCOUT and -EXCOUT pins. The differential feedback pins, +COSIN, -COSIN, +SININ and -SININ take amplitude modulated feedback signals from the physical resolver. The ratio of the input signals allows to calculate the angle (in rad) or speed (in rad/s) of the resolver. The Resolver In block configures the AD2S1210 converter.

The input voltages applied to the +COSIN, -COSIN, +SININ and -SININ input pins are required to be in the range of 2.3 V_{pp} to 4.4 V_{pp}. Each voltage referred to ground must be in the range of 0.15 V to 4.8 V.

The RT Box 2 or 4 provides one resolver input, and the RT Box 3 provides two of them. There is no resolver input available on the RT Box 1 or CE.

The minimum readout time for one resolver is around 2 μs. This time is doubled when both resolver inputs of an RT Box 3 are active. If the model step size is smaller than the resolver readout time, a value of 1 at the output **v** signals the arrival of valid data.

The settling time and the maximum tracking speed are resolution dependent:

Resolution	Maximum tracking speed	Settling time 10° step	Rec. exciter frequency
10 bit	5000 π/s	0.6 ms	10 - 20 kHz
12 bit	2000 π/s	2.2 ms	6 - 20 kHz
14 bit	1000 π/s	6.5 ms	3 - 12 kHz
16 bit	250 π/s	27.5 ms	2 - 10 kHz

The output **e** signals the error state of the resolver. It consists of 8 bit where each bit indicates a separate fault condition:

Bit	Description
D7	Sine/cosine inputs clipped
D6	Sine/cosine inputs below threshold
D5	Sine/cosine inputs overrange
D4	Sine/cosine inputs mismatch
D3	Tracking error
D2	Velocity exceeds range
D1	Phase error
D0	Configuration parity error

For more details consult the AD2S1210 datasheet, available from the Analog Devices website (https://www.analog.com).

Parameters

Resolver module
Specifies which resolver input to use. The second resolver input is available on the RT Box 3 only.

Measurement
Specifies if the block output is the position (in rad) or the speed (in rad/s).

Resolution
The resolver resolution can be 10, 12, 14 or 16 bit. The resolution influences the maximum tracking speed and the settling time (see table above).

Exciter frequency
The frequency of the exciter output voltage (see table above for recommended settings).

Exciter voltage
The amplitude (measured peak-to-peak) for the exciter voltage. The exciter voltage should be chosen depending on the attenuation of the connected resolver such that the SIN and COS feedback voltages are within their allowed range.

Resolver Out

Purpose

Emulates a two-pole transmitter resolver by modulating the amplitude of an external exciter voltage.

Description



The Resolver Out block emulates a two-pole transmitter resolver. A differential exciter voltage is fed into the RT Box and multiplied by the sine and cosine of the angle signal at the block's input terminal. The angle is measured in rad. The resulting output voltages are available as differential signals on the +COSOUT, -COSOUT, +SINOUT and -SINOUT pins at the **Resolver Out** connector. A common DC offset can be added to the differential output signals to adjust the voltage range for resolver-to-digital converter that only allow positive input voltages (as, for example, the AD2S1210 used for the Resolver In (p. 93) in the RT Box 2, 3 and 4).

Absolute maximum rating for +EXCIN and -EXCIN pin is -24 V to +24 V with respect to GND (electrical damage may occur if the maximum rating is exceeded). For correct functionality, neither +EXCIN nor -EXCIN pin may exceed -10 V to +10 V. Additionally, +EXCIN and -EXCIN input should be chosen such that $|(+\mathrm{EXCIN}) - (-\mathrm{EXCIN})|/2 * \mathrm{Gain} + \mathrm{Offset} < 10\,\mathrm{V}$ at all times (output will be incorrect otherwise).

Outputs are generated as follows:

$$+\mathrm{COSOUT} = ((+\mathrm{EXCIN}) - (-\mathrm{EXCIN}))/2 * \mathrm{Gain} * cos(\theta) + \mathrm{Offset}$$

$$-\mathrm{COSOUT} = -((+\mathrm{EXCIN}) - (-\mathrm{EXCIN}))/2 * \mathrm{Gain} * cos(\theta) + \mathrm{Offset}$$

$$+\mathrm{SINOUT} = ((+\mathrm{EXCIN}) - (-\mathrm{EXCIN}))/2 * \mathrm{Gain} * sin(\theta) + \mathrm{Offset}$$

$$-\mathrm{SINOUT} = -((+\mathrm{EXCIN}) - (-\mathrm{EXCIN}))/2 * \mathrm{Gain} * sin(\theta) + \mathrm{Offset}$$

The RT Box 2 or 4 provides one resolver output, and the RT Box 3 provides two of them. There is no resolver output available on the RT Box 1 or CE.

Parameters

Resolver module

Specifies which resolver output to use. The second resolver input is available on the RT Box 3 only.

Gain

The exciter voltage may be attenuated by the given factor in addition to the sine/cosine attenuation. The maximum resolver resolution is achieved with a gain of 1.

DC output offset

Specifies an additional offset voltage to be applied to the +COSOUT, -COSOUT, +SINOUT and -SINOUT signals. The value is specified in Volt.

Interpolation

If interpolation is enabled, the +COSOUT, -COSOUT, +SINOUT and -SINOUT signals are generated in smaller interpolated steps between simulation steps of the RT Box. If an FPGA simulation is running, the interpolation step size is identical to the FPGA simulation step size. If no FPGA simulation is running, the interpolation step size is automatically selected as the minimum integer fraction of the CPU discretization time that is supported by the RT Box.

SFP In

Purpose

Receives signals from another RT Box

Description



For distributed models, the SFP In block receives signals from an SFP Out (p. 98) 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” (p. 45) 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.

SFP Out

Purpose

Sends signals to another RT Box

Description



For distributed models, the SFP Out block sends the signals at its input terminal to an SFP In 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

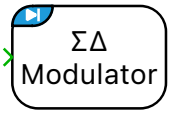
SFP port
The name of the SFP port which is connected to the remote RT Box.

Sigma-Delta Modulator

Purpose

Generate 2nd order sigma-delta modulator signals on digital outputs.

Description



RT Box 2,3 and 4 each has 8 sigma-delta modulators,grouped in 2 units. Each unit can output a CLK and up to 4 DATA signals on digital outputs. While unit 1 can output signals to digital output channels 0 to 7,unit 2 can output to digital output channels 16 to 23. This channel mapping is optimized for the RT Box LaunchPad-Nucleo Interface.

The modulator input range is from -1 to 1,but for stability reasons,dynamic signals should be kept below 70-90% full scale. The minimum and maximum modulator input settings allow such clipping.

The modulator input is calculated as input*scale+offset.

Parameters

Modulator unit
Select the desired modulator unit.

Modulator frequency
The desired frequency of the modulator in hertz (Hz).

Digital output channel for CLK
Index of the digital output channel used for the CLK signal. Can be left empty if a CLK signal is not needed.

Digital output channel(s) for DATA
Index of the digital output channel used for the DATA signal. For vectorized output signals a vector of output channel indices must be specified. A modulator unit supports up to 4 DATA output elements. All 4 modulator channels run at the same modulator frequency.

Offline simulation
Enables or disables data inspection in an offline simulation. If set to **enabled**, terminals are added to the subsystem for which code is generated. In an offline simulation the CLK and DATA output signals are available at these terminals. The block must be placed in the root schematic of the subsystem for which code is generated. If set to **disabled**, no such terminals will be added and the block can be used at any subsystem level.

Scale
A scale factor for the modulator input signal.

Offset
An offset for the scaled modulator input signal.

Minimum modulator input
The lowest value that the modulator input signal can reach. Cannot be lower than -1.

Maximum modulator input
The highest value that the modulator input signal can reach. Cannot exceed 1.

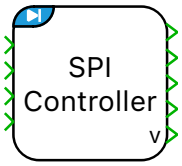
SPI Controller

Purpose

Implement SPI communication via digital outputs/inputs.

SPI enables synchronous communication with peripheral devices. The SPI Controller block provides a clock signal which generates a configurable number of clock pulses. 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.

Description



The RT Box contains two independent SPI modules that can be configured either as an SPI controller or SPI peripheral. When acting as an SPI controller the module provides a single clock and a single chip select signal. The chip select signal is held low during the data transfer. Each SPI module can receive and transmit data on up to four data lines in parallel.

An output value of 1 at the **v** 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 **v** port is set to 0 and the data 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 **v** port is always 1.

The RT Box contains two independent SPI units. Each unit may be used as SPI Controller or SPI Peripheral.

Parameters

Setup

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

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:

Table 9.8: 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 and received first.

Skew-matched clock input
Enables the clock input for clock feedback. Typically required if the signal delay from the controller 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 used.

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

Sample time
The sample time determines how often data is sent (and received) on the SPI bus. It can be set to an integer multiple of the base step size. Use 0 to transfer data in every simulation step and -1 to inherit the sample time from the blocks feeding data into the SPI block.

Input

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

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

Output

Digital output channel for SPI clock
Clock output channel.

Digital output channel for CS
Chip select output channel.

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

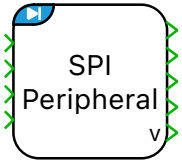
SPI Peripheral

Purpose

Implement SPI communication via digital outputs/inputs.

SPI enables synchronous communication with peripheral devices. Data is read and written on the edges of the clock signal while the chip select input is low. For high-speed communication, the internally processed clock signal, which is synchronized to the output data, can be provided on an output.

Description



The RT Box contains two independent SPI modules that can be configured either as an SPI controller or SPI peripheral. When acting as an SPI peripheral the module provides a single clock input and a single chip select input. The chip select signal must be set to low during the data transfer.

Each SPI module can receive and transmit data on up to four data lines in parallel.

The operation of the SPI peripheral is not synchronized with the simulation. A new SPI cycle is started with the falling edge of the chip select input signal. During the SPI cycle, the device sends the data that was last received at its model inputs. To ensure data integrity, once the SPI cycle is started, the send data cannot be changed by the simulation model until the SPI cycle is complete. After the SPI cycle is complete and the correct number of clock pulses have been received, the received data from the SPI bus will be available at the model outputs of the block in the next simulation step. The availability of new data is indicated by a 1 at the **v** connector. As long as no new data is received on the SPI bus, the previously received data is available at the model outputs of the block.

The RT Box contains two independent SPI units. Each unit may be used as SPI Controller or SPI Peripheral.

Parameters

Setup

SPI clock frequency (approx) [Hz]
The approximate frequency of the SPI clock. This value is used for an internal error correction

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:

Table 9.9: 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 and received first.

Skew-matched clock output
Enables a feedback clock output, typically required for SPI frequencies above 5 MHz if the connected SPI controller provides an input for a feedback clock. The feedback clock is synchronized with the data output signals.

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

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

Sample time
The sample time determines how often the SPI data is updated. It can be set to an integer multiple of the base step size. Use 0 to transfer data in every simulation step and -1 to inherit the sample time from the blocks feeding data into the SPI block.

Input

Digital input channel for SPI clock
SPI clock input.

Digital input channel for CS
Chip select input channel.

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

Output

Digital output channel for skew-matched SPI clock
SPI feedback clock output channel.

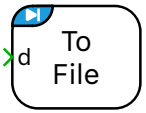
Digital output channel(s) for SPI data
Data output (MISO) channel/s as a scalar or vector. The width of this parameter corresponds to “Number of parallel data channels”. “-1” denotes an unused output.

To File

Purpose

Write signal values to a file.

Description



While a simulation on the RT Box is running the To File block continuously writes the values of its input signals to a file. The data is written in single precision format. The file format can be either a text file with comma separated values (csv) or a MATLAB data file (mat).

To ensure consistent data the simulation must be stopped before retrieving the data file.

On the RT Box 1 and CE files can only be stored on an external USB flash drive. On the RT Box 2, 3 and 4 files can be stored on the internal SSD or an external USB flash drive. For high data throughput a USB 3.0 device should be used. Removing a USB flash drive during the write process may result in inconsistent files, especially if the MATLAB format (mat) is selected.

To retrieve files from the internal SSD or an external USB flash drive, establish a WebDAV connection to `http://<rtbox name>/dav`. On Windows and Mac OS this can be done from the operating system directly. As an alternative, a web interface is available at `http://<rtbox name>/dav`. (Replace `<rtbox name>` with the hostname of your rtbox, e.g. `rtbox-20b0f70361c2.local`.)

Parameters

Filename

The name of the data file to write to. Files are stored in a new directory named after the model currently executed on the RT Box, with four digit number appended. The appended number is increased automatically on each simulation run. A file named `latest.txt` is created containing the newest directory name. The file is overwritten each time a simulation is started if it already exists.

The maximum file size is limited to `2 GB`. If the file size is exceeded, a new file with an increasing number appended is created automatically. File rotation may lead to data loss.

If the option **literal** is chosen, the string that is entered is taken literally as the basename of the data file. A suffix `.csv` or `.mat` is added depending on the file type. If the option **evaluate** is chosen, the string that is entered is interpreted as a MATLAB/Octave expression that must yield a string, which in turn is taken as the basename of the data file. If, for example, the value `['MyFile' num2str(index)]` is entered and the current workspace contains a variable `index` with a value of `2`, the file name will be `MyFile2.csv` or `MyFile2.mat`.

File Type
The file format to use for the data file. The file can be written as a text file with comma separated values (csv) or as a MATLAB data file (mat). Writing to a MATLAB data file is significantly faster compared to a csv file. The minimum `Sample times` for different RT Box versions are given in Tab. 9.10.

Table 9.10: Minmimum supported sample times for different RT Box versions

Target	.csv	.mat
RT Box 1/CE	100 μs	10 μs
RT Box 2/3/4	10 μs	2 μs

Data type
Selects whether the data is stored in single or double precision.

Write device
Selects whether to store files on the internal SSD (`Internal SSD`) or an external USB flash drive (`USB Flash Drive`).

Sample time
The input data will be written to the data file once in the given time period.

File rotation
If enabled, the current output file is closed and a new output file is created when a transition from a boolean **false** to a boolean **true** is detected at the rotation input port.

UDP Receive

Purpose

Receives data sent as UDP packets over the network.

Description



The block opens a UDP network port with the specified port number. If a packet with the required amount of data is received on the UDP port, the data is interpreted as the specified data type and made available on the **d** block output. The output **v** is **1** in each simulation step where new data is received, **0** otherwise.

The received data is interpreted as a contiguous block of data of the specified data type with no padding. Little endian byte order is assumed for integer data types. Packets that don't have the required data size are discarded.

Parameters

Port

The UDP port to open. Ports below 1024 are reserved or system services and should not be used.

Width

The number of values that are expected in the UDP packet.

Data type

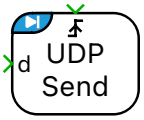
The data type of the data in the UDP packet.

UDP Send

Purpose

Sends data as UDP packets over the network.

Description



The block sends data to a network device using UDP packets. The data at the input port **d** is casted to the specified data type and packed contiguosly with no padding. Little endian byte order is used for integer data types.

Data can either be sent regularly with a fixed sample time or on demand when the trigger input changes. When configured for triggered execution, packets are sent when the trigger signal changes in the manner specified by the **Trigger type** parameter:

rising
: Data is sent when the trigger signal changes from 0 to a non-zero value.

falling
: Data is sent when the trigger signal changes from a non-zero value to 0.

either
: Data is sent when the trigger signal changes from 0 to a non-zero value or vice versa.

Parameters

Remote host name or IP address
: The hostname or IP address of the remote network device where the data is sent to. The IP address must be specified as 4 bytes separated by dots, e.g. **127.0.0.1**. If a host name is specified the RT Box must have access to a name server (DNS).

Remote IP port
: The port of the remote network device where the data is sent to. Ports below 1024 are reserved for system services and should not be used.

Data type
: The data type of the data in the UDP packet.

Execution
: Selects between **regular** and **triggered** execution.

Sample time
: The time period for regular packet transmission (for regular execution only).

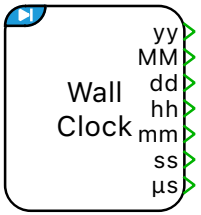
Trigger type
: The direction of the edges of the trigger signal upon which the data is sent, as described above (for triggered execution only).

Wall Clock

Purpose

Outputs the current time and date.

Description



The Wall Clock Block allows access to the real-time clock of the RT Box. It delivers the following output signals:

yy
: Current year

MM
: Current month

dd
: Day of month

hh
: Current hour

mm
: Current minute

ss
: Current second

μs
: Current microsecond

The RT Box 2, 3 and 4 contain an internal real time clock, which provides the time immediately after power-on. The clock is synchronized over the network if the RT Box can contact an NTP (Network Time Protocol) server. The RT Box 1 does not contain an internal real time clock. To get the current time, the box must contact an NTP server before the simulation is started.

To change the default NTP time server, place a file named **ntp.conf** in the directory **config/etc** of the SD card of the RT Box. The file should contain one or more lines with NTP server entries of the form

```
server servername iburst
```

where *servername* is the domain name or the IP address of the NTP server.

Parameters

Offset to UTC [h]
: The offset of the local timezone to UTC (Universal Time Coordinated), in hours. The RT Box does not provide automatic adjustment to daylight saving time.

plexim

electrical engineering software



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