

RT Box

DEMO MODEL

Demo Application for the XML/JSON-RPC Scripting Interface of the RT Box

Using Python or Matlab and a RLC circuit in PLECS

Last updated in RT Box Target Support Package 4.0.1

www.plexim.com

- Request a PLECS and PLECS Coder trial license
- Get the latest RT Box Target Support Package
- Check the PLECS and RT Box TSP documentation

1 Overview

This demo model is aimed at demonstrating the basic usage of the XML/JSON-RPC interface of the RT Box using a Python script or a Matlab script. The script features basic interactions such as:

- Uploading an executable to the RT Box
- Starting a real-time simulation
- Setting a Programmable Value block
- Reading back data from a Data Capture block

> **Note**
>
> This model contains model initialization commands that are accessible from:
> *PLECS Standalone:* the menu **Simulation > Simulation Parameters... > Initializations**
> *PLECS Blockset:* right click in the **Simulink model window > Model Properties > Callbacks > InitFcn***

1.1 Required Software and Hardware

The following section lists the needed software and hardware to fully run this XML/JSON-RPC demo example.

Source Files

There are two main types of files included with this demo: a PLECS model (`.plecs` for Standalone and `.slx` for Blockset) and a script file (Python script named `rlc_network_scripting.py` and the Matlab script named `rlc_network_scripting.m`). These files can be found in the folder of this demo.

Generating an .elf File from PLECS

To generate a new executable and linking format (`.elf`) file, a PLECS and PLECS Coder license are needed. To request a trial license for these products, please visit www.plexim.com/trial[1].

To generate an `.elf` file, please follow the steps below:

- Open the PLECS Standalone or the PLECS Blockset model of interest.
- Select the appropriate **System** from the **Coder + Coder options...** window. Then, from the **Target** tab, select `PLECS RT Box 1`, `PLECS RT Box 2`, `PLECS RT Box 3` or `PLECS RT Box 4` as the target from the dropdown menu.
- Leave the **Target Device** field empty and click on the **Build** button. This generates an `.elf` file without uploading a model to a specific target device. The generated `.elf` file is placed within a folder called `rlc_network_scripting_codegen` at the same directory as the respective PLECS model file, by default.

For Executing a Python Script on the RT Box

Once the `.elf` file is generated, a license for PLECS or the PLECS Coder is no longer necessary to execute the Python script. An RT Box connected to the host computer via Ethernet is required.

> **Note**
>
> For Windows, Linux/UNIX, macOS, and other operating systems, Python 3.x can be newly installed or updated from https://www.python.org/downloads/.

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

For Executing a Matlab Script on the RT Box

Similarly to above, once the `.elf` file is generated, a license for PLECS or the PLECS Coder is no longer necessary. However, a valid Matlab installation with Matlab license is needed to execute the Matlab script. Also, An RT Box connected to the host computer via Ethernet is required.

2 Model

The modeled electrical system is a simple RLC network, as shown in Fig. 1. A capacitor is charged by a DC voltage source via an RL branch and its voltage and current are monitored using a Voltmeter and Ammeter, respectively. The target of this application is to find the maximum voltage that occurs during the transient when an input voltage step is applied.

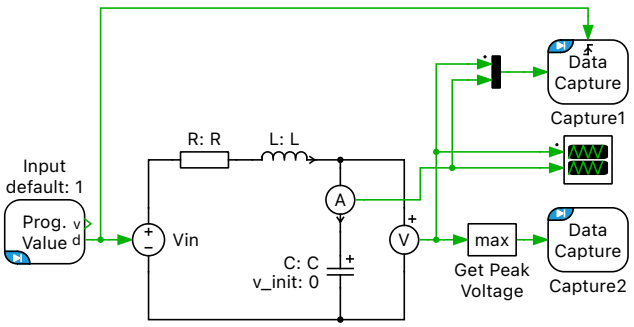


Fig. 1: RLC network with scripting

3 Scripting

The operation of the RT Box can be controlled over its inbuilt XML-RPC or JSON-RPC interface. An overview of the scripting interface is given in Fig. 2. It processes requests and sends back the required data using Extensible Markup Language (XML) or JavaScript Object Notation (JSON). XML/JSON-RPC are supported by a variety of programming languages, such as Python, MATLAB, C and Java. This particular example features basic interactions with the RT Box, such as loading an `.elf` file on the RT Box, starting a simulation, and reading data from or sending data to the real-time simulation. In general such a scripting environment can be used to implement automated testing procedures.

> **Note**
>
> The XML/JSON-RPC interface has a non-deterministic latency. Therefore it is not capable to perform time critical tasks such as control actions in closed-loop fashion, where time delay affects stability.

More information on the RT Box's XML/JSON-RPC interface can be found in the "Scripting" section[2] of the RT Box User Manual[3].

3.1 The Python Script

The following section explains selective parts of the Python script `rlc_network_scripting.py` that is used in this demo model.

The XML/JSON-RPC socket is set up to listen for incoming data on TCP port 9998. Before executing this script the RT Box hostname in the script must be changed from `rtbox-123.local` to the hostname

[2] https://docs.plexim.com/tsp/rtbox/latest/scripting/#rtbox-scripting
[3] https://www.plexim.com/sites/default/files/rtboxmanual.pdf

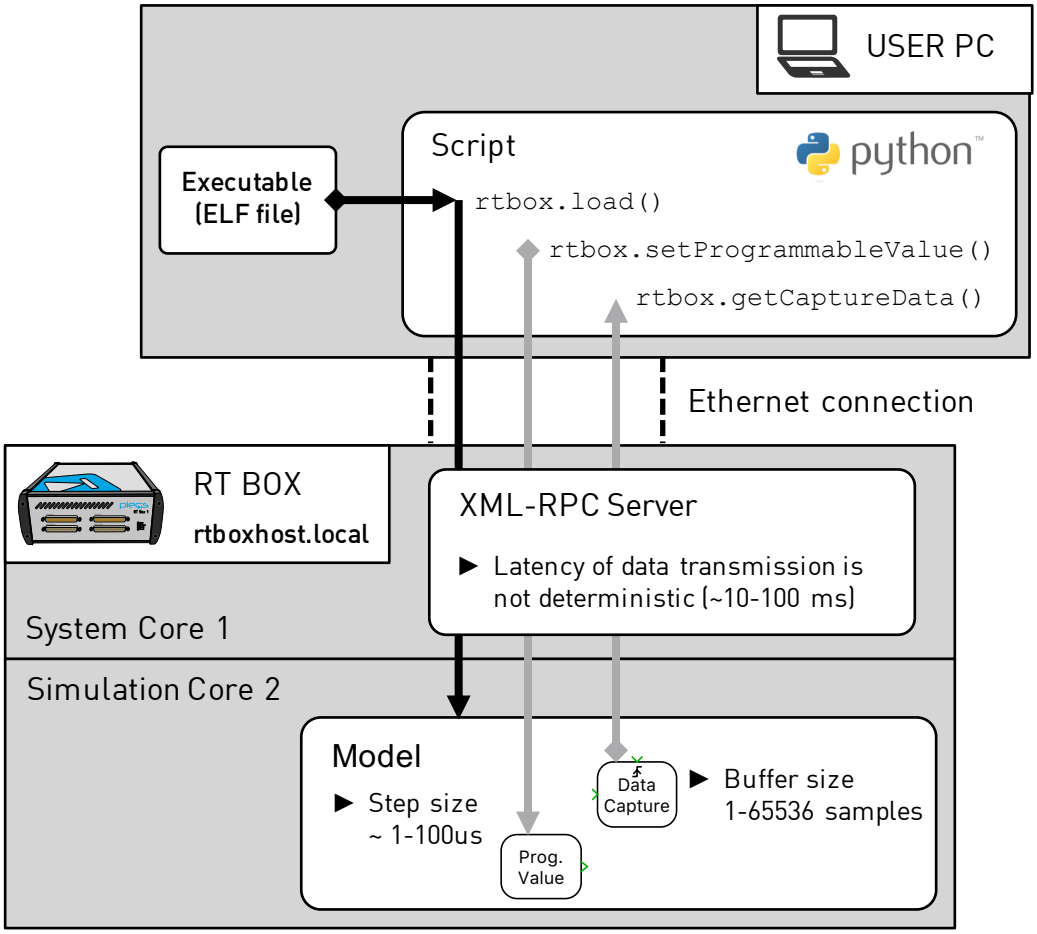


Fig. 2: XML-RPC interface of the RT Box connected to the host computer using Ethernet

of the RT Box that is used to run this example. At first the necessary modules are loaded and the model name and communication method are defined.

```
import socket
import time
import base64

HOST_NAME         = "rtbox-123.local"
ip                = socket.gethostbyname(HOST_NAME)
HOST_ADDRESS      = "http://" + ip + ":9998/RPC2"
MODEL_NAME        = "rlc_network_scripting"
METHOD            = "XML"    # choose "XML" or "JSON"
```

Then the generated `.elf` file is read and uploaded to the configured RT Box depending on the selected communication method, XML-RPC or JSON-RPC.

```
if METHOD == "JSON":
   import jsonrpc_requests
   import collections.abc      # to make jsonrpc_requests usable for Python 3.10+
   collections.Mapping = collections.abc.Mapping
   server = jsonrpc_requests.Server(HOST_ADDRESS)
elif METHOD == "XML":
   import xmlrpc.client
   server = xmlrpc.client.Server(HOST_ADDRESS)

with open("rlc_network_scripting_codegen/" + MODEL_NAME + ".elf", "rb") as f:
   print("Uploading executable")
   server.rtbox.load(base64.b64encode(f.read()).decode())
```

The following command starts the real-time simulation:

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

The commands `getProgrammableValueBlocks()` and `getDataCaptureBlocks()` query the Programmable Value and Data Capture blocks that are placed in the PLECS model, respectively, as shown in Fig. 1 on p. 3.

```
inputblocks = server.rtbox.getProgrammableValueBlocks()
outputblocks = server.rtbox.getDataCaptureBlocks()
```

The command `setProgrammableValue()` sets the value of the DC input voltage V_{in} from the specified initial value of 1 V in the PLECS model to 2 V.

```
Vin = 2
server.rtbox.setProgrammableValue('Input', [Vin])
```

In the parameter window of the "Capture1" block, a `rising` type **Trigger** with a **Trigger level** greater than 1 is specified. When the command `getCaptureTriggerCount()` reads a value greater than 0, the command `getCaptureData()` reads the input voltage step response across the RLC network from 1 V to 2 V.

```
while server.rtbox.getCaptureTriggerCount('Capture1')==0:
   print("Waiting for data")
   time.sleep(1)
data1 = server.rtbox.getCaptureData('Capture1')
data2 = server.rtbox.getCaptureData('Capture2')
```

The following code tabulates the captured data and stops the real-time communication with the RT Box server:

```
Vm = [row[0] for row in data1['data']]
Am = [row[1] for row in data1['data']]
VmMax = data2['data'][0][0]

server.rtbox.stop()
```

3.2 The Matlab Script

If you haven't installed Plexim's JSON-RPC client for Matlab before, please do so first.

Install the JSON-RPC client for Matlab

- Please visit Plexim's GitHub page https://github.com/plexim/matlab-jsonrpc and download the latest version of the JSON-RPC client by clicking on `Code` and then `Download ZIP`.
- Extract the `matlab-jsonrpc-main.zip` file.
- Open Matlab and add the folder `matlab-jsonrpc-main` that contains the JSON-RPC client to the Matlab Search Path (see for example this link[4] for an explanation on how to do this).

The structure of the Matlab script is very similar to that of the Python script. The following section explains only the first two special parts of the Matlab script `rlc_network_scripting.m` used in this demo.

```
HOST_NAME      = "rtbox-123.local";
HOST_ADDRESS   = "http://" + HOST_NAME + ":9998/RPC2";
MODEL_NAME     = "rlc_network_scripting";
```

Then the generated `.elf` file is read and uploaded to the configured RT Box using the JSON-RPC communication method.

[4] https://www.mathworks.com/help/matlab/matlab_env/add-folders-to-matlab-search-path-at-startup.html

```
%Initialize server and load executable
server = jsonrpc(HOST_ADDRESS);

f = fopen('rlc_network_scripting_codegen/' + MODEL_NAME + '.elf', 'rb');
fprintf("Uploading executable\n");
server.rtbox.load(matlab.net.base64encode(fread(f, '*uint8')));
fclose(f);
```

4 Simulation

The Python script can be executed in any Python IDE, or as described below. The m-file can be executed in Matlab environment. Either script loads the `.elf` executable on the RT Box and starts the real-time simulation.

Note

As described in section “Required Software and Hardware” (p. 2), please generate the .elf file before running the script. In addition, the RT Box hostname in the script must be changed from the default `rtbox-123.local` to the hostname corresponding to the RT Box used, as described in section “The Python Script” (p. 3).

4.1 Executing the Python Script

To access the Python script from either the Windows Command Prompt or Mac Terminal, change to the directory where the `.py` file is saved (i.e. inside the RT Box target support package folder). Here it is assumed that the folder is located on the Desktop.

For example, using the Windows Command Prompt:

```
cd C:\Desktop\PLECS_RT_Box\demos\rlc_network_scripting
```

Or using the Mac Terminal:

```
cd ~/Desktop/PLECS_RT_Box/demos/rlc_network_scripting
```

Then enter the following command to execute the script using Python 3.x in the Windows Command Prompt:

```
py -3 ./rlc_network_scripting.py
```

Or using the Mac Terminal:

```
python3 ./rlc_network_scripting.py
```

The output from the Python script `rlc_network_scripting.py` should then be as follows:

```
Uploading executable
Starting executable
Real-time simulation running
Available input blocks are:
['Input']
Available output blocks are:
['Capture1', 'Capture2']
Setting Vin as 2.00V
Waiting for data
Stopping executable
Real-time simulation stopped
Max value of Vm = 2.54V
```

4.2 Executing the Matlab Script

Run the `rlc_network_scripting.m` script in Matlab. The output should be very similar to that of the Python script.

5 Conclusion

This model demonstrates the basic working principle of the XML/JSON-RPC interface of the RT Box, including the Programmable Value and Data Capture blocks from the PLECS RT Box[5] component library.

6 Appendix

To view the Voltmeter and the Ammeter readings captured by "Capture1" block of Fig. 1 on p. 3, `matplotlib`, a freely available 2D plotting library for Python can be used. Please refer to the additional Python file named `rlc_network_scripting_matplotlib.py` included in the folder of this demo.

Fig. 3 displays the capacitor voltage and current waveforms when the input voltage of the RLC network is changed from 1 V to 2 V.

6.1 Installing matplotlib

Install `matplotlib` by entering the following commands:

Using the Windows Command Prompt:

```
py -3 -m pip install -U matplotlib
```

If you receive an error message that the `pip` module is unknown, you will need to install it first, before installing `matplotlib`:

```
py -3 -m pip install -U pip
```

Using the Mac Terminal:

```
python3 -m pip install -U matplotlib
```

6.2 Python Script

The script for plotting using Python is given below:

```
import matplotlib.pyplot as plt
plt.close('all')
x = [i * data1['sampleTime'] for i in range(0, len(data1['data']))]
fig, ax1 = plt.subplots()
ax1.plot(x, Vm, 'b')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Vm (V)', color='b')
ax1.tick_params('y', colors='b')
ax2 = ax1.twinx()
ax2.plot(x, Am, 'r')
ax2.set_ylabel('Am (A)', color='r')
ax2.tick_params('y', colors='r')
plt.title("Voltmeter and Ammeter Readings")
plt.show()
```

7 Bibliography

[1] *RT Box User Manual*, Plexim GmbH, Online: https://www.plexim.com/sites/default/files/rtboxmanual.pdf

[5] https://www.plexim.com/products/rt_box

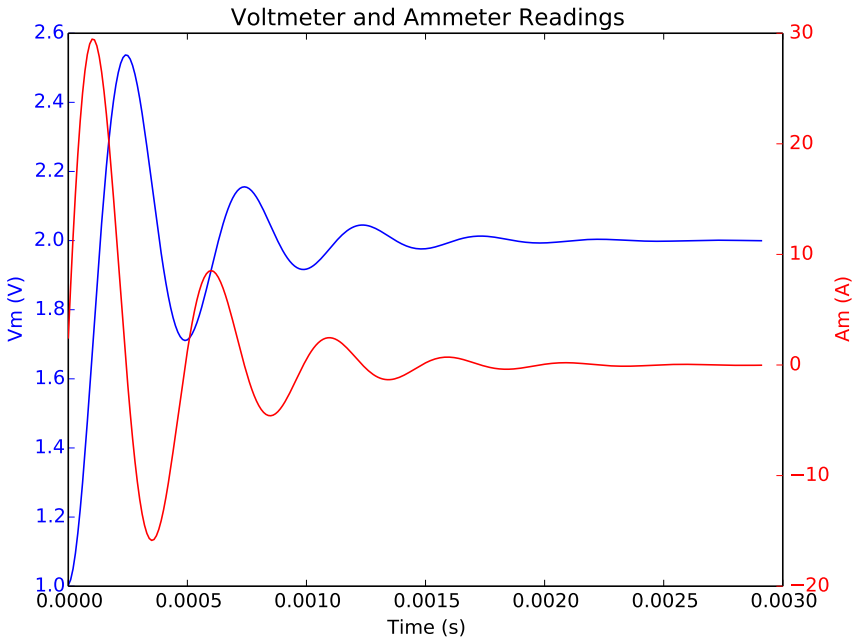


Fig. 3: Capacitor voltage and current waveforms

Revision History:

RT Box TSP 1.8.3	First release
RT Box TSP 2.2.1	Add JSON-RPC feature for Python scripting
RT Box TSP 4.0.1	Add the .m file to support JSON-RPC for Matlab scripting

How to Contact Plexim:

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

RT Box Demo Model

© 2002–2026 by Plexim GmbH

The software PLECS described in this document 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 is a registered trademark of Plexim GmbH. MATLAB, Simulink and Simulink Coder are registered trademarks of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.