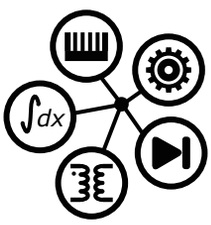


RT Box

DEMO MODEL

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

RT BoxのXML/JSON-RPCスクリプトインタフェースのデモアプリケーション

- PLECSでPythonとRLC回路を使用 -

Last updated in RT Box TSP 2.2.1

KESCO KEISOKU ENGINEERING SYSTEM
計測エンジニアリングシステム株式会社
https://kesco.co.jp

1 はじめに

このデモ モデルは、Pythonスクリプトを使用してRT BoxのXML-RPCインタフェースの基本的な使用方法を示すことを目的としています。このデモモデルには次の機能があります:

- RT Boxへの実行ファイルのアップロード
- リアルタイムシミュレーションの開始
- Programmable Valueブロックの設定
- Data Captureブロックからのデータの読み出し

注意 このモデルには、以下からアクセスできるモデル初期化コマンドが含まれています:
PLECS Standalone: **シミュレーション**メニュー -> **シミュレーション・パラメータ...** -> **初期化**
PLECS Blockset: **Simulinkモデルウィンドウ**で右クリック -> **モデル プロパティ** -> **コールバック** -> **InitFcn***

1.1 必要なハードウェハおよびソフトウェア

以下に、このXML-RPCのデモを完全に実行するために必要なソフトウェアとハードウェアを示します。

ソースファイル

このデモには、PLECSモデル(Standalone版の場合は`.plecs`、Blockset版の場合は`.slx`)と`rlc_network_scripting.py`とPythonスクリプトという2つのメインファイルが含まれています。これらのファイルは、PLECS Help BrowserのRT Box Demo Models sectionにあります。

PLECSから.elfファイルを生成

新しい実行可能ファイルとリンク形式(.elf)ファイルを生成するには、PLECSおよびPLECS Coderのライセンスが必要です。これらの製品の試用ライセンスをリクエストするには、www.plexim.com/trialにアクセスしてください。

`.elf`ファイルを生成するには、以下の手順に従ってください:

- 対象のPLECS StandaloneまたはPLECS Blocksetモデルを開きます。
- **Coder** -> **Coder オプション...**ウィンドウから適切な**システム**を選択します。次に、**ターゲット**タブで、ターゲットドロップダウンメニューから`PLECS RT Box 1`を選択します。
- **ターゲットデバイス**フィールドを空のままにして、**ビルド**ボタンをクリックします。これにより、特定のターゲットにモデルをアップロードせずに`.elf`ファイルを生成します。生成した`.elf`ファイルは、デフォルトでは、それぞれのPLECSモデルファイルと同じディレクトリにある`rlc_network_scripting_codegen`というフォルダ内に配置されます。

RT BoxでPythonスクリプトを実行するために

`.elf`ファイルが生成されると、Pythonスクリプトを実行するためにPLECSまたはPLECS Coderのライセンスは必要ありません。イーサネット経由でホストコンピュータに接続された RT Boxが必要です。

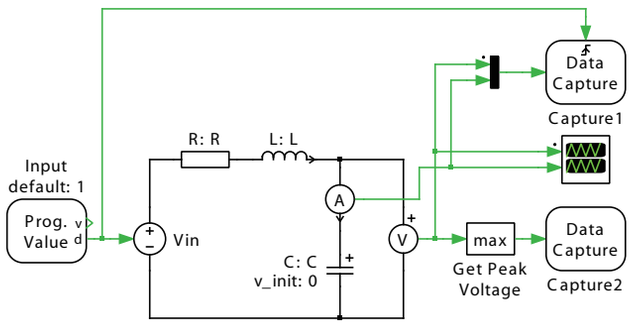
Pythonのインストール

Windows、Linux/UNIX、macOS、その他のオペレーティングシステムの場合、Python 3.xをhttps://www.python.org/downloads/から新規インストールまたはアップデートしてください。

2 モデル

モデル化した電気システムは、図1に示すように、単純なRLCネットワークです。キャパシタはDC電圧源からRL回路を介して充電され、その電圧と電流はそれぞれ**電圧計**と**電流計**を使用して監視しています。このアプリケーションの目的は、入力電圧ステップが印加されたときの過渡中に発生する最大電圧を検出することです。

図1: スクリプト機能付きRLCネットワーク



3 スクリプト

RT Boxの操作は、内蔵のXML-RPCまたはJSON-RPCインタフェースを介して制御できます。スクリプトインタフェースの概要を図2に示します。リクエストを処理し、拡張可能なマークアップ言語(Extensible Markup Language: XML)またはJavaScriptオブジェクト表記法(JavaScript Object Notation: JSON)を使用して必要なデータを返します。XML/JSON-RPCは、Java、Python、C、MATLABなどのさまざまなプログラミング言語でサポートされています。このデモでは、RT Boxへの`.elf`ファイルの読み込み、シミュレーションの開始、Python 3.xを使用してリアルタイムシミュレーションからのデータの読み取りやリアルタイムシミュレーションへのデータの送信など、RT Boxとの基本的なやり取りについて説明します。一般的に、このようなスクリプト環境は、自動テスト手順の実装に使用できます。

注意 XML/JSON-RPCインタフェースには非決定的な遅延があります。したがって、時間遅延が安定性に影響する閉ループ方式の制御動作など、時間的制約のあるタスクを実行することはできません。

RT BoxのXML/JSON-RPCインタフェースの詳細については、RT Boxユーザマニュアル[1]を参照してください。

3.1 Pythonスクリプト

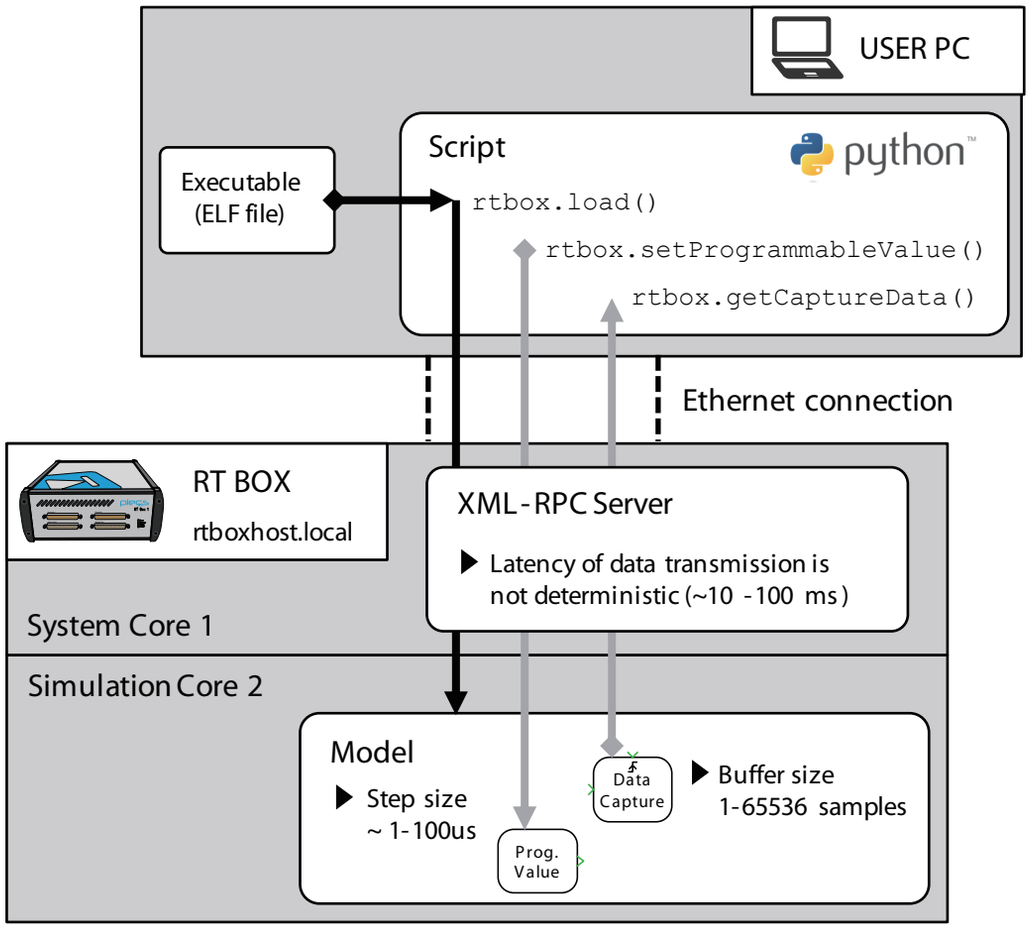
このセクションでは、このデモモデルで使用しているPythonスクリプトの選択された部分について説明します。

XML/JSON-RPCソケットは、TCPポート`9998`で受信データをリッスンするように設定されています。このスクリプトを実行する前に、スクリプト内のRT Boxホスト名を`rtbox-123.local`から、このデモの実行に使用するRT Boxのホスト名に変更する必要があります。最初に必要なモジュールをロードし、モデル名と通信方法を定義します。

```
import socket
import time
import base64

HOST_NAME  = "rtbox-123.local"
ip         = socket.gethostbyname(HOST_NAME)
```

図2: イーサネットを使用してホストコンピュータに接続されたRT BoxのXML-RPCインタフェース




```
HOST_ADDRESS      = "http://" + ip + ":9998/RPC2"
MODEL_NAME        = "rlc_network_scripting"
METHOD            = "XML" # choose "XML" or "JSON"
```

次に、生成された.elf ファイルが読み取られ、選択した通信方法(XML-RPCまたはJSON-RPC)に応じて構成されたRT Boxにアップロードされます。

```
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())
f.closed
```

次のコマンドはリアルタイム シミュレーションを開始します:

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

`getProgrammableValueBlocks()`コマンドと`getDataCaptureBlocks()`コマンドは、図1に示すように、それぞれPLECSモデルに配置されているProgrammable ValueブロックとData Captureブロックをクエリします。

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

コマンド`setProgrammableValue()`は、PLECSモデル内でDC入力電圧V_{in}の値を指定された初期値1Vから2Vに設定します。

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

"Capture1"のData Captureブロックのパラメータウィンドウでは、**Trigger level**が1より大きい`rising`の**Trigger type**が指定されています。コマンド`getCaptureTriggerCount()`が0より大きい値を読み取ると、コマンド`getCaptureData()`はRLCネットワークにおける入力電圧のステップ応答を1Vから2Vまで読み取ります。

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

次のコードは、RT Boxサーバとのリアルタイム通信を停止し、取得したデータを表形式で出力します:

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

server.rtbox.stop();
```

4 シミュレーション

Pythonスクリプトは、任意のPython IDEで実行することも、以下に説明するように実行することもできます。Pythonスクリプトは、RT Boxに`.elf`実行可能ファイルをロードし、リアルタイムシミュレーションを開始します。

注意 セクション1.1で説明したように、スクリプトを実行する前に`.elf`ファイルを生成してください。さらに、Pythonスクリプト内のRT Boxのホスト名は、セクション3.1で説明しているように、`rtbox-123.local`から、使用するRT Boxに対応するホスト名に変更する必要があります。

Pythonの実行

WindowsのコマンドプロンプトまたはMacのターミナルから Pythonスクリプトにアクセスするには、`.py`ファイルが保存されているディレクトリ(つまり、RT Box Target Support Packageフォルダ内)に変更します。ここでは、フォルダがデスクトップにあると想定しています。

Windowsのコマンドプロンプトを使用する場合:

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

Macのターミナルを使用する場合:

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

次に、Windowsのコマンドプロンプトで Python 3.xを使用してスクリプトを実行するには、次のコマンドを入力します:

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

Macのターミナルを使用する場合:

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

Pythonスクリプト"rlc_network_scripting.py"からの出力は次のようになります:

```
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
```

5 まとめ

このモデルは、PLECS RT Boxコンポーネント ライブラリのProgrammable ValueブロックとData Captureブロックを含む、RT BoxのXML/JSON-RPCインタフェースの基本的な動作原理を示しています。

6 付録

図1の"Capture1"のData Captureブロックで取得した電圧計と電流計の読み取り値を表示するには、Python用の無料で利用可能な2Dプロットライブラリである`matplotlib`を使用します。このデモのフォルダに含まれている`rlc_network_scripting_matplotlib.py`という追加のPythonファイルを参照してください。

図3は、RLCネットワークの入力電圧が1Vから2Vに変化したときのキャパシタの電圧と電流の波形を示しています。

matplotlibのインストール

次のコマンドを入力して`matplotlib`をインストールします:

Windowsのコマンドプロンプトを使用する場合:

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

`pip`モジュールが不明であるというエラーメッセージが表示される場合は、`matplotlib`をインストールする前に、まずそれをインストールする必要があります:

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

Macのターミナルを使用する場合:

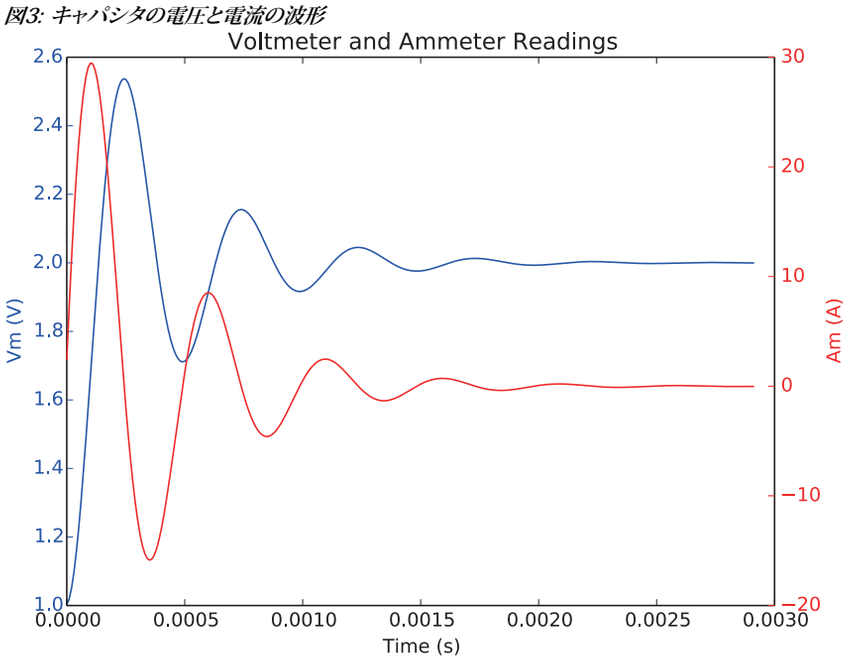
```
python3 -m pip install -U matplotlib
```

Pythonスクリプト

Pythonを使用してプロットするためのスクリプトを以下に示します:

```
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()
```

図3: キャパシタの電圧と電流の波形



7 参考文献

[1] *RT Box User Manual*, Plexim GmbH, Online: https://www.plexim.com/sites/default/files/rtboxmanual.pdf
日本語版: https://adv-auto.co.jp/products/plexim/manual.html

改訂履歴:

RT Box TSP 1.8.3　初版

RT Box TSP 2.2.1　JSON-RPC機能を追加

Pleximへの連絡方法:

☎ +41 44 533 51 00　Phone

+41 44 533 51 01　Fax

✉ Plexim GmbH　Mail
Technoparkstrasse 1
8005 Zurich
Switzerland

@ info@plexim.com　Email

http://www.plexim.com　Web

KESCO 計測エンジニアリングシステムへの連絡方法:

☎ +81 3 6273 7505　Phone

+81 3 6285 0250　Fax

✉ Keisoku Engineering Systm CO.,LTD.　Mail
1-9-5 Uchikanda, Chiyoda-ku
Tokyo, 101-0047
Japan

https://kesco.co.jp　Web

RT Box Demo Model