

RT Box

DEMO MODEL

Data Logging Demo

Different methods to log RT Box simulation data

Last updated in RT Box Target Support Package 2.2.1

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

An RT Box running a model in real time can connect the External Mode in order to visualize the running waveforms inside the PLECS Scope placed in the circuit.

Very often, users would like to log a range of simulation data for futher processing. The RT Box is capable of doing so via various protocols. This demo model showcases the following 5 methods:

- To File - writing .csv or .mat file to the internal SSD of an RT Box 2/3/4, or a plug-in USB stick on all RT Boxes,
- Data Capture - via XML/JSON-RPC protocol, the client is implemented in a Python script,
- UDP - via User Datagram Protocol, the client is implemented in a Python script,
- XCP - Universal Measurement and Calibration Protocol is an interface for read and write access to the memory of an ECU (Electronic Control Unit). CANape [1] from Vector Informatik is a widely used XCP master in the automotive industry.
- PLECS Scope - In the end, exporting waveform data from PLECS Scope captured under External Mode is also possible.

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 Requirements

- One PLECS RT Box[1] and one PLECS Coder[2] license.
- The RT Box Target Support Package[3] (minimum version 2.2.1).
- Follow the step-by-step instructions on configuring PLECS and the RT Box in the Quick Start guide of the RT Box User Manual[4].

2 Model

The circuit schematic is shown in Fig. 1.

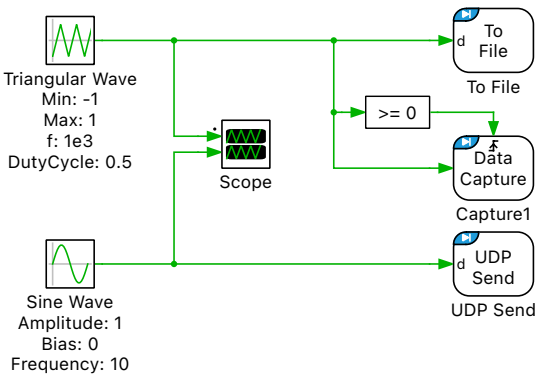


Fig. 1: Circuit schematic for the data logging demo

[1] https://www.plexim.com/products/plecs/rt_box
[2] https://www.plexim.com/products/plecs/plecs/plecs_coder
[3] https://plexim.com/download/rt_box
[4] https://www.plexim.com/sites/default/files/rtboxmanual.pdf

- To File block can continuously write data down to a sample step of several microseconds, reaching the RT Box discretization step size.
- Data Capture block transfers a data package where data points are at the RT Box discretization step size, but each data package has a limited number of samples.
- UDP block transmits data packets according to its configured sample time, which is usually in the range of several milliseconds.
- XCP transmits data at the RT Box discretization step size.

Below is a summary in Tab. 1:

Table 1: Comparison for RT Box data logging methods

Method	Sample Time (min.)		Buffer	Continuous Streaming	Physical Layer	Host Inferface (demonstrated)
To File	RT Box 1/CE	.csv $100\mu s$	No	Yes	-	-
		.mat $10\mu s$				
	RT Box 2/3/4	.csv $10\mu s$				
		.mat $2\mu s$				
Data Capture	RT Box disc. step		Yes	No	Ethernet	Python
UDP Send	1 ms		No	Yes	Ethernet	Python
XCP	RT Box disc. step		Yes	Yes	Ethernet	CANape
PLECS Scope (External Mode)	RT Box disc. step		Yes	Yes	Ethernet	PLECS

Therefore as a demonstration, a rather fast 1 kHz Triangular Wave is logged by To File block and Data Capture at $10\,\mu s$ sample time. Meanwhile a slowly-changing Sine Wave of 10 Hz is transmitted via UDP by a sample time of 10 ms. Both the Triangular Wave and the Sine Wave are connected to a PLECS Scope, therefore their values are also transmitted via XCP and PLECS Scope External Mode.

2.1 To File

This block works on

- RT Box 1 and CE - only to the external USB Flash Drive,
- RT Box 2, 3 and 4 - both to the Internal SSD or the external USB Flash Drive.

The To File block continuously writes the values to a file, while a simulation is running on the RT Box. The **File type** can be either `.csv` or `.mat` format.

The **Sample time** of this block defines the time step with which the input signals are written to a file. Different file formats implicates different minimum sample times on different RT Boxes. It is concluded in Tab. 1. For more details please refer to the Help page of the To File block.

The field **Write Device** of this block defines the medium to write files onto. A WebDAV connection can be established with the RT Box by accessing `http://<rtbox name>/dav` in a web browser. Replace `<rtbox name>` with the host name of your RT Box, e.g. `rtbox-20b0f7049846.local`. Fig. 2 shows the view of the folder structure via WebDAV access.

- Internal SSD

  Clicking into the `ssd` folder, Fig. 3 on p. 5 shows an example folder structure after executing and stopping this demo three times. The file named `latest.txt` specifies the directory name of the latest run. In this case, the file `data.csv` inside the folder `data_logging_0003` stores the data for the most recent run.

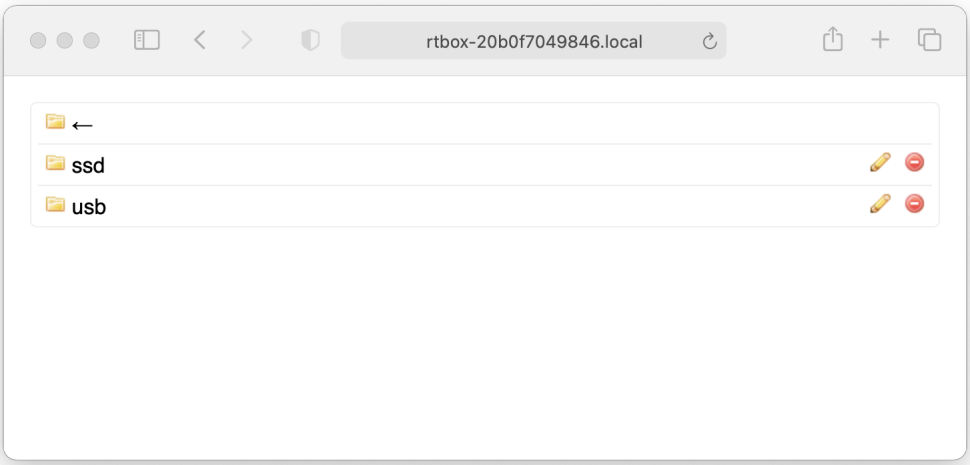


Fig. 2: The web interface accessing both the internal SSD and the plugged USB Flash Drive of an RT Box 2/3/4

- External USB Flash Drive
  Writing to a USB Flash Drive results in similar folder structure as shown in Fig. 3. One can access the USB content via WebDAV by clicking into the `usb` folder. One can also unplug the USB flash drive from the RT Box after stopping the real-time simulation and plug it onto a PC to browse the files.
  Note that for high data throughput a USB 3.0 device should be used.

In this demo, the 1 kHz Triangular Wave data is written as a `data.mat` file to the internal SSD of the RT Box 2/3/4 with a sample time of $10\,\mu s$. Build the model onto the RT Box, after a while, **Stop** the running from the RT Box Web Interface.

Check out the file `data.mat` inside the folder of the latest run. The file should have only one row of data, where an increment/decrement of 0.04 are observed between each sample step. And every 100 points complete a 1 kHz Triangular Wave period.

If the user has only an RT Box 1/CE, one should change the setting in the To File block as **Write Device:** `USB Flash Drive`, then build the model onto the RT Box 1/CE with an external USB stick plugged in. The generated `data.mat` should also have one row of data with the same increment/decrement of 0.04 between each data point.

2.2 Data Capture

The Data Capture block captures data from real-time simulations that are controlled by an external script.

All captured data by this block are written into an internal buffer first. The **Number of samples** in this block defines the size of the buffer. In this demo, the data capture starting point is triggered by the Triangular Wave counting up crossing 0 moment. The Data Capture buffer size is set as `50` in order to capture exactly one positive half cycle of the Triangular Wave.

> **Note**
>
> The number of samples multiplied by the signal width must not be larger than 65536.

A Python script named `data_logging_datacapture.py` can be found by clicking on “Folder” at the very top of this page. The Python code used is very similar to the Example Script inside the Help page of the Data Capture block.

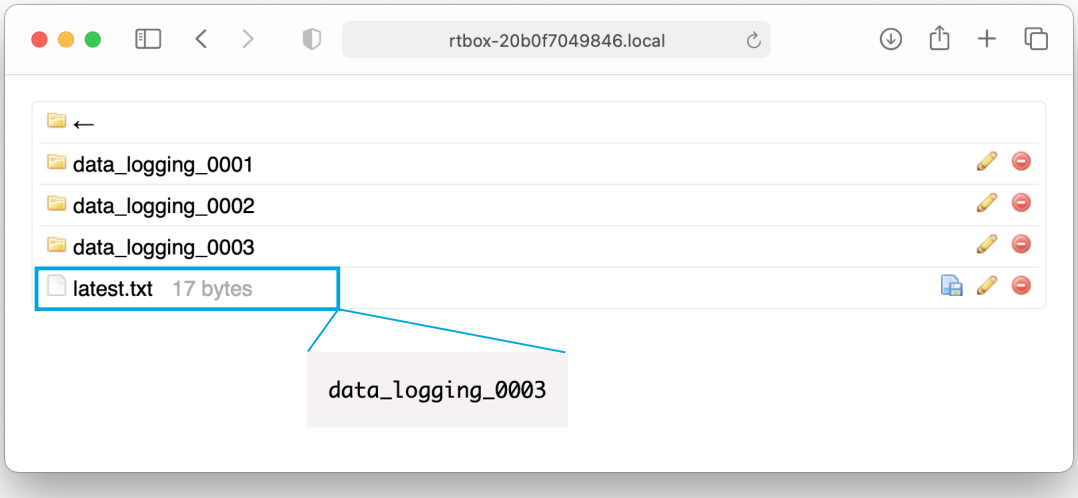


Fig. 3: The web interface accessing the internal SSD of an RT Box 2/3/4

Simulation

Before running the Python script, the user needs to change the following line of code to fit their own RT Box hostname.

```
HOST_NAME = "rtbox-123.local"
```

If the model has not been built onto the RT Box previously, please build it once before running the Python script.

Executing the Python script loads the pre-built RT Box executable (in `.elf` format) and starts the simulation. Once the Data Capture is triggered and it has captured 50 samples, the real-time simulation is stopped. The output from the Python script should look like the following:

```
Uploading executable
Starting executable
Real-time simulation running
Waiting for data
Stopping executable
Captured Data are:
[0.0, 0.040000000000000036, 0.08000000000000007, 0.1200000000000001,
0.15999999999999992, 0.19999999999999996, 0.24, 0.28, 0.32000000000000006,
0.3600000000000001, 0.40000000000000013, 0.43999999999999995, 0.48, 0.52,
0.56, 0.6000000000000001, 0.6400000000000001, 0.6799999999999999, 0.72, 0.76,
0.8, 0.8400000000000001, 0.8800000000000001, 0.9199999999999999, 0.96, 1.0,
0.96, 0.9199999999999999, 0.8799999999999999, 0.8400000000000001, 0.8, 0.76,
0.72, 0.6799999999999999, 0.6399999999999999, 0.6000000000000001, 0.56, 0.52,
0.48, 0.43999999999999995, 0.3999999999999999, 0.3599999999999999,
0.32000000000000006, 0.28, 0.24, 0.19999999999999996, 0.15999999999999992,
0.11999999999999988, 0.08000000000000007, 0.040000000000000036]
```

A total number of 50 data points are printed out in the Console. With a $10\,\mu s$ sample step capturing a 1 kHz Triangular Wave with a range of `[-1,1]`, an increment/decrement of 0.04 is observed between two consecutive samples.

2.3 UDP

The UDP traffic is typically used for streaming media applications where an occasional lost packet does not matter. Therefore it is suitable for transmitting slowly-varying signals in the real-time simulation.

In this demo, the RT Box sends UDP packets with the information of 10 Hz Sine Wave data at the **Sample time** of 10 ms. The RT Box sends UDP packets to a remote UDP client, whose IP address is defined in the **Remote IP address** field of the UDP Send block. In this application, it is the IP of the host PC connected with the RT Box. Both the **Remote IP address** and the **Remote IP port** configured in the UDP Send block has to be reflected correctly on the UDP client Python script side, where it writes:

```
UDP_clientIP = "10.0.0.103" #UDP client IP
UDP_clientPORT = 52345
```

In the UDP client Python script, the UDP packets are unpacked and the data are printed out in the Python Console. The Python script named `data_logging_udpclient.py` can be found by clicking on "Folder" at the very top of this page.

Simulation

First make sure the demo is built onto the RT Box and starts running.

Then execute the Python script, the output to the Python Console should continue as the following:

```
UDP received data: (4.7505016142600914e-14,)

UDP received data: (0.5877852439880371,)

UDP received data: (0.9510565400123596,)

UDP received data: (0.9510565400123596,)

UDP received data: (0.5877852439880371,)

UDP received data: (-4.7505016142600914e-14,)

UDP received data: (-0.5877852439880371,)

UDP received data: (-0.9510565400123596,)

UDP received data: (-0.9510565400123596,)

UDP received data: (-0.5877852439880371,)

UDP received data: (4.7505016142600914e-14,)

UDP received data: (0.5877852439880371,)
...
```

Since a 10 ms UDP transmission step is used to log a Sine Wave of 10 Hz, every 10 data points of the UDP received data represent one full Sine cycle.

2.4 XCP

XCP (Universal Measurement and Calibration Protocol) is a network protocol originating from ASAM (Association for Standardization of Automation and Measuring Systems) for connecting calibration systems to electronic control units, ECUs. The first letter X in XCP expresses the fact that the protocol is designed for a variety of bus systems [2].

RT Box 2, 3 and 4 can be used as an XCP slave to deliver measurements to an XCP master in real-time. In this demo, CANape is used as the XCP master.

PLECS model coder options

XCP can be enabled in the **Coder Options** dialog of this demo model. Go to **Target** tab, and tick the option **Enable XCP**. Afterwards in the field **XCP slave identity**, choose `Specify IP address`. Next in the field **XCP slave**, fill in the IP address of the RT Box 2/3/4 in use.

Click **Build**, the demo is built onto the RT Box and starts running. Inside the folder `data_logging_codegen` at the same directory as the demo file `data_logging.plecs`, a `data_logging.a2l` file is generated and is the one that needs to be imported into CANape environment.

> **Note**
>
> The signals transmitted over XCP protocol are the signals that are connected to a PLECS Scope in the PLECS schematic.

CANape configuration

CANape version 20.0 is used for demonstration below.

Create a new project. Browse to the `data_logging.a2l` file mentioned above in the Windows File Explorer. Drag the `data_logging.a2l` file, hoover over the folder called **Devices** under CANape view and then release the mouse.

Click the button **New network** as shown in Fig. 4.

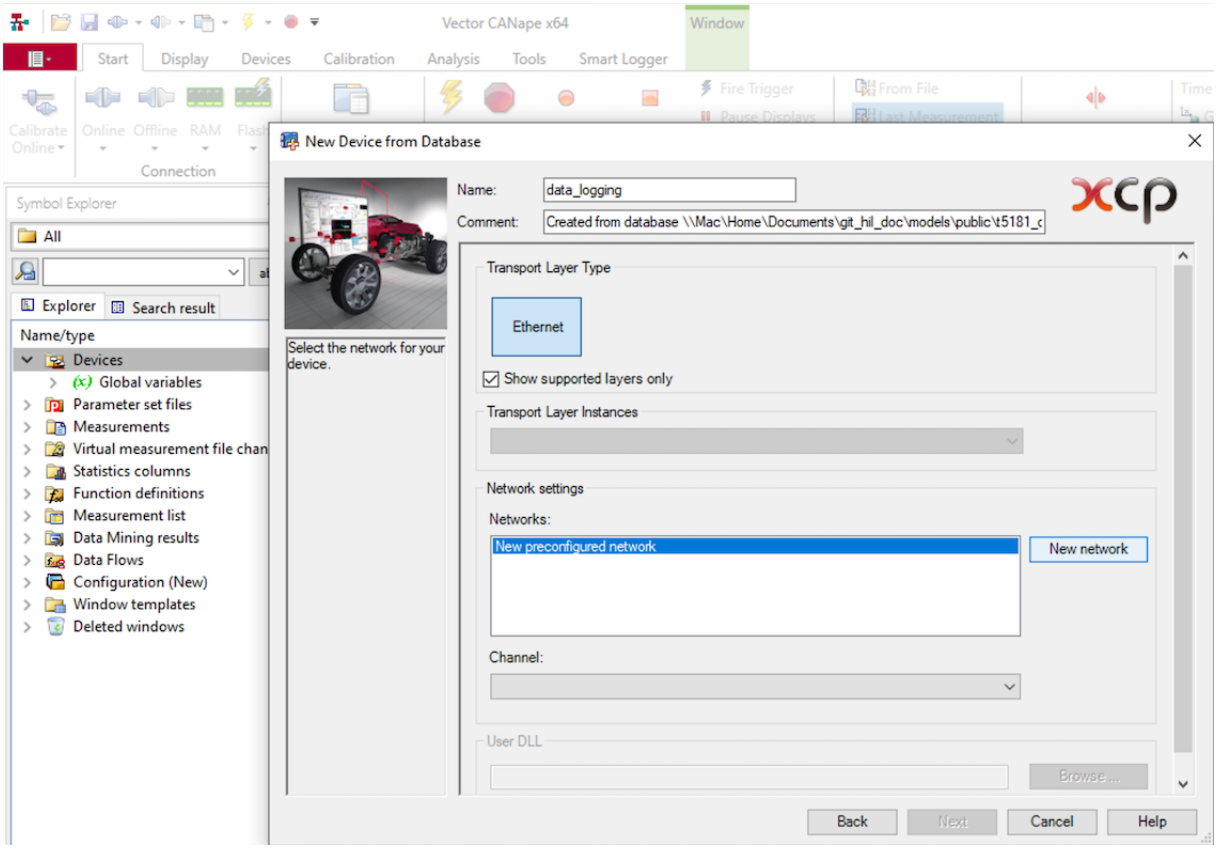


Fig. 4: Create new network in CANape

In the popped-up `Settings for ETH_Network` window, use all default settings as shown in Fig. 5, and then click **Close window accept changes** at the left up corner. Keep clicking **Next** and finally **OK** until the import process is finished.

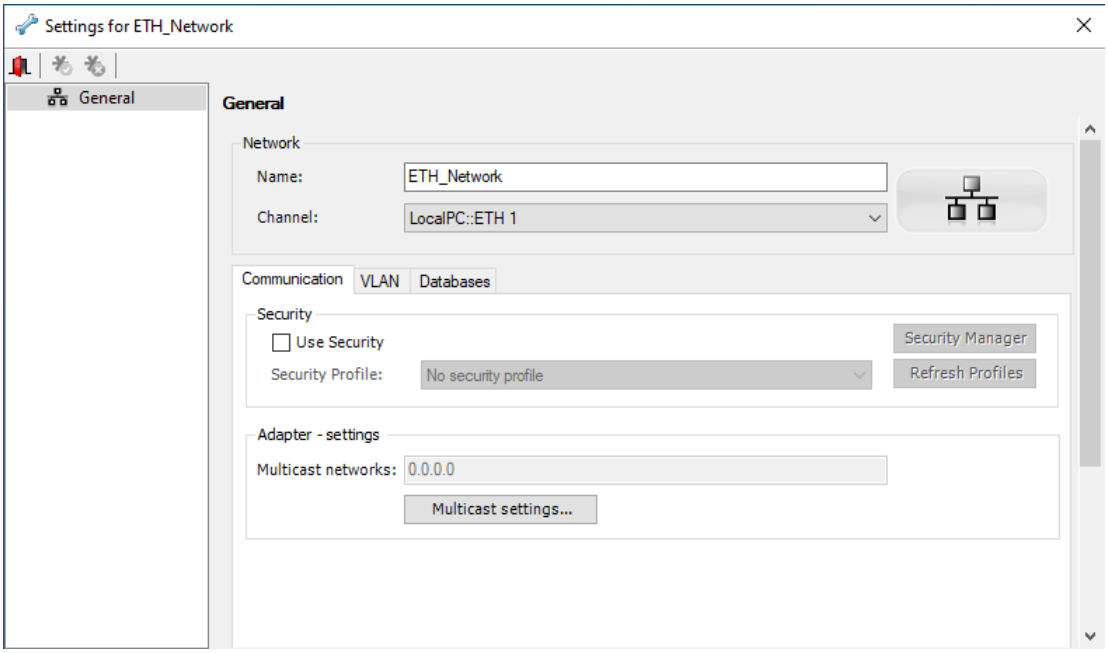


Fig. 5: Default settings for the new network in CANape

A window named `Settings for data_logging` will pop out. The following two attributes need to be changed.

Go to Protocol + Event List + Expert settings, the field **TIME_CORRELATION_GETDAQCLK** needs to be changed into `Extended response` from the default `Multicast`, shown as Fig. 6.

Go to Protocol + Transport Layer + Expert settings, the field **COUNTER_HANDLING** needs to be changed into `Exclude command response` from the default `Include command response`, shown as Fig. 7 on p. 10.

In the end, click **Close window accept all changes** at the top left corner of the window.

Next, double click in the Project Explorer **Devices** + **data_logging** + **data_logging.a2l** + **Scope.Plot_1** + **Triangular_Wave**, and double click on the **Devices** + **data_logging** + **data_logging.a2l** + **Scope.Plot_2** + **Sine_Wave** so that both signals are added to the Graphic window in CANape.

Up to now the preparation work in CANape is done. Make sure the model is running properly on the RT Box if it hasn't started running previously. Click **Start** in the menu bar of CANape, waveforms of the two signals should start streaming in real-time. One can adjust the Y-axis and time-axis minimum/maximum values, so that the frequency and the amplitude of the two signals are easily readable.

Click **Start Recording** button in the menu bar, and after a while click **Stop Recording**. The data in between are now recorded and ready to be exported.

Fig. 8 on p. 11 gives an example of the recorded Triangular and Sine Wave data. The data points for both the Triangular and the Sine wave are captured at the RT Box discretization step size, i.e. $10\,\mu$s.

2.5 PLECS Scope

After building the model onto the RT Box, one can **Connect** the **External Mode**, and **Activate autotriggering**.

In the **External Mode** tab of the Coder Options dialog, the **Number of samples** is set as `5000` in this demo. This number defines how many data points are shown on the PLECS Scope in external mode, with the RT Box discretization step size ($10\,\mu$s in this demo) between each point. Therefore the full length of the Scope Time-axis is $10\,\mu s \cdot 5000 = 0.05\,s$. Fig. 9 on p. 11 depicts the result.

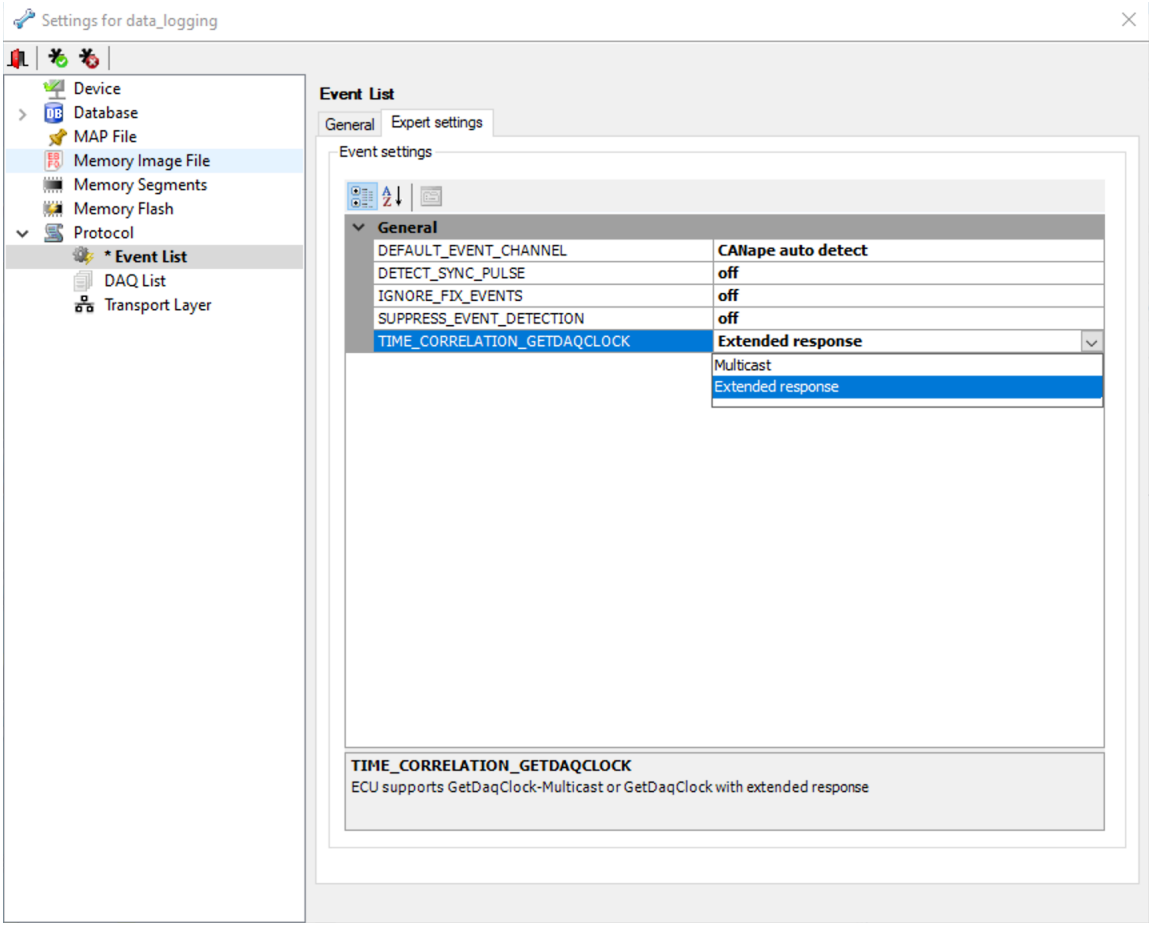


Fig. 6: Change in Protocol + Event List + Expert settings of the project settings in CANape

One can also consider using an integer number `n` in the **Decimation** field to downsample at every `n` RT Box discretization steps, so that with the same **Number of samples** the effective PLECS Scope time-axis is longer.

Click **Stop autotriggering** under **External Mode**, so that the data in PLECS Scope are not updating any more. Next, under the view of the Scope, go to menu **File + Export + as CSV + All...** and save as, for example `data.csv`. In the generated csv file the first column stores the Time-axis value at every RT Box discretization step size (i.e. $10\,\mu s$), followed by a second column of the Triangular Wave data, then a third column of the Sine Wave data.

3 Conclusion

This demo showcases the possible ways of logging real-time simulation data running on the RT Box.

4 Bibliography

[1] CANape, [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/canape. [Accessed: Feb. 16, 2022].

[2] ASAM MCD-1 XCP, [Online]. Available: https://www.asam.net/standards/detail/mcd-1-xcp/. [Accessed: Feb. 18, 2022].

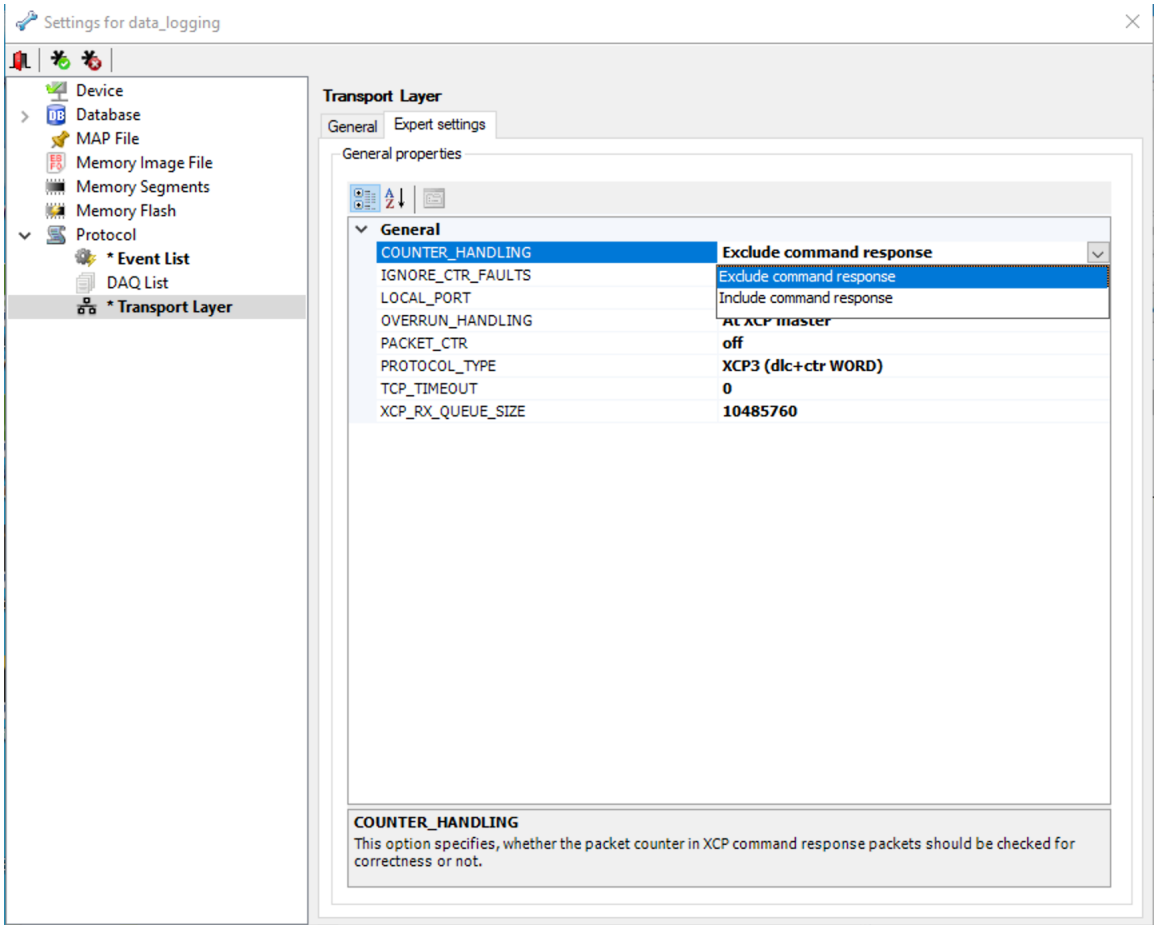


Fig. 7: Change in Protocol + Transport Layer + Expert settings of the project settings in CANape

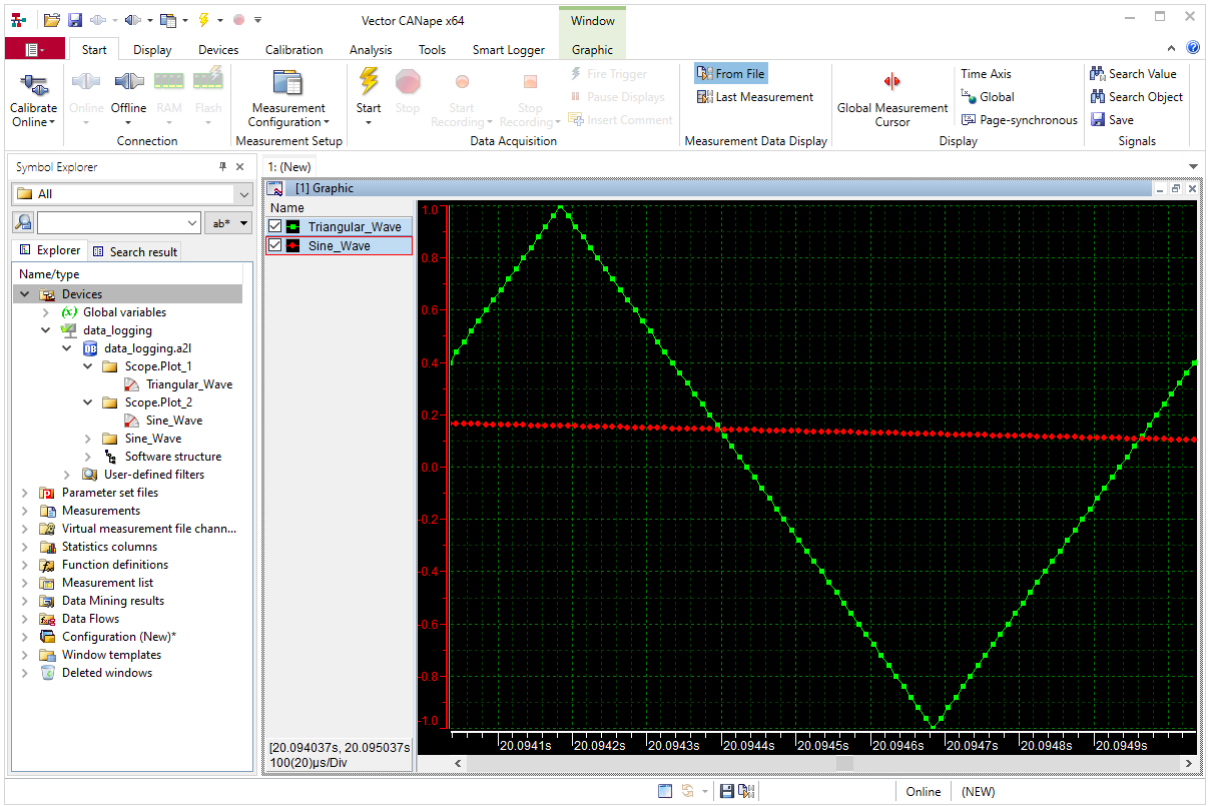


Fig. 8: Recorded real-time simulation result in CANape via XCP protocol

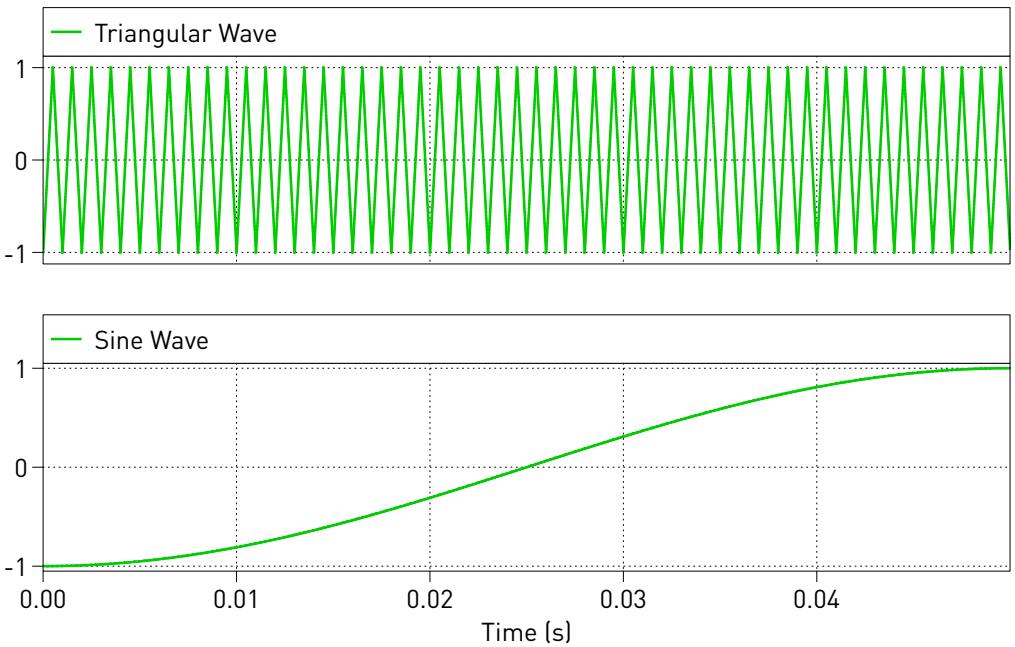


Fig. 9: PLECS Scope waveforms under the RT Box external mode

Revision History:

RT Box TSP 2.2.1	First release

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.