

RT Box

DEMO MODEL

CAN Channel Loop-Back Demo on a Single RT Box

Last updated in RT Box Target Support Package 2.0.5

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

A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each without a central host computer. A CAN connector is available on RT Box 2, 3 and 4, and since hardware revision 1.2 of the RT Box 1. The connector drawing and pin-out can be found in the CAN interface section of the RT Box User Manual[1]. There are two CAN channels available on the connector, each having a pair of High (CAN_H) and Low (CAN_L) signals. This demo model shows:

- a simple loop-back scenario that connects the two CAN channels,
- how to use data base CAN (`.dbc`) files to configure the CAN Pack and Unpack blocks,
- how to use the valid port of the Can Receive block to trigger a calculation upon arrival of new data.

1.1 Requirements

To run this demo model, the following items are needed (available at www.plexim.com[2]):

- One PLECS RT Box[3] and one PLECS Coder[4] license. For the RT Box 1 the minimum hardware revision is 1.2.
- The RT Box Target Support Package[5]
- Follow the step-by-step instructions on configuring PLECS and the RT Box in the Quick Start guide of the RT Box User Manual[6].
- Wires to connect the CAN_H pin of channel 1 and 2, and CAN_L pin of channel 1 and 2 together. See Fig. 1 for an illustration.

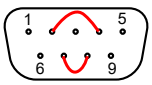


Fig. 1: Loop-back wiring of CAN channel 1 and 2 on the rear panel of the RT Box

2 Model

This demo model shows the CAN transmitting and receiving functions on two different channels inside the same subsystem. With the wiring setup shown in Fig. 1, a loop-back of the CAN message between two channels is formed. The circuit schematic is shown in Fig. 2. The CAN Pack block generates a CAN message with PLECS signals as its inputs. Next, it interfaces with the CAN Transmit block to send out the CAN message. On the receiver side, after the CAN Receive block a CAN Unpack block is used to unpack the CAN message into PLECS signals that are displayed in a scope.

Note

The CAN Pack block forces data type casting on the signals feeding into it. Data type inconsistencies between the “Output data type” of the feeding signals and the signal data type defined inside the CAN Pack block should therefore be avoided.

2.1 CAN Transmitting

The CAN Pack block generates a CAN message by packing the input signals into a byte vector (see . Fig. 3). The **CAN ID** field specifies the ID of the CAN message. In this demo, it is set as `0x1`. The CAN

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

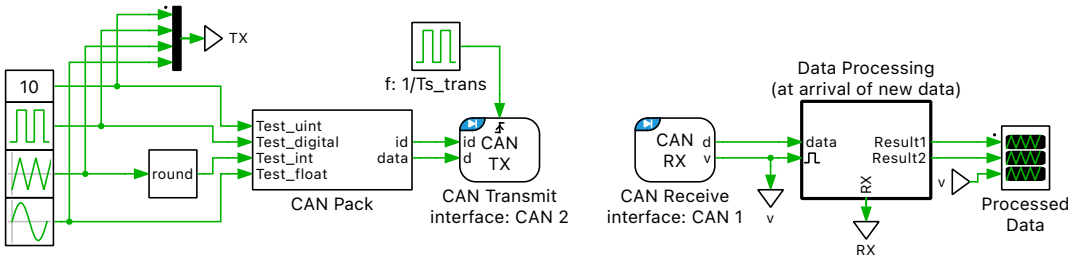


Fig. 2: Subsystem circuit schematic for CAN loop-back demo

ID can be supplied as either an 11-bit value (for CAN 2.0A compliance) or 29-bit value (for CAN 2.0B compliance). The **id** terminal of the block must be connected to the **id** terminal of the CAN Transmit block to use this parameter in the generated CAN message. A signal in a CAN message is defined by its data type, its byte order (Little Endian / Big Endian) and its start bit and length within the 64-bit CAN message. A signal can be scaled and an offset can be applied to efficiently send floating point signals as integers. The bits within the CAN message are numbered from 0 (least significant bit of the first byte) to 63 (most significant bit of the last byte).

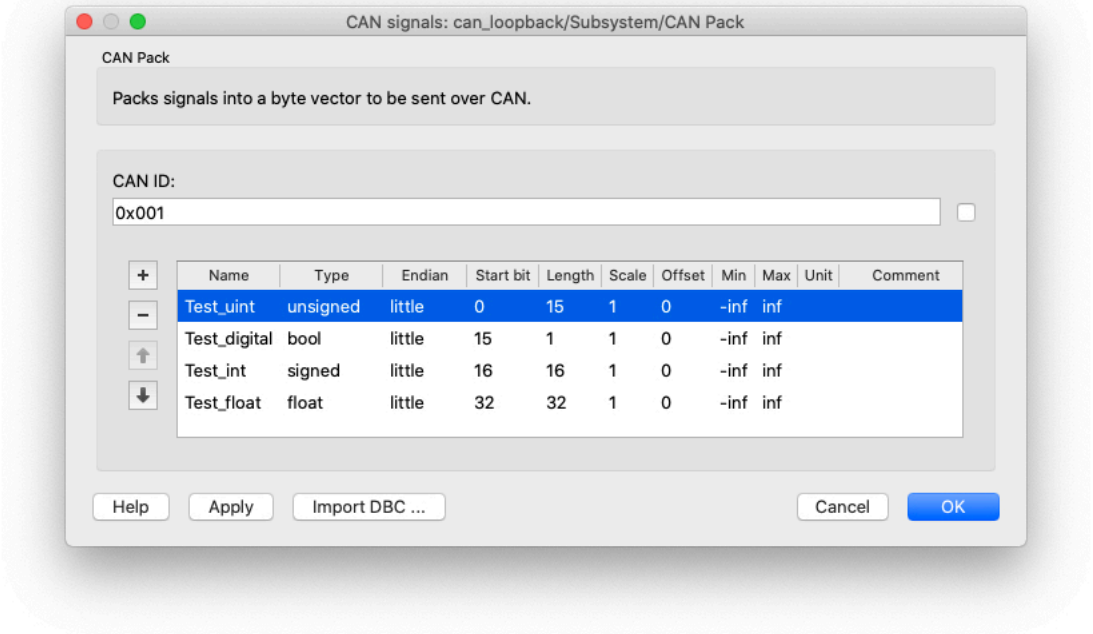


Fig. 3: Mask content of the CAN Pack block

The signal definitions can optionally be imported from a DBC (data base CAN) file by clicking the **Import DBC ...** button. Then browse to the `can_loopback.dbc` file provided with this demo model and click **Open**. One should see the same CAN message signal definitions as originally listed in CAN Pack block. Clicking **Import** will finish the import DBC process.

Most CAN networks are proprietary in the sense that only the Original Equipment Manufacturer (OEM) has the DBC file required to decode the data. This is the case, for example for raw CAN data in most cars, bikes, EVs, production machinery, etc.

In this demo model, we send out:

1. an `unsigned integer` type from bit 0 to bit 14 (15-bit length) provided by a Constant block,

2 a `bool` type digital signal on bit 15 provided by a Pulse Signal of 1 Hz and 0.5 duty cycle with values alternating between 0 and 1,

3 a 5 Hz Triangular Wave between -5 and +5 as a `signed integer` type from bit 16 to bit 31 (16-bit length),

4 a 5 Hz Sine Wave with amplitude of 1 as a `float` type from bit 32 to bit 63 (32-bit length).

This information is stored in a total of 64 bits which corresponds to the maximum amount of data in one package. Note that the Constant block for the first test signal is added into the **Exceptions** field of the **Parameter Inlining** tab of **Coder options...** window. This means that once the model is running in real-time, the value can be tuned during a real-time simulation in the External Mode.

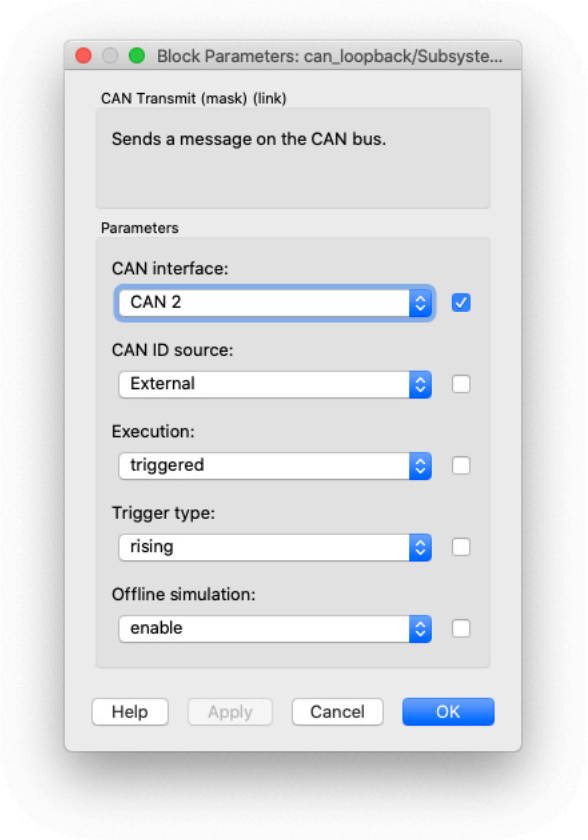


Fig. 4: Mask content of the CAN Transmit block

The CAN Transmit block sends out a message on the CAN bus. Fig. 4 shows the mask content of the CAN Transmit block.

- CAN channel 2 is used as the transmitter channel and is therefore chosen under **CAN interface** field.
- The **CAN ID source** field is chosen as `External` because it is provided from the CAN Pack block as an input signal. One can also choose the ID source as a `Parameter` and specify the CAN ID here.
- The **Execution** field can be chosen as `regular` or `triggered`. If `regular` is chosen, in the next field, **Sample time**, one can specify the time period for regular packet transmission. This demo uses a `triggered` execution with rising trigger sensitivity. A Trigger Port is shown at the mask level of the CAN Transmit block to which a Pulse Generator is connected with frequency equal to the CAN transmission sample time (`0.01` Seconds).
- The **Offline simulation** field enables or disables the Target Inports on the root schematic that allow the simulation of CAN Messages in an offline simulation.

2.2 CAN Receiving

The CAN Receive block receives a CAN message. 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 (and valid) data is received, and 0 otherwise. Fig. 5 shows the mask content of the CAN Receive block.

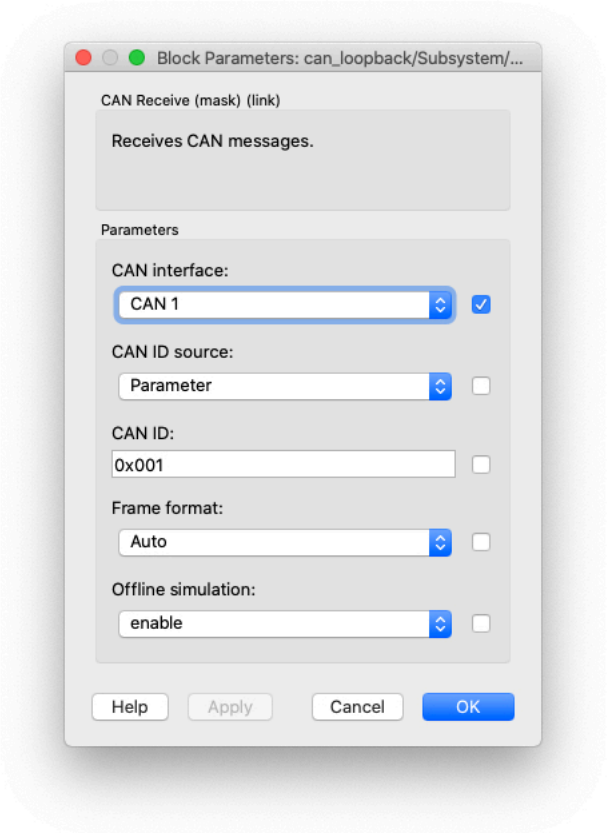


Fig. 5: Mask content of the CAN Receive block

- CAN channel 1 is the receiver channel and set in the **CAN interface** parameter field of the CAN receive block.
- Note that the CAN Transmit and CAN Receive blocks have to use the same CAN ID to realize the loop-back of data.
- Setting the **Frame format** to `Auto` means that it uses the `Base` format if the specified CAN ID is smaller than `2047`. Otherwise, the `Extended` format is used.
- The **Offline simulation** field enables or disables the Target Inports on the root schematic that allow the simulation of CAN Messages in an offline simulation.

The CAN Unpack block decodes signals from a byte vector received over CAN into the original message. Its CAN ID and signal definitions should be set in the exact same way as the CAN Pack block shown in Fig. 3 on p. 3. One can do it by manually editing the content of CAN Unpack block or simply by selecting **Import DBC ...** and specifying the same DBC file as is used on the CAN Pack side.

3 Simulation

The selected CAN interface must be enabled and configured in the RT Box Coder Options Dialog. From the **System** tab of the **Coder options...** window, select the “Subsystem” and go to the **Target** tab. Under the subtab **CAN**, make sure to check both **Enable CAN1** and **Enable CAN2**.

- **CAN1 baud rate** specifies the baud rate for CAN transmission. Note that all devices on a CAN bus must be configured to use the same baud rate. Here in this demo the default value of `500 kHz` is used.
- **CAN1 tx/rx pin** specifies which I/Os to use for CAN communication. The option `Internal CAN interface` is available on all RT Box versions except RT Box 1 with hardware revision earlier than 1.2. For older revisions the digital in and out pins 26/27 can be used to set up a CAN interface. Please note that an external CAN transceiver chip needs to be added by the user.
- If the `Internal CAN interface` is chosen above, a new field **Termination for CAN1** will appear. Each end of a CAN Bus should have a termination resistance of 120 Ω between the CAN-H pin and the CAN-L pin. As this example discusses a direct point-to-point connection, the termination resistance should be enabled for both CAN interfaces.

Build the "RT Box" model onto the RT Box. Once the model is uploaded, from the **External Mode** tab of the **Coder options...** window, **Connect** to the RT Box and **Activate autotriggering**. The real-time simulation results are shown in Fig. 6.

- The first signal received in an unsigned integer with a value of 10. By changing the Constant value in front of the CAN Pack block to another value (for example: from 10 to 20) during run-time, one should see the change reflected on the first received signal of the Scope in Fig. 6 immediately.
- The second received signal is a 1 Hz, 0.5 duty cycle digital pulse alternating between 0 and 1.
- The third signal is a 5 Hz Triangular waveform with duty cycle of 0.5 changing between -5 and +5.
- The fourth signal is a 5 Hz Sine waveform with amplitude of 1.
- The fifth signal shows the valid flag asserted when new valid data arrives

One can also see in the last two signals that the 10 ms staircase effect of quantization correctly reflects the **Sample time** setting in the CAN Transmit block.

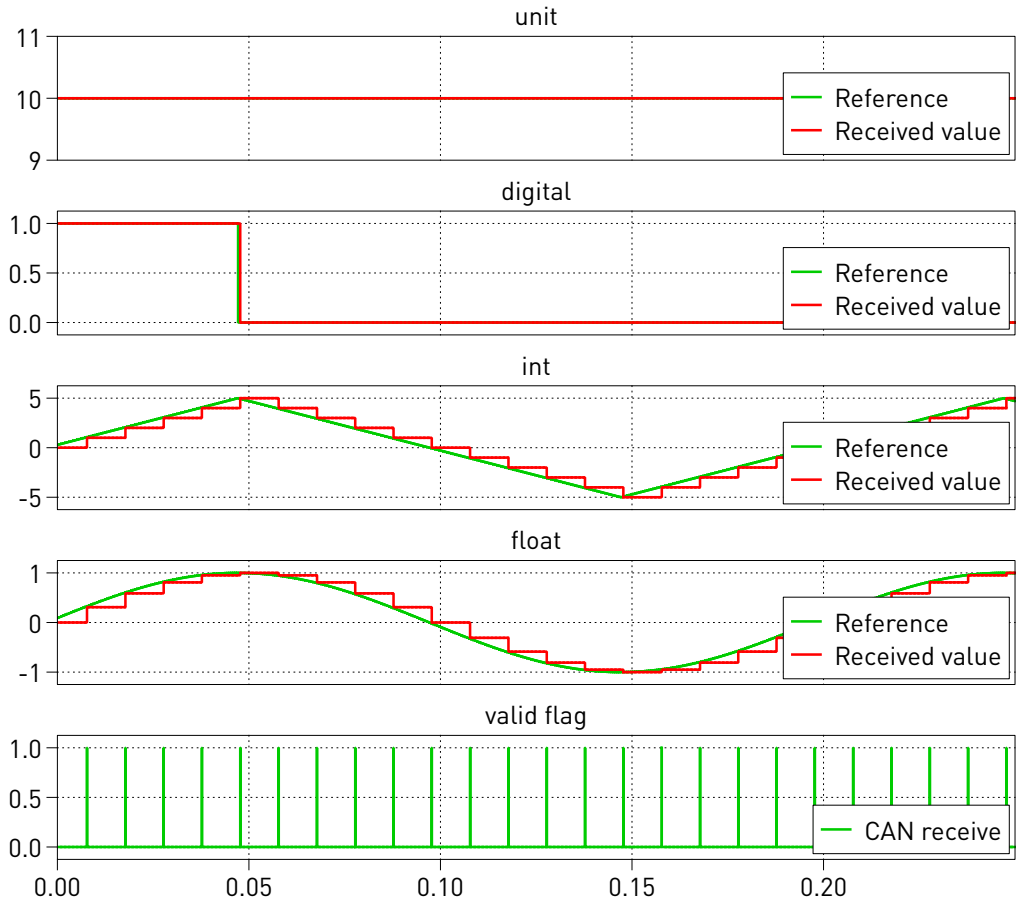


Fig. 6: Real-time waveform showing the four received signals after CAN loop-back test

The “v” port signal of the CAN Receive block goes high only when new data arrived. The signal can be used to trigger potentially time consuming calculations implemented in a triggered/enabled subsystem. This demonstration uses a simple setup where the first two signals are added and the last two signals are multiplied.

4 Conclusion

This RT Box demo model demonstrates a loop-back test using the integrated CAN interface on the RT Box. It shows how to set up the CAN interface in the Coder settings and how CAN messages are formed in a PLECS model using CAN Pack and Unpack blocks. The demo model runs in both offline and in real-time simulation.

Revision History:

RT Box TSP 2.0.5	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.