

OneEye_UART_Oscilloscope_1 for KIT_AURIX_TC334_LK

Oscilloscope over UART using OneEye

AURIX™ TC3xx Microcontroller Training
V1.0.0



[Please read the Important Notice and Warnings at the end of this document](#)

Scope of work

Demonstrate how to implement the OneEye oscilloscope over the UART (USB) interface

After configuring the OneEye UART interface, a OneEye oscilloscope is created. The signals are updated and sampled every millisecond. OneEye is used to visualize the signal values.

Introduction

- › **OneEye** is a GUI that enables the creation of interactive Graphical User Interface. Graphical elements can be drag from a toolbox and drop onto the GUI. The behavior of the created GUI can be customized. Different communication interfaces like UART, Ethernet, CAN, DAS can be used to interact with the embedded system
- › **SyncProtocol / ProtocolBB** is a synchronous protocol that enables data streaming between the target microcontroller and OneEye. It enables to open multiple communication channels, provide packet acknowledge and packet checksum. Data are transported within a message with a message ID and a message payload. See the OneEye help for more information.

Single frame

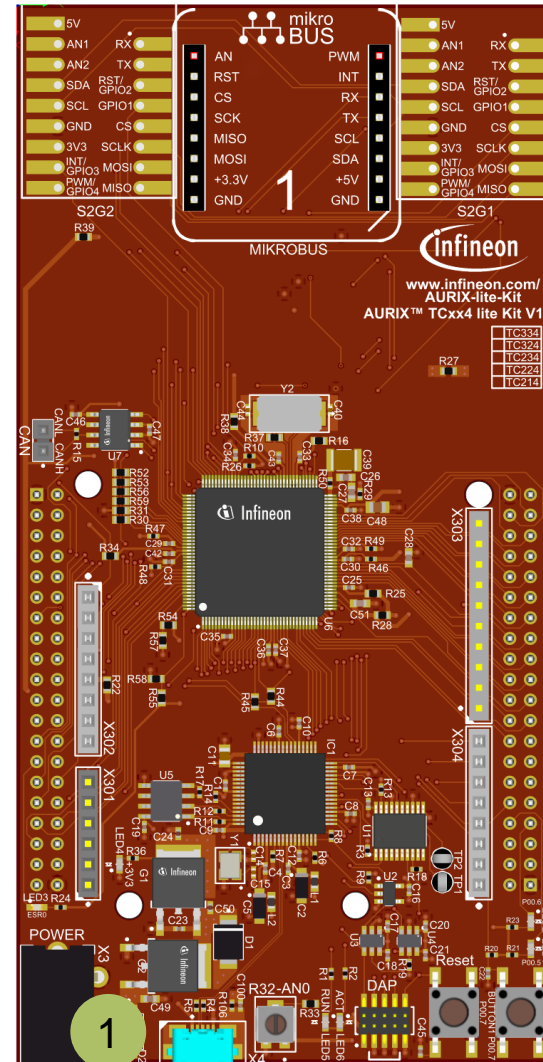
Offset	0	1	2	3
0	Start Byte	Sender	Receiver	Payload length
4	Flags (frameType=data)		Checksum payload	Checksum header
8	Message ID		(Reserved)	
12	Message length			
16	Message payload			
...	... (Message payload)			

- › **Note:** It is recommended to go through some of the **basic tutorials** listed in the help embedded in OneEye (Menu: Help -> OneEye help). This enables a quicker ramp-up in the OneEye concept and ensures a nice journey with OneEye

Hardware setup

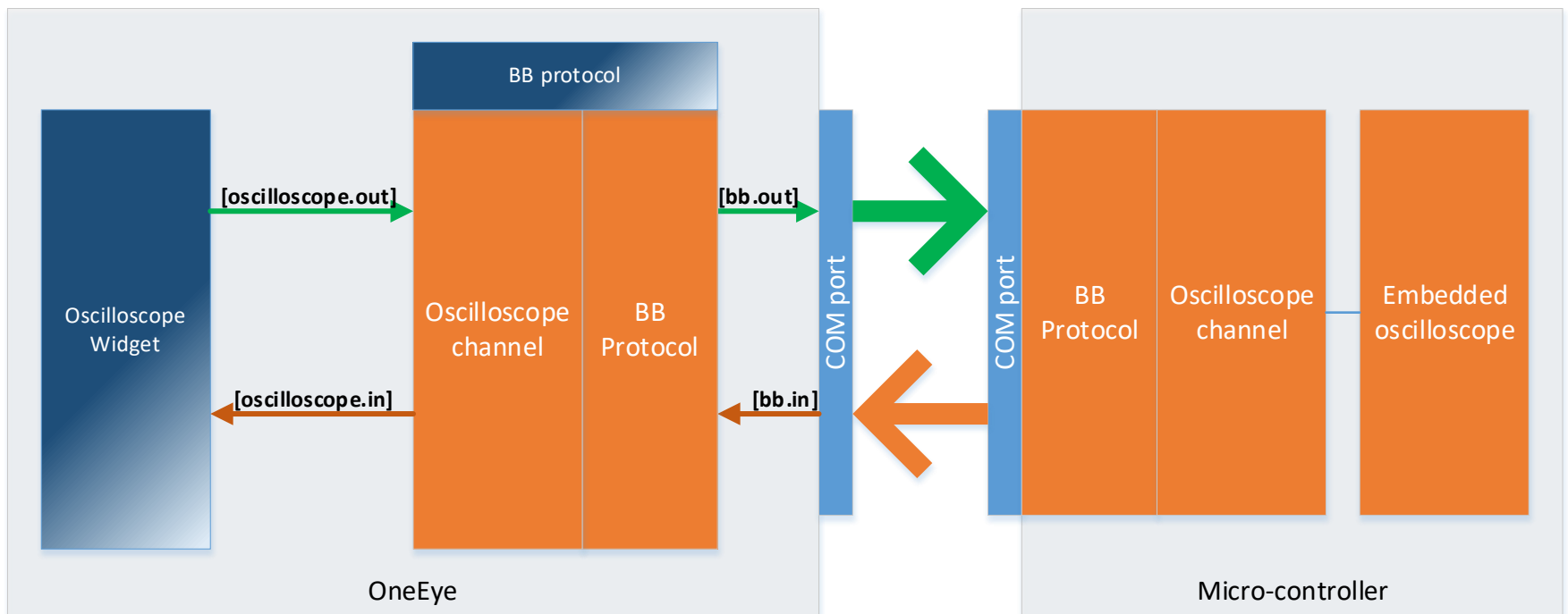
This code example has been developed for the board KIT_A2G_TC334_LITE.

The board should be connected to the PC through the USB port 1



Configuration overview

In this configuration an oscilloscope running on the microcontroller is connected to the COM port.
 In OneEye, two signals **bb.in** and **bb.out** are used to connect the COM port data stream to the BB protocol.
 The BB protocol is configured to open a channel reserved for the oscilloscope. This channel connects to the oscilloscope with the **oscilloscope.in** and **oscilloscope.out** signals.



Implementation - AURIX

Enabling the OneEye library

The OneEye library must be enabled by adding the following line to *lfx_Cfg.h*:
#define IFX_OE_AL_USE_AURIX_ILLD

Configuring the OneEye Oscilloscope

A OneEye oscilloscope (*lfx_Oe_Osci*) is an object that enable data sampling and provide triggering functionality.

The OneEye oscilloscope communication interface (*lfx_Oe_OsciBb*) enables streaming of data and control of the oscilloscope using the BB protocol (*lfx_Oe_SyncProtocol*).

The OneEye oscilloscope is initialized with *initOscilloscope()* / *lfx_Oe_Osci_init()*.

The ***autoAddChannels*** parameter enables to automatically add channels for each created oscilloscope signal.

The sample period (***samplePeriod***) is set to 1ms and provides OneEye information about sample timing.

The ***triggerMode*** is set to automatic, note that this value can be changed from the OneEye oscilloscope interface later.

The *lfx_oe_osci.h* file can be found in the Libraries\OneEye directory.

Adding signals to the oscilloscope

Oscilloscope signals are mainly pointers that the oscilloscope can use for data sampling. The signals are added using *lfx_Oe_Osci_addSignal()*. The function takes as parameter the signal **name** displayed by the oscilloscope, with an optional unit string in parenthesis, the signal **type** which informs the oscilloscope how to read the pointer value and a **source** pointer to the data. The last parameter corresponds to the **q format** used in case of fix point data, or 0 if not used.

Implementation - AURIX

Starting the oscilloscope

The oscilloscope is started with the function *lfx_Oe_Osci_start()*.

Configuring the signal generator

A signal generator is used to provide the user with some value to read / write. The signal generator does nothing more than incrementing two signals, *signalA* and *signalB*, stored in the structure *g_signalGenerator* up to a maximum value before resetting them.

The initialization of the signal generator is done with *initSignalGenerator()*.

Running the signal generator and the oscilloscope

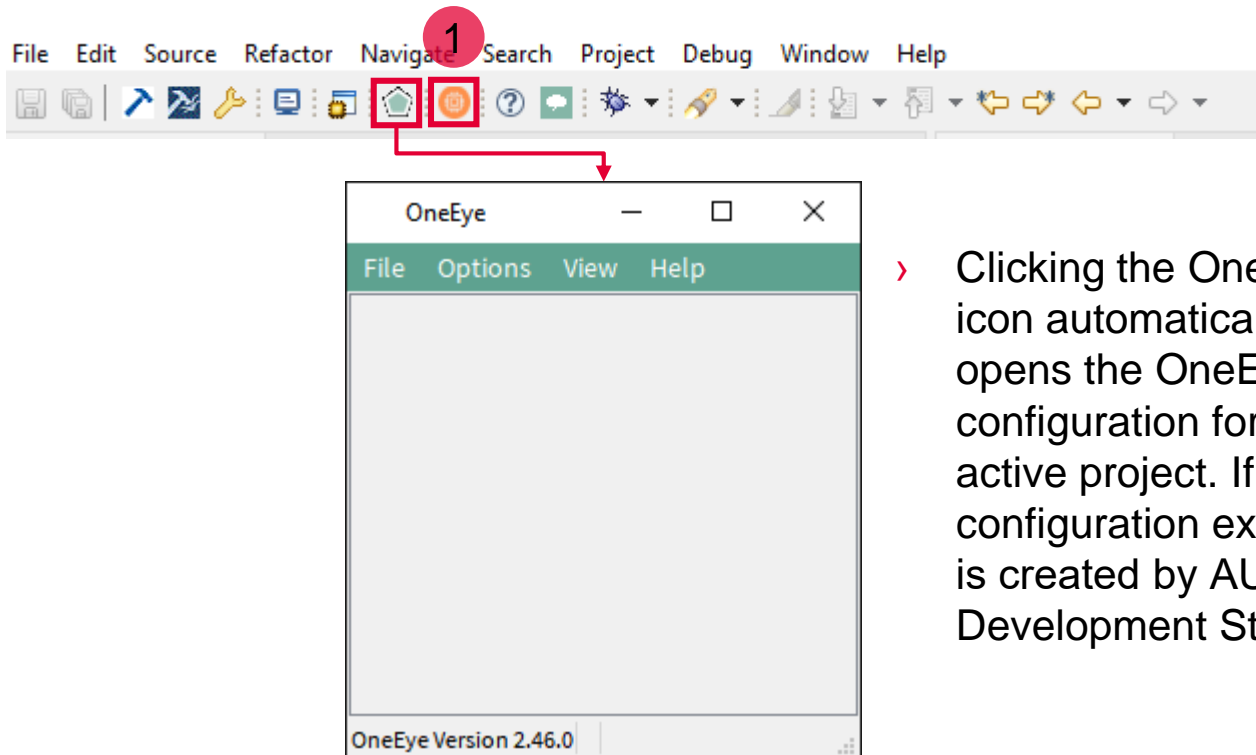
The signal generator is executed in the background loop every 1ms with *processSignalGenerator()*. To ensure the timing, a **deadline** variable is periodically updated with *lfx_Oe_Time_add()* to obtain the 1ms period.

The oscilloscope runs in the same background loop with *sampleOscilloscope() / lfx_Oe_Osci_step()*. This function handles the triggering and sampling of data.

Note: the call to *lfx_Oe_Osci_step()* can be moved to an interrupt service routine if required by the application use case.

Run and Test

- › After code compilation, flash the device using the Flash button **1** to ensure that the program is running on the device
- › For this training, the OneEye application is required for visualizing the values. OneEye can be opened inside the AURIX™ Development Studio using the following icon:



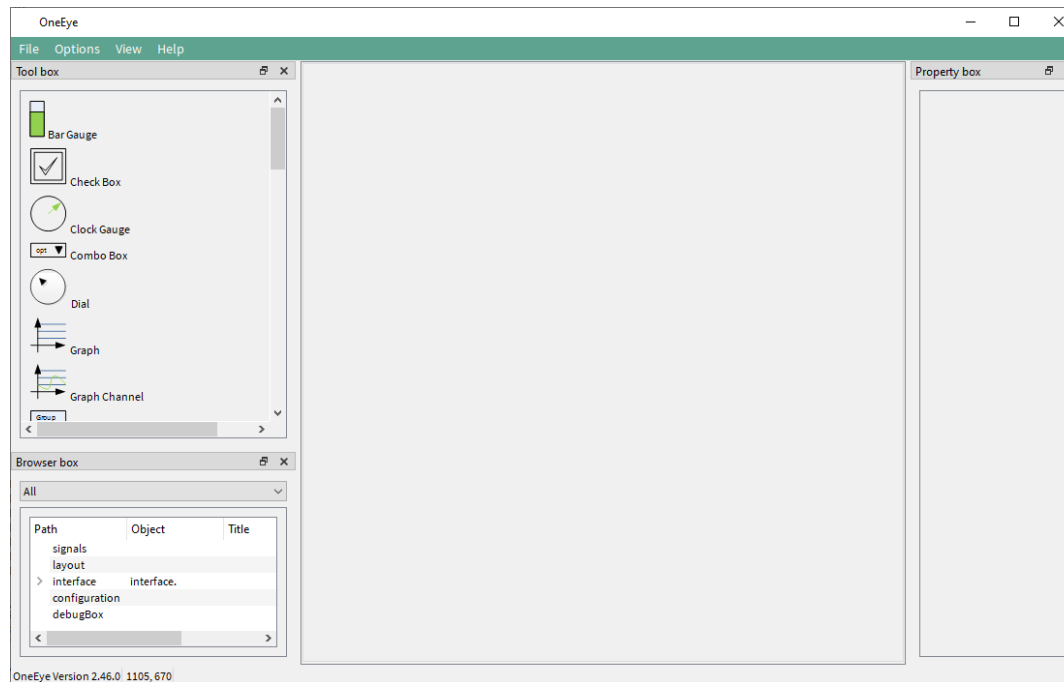
- › Clicking the OneEye icon automatically opens the OneEye configuration for the active project. If no configuration exists, it is created by AURIX™ Development Studio

Implementation - OneEye

In this training, the OneEye configuration is provided inside the Libraries folder. The following steps are needed to configure the oscilloscope from a brand-new configuration.

Setup OneEye for editing

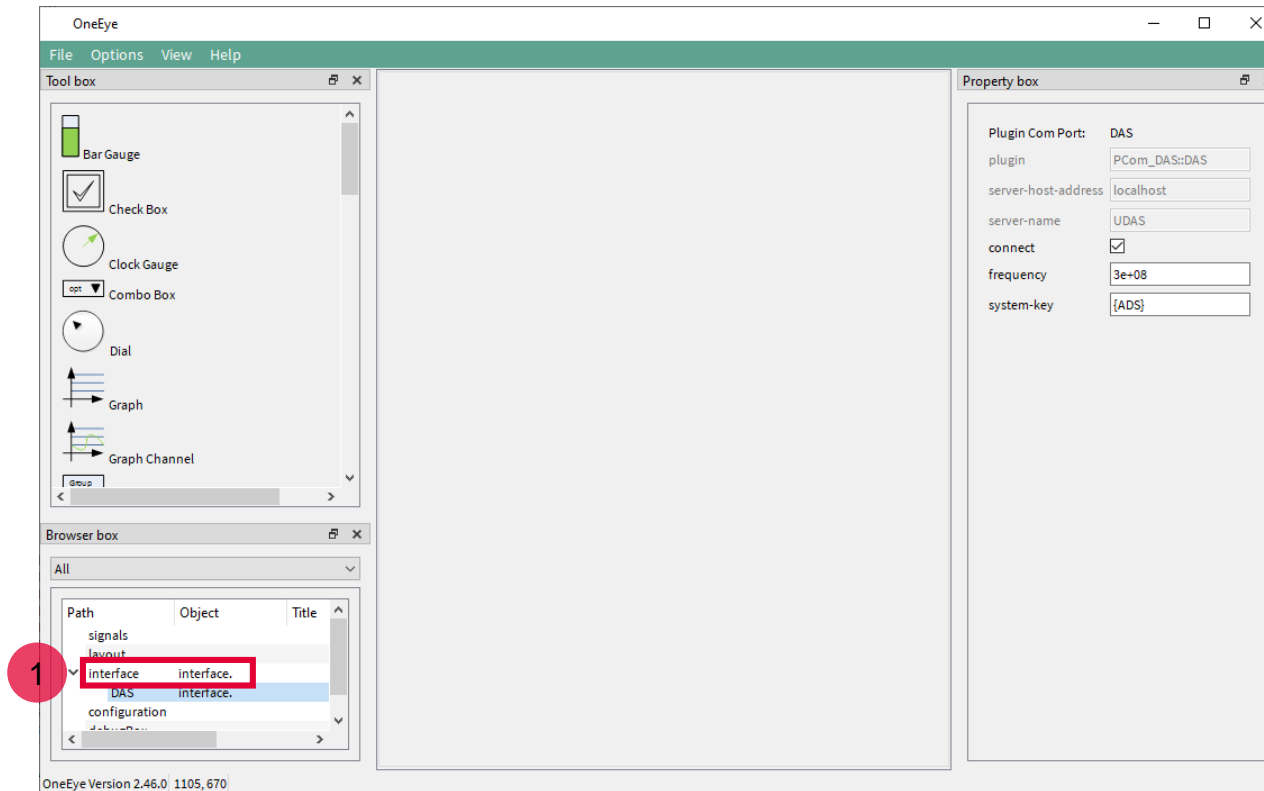
Select the OneEye menu **“Options -> Edit mode”** (if not already checked) to enable the edit mode. Select the OneEye menu **“View -> Browser box”**, **“View -> Property box”**, **“View -> Tool box”** (if not already checked) to display the browser, property box and tool box. Note that the box can be moved around.



Implementation - OneEye

Removing the default DAS interface

When the OneEye configuration is created by ADS, it is already setup with a DAS interface. Select the interface in the Browser box **1** and delete it with “right click and remove” as it is not required in this example.

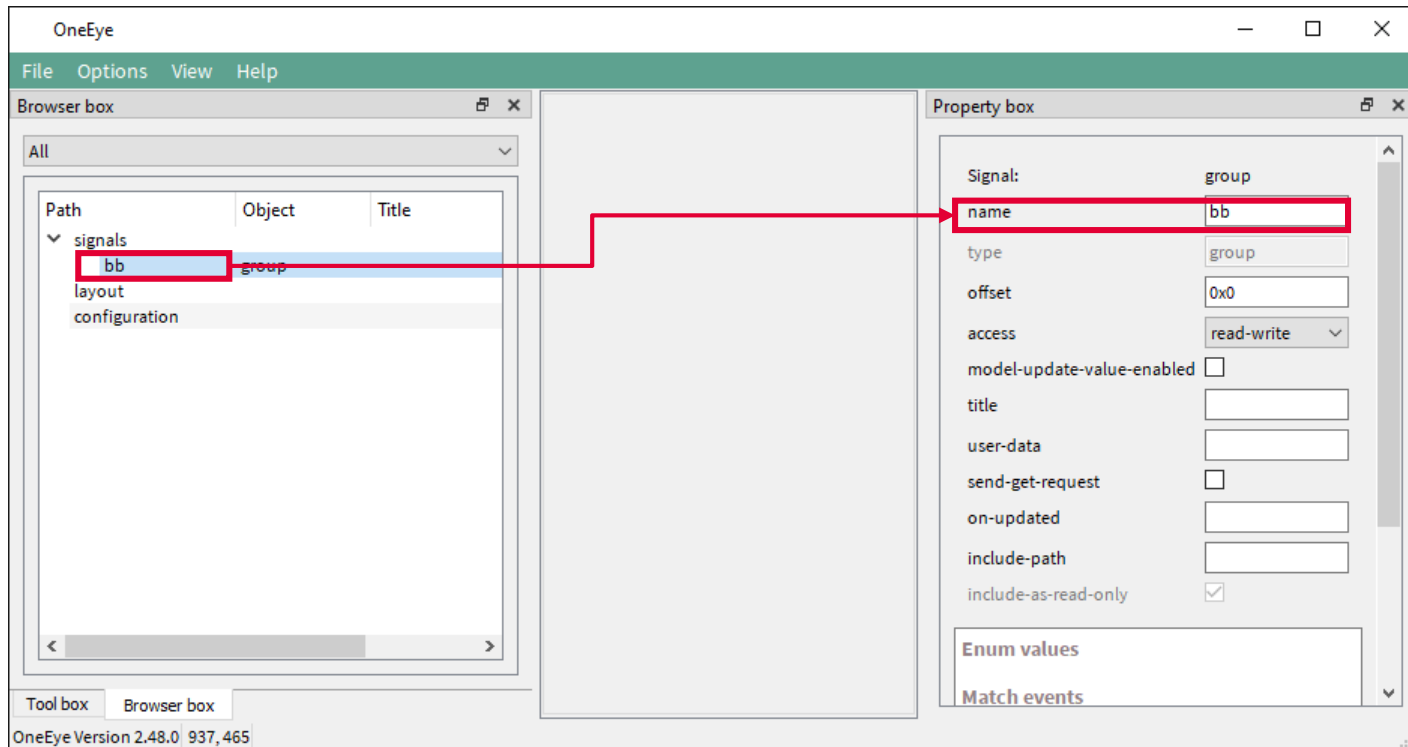


Implementation - OneEye

Configuring the UART interface: Signal creation

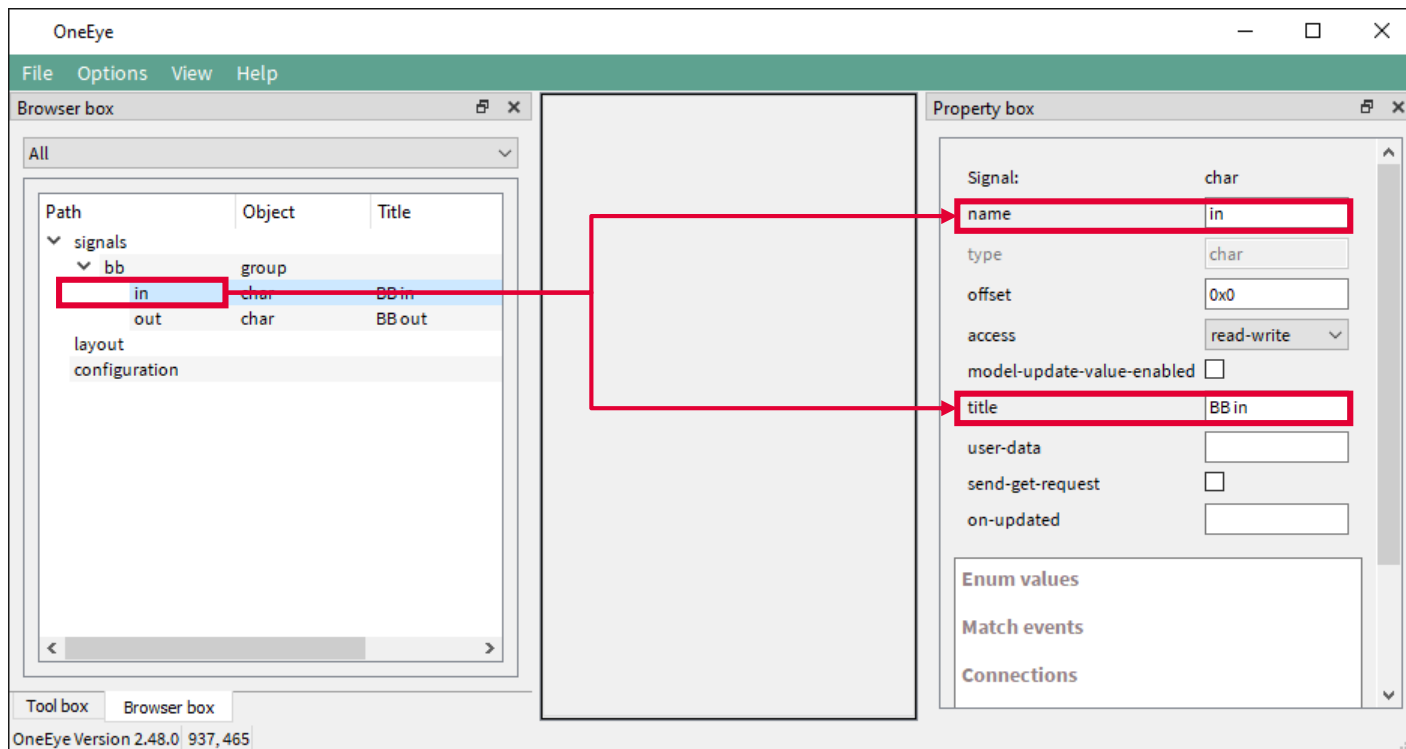
The first step is to create 2 signals to connect the received and transmit data over the UART.

Create a signal group and set its **name** property to **bb**.



Implementation - OneEye

Add two signals of type **char** into the **bb** group, name them **in** and **out**, and set their **title** property to respectively **BB in** and **BB out**.

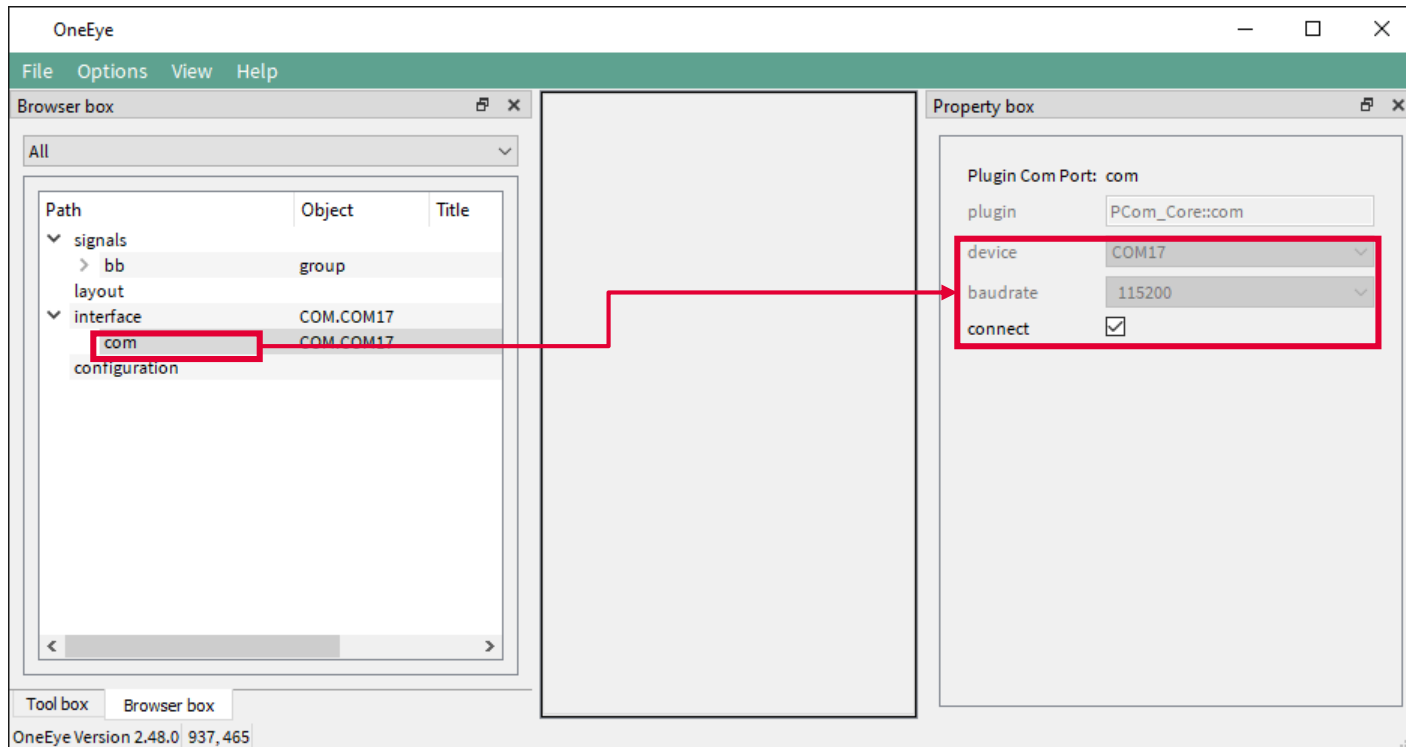


Implementation - OneEye

Configuring the UART interface: COM port

Right click in an empty area of the Browser box, and select **Add child -> Interface**. Then right click on the created interface and select **Add child -> com**. Select the **com** item and set its **device** property to the COM port connected to the AURIX board. Set the **baudrate** property to **115200** and click **connect**.

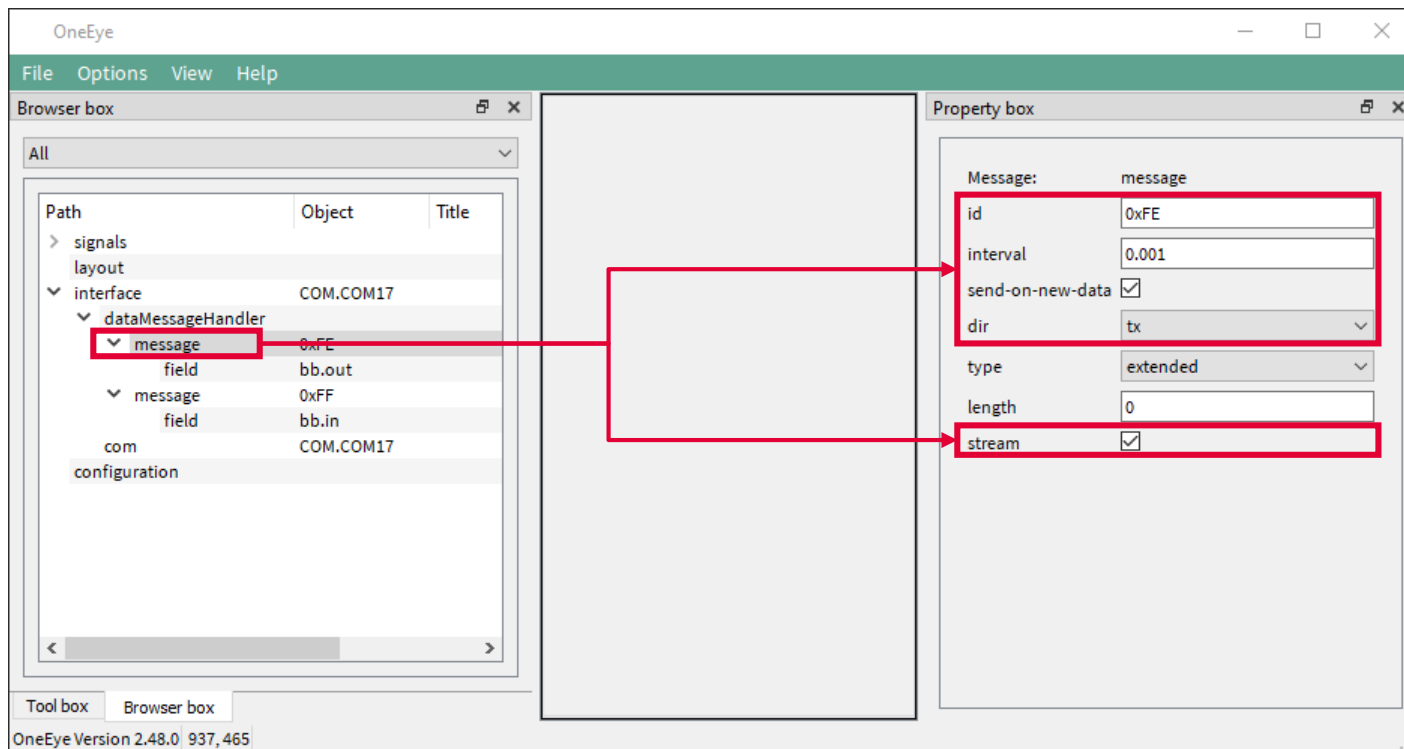
The COM port is now opened and ready for communication.



Implementation - OneEye

Configuring the UART interface: Transmit stream

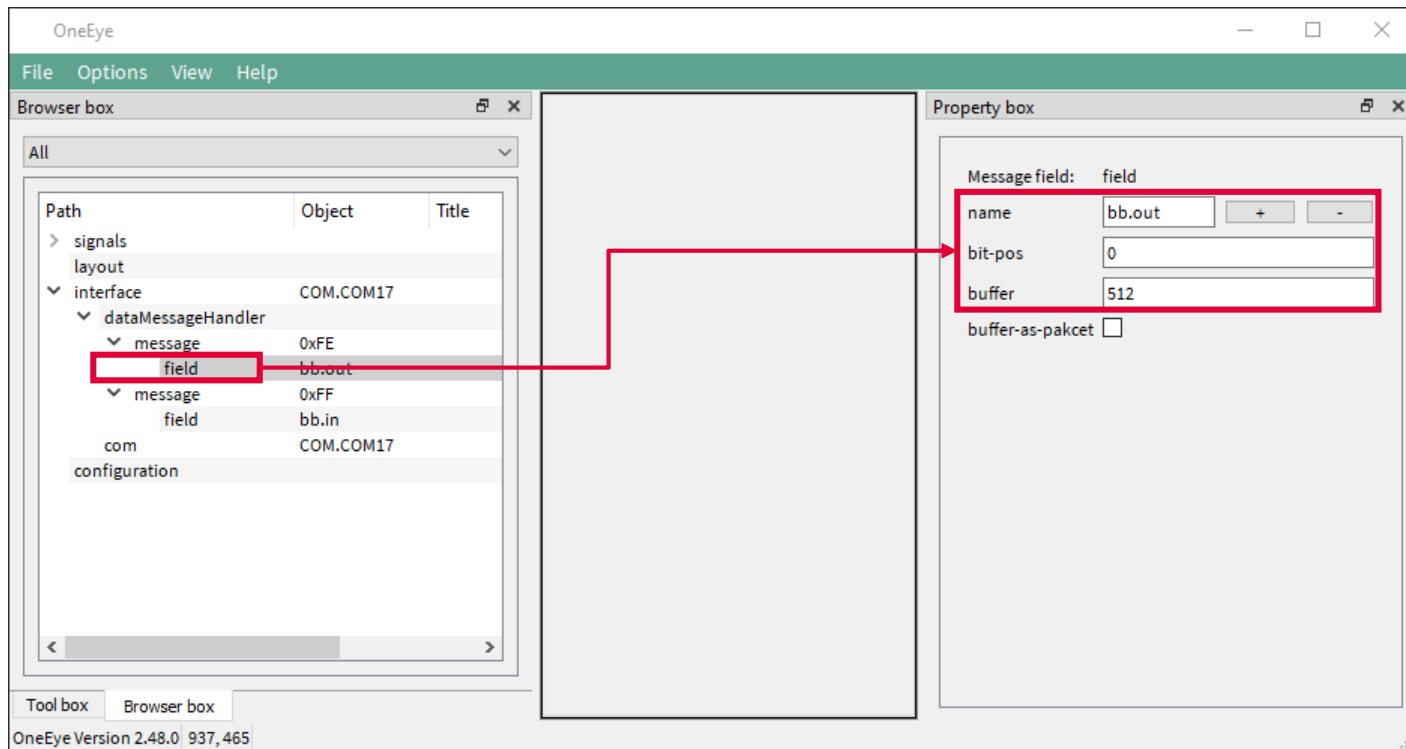
Right click on the **interface** in the Browser box, and select **Add child -> dataMessageHandler**. Then right click on the created **dataMessageHandler** and select **Add child -> message** to create a message item. Configure the **message** with the **id=0xFE**, **interval=0.001**, **send-on-new-data** checked, **dir=tx**, **stream** checked.



Implementation - OneEye

Right click on the **message**, and select **Add child -> field**.
Configure the field with **name=bb.out**, **bit-pos=0**, **buffer=512**.

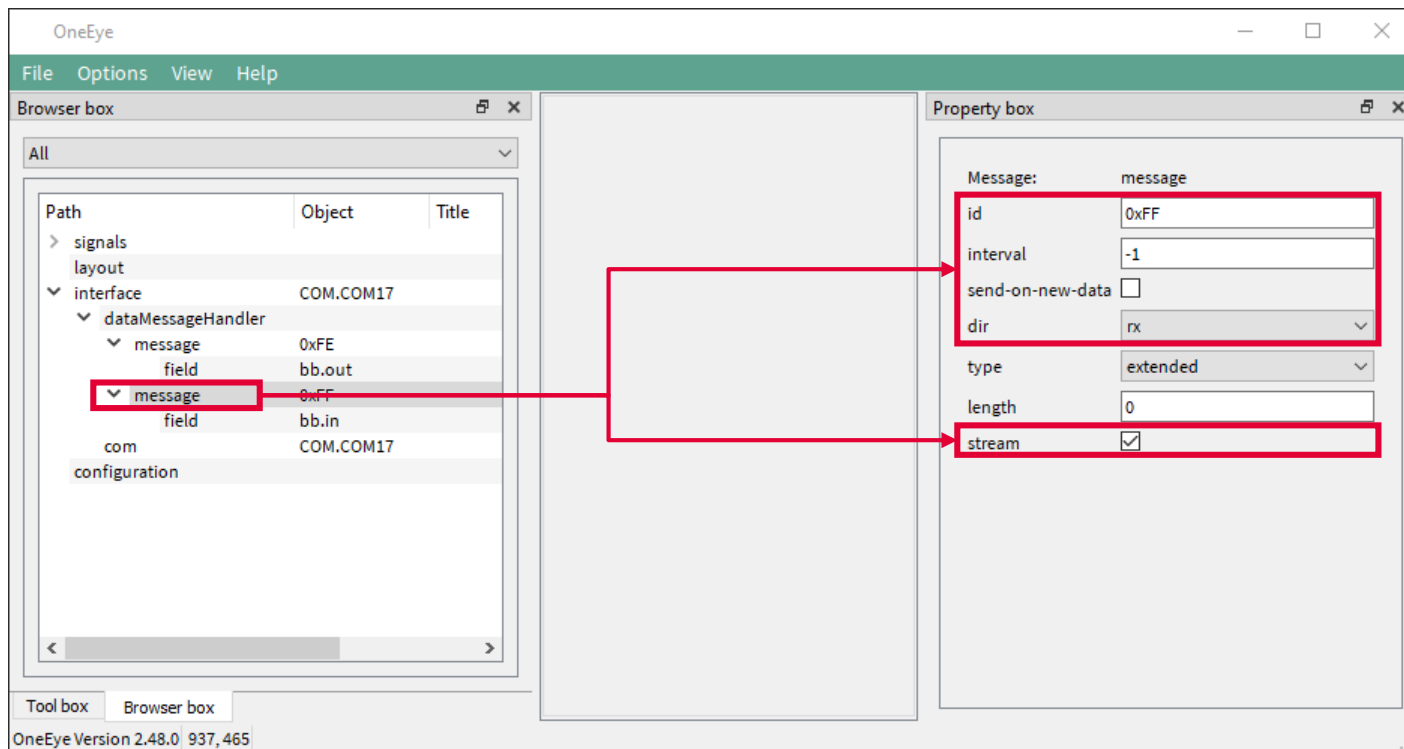
Now, data will be transmitted over the UART each time the **bb.out** signal is written with some data.



Implementation - OneEye

Configuring the UART interface: Receive stream

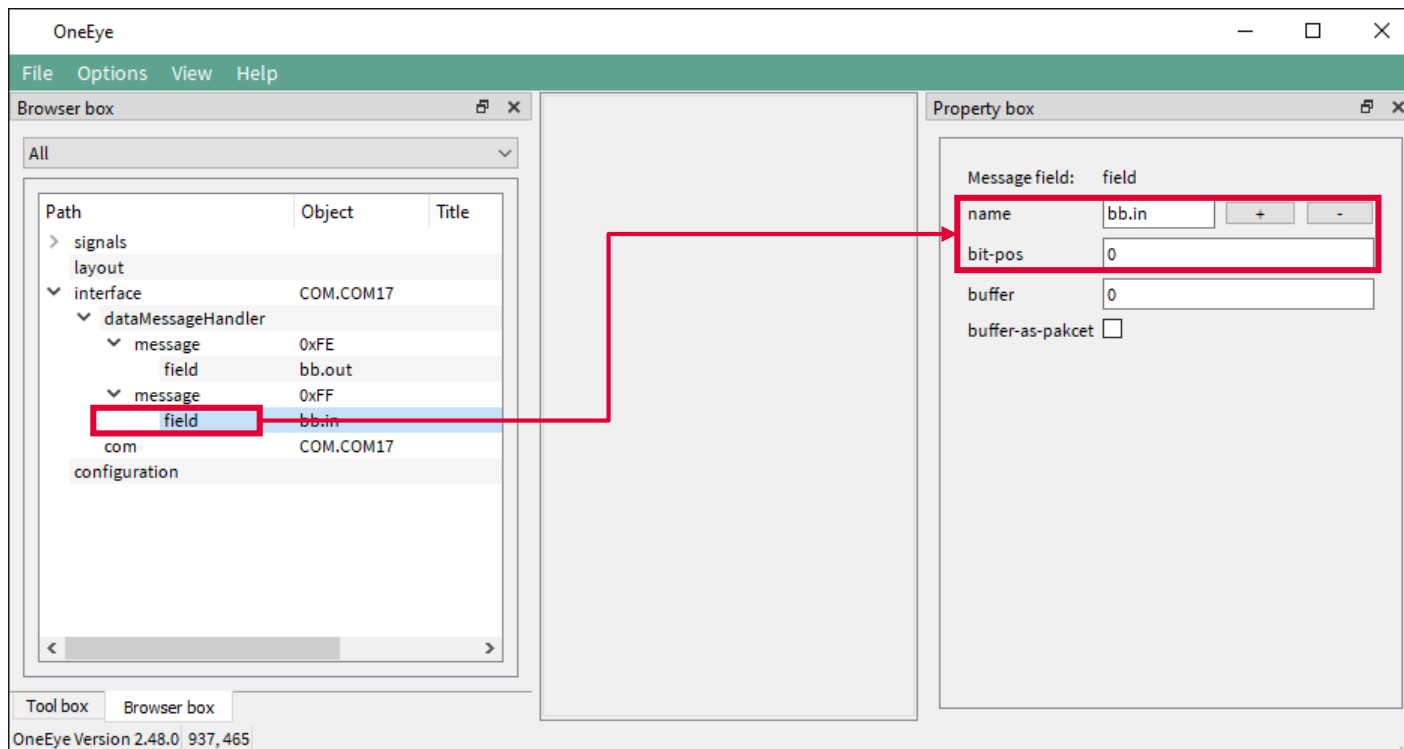
Right click on the **dataMessageHandler** and select **Add child -> message** to create a second message item. Configure the message with the **id=0xFF**, **interval=-1**, **dir=rx**, stream checked.



Implementation - OneEye

Right click on the **message**, and select **Add child -> field**.
Configure the field with **name=bb.in**, **bit-pos=0**.

Now each time data are received over the UART, the **bb.in** signal will be updated.

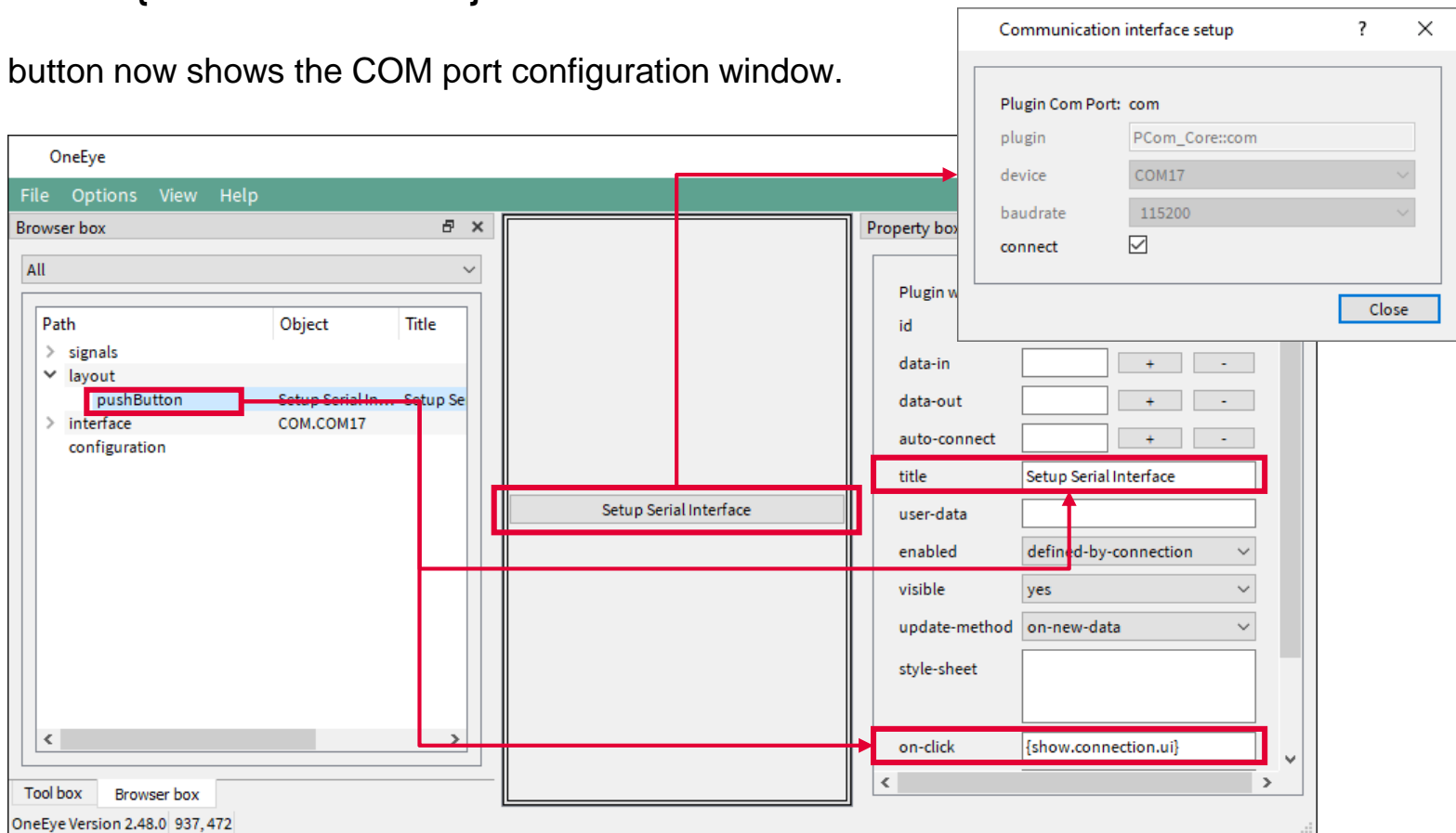


Implementation - OneEye

Configuring the UART interface: Push button

Drag and drop a **pushButton** widget from the toolbox onto the layout, configure it with **title=Setup Serial Interface**, **on-click={show.connection.ui}**.

Clicking the button now shows the COM port configuration window.

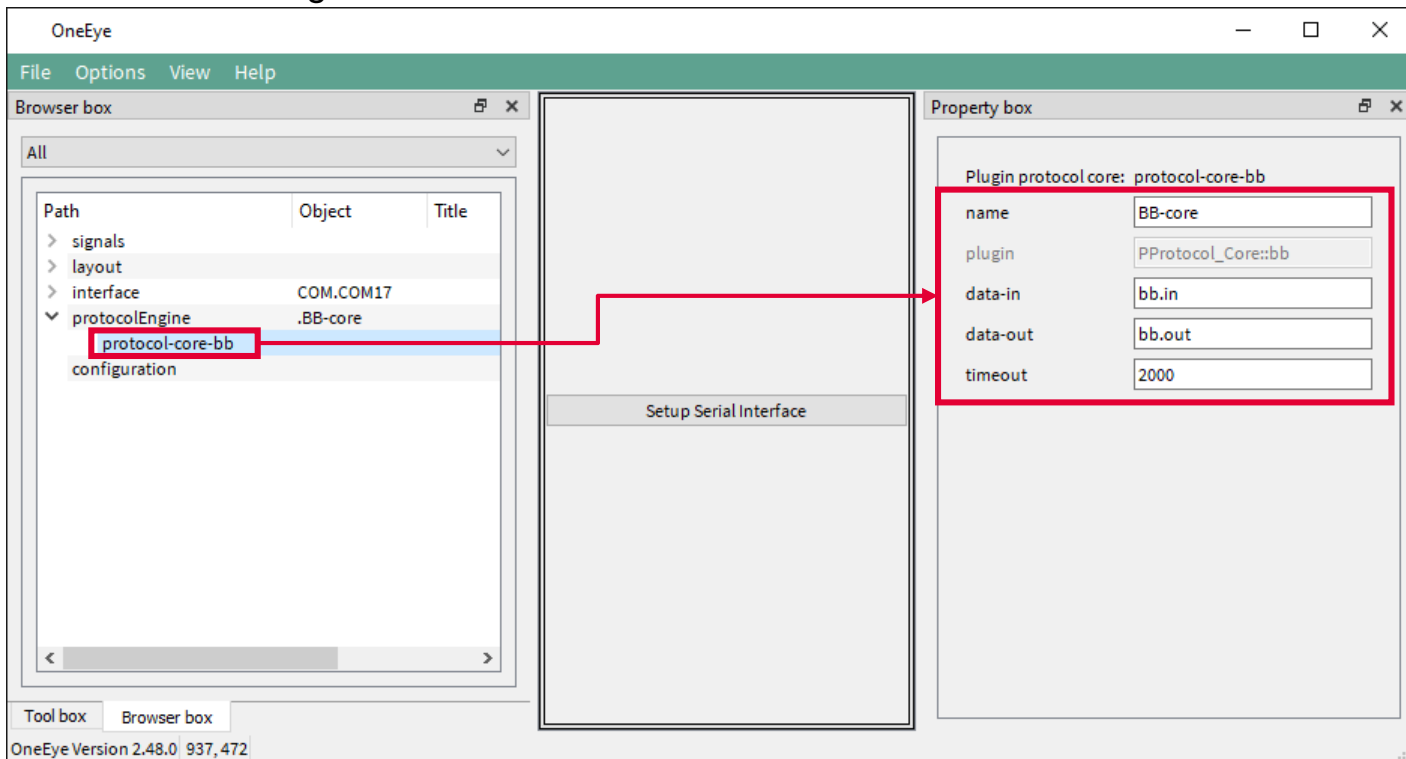


The screenshot illustrates the configuration process in the OneEye software. The main window shows a 'Browser box' with a tree view containing 'signals', 'layout', 'interface', and 'configuration'. A 'pushButton' widget is highlighted in the 'layout' folder, with its title set to 'Setup Serial Interface'. The 'Property box' on the right shows the widget's configuration, including the title 'Setup Serial Interface' and the 'on-click' event set to '{show.connection.ui}'. A red box highlights the 'on-click' property. A red arrow points from the 'pushButton' in the browser box to the 'Setup Serial Interface' button in the layout. Another red arrow points from the 'on-click' property in the property box to the 'on-click' property in the 'Communication interface setup' dialog. The dialog shows the configuration for the communication interface, including the plugin 'PCom_Core::com', device 'COM17', baudrate '115200', and a checked 'connect' checkbox. A red box highlights the 'title' property in the dialog, which is set to 'Setup Serial Interface'. A red arrow points from the 'title' property in the property box to the 'title' property in the dialog. The 'Communication interface setup' dialog is also highlighted with a red box.

Implementation - OneEye

Configuring the BB protocol

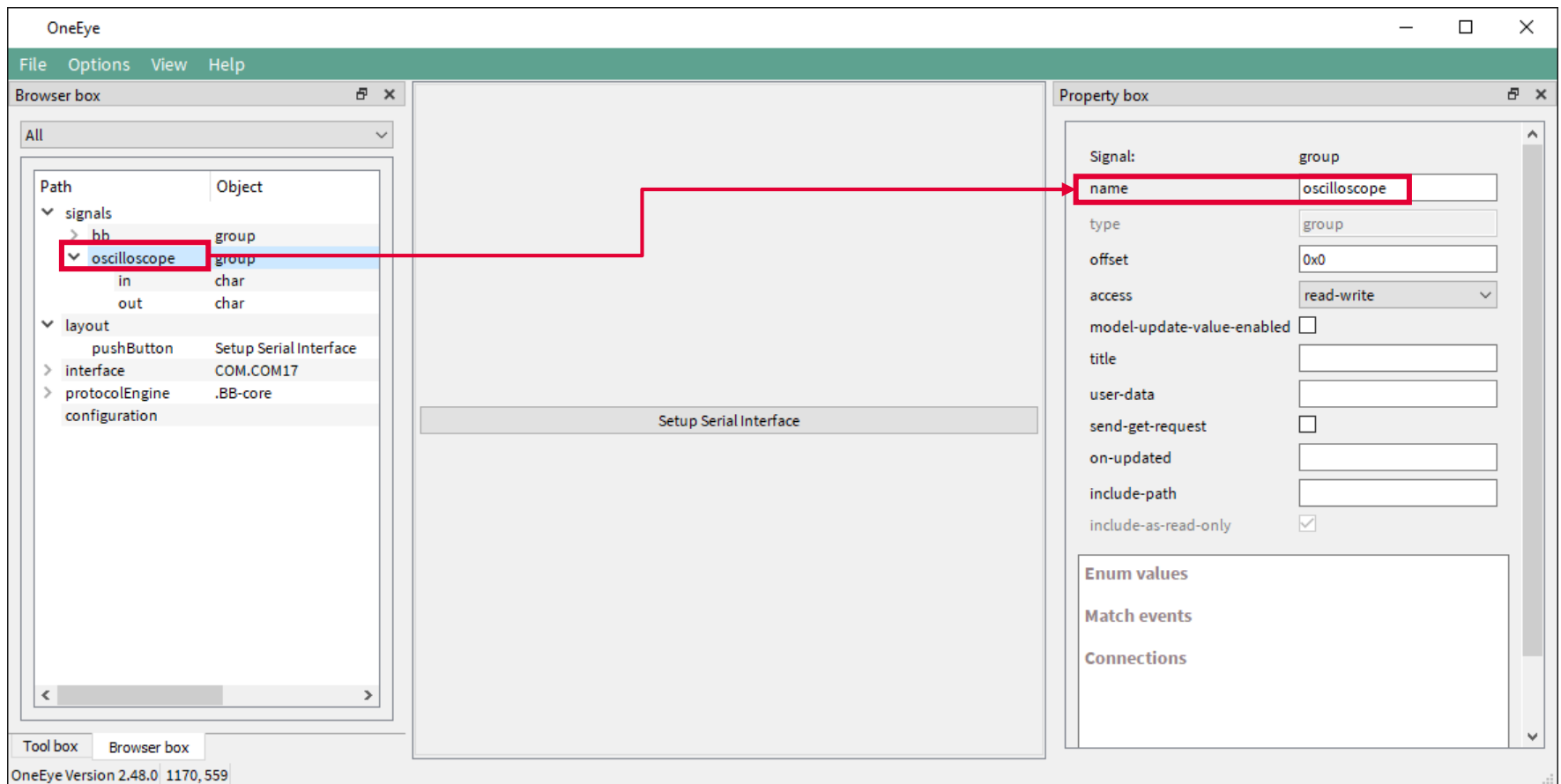
Right click in an empty area of the Browser box, and select **Add child -> protocolEngine**. Then right click on the created **protocolEngine** and select **Add child -> protocol-core-bb**. Connect the BB protocol stream to the **bb.in** and **bb.out** signals by setting respectively the **data-in** and **data-out** properties. Set the **name** property to **BB-core**. And set the **timeout** to **2000** ms so that frames are dropped after 2 seconds in case the microcontroller is not answering.



Implementation - OneEye

Configuring the Oscilloscope: signals creation

Create a signal group and set its **name** property to **oscilloscope**.

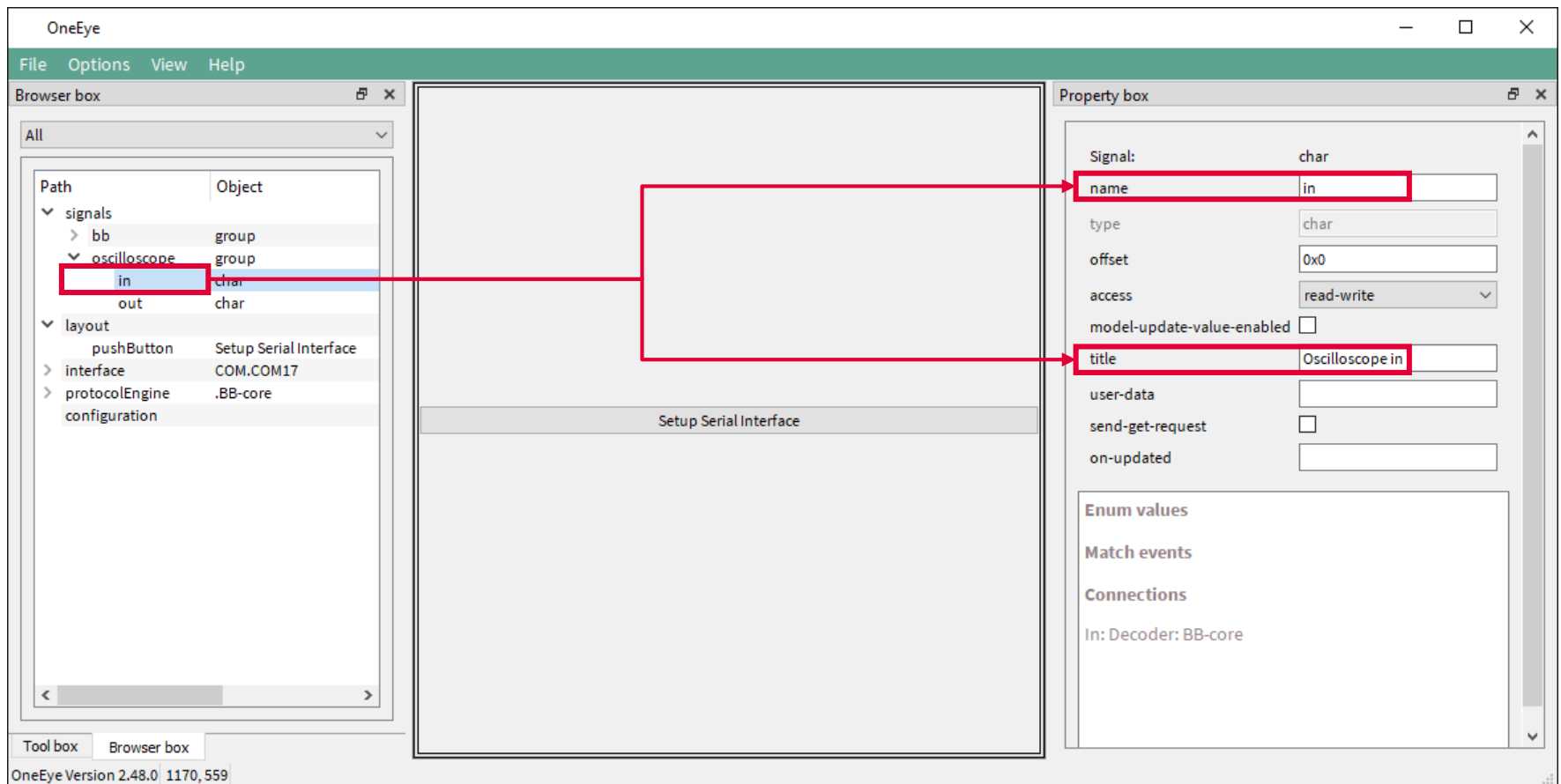


The screenshot shows the OneEye software interface with the following components:

- Browser box:** A tree view on the left showing a hierarchy of objects. The 'signals' folder is expanded, and the 'oscilloscope' group is selected and highlighted with a red box. A red arrow points from this box to the 'name' property in the Property box.
- Property box:** A panel on the right showing the configuration for the selected 'oscilloscope' group. The 'name' property is set to 'oscilloscope' and is highlighted with a red box. Other properties include 'type' (group), 'offset' (0x0), 'access' (read-write), and 'include-as-read-only' (checked).
- Central Canvas:** A large area in the middle showing a 'Setup Serial Interface' component.
- Bottom Bar:** Shows 'Tool box' and 'Browser box' tabs, and the version information 'OneEye Version 2.48.0 1170, 559'.

Implementation - OneEye

Add two signals of type **char** into the **oscilloscope** group, name them **in** and **out**, and set their **title** property to respectively **Oscilloscope in** and **Oscilloscope out**.



Implementation - OneEye

Create the oscilloscope widget

Drag and drop an **oscilloscope** widget from the toolbox onto the layout, set the oscilloscope properties **data-in** and **data-out** to respectively **oscilloscope.in** and **oscilloscope.out**. Set the **protocol-type** property to **ProtocolBB**. Set the **unit-x** property to **s**.

The screenshot shows the OneEye software interface with the following components and configurations:

- Browser box:** A tree view showing the project structure. The **oscilloscope** widget is highlighted in red in the **layout** folder.
- Setup Serial Interface:** A control panel for the oscilloscope. The **Control** tab is active, showing settings like **Run** (Disabled), **Buffer depth** (512), **Sampling interval** (1), **Sampling period (s)** (0.001), and **Time/Div** (10ms/Div). The **Trigger** section is set to **Automatic** mode with a **Slope** of **Rising edge** and **Channel** of **CH0: Signal A (V)**. The **Level** is set to 0.
- Property box:** A list of properties for the **oscilloscope** widget. The following properties are highlighted with red boxes and arrows:
 - data-in:** Set to **oscilloscope.in**
 - data-out:** Set to **oscilloscope.out**
 - protocol-type:** Set to **ProtocolBB**
 - unit-x:** Set to **s**
- Oscilloscope Plot:** A central workspace showing an oscilloscope plot with two signals: **0: Signal A (V)** (red) and **1: Signal B** (blue). The plot shows a rising edge for Signal A. The **Y-axis** is set to **200 (V)/Div** and the **X-axis** is set to **0.01 s/Div**.

Implementation - OneEye

Connect the oscilloscope widget to the BB protocol

Right click on the **protocol-core-bb** and select **Add child -> target**. Select the **target** item and set **local-port** and **remote-port** to **3** to match the AURIX settings, **set signal-in=oscilloscope.out, signal-out=oscilloscope.in, forward=checked**.

The screenshot displays the OneEye software interface with three main panels: Browser box, Setup Serial Interface, and Property box.

- Browser box:** Shows a tree view of the project structure. The path is expanded to `protocol-core-bb`, and the `target` item is selected and highlighted with a red box.
- Setup Serial Interface:** Shows the configuration for the selected `target`. The `Oscilloscope in` section is visible, with two channels (0 and 1) highlighted by red and blue boxes respectively. The `Control` section includes a `Run` button. The `Trigger` section includes `Single mode`, `Trigger mode`, `Slope`, `Channel`, and `Level` settings.
- Property box:** Shows the configuration for the selected `target`. The `local-port` and `remote-port` are set to `3`. The `signal-in` is set to `oscilloscope.out` and the `signal-out` is set to `oscilloscope.in`. The `forward` checkbox is checked.

Red arrows indicate the connection between the `target` item in the Browser box and the `target` configuration in the Property box, and between the `Oscilloscope in` section in the Setup Serial Interface and the `signal-in` and `signal-out` fields in the Property box.

OneEye Version 2.48.0 1170, 559

Implementation - OneEye

Test the oscilloscope

The oscilloscope Control tab provides configuration for the trigger and information about the oscilloscope state (armed, triggered, uploading).

Click on the **Control** button **1** and the **Refresh State** button to retrieve the oscilloscope settings (channels, timings, ...).

The screenshot displays the OneEye software interface. The main window is titled "OneEye" and contains several panels:

- Tool box:** Contains various components like Label, Line Edit, Logger, Logger Channel, Mux-Demux, Named Pipe Client, Named Pipe Server, Oscilloscope, and Oscilloscope Channel.
- Browser box:** Shows a tree view of objects under "signals" and "bb".
- Setup Serial Interface:** A central panel with a grid for the oscilloscope. Below the grid, the "Control" tab is active, showing settings for Buffer depth (512), Sampling interval (1), Sampling period (0.01), Time/Div (1s/Div), and Trigger mode (Automatic). The "Refresh State" button is highlighted with a red box.
- Property box:** Shows configuration for the target, including local-port (3), remote-port (3), signal-in (oscilloscope.out), signal-out (oscilloscope.in), forward (checked), and send-parent-value (unchecked).

At the bottom of the oscilloscope grid, the "Control" button is highlighted with a red box and a red circle containing the number "1". Other buttons like "Channel", "Save", "Info", and "Hide" are also visible.

OneEye Version 2.48.0 1310, 645

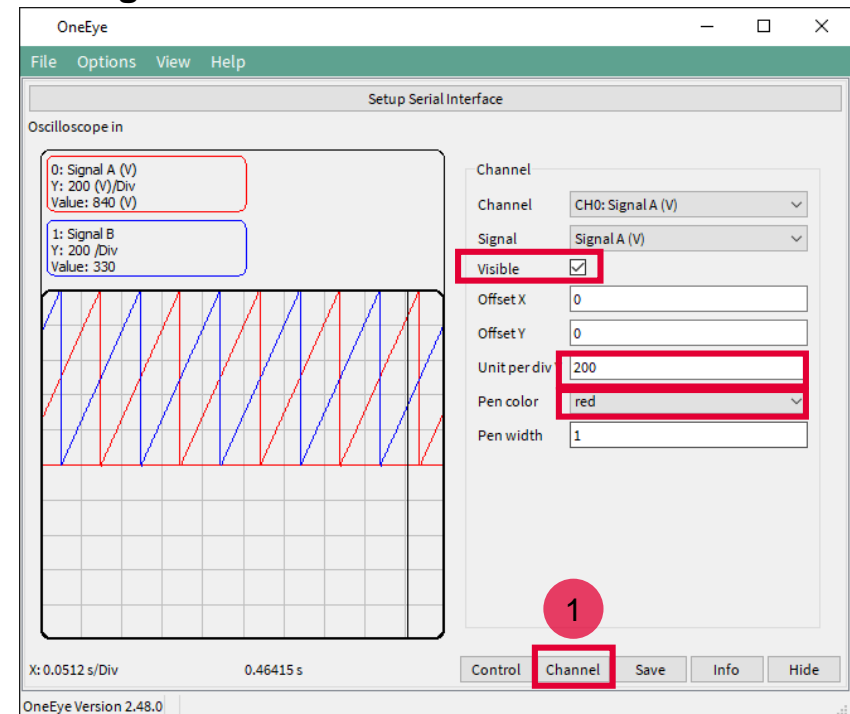
Implementation - OneEye

Test the oscilloscope

In the oscilloscope Channel tab, click on the Channel button **1** and check the **visible** check box for both **CH0: Signal A** and **CH1: Signal B** to display the two channels.

Set the **Unit per div Y** to **200** for both **CH0: Signal A** and **CH1: Signal B**.

Select the **Pen color red** for **CH0: Signal A** and **blue** for **CH1: Signal B**.



Implementation - OneEye

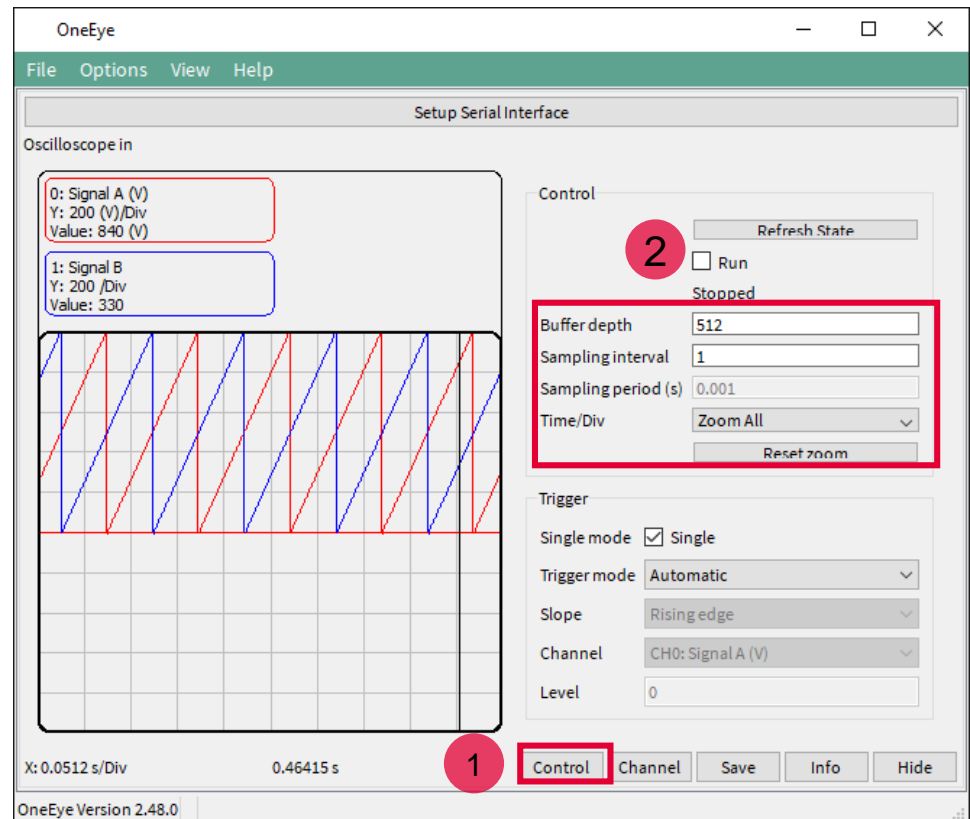
Test the oscilloscope

Click on the Control button **1**, check the run button **2**, the values for **signalA** and **signalB** should be updating in the oscilloscope.

Set the **Time/Div** value to **Zoom All** to configure the horizontal scale to use the full screen of the oscilloscope window.

The **Buffer depth** configures the oscilloscope buffer depth, here 512 points are used to fill the buffer. This value can be changed within the limit set by the software.

The **Sampling interval** provides the information whether to sample at each interval (1) or not (>1) to the oscilloscope.



Implementation - OneEye

Advanced options

Advanced configuration can be added to the file *Ifx_Cfg.h* or *ifx_oe_cfg.h* to tune the oscilloscope capabilities, this includes:

- › ***IFX_CFG_OE_OSCI_MAX_NUM_OF_SIGNALS***: the maximum number of signals that can be declared by the user
- › ***IFX_CFG_OE_OSCI_MAX_NUM_OF_CHANNELS***: the maximum number of channels that can be buffered
- › ***IFX_CFG_OE_OSCI_NUM_OF_SAMPLES***: the maximum number of sample per channel

Note: the memory used by the oscilloscope is mainly defined by

IFX_CFG_OE_OSCI_MAX_NUM_OF_CHANNELS * IFX_CFG_OE_OSCI_NUM_OF_SAMPLES * 4

Default values for the above mentioned macros are provided in *ifx_oe_oscicfg.h* under Library/OneEye.

References



- › AURIX™ Development Studio is available online:
- › <https://www.infineon.com/aurixdevelopmentstudio>
- › Use the „*Import...*“ function to get access to more code examples.



- › More code examples can be found on the GIT repository:
- › https://github.com/Infineon/AURIX_code_examples



- › For additional trainings, visit our webpage:
- › <https://www.infineon.com/aurix-expert-training>



- › For questions and support, use the AURIX™ Forum:
- › <https://www.infineonforums.com/forums/13-Aurix-Forum>

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2022-06

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2021 Infineon Technologies AG.

All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

OneEye_UART_Oscilloscope_1

_KIT_TC334_LK

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”).

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.