

## Contents

Product Overview .....	3
<b>Quick Start Guides</b> .....	7
Configuration Setup .....	7
The structure of X-Analyser Configuration setups.....	7
Classification of Components.....	9
Anchoring.....	10
Quick Start for CAN .....	13
Raw Trace Features.....	15
CAN Example Project .....	16
Quick Start for LIN.....	16
Importing legacy files from X-Analyser 2 .....	19
Loading Capture files - .cpr .....	19
Loading Signal files - .sig and .dbc.....	21
Loading Transmitter files - .otr and .tmr.....	23
Inject Error Frames.....	25
Injecting Error Frames Using a Kvaser CAN Interface .....	25
Quick Start for Picoscope Configuration.....	26
Supported Devices .....	26
Picoscope Configuration .....	26
Collecting/Analysing Waveforms.....	30
Picoscope Settings .....	32
Decoding from CAN_H or CAN_L only .....	33
Installation of X-Analyser .....	34
Main Features .....	38
Object Transmitter Taskbar .....	44
LIN Object Transmitter.....	44
Display Filters.....	45
Invoke the Filter Editor. ....	46
How to Construct Filter Criteria with Multiple Conditions Joined by One Logical Operator.....	52
Column Options .....	56
Virtual Interfaces.....	57

Real Time Playback .....	58
ISO15765 Support .....	58
Adding ISO15765 Component for an ECU.....	59
Setting Up an ISO15765 Object Transmitter.....	61
Interpreting ISO15765 Messages.....	62
UDS DTC Reader.....	62
Frame Filtering .....	63
Example Setup .....	65
Stop Filter Example .....	65
Pass Filter .....	66
J1939 Source Address Pass Rule .....	67
J1939 Higher Layer Protocol .....	67
Starting a J1939 Project .....	68
J1939 Message & Signal Interpretation .....	68
J1939 Message Transmission with Interactive Generator.....	68
J1939 Signal Panels, Scope and Gauges.....	68
J1939 Higher Layer Protocol .....	69
Starting a J1939 Project .....	69
J1939 Message & Signal Interpretation .....	69
J1939 Message Transmission with Interactive Generator .....	69
J1939 Signal Panels, Scope and Gauges.....	70
CAN Gateway .....	70
Adding and Configuring a Gateway .....	70
See Also.....	72
CANopen NMT Transmitters.....	73
CANFD Configuration .....	74
CANFD Transmitters.....	76
NMEA2000 Higher Layer Protocol .....	77
NMEA2000 Layout Setup .....	77
NMEA2000 Protocol Layer Transmitters .....	79
CANopen Higher Layer Protocol .....	81
Starting a CANopen Project .....	81
CANopen Message Interpretation .....	81

Load LDF Schedule in Object Transmitter .....	81
Interactive Generator .....	83
Introduction to CAN .....	88
CAN Layered Architecture .....	89
CAN Physical Layer .....	92
CAN Wiring .....	92
CAN Data Format .....	92
CAN-bus Digital Signaling .....	93
Bit Stuffing .....	95
CAN Frame Message Format .....	96
CAN Frame Message Format .....	98
CAN Higher Layer Protocols .....	101
Introduction to LIN .....	101
The LIN Communication Concept .....	102
Key Features of LIN .....	102
Inside the LIN Frame .....	103
Sync Break .....	103
Sync Field .....	104
Identifier Field .....	104
Data Field .....	105
Checksum Field .....	105
Applications for LIN .....	106
Frequently Asked Questions .....	107
FAQ: What is the format of the timestamp? .....	107

## Product Overview

X-Analyser provides the following benefits:

- Modernised Graphical User Interface (reconfigurable)
- Improved message filter (stop and pass)

- Dials and Gauges graphical display for CAN/LIN Signals
- Unified Diagnostic Services – Object Transmitter
- Multiple Message Trace, Signals, Scope Displays
- Interactive Signal Editor – for reverse engineering
- Touch screen support – great for in-vehicle use
- Easy to use - Great as a training tool

The X-Analyser is a versatile Windows based development tool for Controller Area Network (CAN) and Local Interconnect Network (LIN) based systems. The aim of the X-Analyser is to offer to the user ease of use, flexibility, increased performance and more powerful features. The X-Analyser allows the user to monitor, analyse and log bus data. It also allows CAN transmissions and a host of other facilities.

The table below gives a full comparison of the features available in each version. It should be noted that features available depend upon the level of the license and options that have been purchased.

	<b>X-Analyser</b>		
	<b>Professional Edition</b>	<b>Standard Edition</b>	<b>Eco Edition</b>
Multi-Raw Data Trace Display	Yes	Yes	Yes
Engineering Signals	CAN & LIN	CAN & LIN	None
Signals/Database Editor	Integrated & Standalone		No
Multi-Signal Display	Yes	Yes	No
Multi-Signal Scope Display	Yes	Yes	No
Dials & Gauges Signal Display	Yes	Yes	No
Stop /Pass Filters	Stop & Pass	Stop & Pass	Stop & Pass
CAN-FD	Virtual Bus	Virtual Bus	Virtual Bus
CANdb/LDF Support	Yes	Yes	No
Import ASC Log Files	Yes	Yes	Yes
Vector CAN Interface Support	Yes	Yes	Yes
Kvaser CAN Interface Support	Yes	Yes	Yes

	<b>X-Analyser</b>		
	<b>Professional Edition</b>	<b>Standard Edition</b>	<b>Eco Edition</b>
Kvaser LIN Interface Support	Yes	Yes	Yes
Network Channels (1)	Unlimited	Single Channel	Single Channel
CAN/LIN Object Transmitter	Unlimited	Unlimited	Unlimited
ISO15765 Object Transmitter	Yes	Yes	No
Interactive Generator	Yes - CAN & J1939	No	No
Real-time CAN/LIN Playback	Yes	Yes	Yes
Configurable GUI	Yes	Yes	Yes
Windows OS	Vista  Windows 7  Windows 8  Windows 10		
Scripting (C#.net)	Yes	No	No
Scripting (T-Script) (2)	Yes	Yes with compatible Kvaser device	Yes with compatible Kvaser device
J1939 Option	Yes	No	No
CANopen	Yes	No	No



1. One interface needs to be nominated as the licensed interface that also acts as a dongle. For multichannel versions of X-Analyser, unlicensed supported interfaces can be used for subsequent additional channels.
2. Needs compatible Kvaser interface



## Quick Start Guides

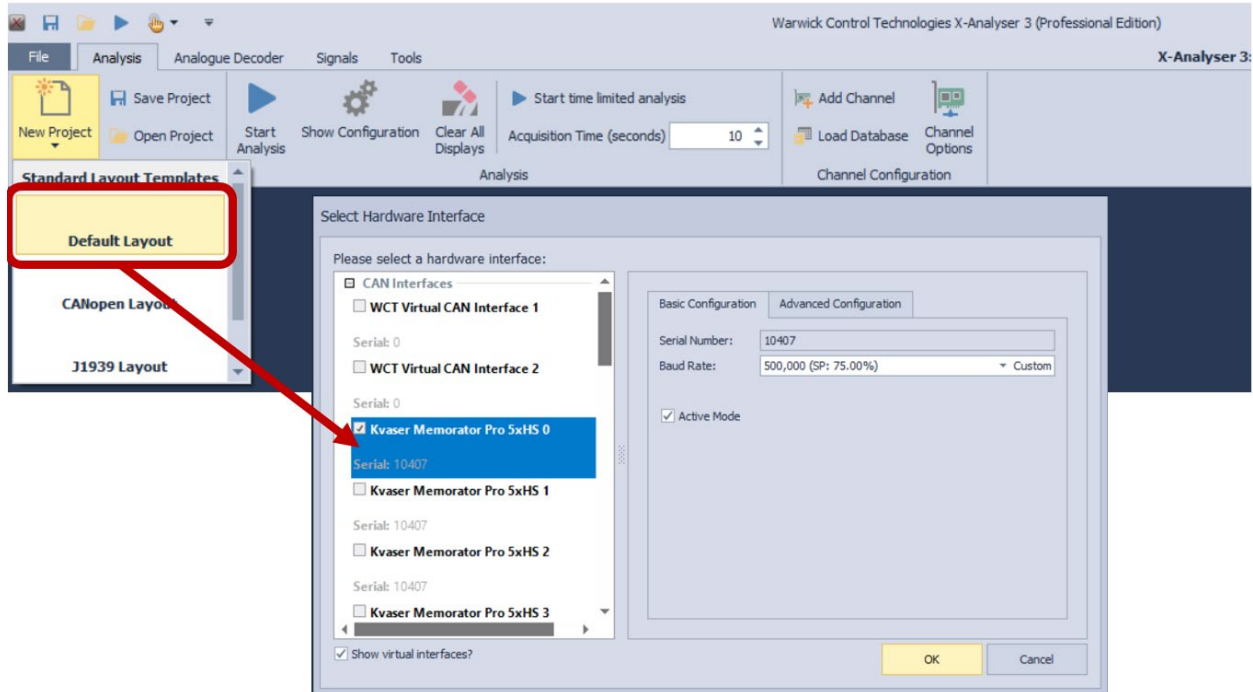
### Configuration Setup

[Product Overview](#) > Configuration Setup

The structure of X-Analyser Configuration setups.

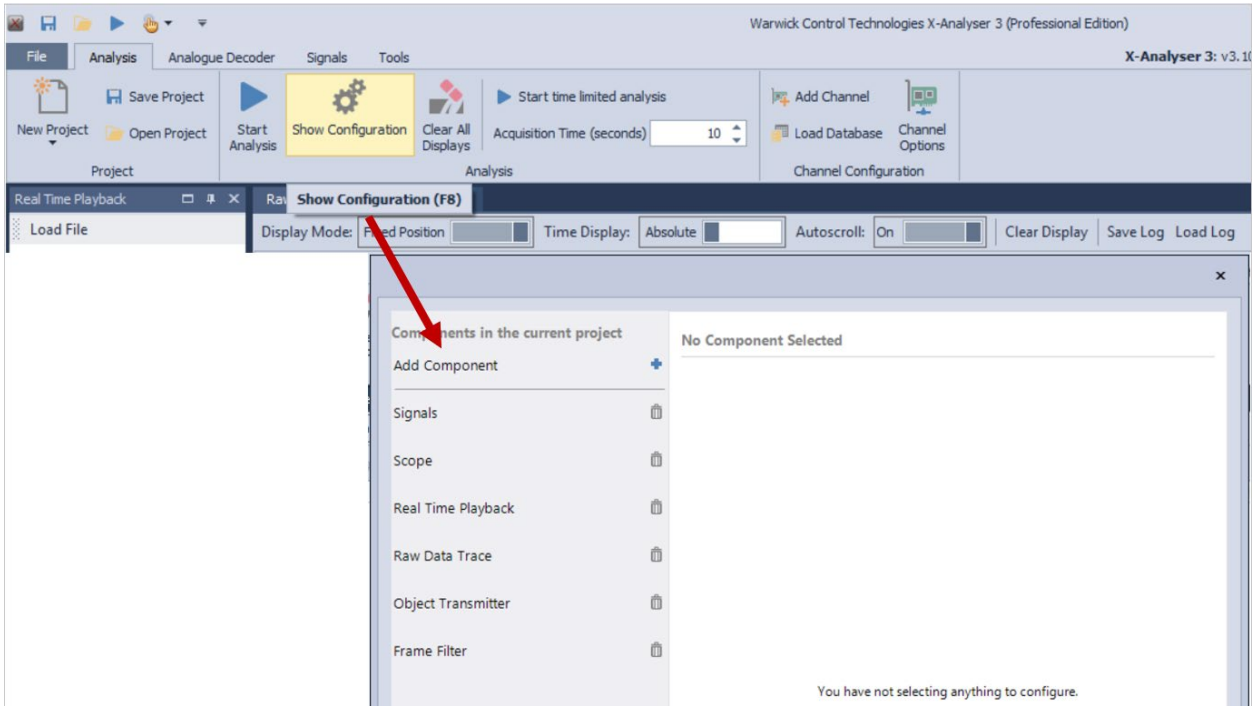
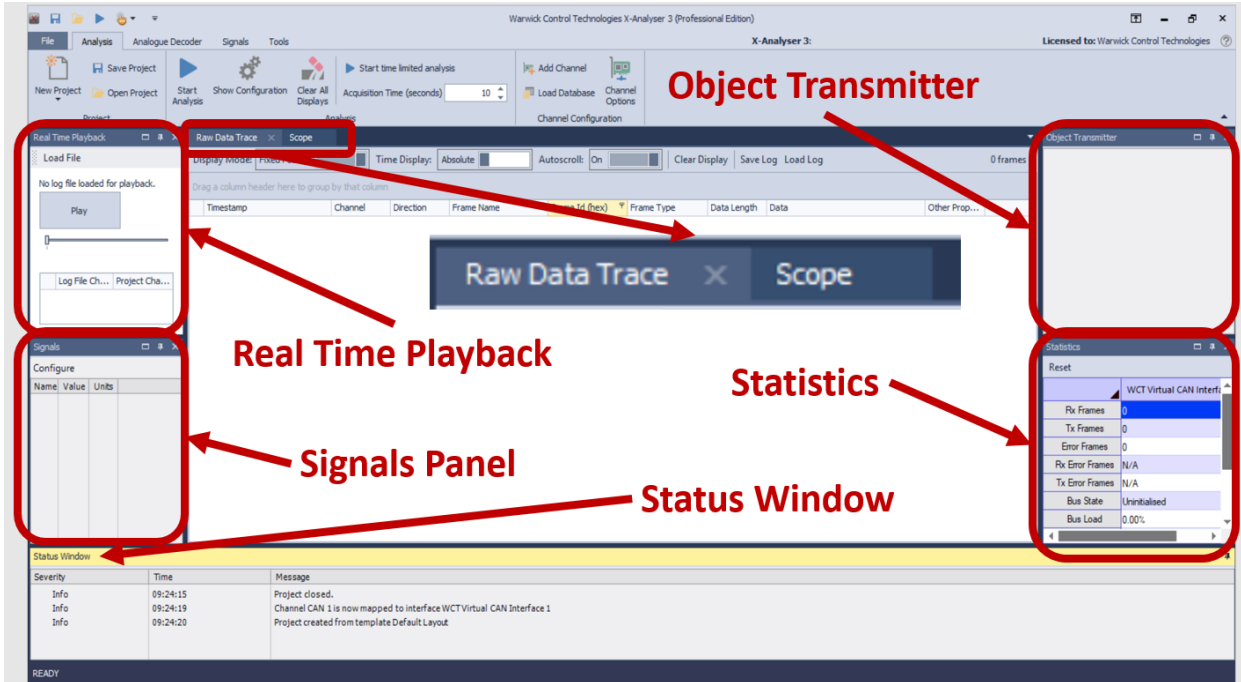
X-Analyser has a much more dynamic appearance in its configuration structure. There are many functions that are of modular forms. The modular components within X-Analyser form many structured features: The Components tend to be broken down in two sub-categories – Panels and Documents. Panels are anchored to the left and right edges of the display, and Documents are the main displays that are featured in the middle.

For your convenience, X-Analyser has a default configuration that places the most commonly used Panels in the display for you. On starting up X-Analyser, go to **New Project** and select **Default Layout**.



Let us consider the default display for the initial X-Analyser configuration. In the illustration below, the default panels are shown, which include:

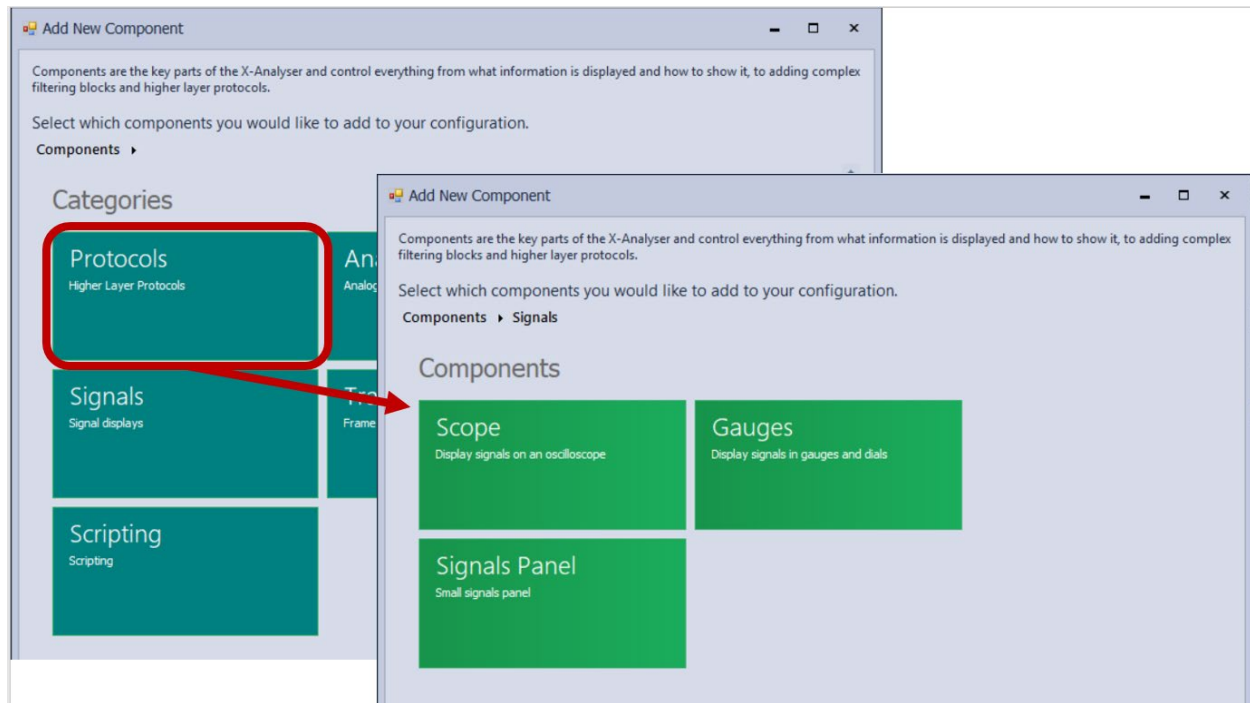
- Raw Data display
- Signals display
- Real Time Playback
- Signals Configuration
- Transmitter Configuration
- Statistics
- X-Analyser Status



If you were to configure these individually from scratch, the set up would appear as shown. This is accessed by selecting **Show Configuration**, and you will see all the default components that are provided in the default configuration. If you wish to delete a panel, click on the bin to right of Component in current project.

To add another component, click on **Add Component**, and you see the choices of components as shown below. Here we will review Components that are possible within X-Analyser.





Here you can see in green the main Components that are immediately available. There more Components with the Catagories (in blue). For example, if you select Signals Category, you will see that there are three possible Components (Scopes, Gauges and Signals Panel). Under the other Catagories:

- Transport Layers will contain ISO 15765, J1939 and CANopen
- Triggering will contain Frame Trigger and Signal data Trigger
- Transmitters will contain Object Transmitters and Real Time Playback

Once any more Components are added or deleted, you may save this configuration for later call up. This is done by clicking on the Save Project icon and saving your .xfg file where you wish.

### Classification of Components

Within X-Analyser, we have classified the Components depending on where they can be anchored within the X-Analyser display. The two classifications are:

#### *Documents*

These are anchored in the middle of the display. Documents include:

- Raw Data Trace
- Scope
- Gauges
- Transport Layers (ISO 15765, J1939, CANopen)

## Panels

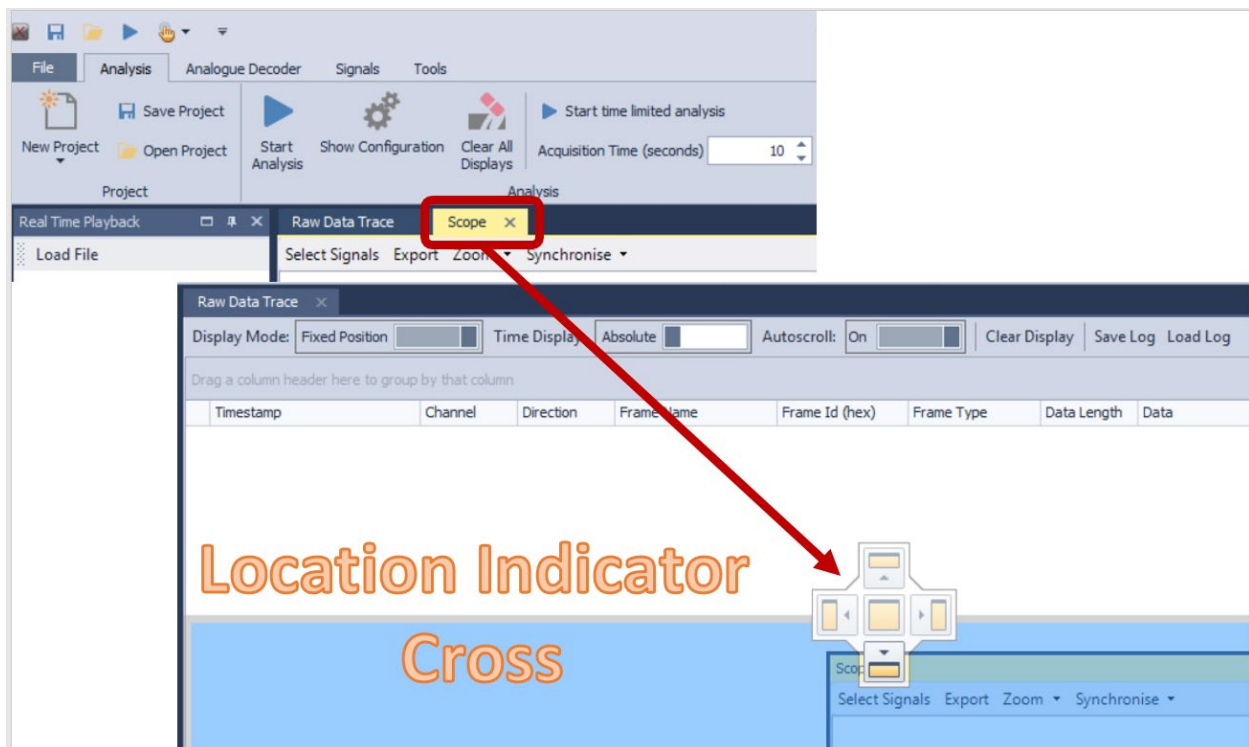
These are anchored on the edges of the display. These include:

- Real Time Playback
- Signals
- Object Transmitter
- Statistics

## Anchoring

### Documents

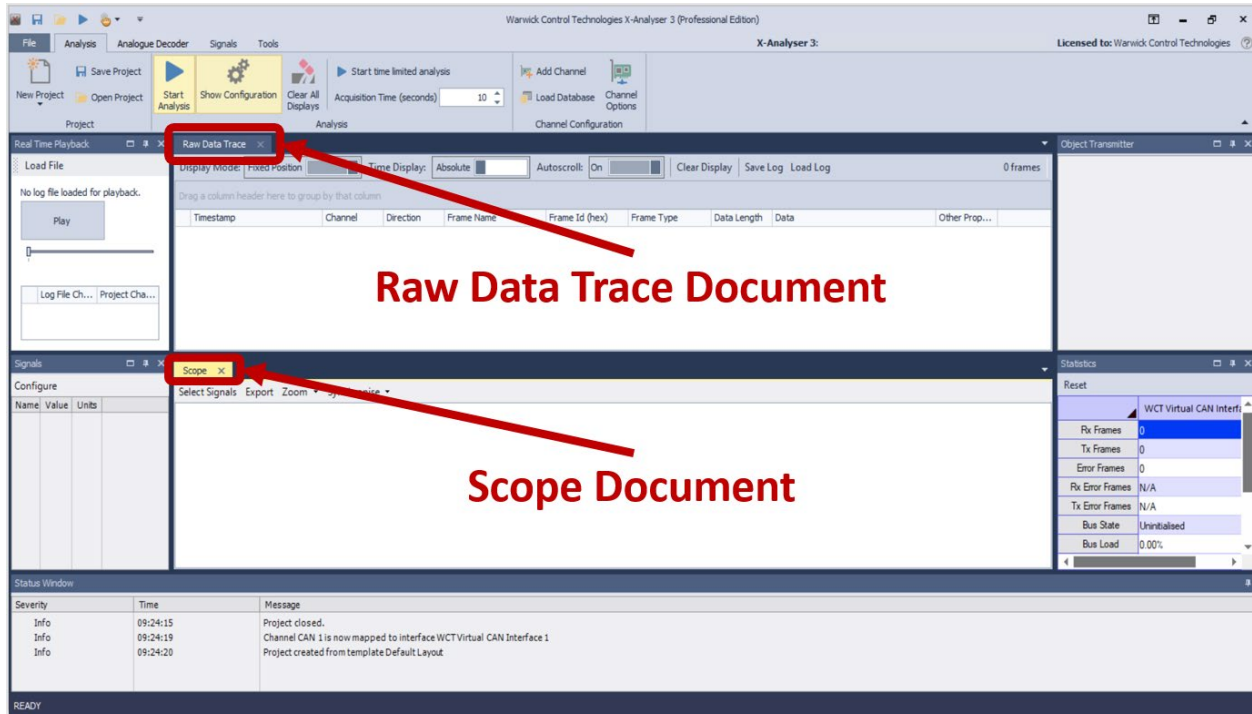
Documents can be grabbed via the mouse and dragged to a predetermined area in the centre area of the main display. For example, to dock the **Scope** display into an area, go to the **Scope** tab on the main display, left click with the mouse and hold. Drag the Scope window using the mouse arrow to indicate location as shown below.



The location indicator cross shows the possible anchoring positions:

- Split main display vertically left or right

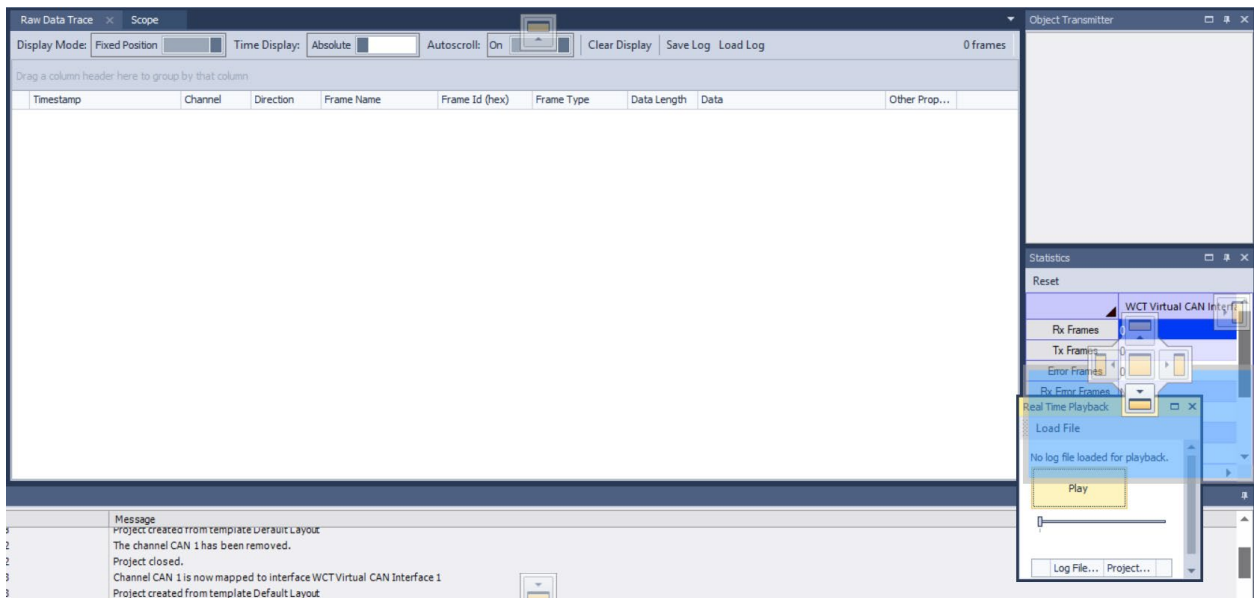
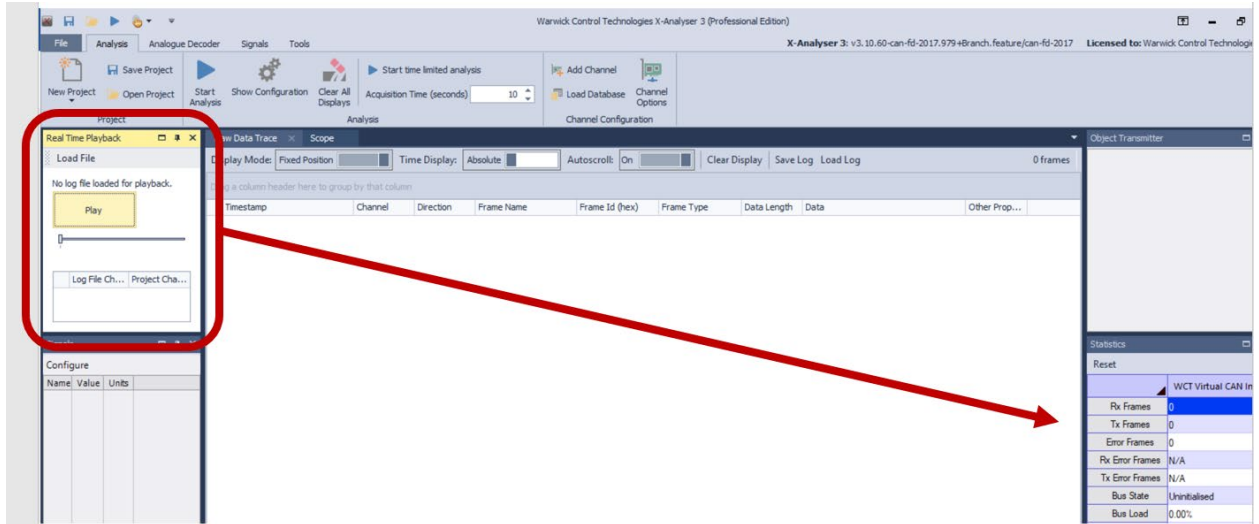
- Split main display horizontally left or right
- Place in middle with Raw display to selected by a tab.



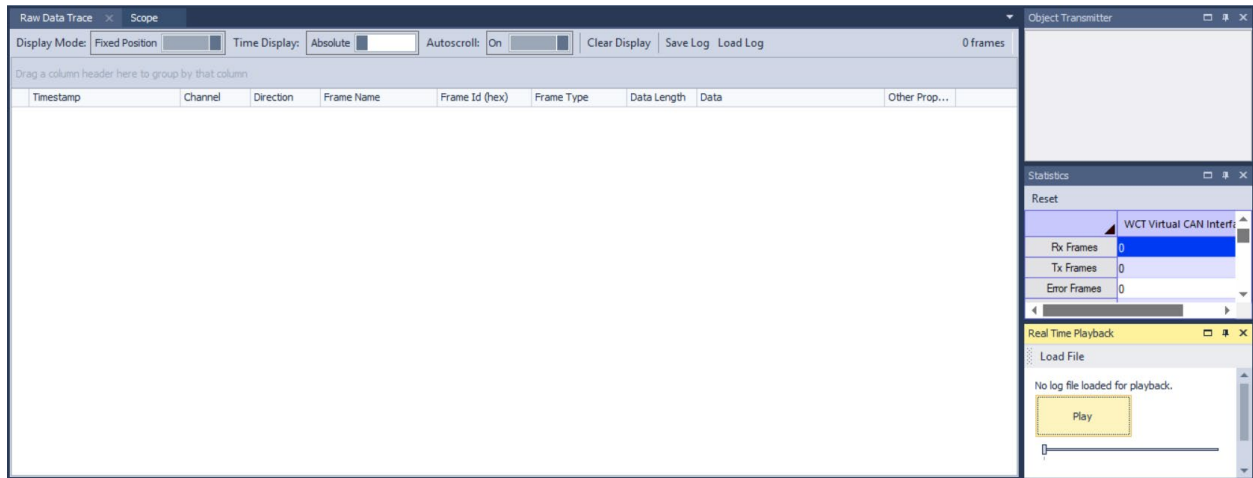
In this example, we have moved the Scope window with the mouse arrow to indicate a horizontal split. The result placement is shown above. Documents as listed above can only be placed in the Centre area.

### Panels

It is recommended that Panels reside in the side areas of the display. Indeed during default settings, loading of Panels, they are arranged in the side areas. Panels can be grabbed via the mouse and dragged to predetermined panel area in the side bars. Also, Panels can be anchored in the centre display. In the example below, we move the **Real Time Playback** from the left top side bar to the right bottom side bar. This will give more area for Signals for example. To perform this, left click on the mouse and hold on the top of the **Real Time Playback** pane. Then drag it to the bottom right and place mouse arrow in bottom box of the position indicator cross as shown below.



So the resulting display will appear as below.



It is possible to anchor Panels into the centre area of the display. The operation is the same as shown in the section on moving Documents.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

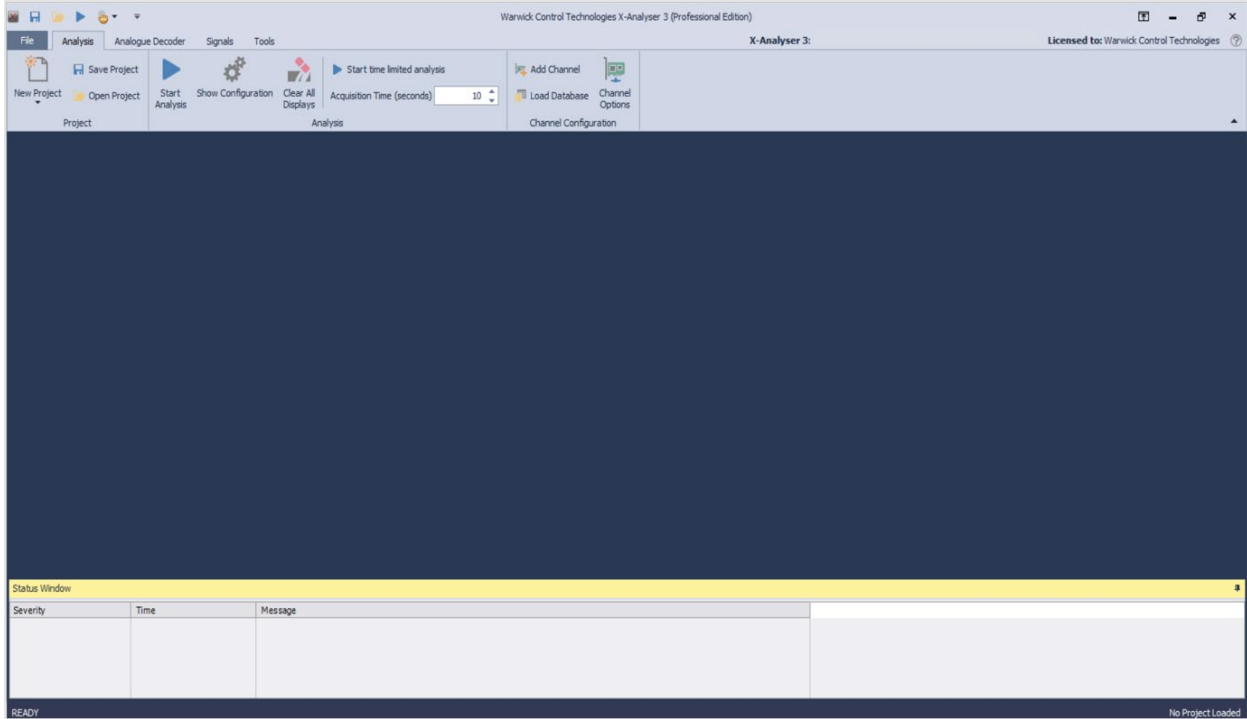
## Quick Start for CAN

Quick Start Guides > Quick Start for CAN

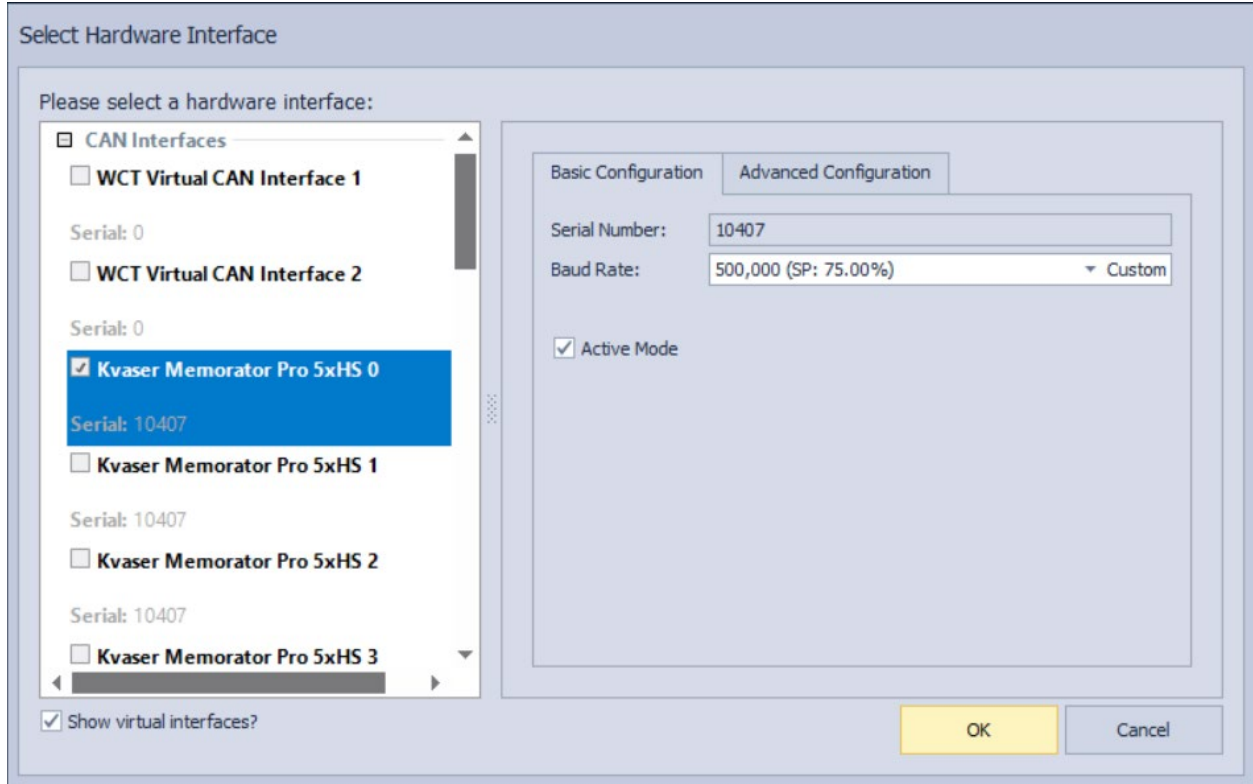
First connect your interface to the computer then install all the necessary drivers. See Drivers for more information.

Open X-Analyser by clicking on the program in the start menu.

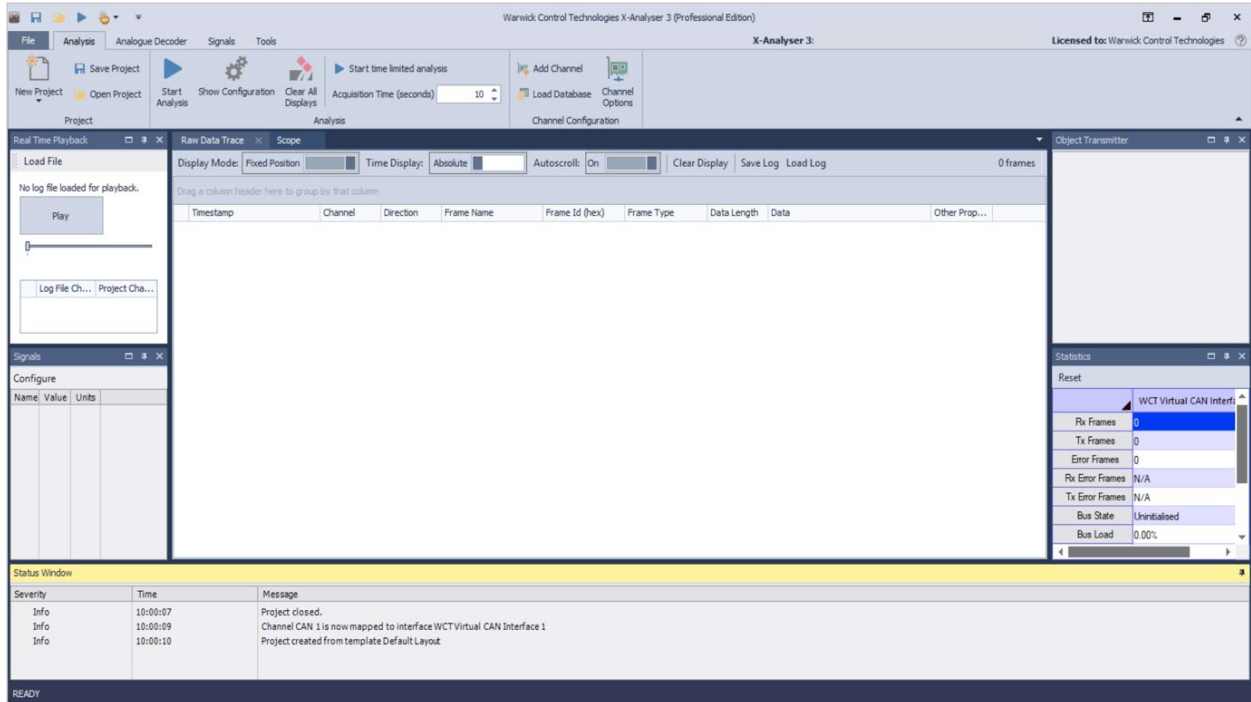
This screen should appear after a short wait.



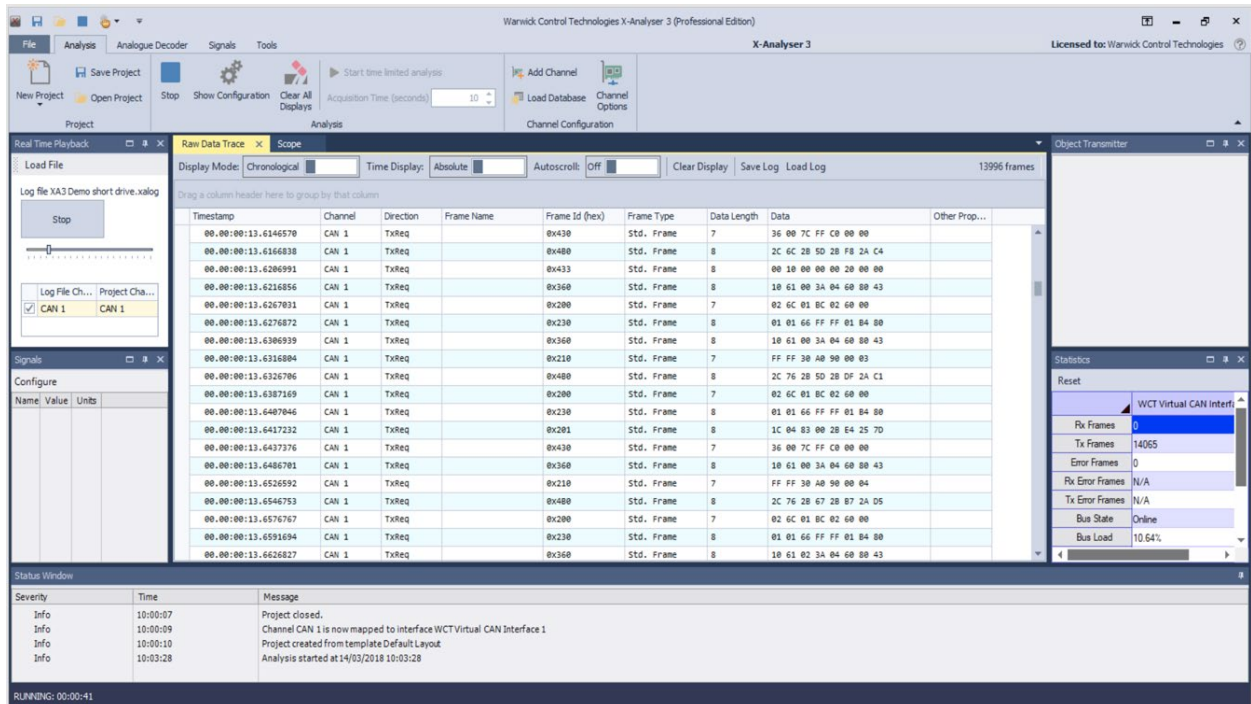
Click on the **New Project** button followed by **Default Layout**. The screen below will appear, select your interface from the list the select the baud rate of the CAN bus that is being analysed.



Then click **OK**, the screen below will appear.



Click **Start Analysis** in the top left and your data should appear if everything is set up correctly.



### Raw Trace Features

- Display Mode - Chronological shows frames in chronological order as frames come through. Fixed Position shows frames grouped by ID.

- Time Display - Relative shows the update rate of the message relative to the previous frame with the same ID. Absolute shows the time since analysis was started.
- Autoscroll - When ON the raw trace display will show the most recent frame at the bottom of the display. When OFF you can scroll through previous frames. NB - when selecting a frame with Autoscroll ON it will turn OFF and the display will stop updating.

## CAN Example Project

We have provided with X-analyser an example project of a short car journey when connected to a Ford Fiesta's CAN bus. This can be accessed via **Documents -> X-Analyser 3 -> Examples -> Drive Project**. This contains a configured project **.xfg** and log file to play back data **.xalog** for the project.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Quick Start for LIN

Quick Start Guides > Quick Start for LIN

X-Analyser can view LIN frames in a similar manner as CAN frames. This section will describe the interface setup, configuration and collecting LIN data with a Kvaser LIN interface.

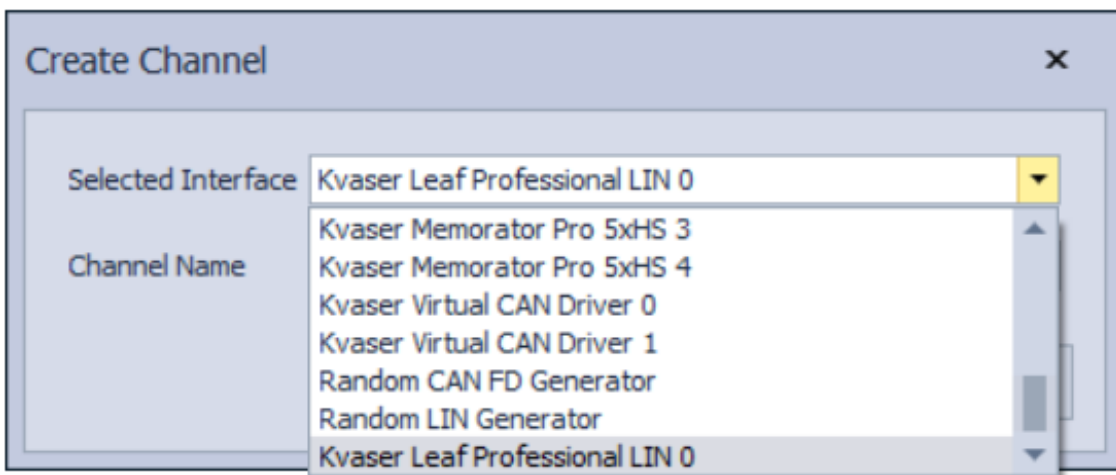
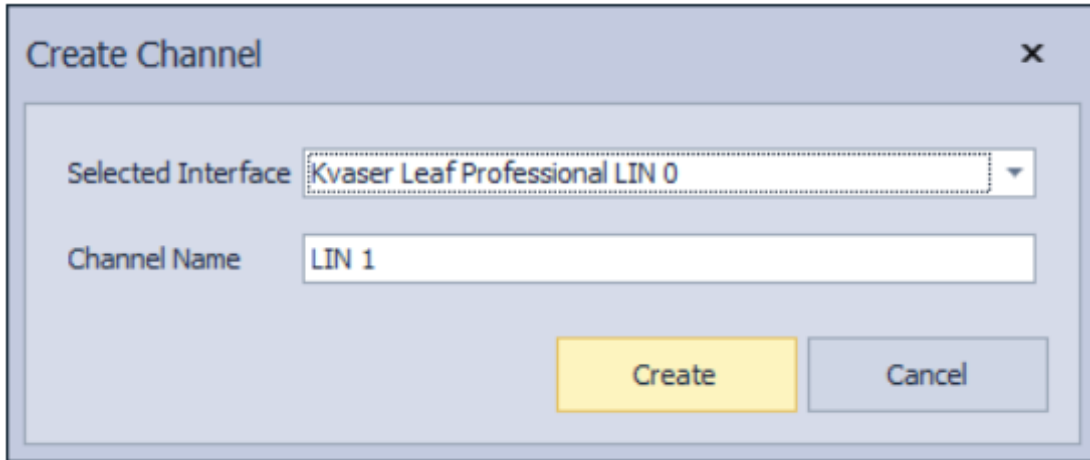
### Interface setup

Ensure the LIN interface is powered from the vehicle or unit to be tested. Refer to the Kvaser hardware notes for full information. The connections are briefly covered here. Ensure the following pins are connected on the D 9 connector of the Kvaser LIN interface:

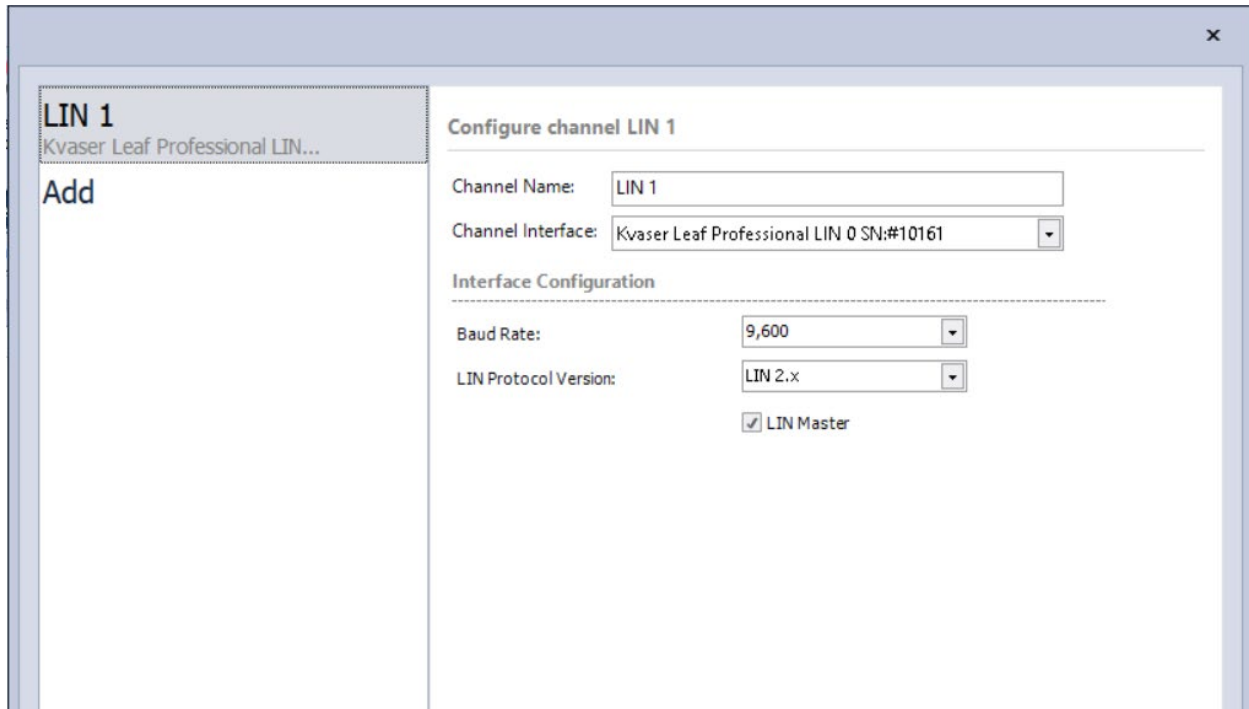
- Pin 9 – 12 volts power
- Pin 3 – Ground
- Pin 7 – LIN bus

Once the interface is connected, powered and plugged into the USB port of the computer, In main display, select **Add Channel**. In here you will see options available to you. If the LIN interface is connected and powered, the drop-down box will include the LIN interface as an option.





Select the LIN interface and give the Channel a Name (e.g. LIN 1). Click on **Create**. To configure the channel to the correct data rate and LIN version, click on **Channel Options**. Here you will see the LIN data rate and LIN version options. Also, it will ask you if the interface is to be used as a Master. In most cases, this box will be ticked. It is possible to have X-Analyser emulate a Slave node for test purposes.



Click on Close, and your LIN channel is ready to go.

If configuring from **New Project** -> **Default Layout**, ensure to configure a LIN channel there. You may add CAN channels to the project by the usual Configure Channel means.

On selecting the start button, you will see LIN frames in the Raw Data Trace display as shown below.

Raw Data Trace						
Display Mode: Fixed Position		Time Display: Absolute		Clear Display   Save Log   Load Log		
Drag a column header here to group by that column						
Timestamp	Channel	Direction	Frame Id (hex)	Frame Type	Data Length	Data
00:02:33.0030000	LIN 1	Rx	0x001	SlaveResponse	2	80 00
00:01:16.0080000	LIN 1	TxReq	0x002	HeaderResponse	8	00 0F 00 00 00 00 00 00

Here you will see the LIN Frame particulars:

- Timestamp
- Assigned Channel
- Direction – This is the direction of the LIN data (message response).
- Frame Identifiers (IDs)
- Frame Types
  - Slave Response – Master Header/Slave Response
  - Master Request – Master Header/Master Response
  - Header Response – Viewing nodes on the bus (Master and its Slaves)
- Data Length Code (DLC)
- LIN Data

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## Importing legacy files from X-Analyser 2

Quick Start Guides > Importing legacy files from X-Analyser 2

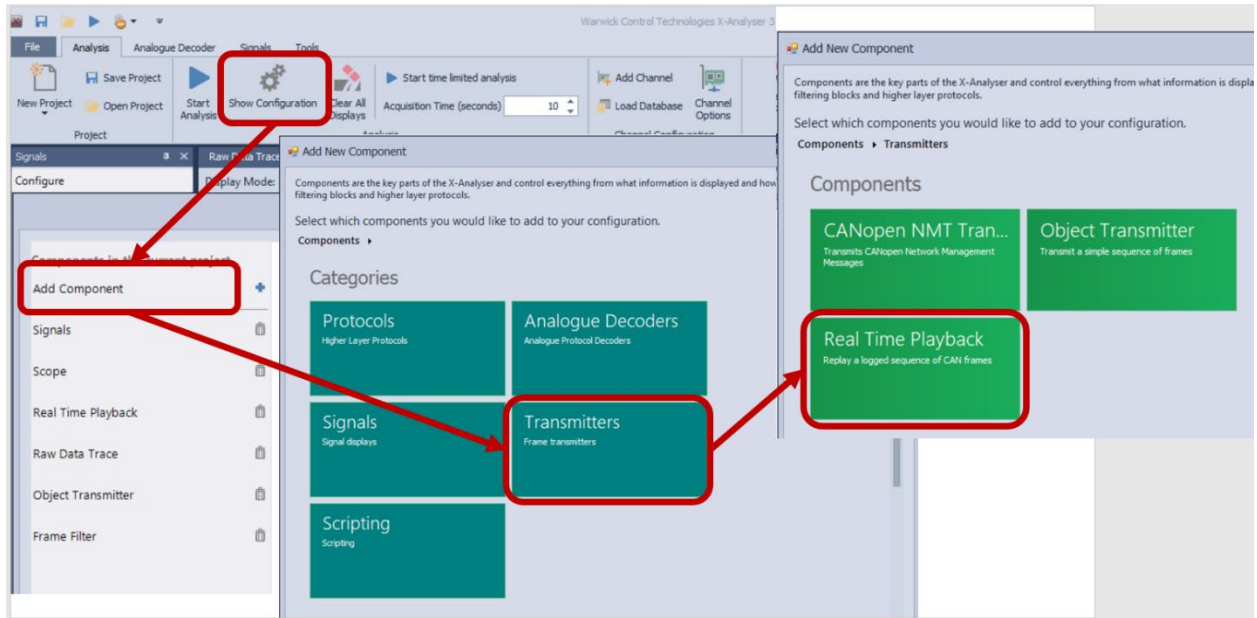
With the new features and enhancement in X-Analyser version 3, there are some areas that need defining when loading in files generated in X-Analyser version 2. In this section, the following file formats are covered

- Capture files - .cpr
  - To view
  - To Playback
- Signal files - .sig or .dbc
- General Transmitter files - .tmr
- Object Transmitter files - .otr
- Filter files - .ftr
- Trigger files – tgr

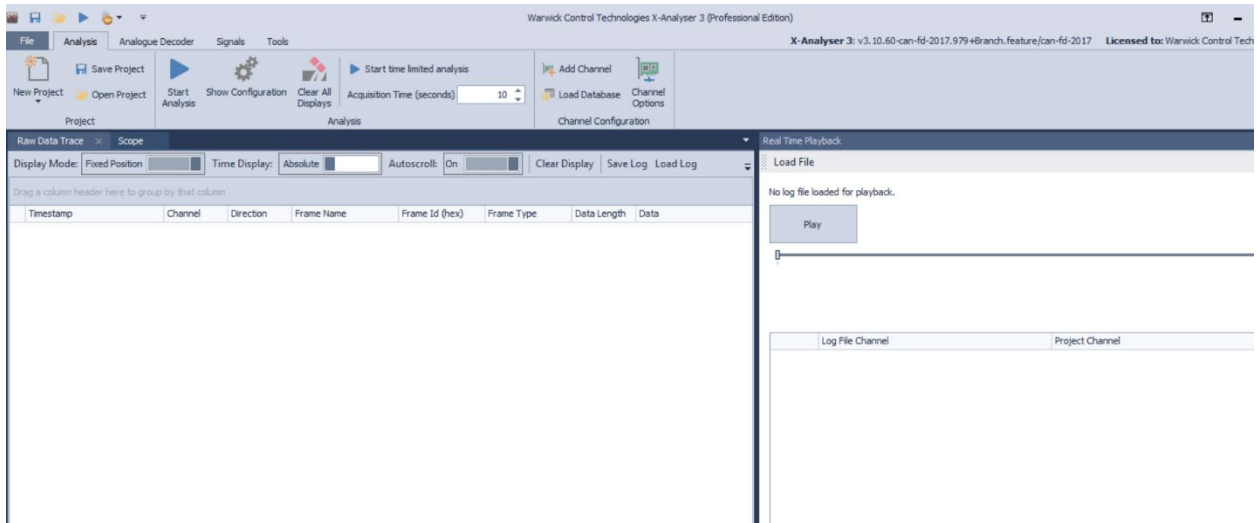
Loading Capture files - .cpr

Capture files that were generated from X-A 2 can be loaded into X-A 3. There are two modes for this. One is to load a .cpr file to view the messages chronologically. The other is to load the .cpr file into the Playback memory for **Real-time Playback**.

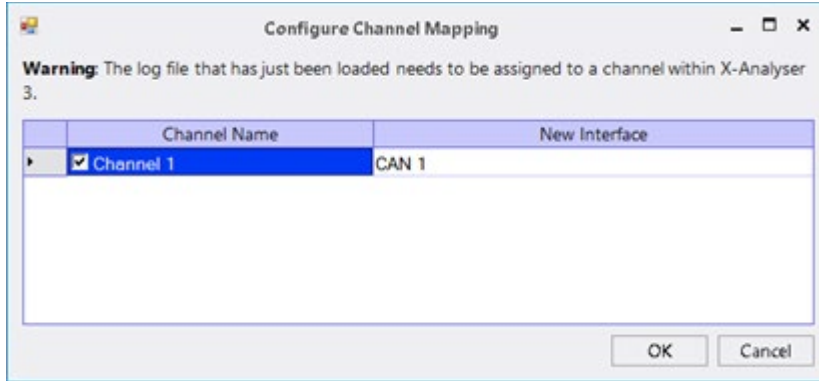
To load the .cpr for playback, ensure there is a **Real-Time Playback** Component in the main display. To do this, go to **Analysis** at the top. In **Analysis** select **Show Configuration**. This will show you what components are available for the main display. If the **Real-Time Playback** component is not there, select **Add Component**, and select the **Transmitter** Category.



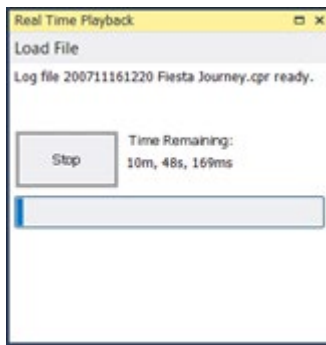
In there you will see the **Real-Time Playback** component. Select that, and it will automatically appear on the main display. Close the configuration windows and you will see the **Real-Time Playback** in the main window.



In the Real-Time Playback window, select load File. Browse to where your .cpr files are and select a file as you would in X-A 2. Once the file is selected, you will see a warning message to ensure that your playback file is assigned a channel. Select a channel, hit OK.



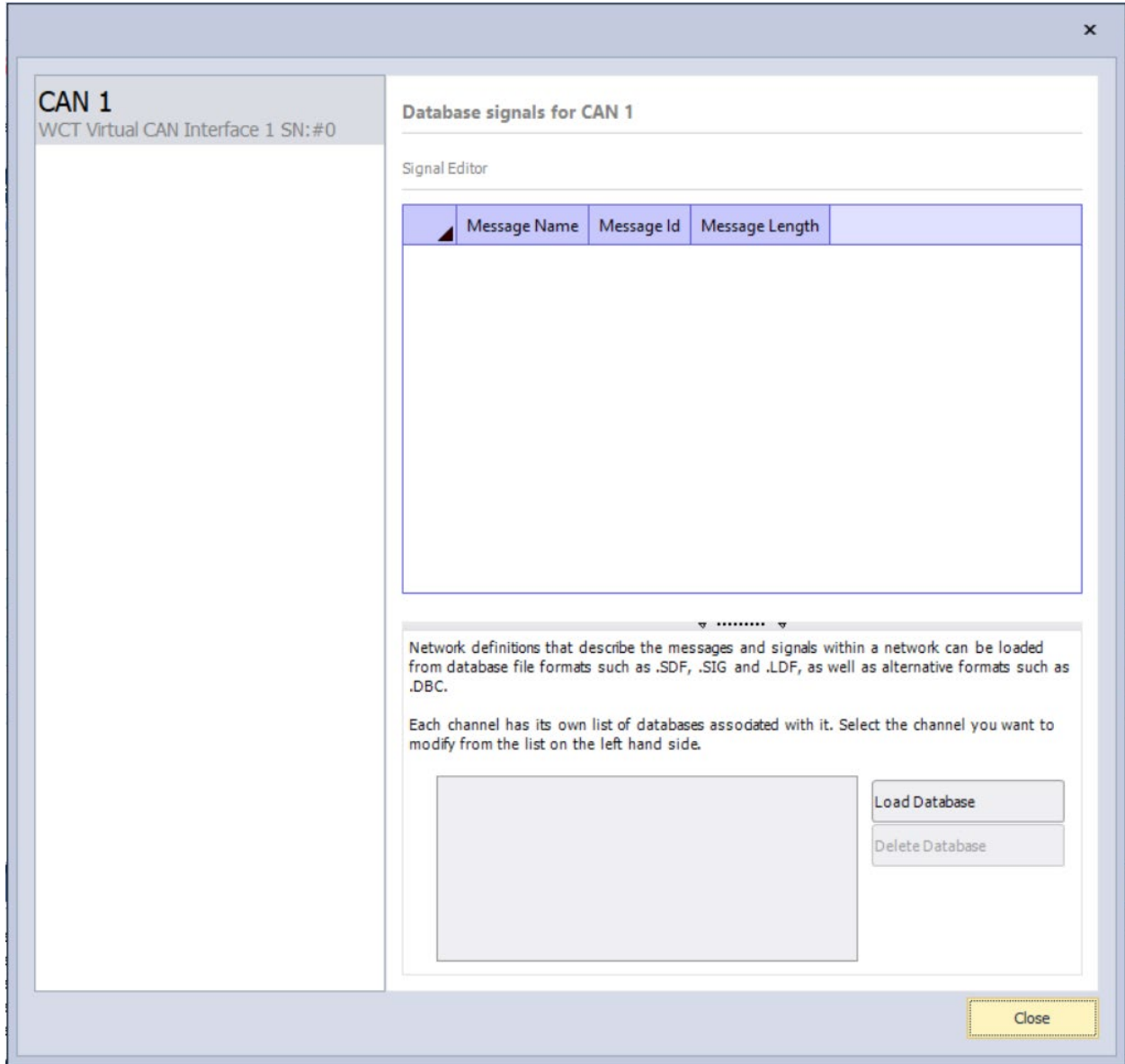
After selecting the Start button in the main display, in the Real Time Playback plan select the Play button. Once the playback starts, you will see a status bar showing the time remaining on the playback.

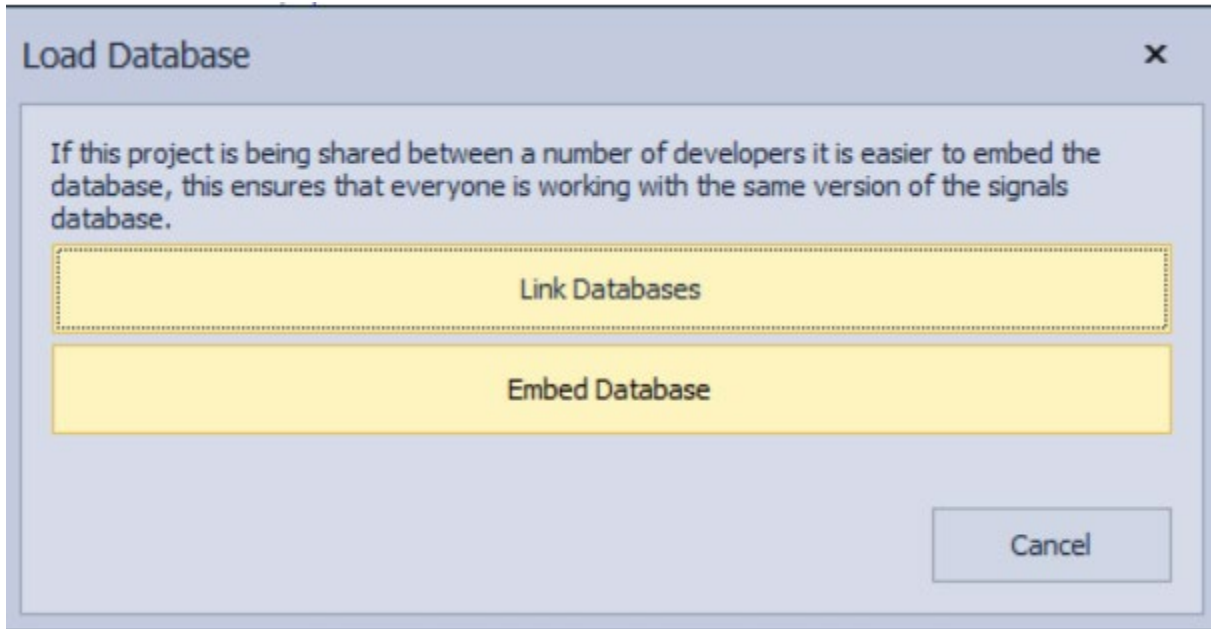


Loading Signal files - .sig and .dbc

Ensure there is a Signals panel on the main display. In the Main panel under Signals, select **Load Database**. Here you will see the display panel for selecting the channel and explanation of file formats that can be utilised besides .sig and .dbc. In this panel select **Load Database**, browse and select a .sig or .dbc file as you would in X-A2.

Once a file is selected, the X-A3 will ask you if you want to **link** the file at its location, or **embed** the file into this session. Most cases, you will embed the file.





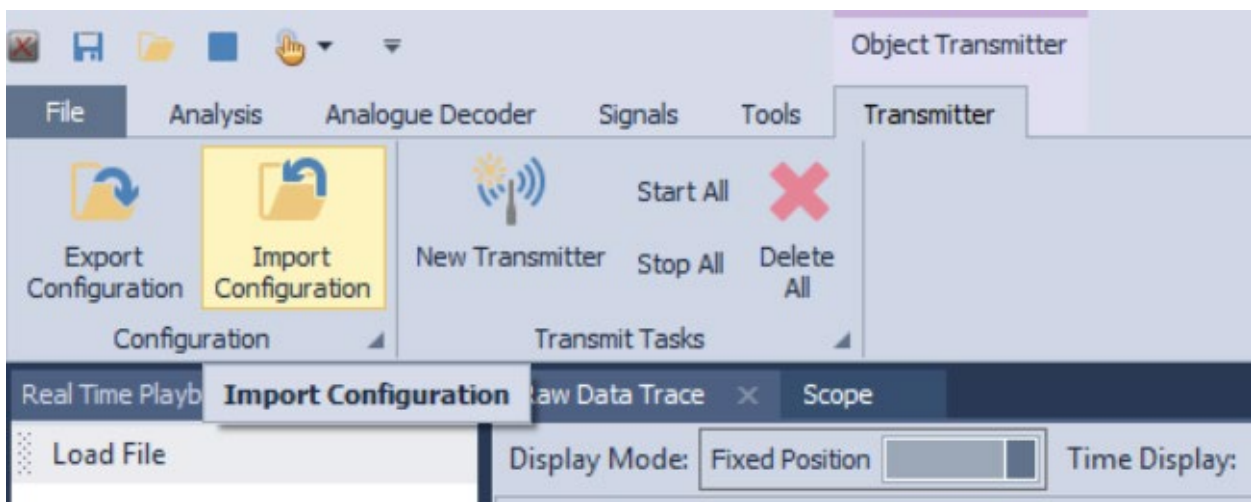
Once you have closed the **Load Database** window, back in the main display and in the Signals plane, configure the desired Signals for viewing by clicking the **Configure** button. Configure your signals as covered in section X.X.

Loading Transmitter files - .otr and .tmr

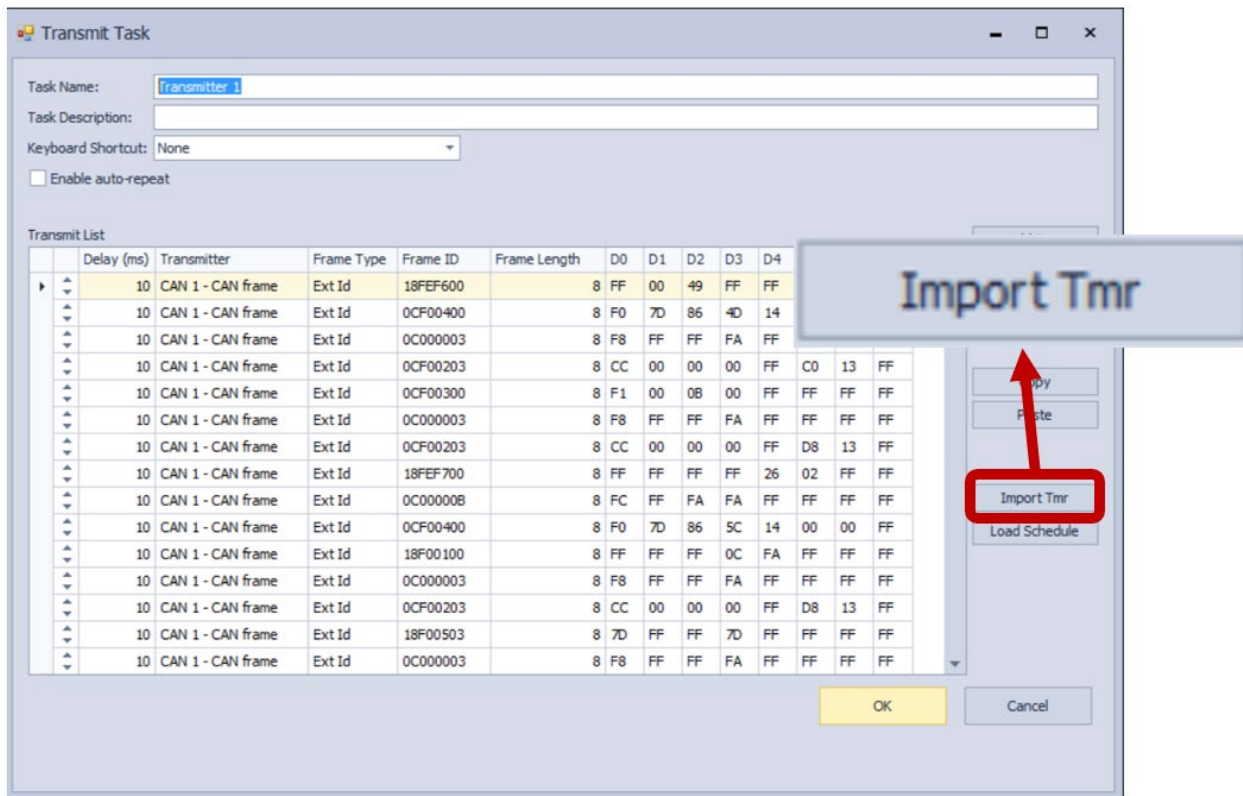
X-Analyser has a different setup than X-Analyser 2 for configuring transmitter functions. Transmitter files are now stored in the Project file - .xfg. Refer to the section on setting up transmitter routines. To load legacy X-A 2 transmit files (.otr & .tmr), the following must be considered.

To utilise existing General Transmitter Files (.tmr), there are two ways of importing.

First method is to click on the **Import Configuration** button in the Object Transmit taskbar.



Then browse until you find the legacy X-A 2 .tmr file you wish to use. You will note in the **Transmit Task** window, that each line is assigned a delay time. This is similar to setting up multiple Object transmitters. It is possible to set each line individually, or perform a global setting for all the messages. This is done by right clicking on the Delay column and selecting **Set all Delays**.



It is possible to have the routine cycle by clicking on the Enable auto-repeat box. Once you click OK and return to the main display, you may start your transmit routine by clicking on the Start button, then clicking on the Transmitter button in the Object Transmitter pane.

While the transmit routine is running, you will see the Transmitter button flashing. It is stepping through each of the messages in sequence of the file loaded.

The other method of loading a legacy General Transmitter (.tmr) file is to click on **Add New Transmitter**, then select the **Import TMR** button. Then follow the same procedure as above.

To utilise legacy Object Transmitter (.otr) files, you may only do this by using the **Import Configuration** button as described above.

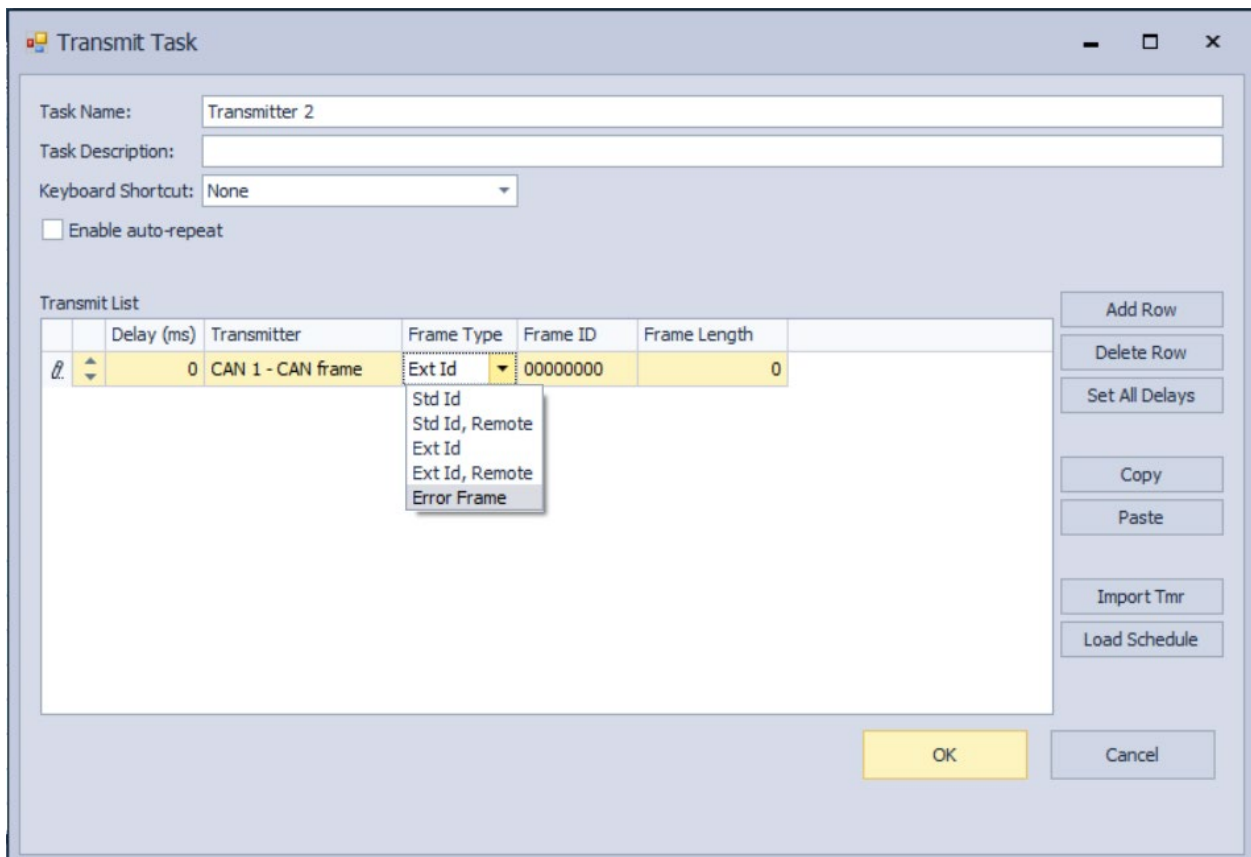


## Inject Error Frames

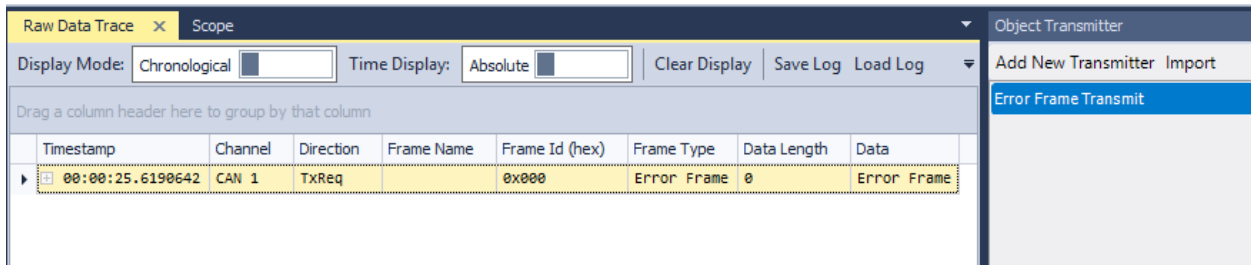
Quick Start Guides > Inject Error Frames

Injecting Error Frames Using a Kvaser CAN Interface

To send an error frame onto the CAN bus you must add an **Object Transmitter** as shown below:



When you activate the Object Transmitter with X-Analyser running you will see something like this:



---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## Quick Start for Picoscope Configuration

Quick Start Guides > Quick Start for Picoscope Configuration

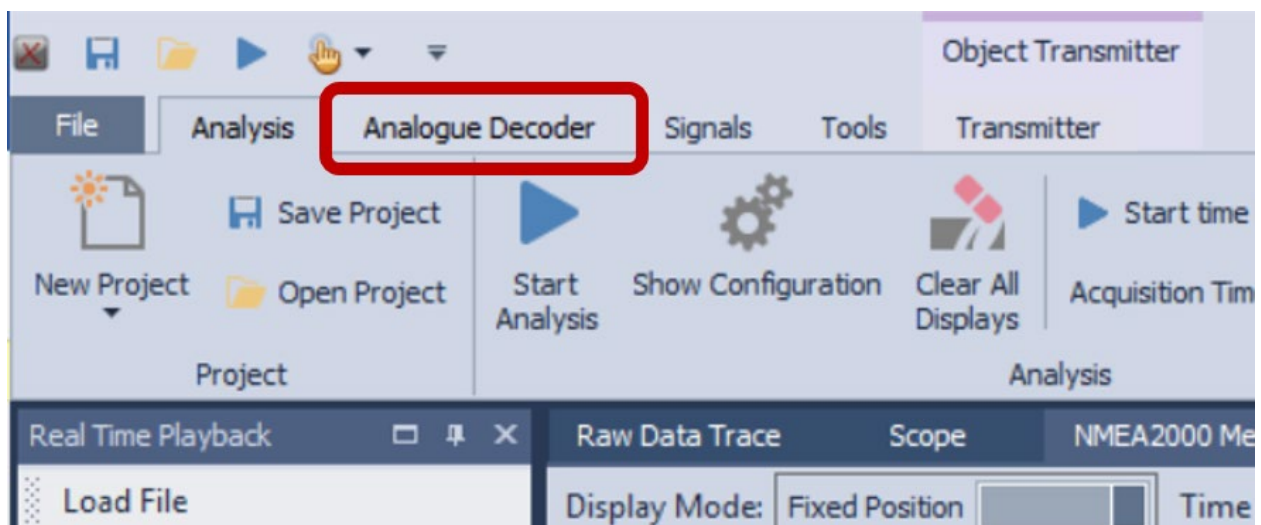
Supported Devices

PicoScope 2206b

Other 2000 series Picoscope's may work but they have not been fully tested.

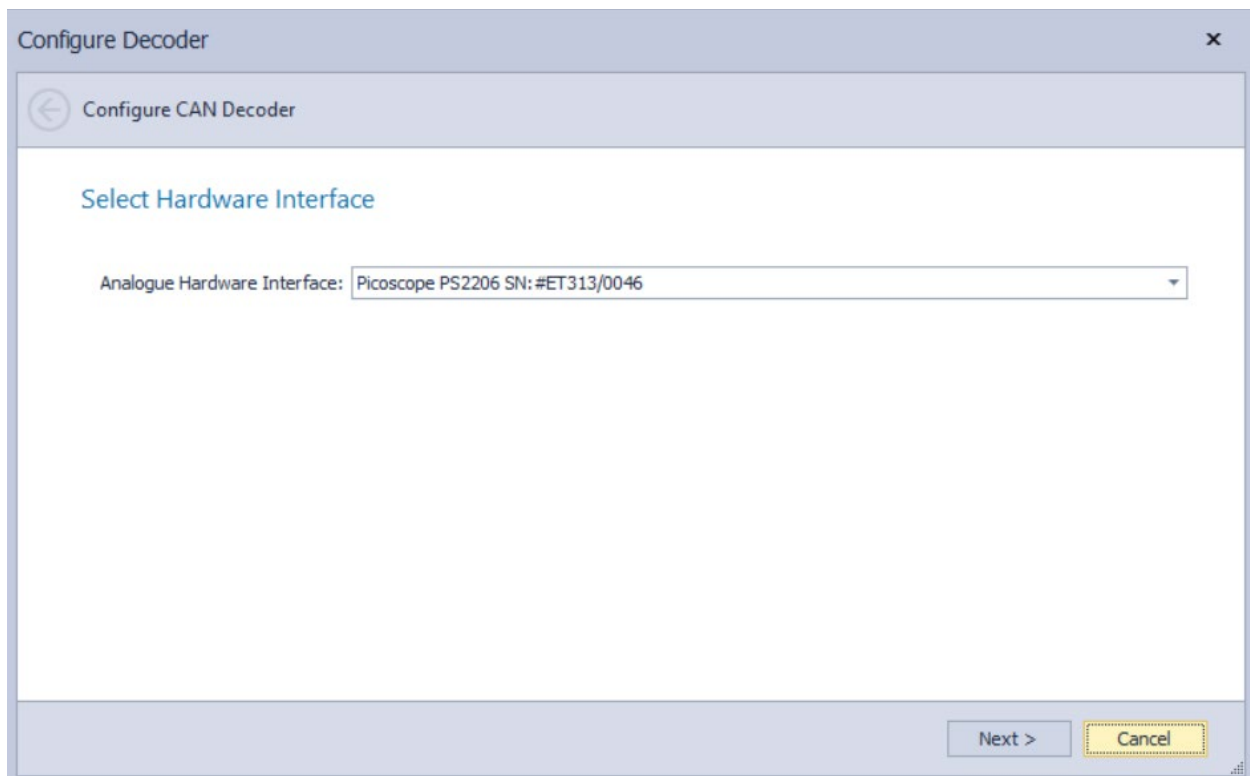
Picoscope Configuration

- Connect a Picoscope to USB and run X-Analyser.
- Select **Analogue Decoder** tab at the top of X-analyser screen.
- Select **Configure Decoder** button in **Analogue decoder** tab.

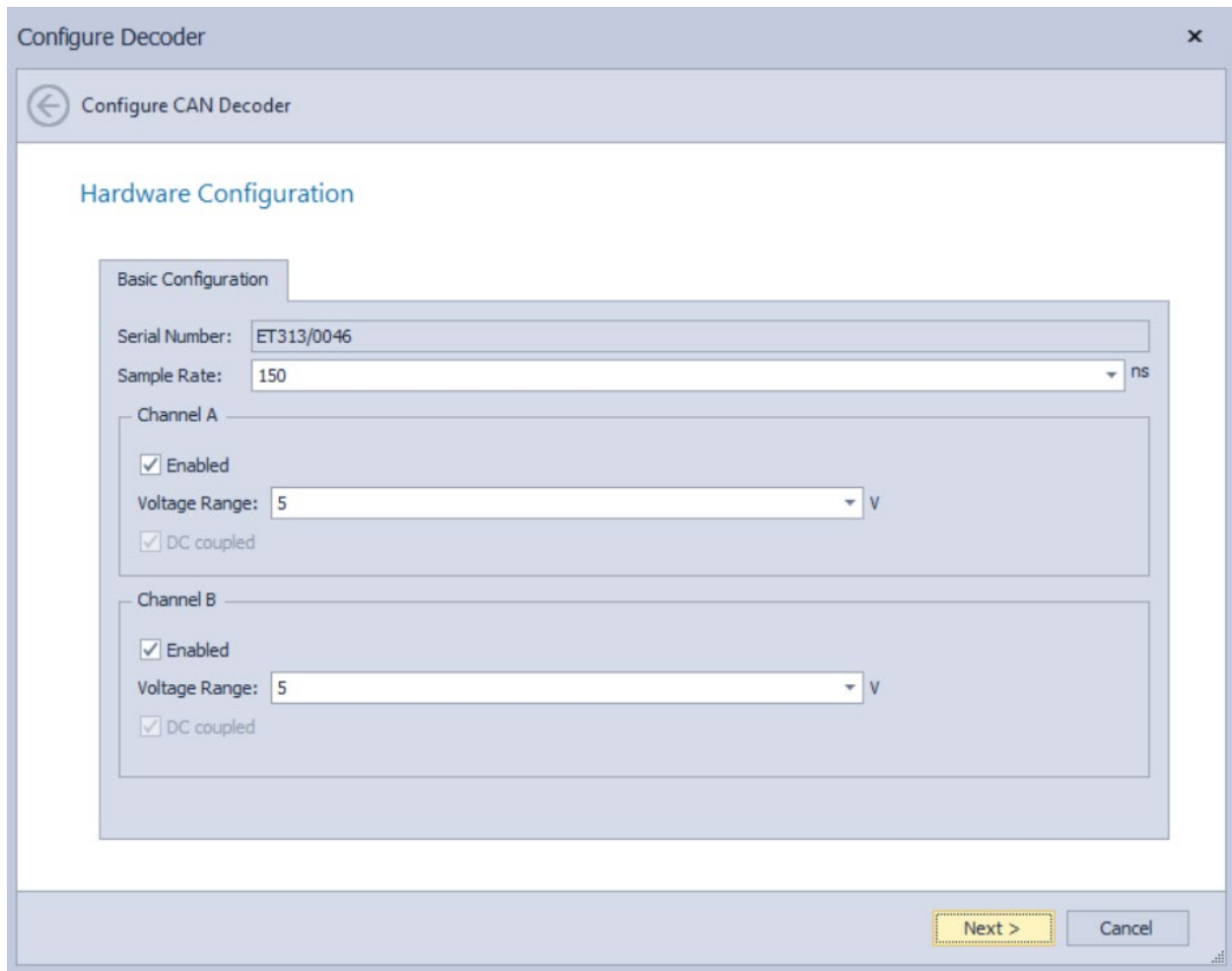




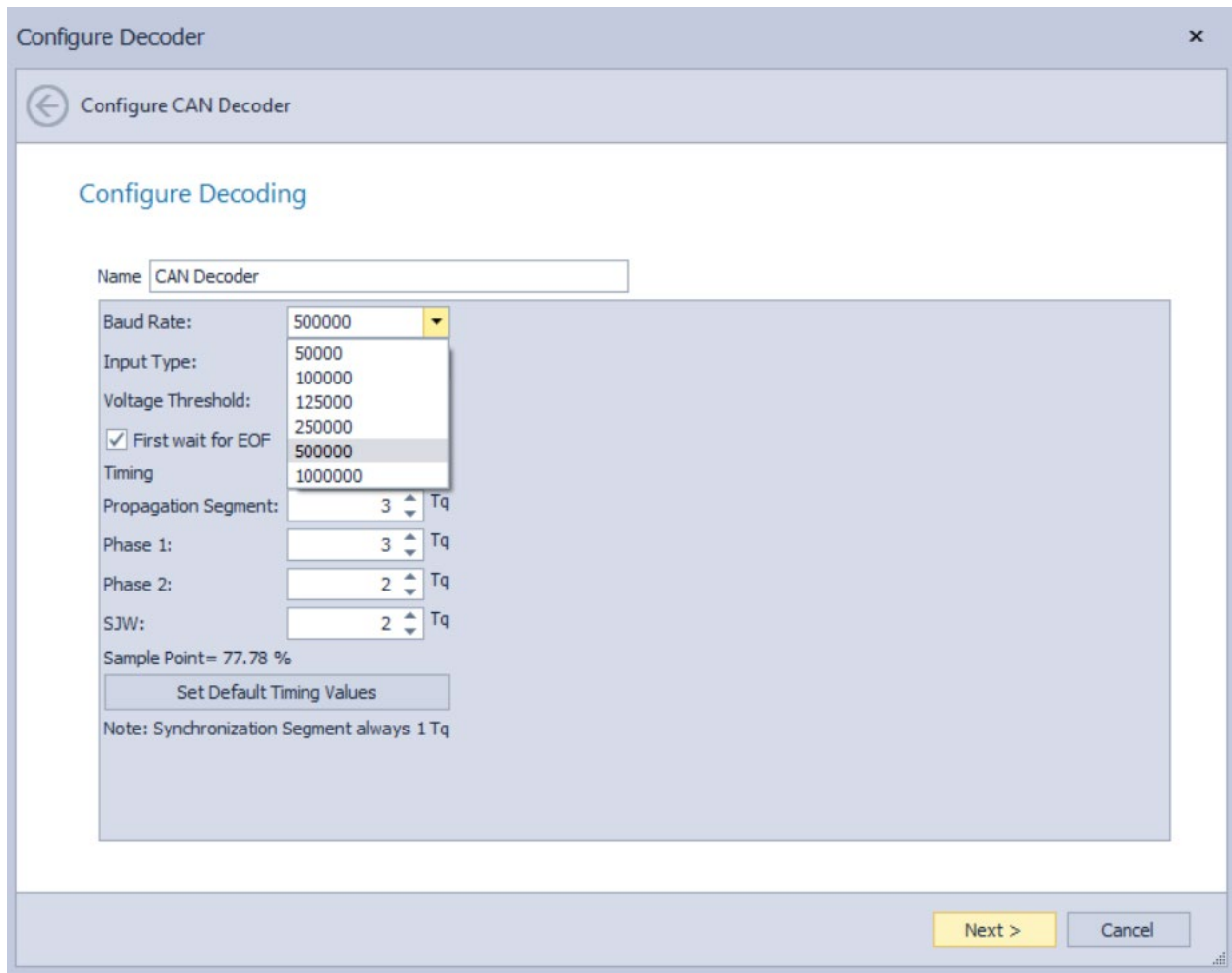
- The **Configure Decoder** window shall then open.
- Firstly make sure the correct picoscope interface is showing through serial number.



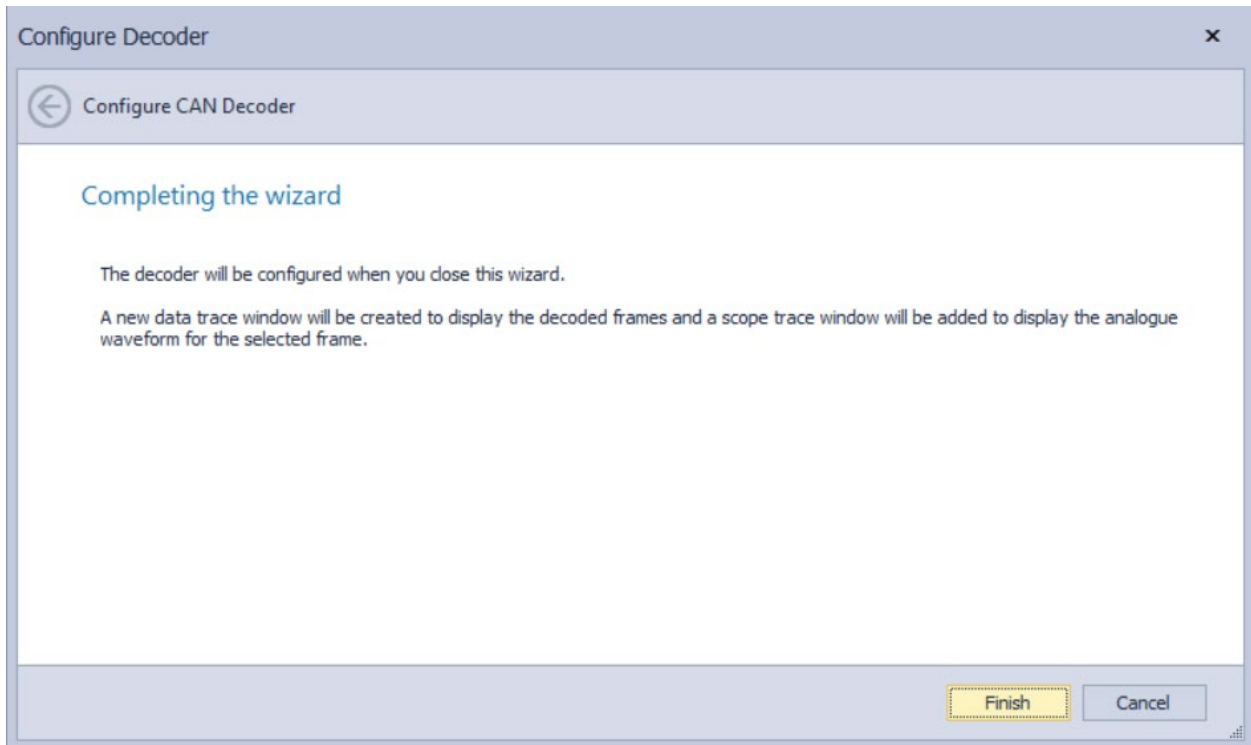
- Then select **Next**.
- The next **Configure Decoder** window shall show settings for **Sample Rate** and which channels are enabled.
- These settings can be left as their default settings for reading both CAN-H and CAN-L. The **Sample Rate** can be adjusted for more or less accurate collections.



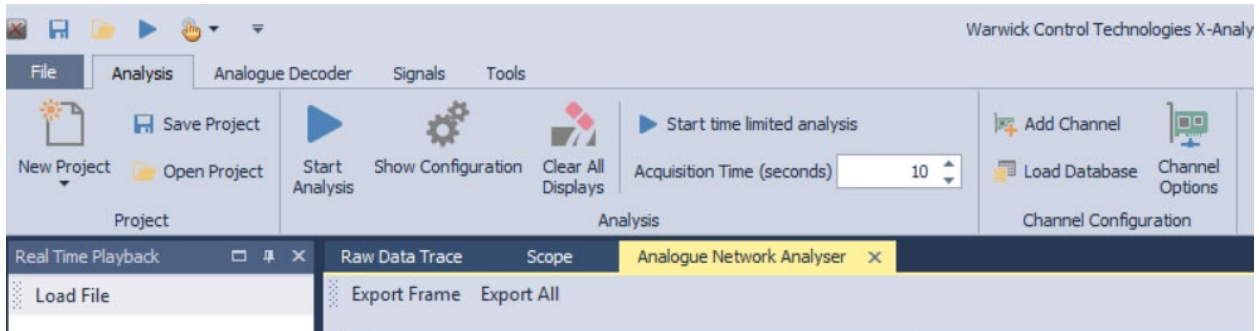
- Then select **Next**.
- The next **Configure Decoder** window shall allow settings for **Baud Rate**, **Input Type** and **Voltage Threshold**. Normally only the **Baud Rate** will need setting. Settings for **Input Type** and **Voltage Threshold** shall be explained later, this is for just reading CAN-H or CAN-L electrical signals. **Timing** settings will also be explained later within the manual.



- Then select **Next**.
- The next **Configure Decoder** window shall appear, the user will just need to select **Finish**.

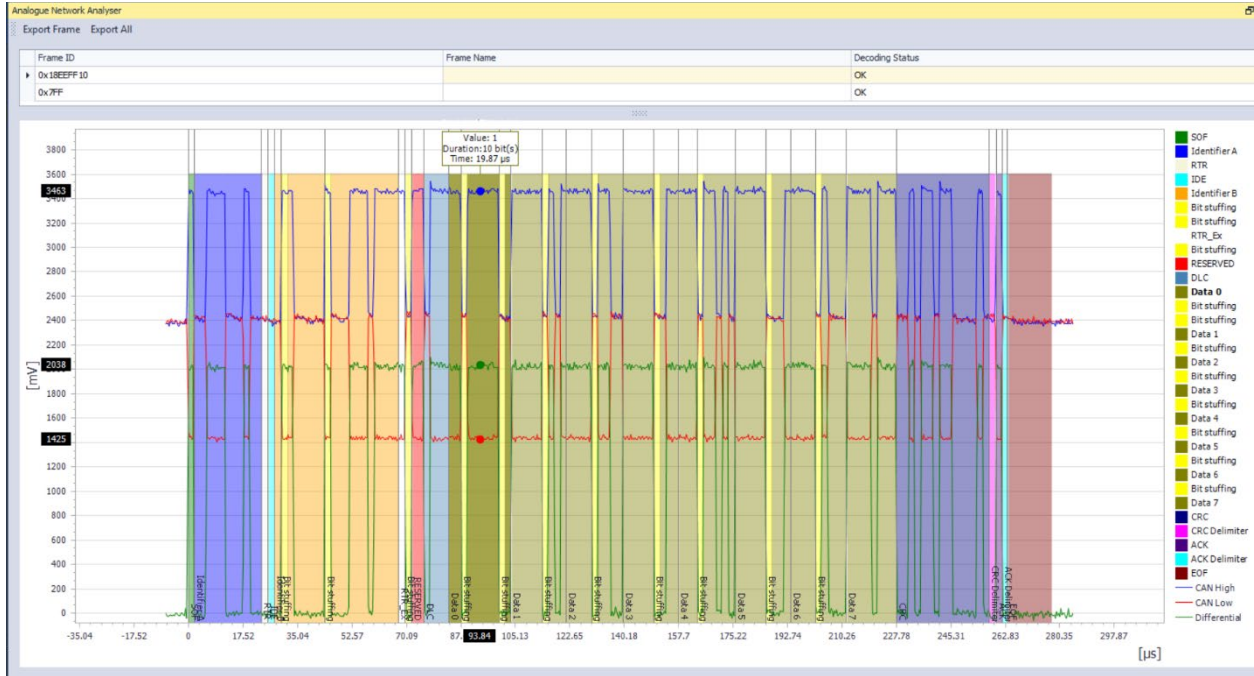


- This is the setup of the Picoscope channel completed you shall now see on your X-analyser application show a **Analogue Network Analyser** tab.

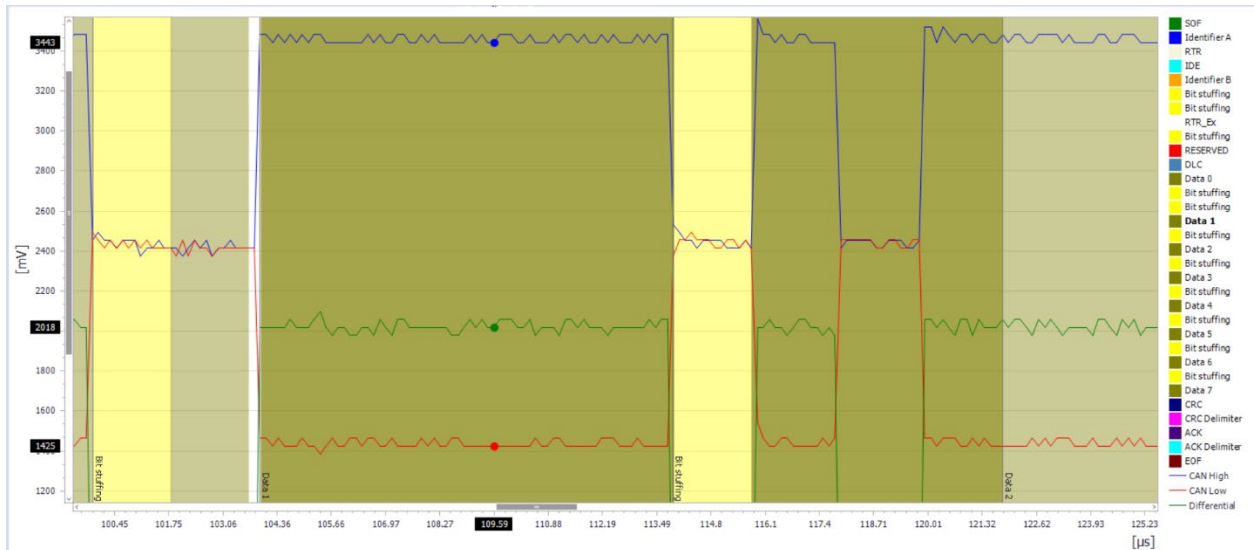


### Collecting/Analysing Waveforms

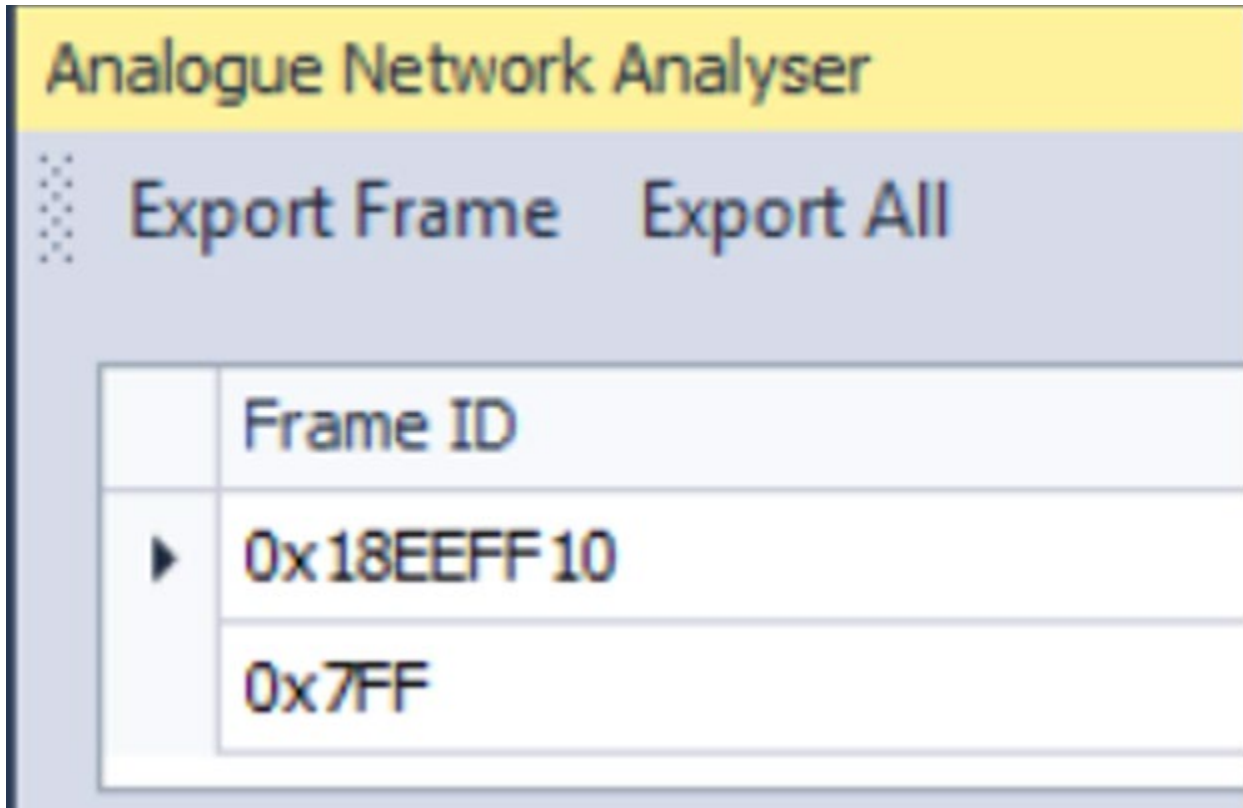
- Once the Picoscope channel is configured it can then be used to collect and analyse waveforms when connected to CAN bus. This is done by selecting **Start Analysis** and watching the frames come through.



- The tool tips will highlight CAN-H, CAN-L and differential voltages when using the cursor. You can also use the Shift key and cursor at the same time to zoom in on part of the waveform.



- You can also use the Ctrl Key and + or – to generically zoom in and out.
- These waveforms can also be exported into an excel file showing Voltage values and other data at the sample point. This is done by using **Export Frame** to export a single frame that is selected and **Export All** to export all frames on that collection.



Picoscope Settings

**PicoScope Configuration:**

**Sample Rate**

150 ns is a default one. If your analysis does not run, it means your PC is not fast enough. Try to increase it to 200 ns or even higher values, but keep in mind we need at least 5 samples per bit. If you want to sample just one channel, you can try lower sample rates – 100 ns.

**Channels**

Select Channel A if you want to sample CAN\_H or CAN\_L.  
Select both Channels if you want to sample both CAN\_H and CAN\_L simultaneously (= differential mode).

**Range**

5V is correct for the CAN bus.

**Baud Rate**

Select the CAN baud rate

**Input Type**

If you sample both CAN wires, select Differential. Otherwise select Low or High.  
Make sure that both Picoscope channels are enabled if you want to use the Differential Mode.



### **Voltage Threshold**

Threshold for logic values, the following are suggested values for all Input Types:

Differential Mode = 1,5 V

CAN\_High = 3 V

CAN\_Low = 2 V

### **First wait for EOF**

Keep it checked for Picoscope.

### **Timing parameters**

Fine tuning of timing parameters of the CAN Decoder. Does not need to be changed.

Decoding from CAN\_H or CAN\_L only

There could be scenarios in which you may wish to examine CAN\_H or CAN\_L only and decode only from this line, e.g.:

- CAN\_H or CAN\_L grounded so you may want to look at the other line
- Decoding from a single line may allow faster sampling rates with the PicoScope (depending on the drivers of the PicoScope model)

To configure the X-Analyser and PicoScope for decoding from CAN\_H only change:

- In the Picoscope channel configuration, **Enable** for **Channel A**, untick **Enable** for **Channel B**
- In the **CAN Decoder**, change **Input Type** from **Differential** to **High** and **Voltage Threshold** to 3 volts (because CAN\_H goes from 2.5V to 3.5V during the dominant bit)

To configure the X-Analyser and PicoScope for decoding from CAN\_L only change:

- In the Picoscope channel configuration, untick **Enable** for **Channel A**, **Enable** for **Channel B**
- In the **CAN Decoder**, change **Input Type** from **Differential** to **Low** and **Voltage Threshold** to 2 volts (because CAN\_L goes from 2.5V to 1.5V during the dominant bit)

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## Installation of X-Analyser

This section of the user guide will guide you through installing X-Analyser on your PC.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## System Requirements

[Installation of X-Analyser](#) > System Requirements

Processor: Intel Core 2 Duo 2.6 GHz

Memory (RAM): 2 GB

Hard drive capacity: ≥ 1.5 GB (depending on options used and required operating system components)

Screen resolution: 1280 x 1024

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## Installation of Software

[Installation of X-Analyser](#) > Installation of Software

Please ensure that you have installed .NET v4.6.2 Runtime framework before installation.  
It can be downloaded from the link below

<https://www.microsoft.com/net/download/all>

- Open up the email that you will have received and click on the download link for X-Analyser. Save the file onto your hard disk and then double click on it to start the installer.
  - A warning message may appear regarding the safety of the file, please click allow.
  - Automatic installation will commence. Follow the instructions on the screen as they appear.
- 
- If at a later date the removal of X-Analyser from the hard disk is required, Add/Remove programs should be used from Control Panel. Within Add/Remove Programs, X-Analyser should be selected before pressing the Add/Remove button. The instructions should then be followed to uninstall all X-Analyser components from the hard disk.
  - Any user files will not be removed during uninstallation or version upgrades.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Licensing

[Installation of X-Analyser](#) > Licensing

A valid license must be present for the X-Analyser to start. A 30-day trial license can be obtained from Warwick Control by contacting [sales@warwickcontrol.com](mailto:sales@warwickcontrol.com).

When X-Analyser is started for the first time it will prompt you to locate your license file. This will have been provided to you by your Warwick Control sales representative.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Driver Downloads

[Installation of X-Analyser](#) > Driver Downloads

Drivers for your interface can be found at the manufacturers website.

Kvaser	<a href="http://www.kvaser.com/en/downloads.html">http://www.kvaser.com/en/downloads.html</a>
Softing	<a href="http://industrial.softing.com/en/downloads.html">http://industrial.softing.com/en/downloads.html</a>

---

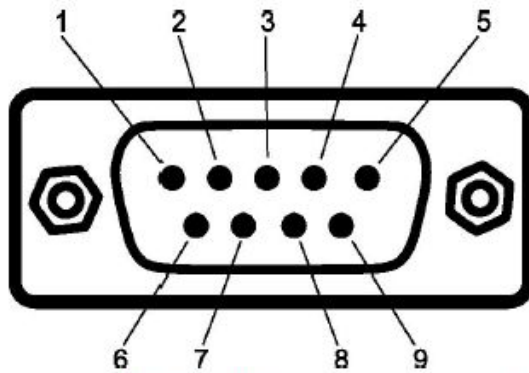
Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Interface Pin Out

[Installation of X-Analyser](#) > Interface Pin Out

Below are the CAN and LIN pin-outs of the D-SUB plug. These are shown in the table below the diagram.

The CAN channel has a 9-pin D-SUB plug.



The D-SUB connector pin numbers

D-SUB Pin number	Low Speed	High Speed	SWC	LIN
1	Not connected	Not connected	Not connected	Not connected
2	CAN_L	CAN_L	Not connected	Not connected
3	GND	GND	GND	GND
4	Not connected	Not connected	Pull-down resistor; see chapter 3.6.1.	Not connected
5	Shield	Shield	Shield	Shield
6	Not connected	Not connected	Not connected	Not connected
7	CAN_H	CAN_H	CAN_H	LIN_BUS
8	Not connected	Not connected	Not connected	Not connected
9	Reference power	Not connected	Reference power	Reference power

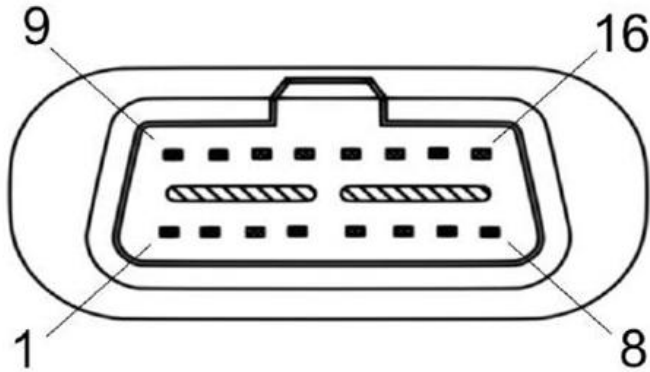


Figure 10: OBDII Connector pin Numbers.

OBDII Pin number	
4	Shield
5	GND
6	CAN_H
14	CAN_L
16	Reference power (not used)

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Main Features

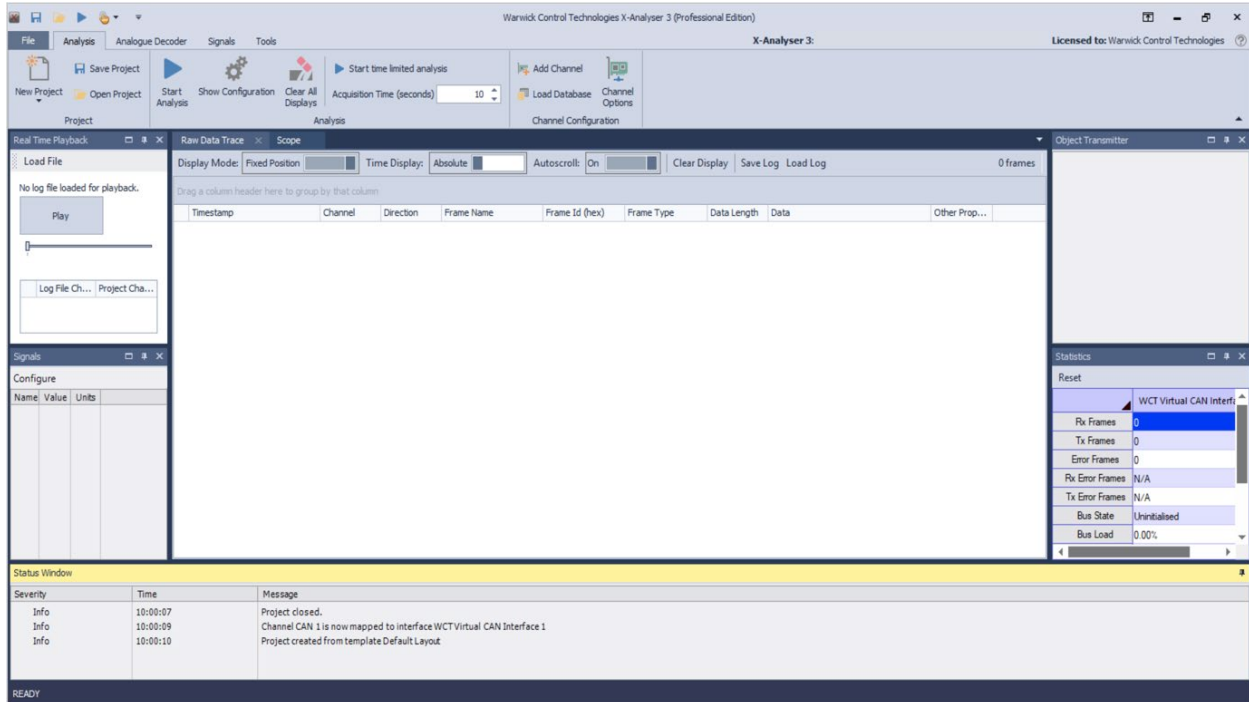
X-Analyser User Manual

### Signals

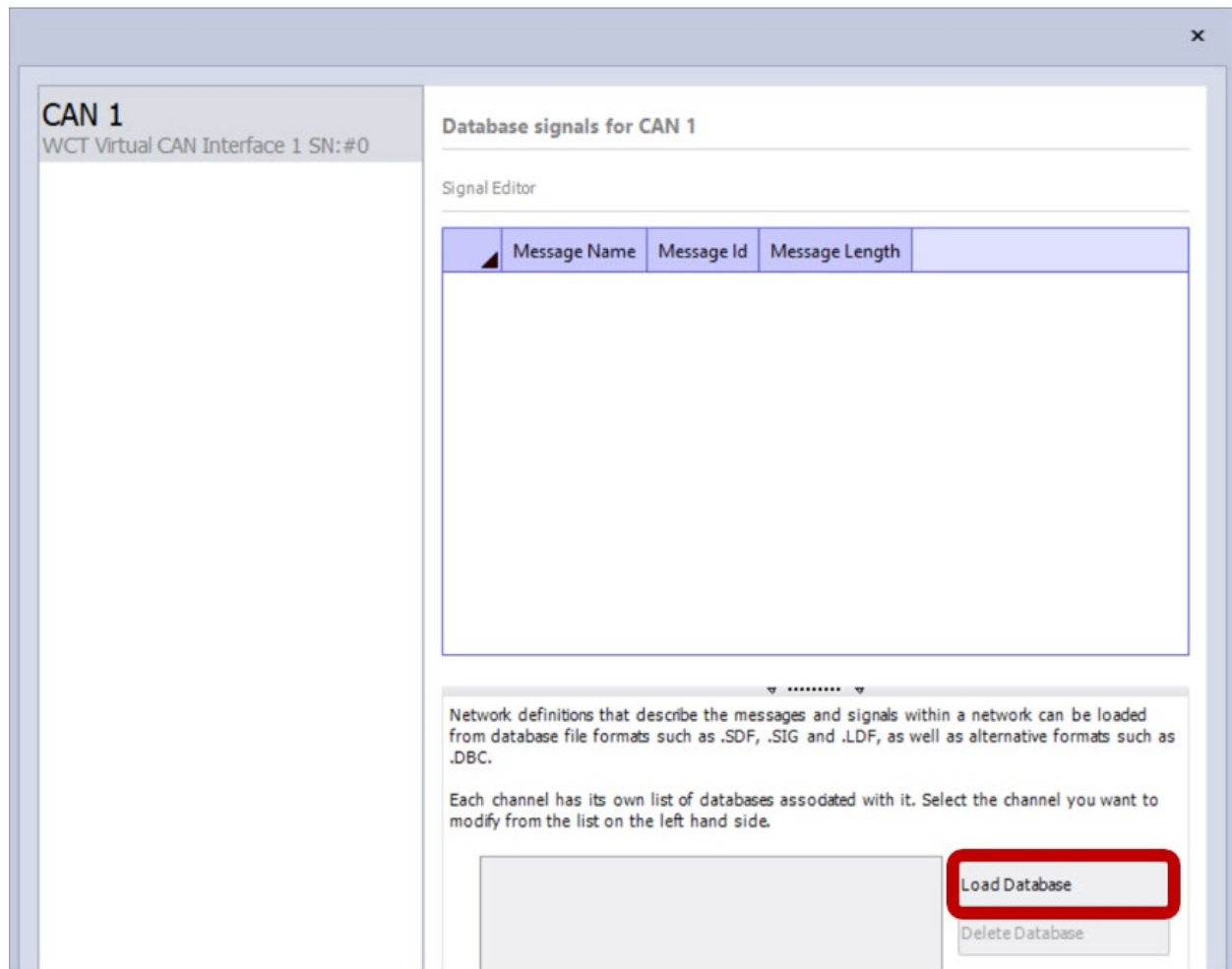
Main Features > Signals

This section will show you how to set up **Signals** on XA3.

Please start from this screen:



Click on the **Load Database** button at the top of the screen.



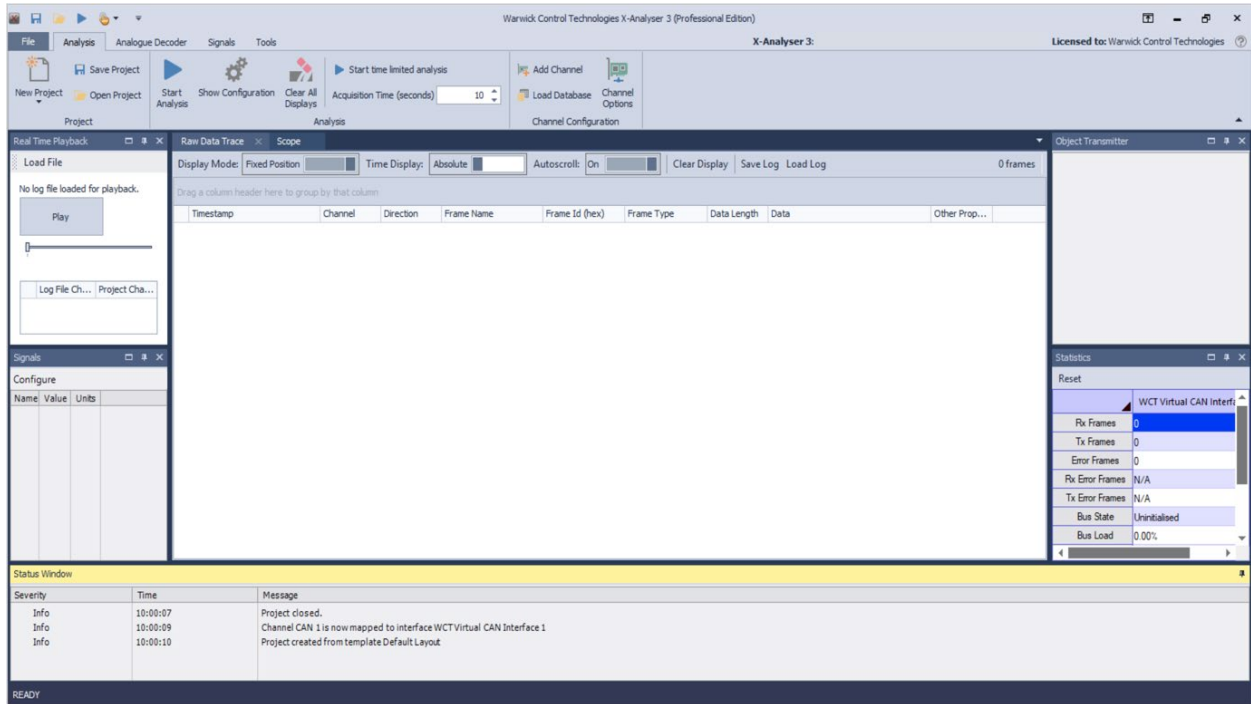
You can then input your signal data or load it from a database. Then click close and it will be loaded into your project.

## Transmitters

Main Features > Transmitters

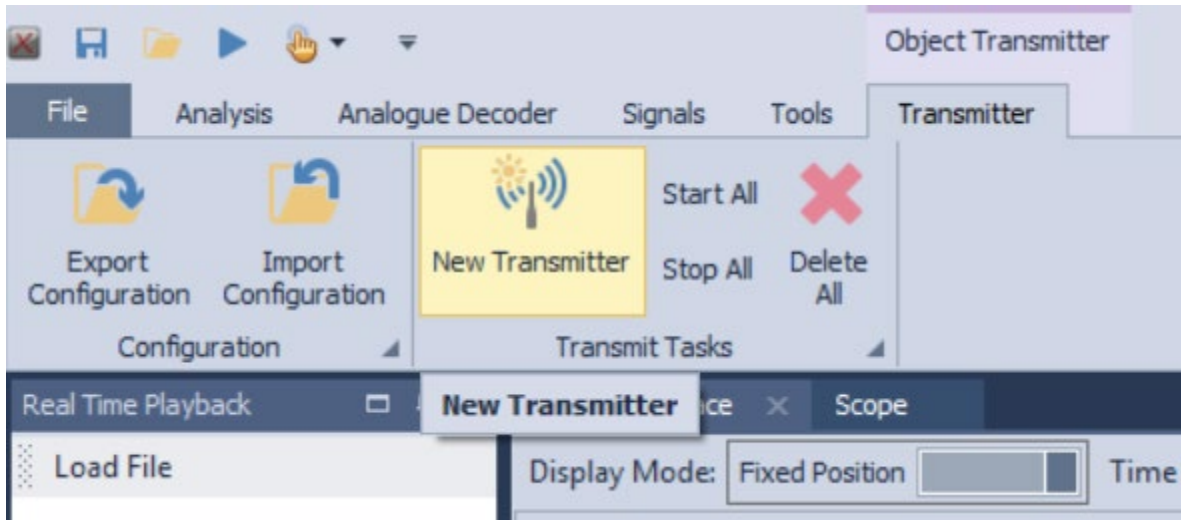


This section will show you how to set up Transmitters on X-Analyser.  
Please start from this screen:



Click on the add new transmitter button

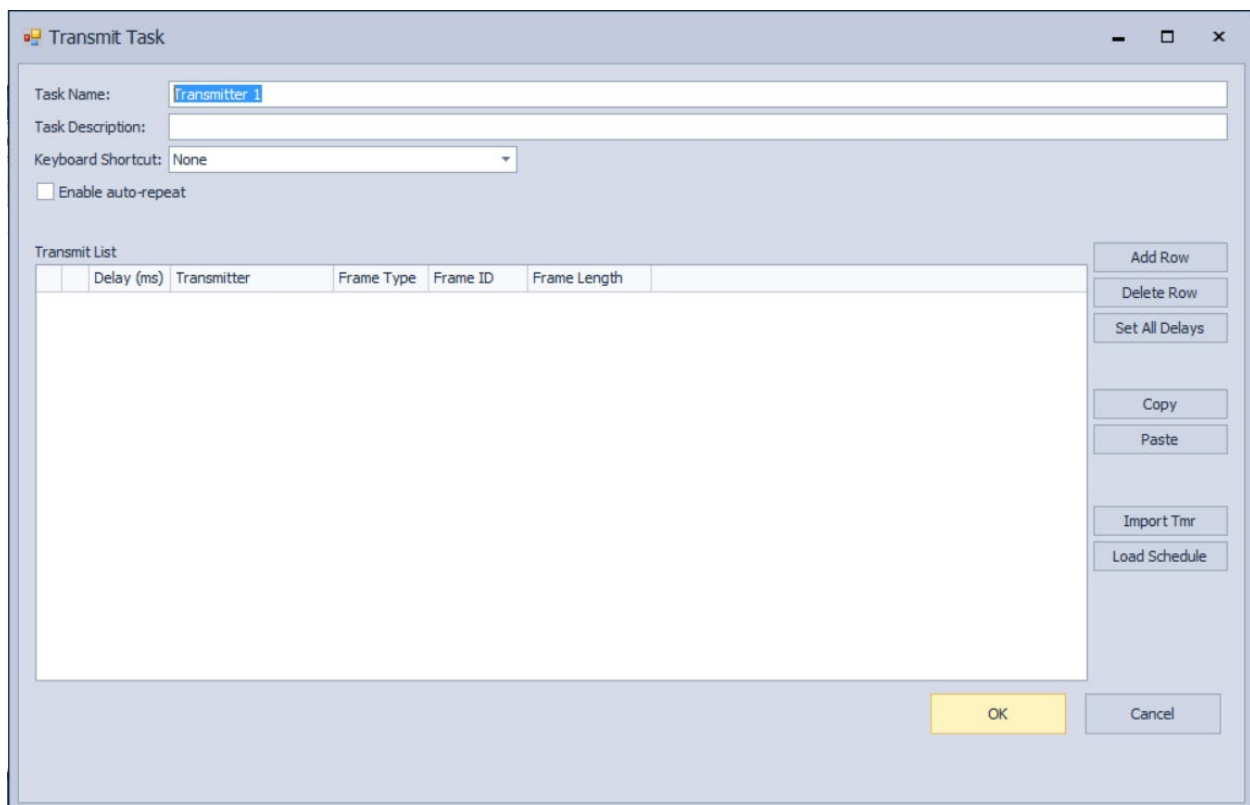
- Transmitter taskbar



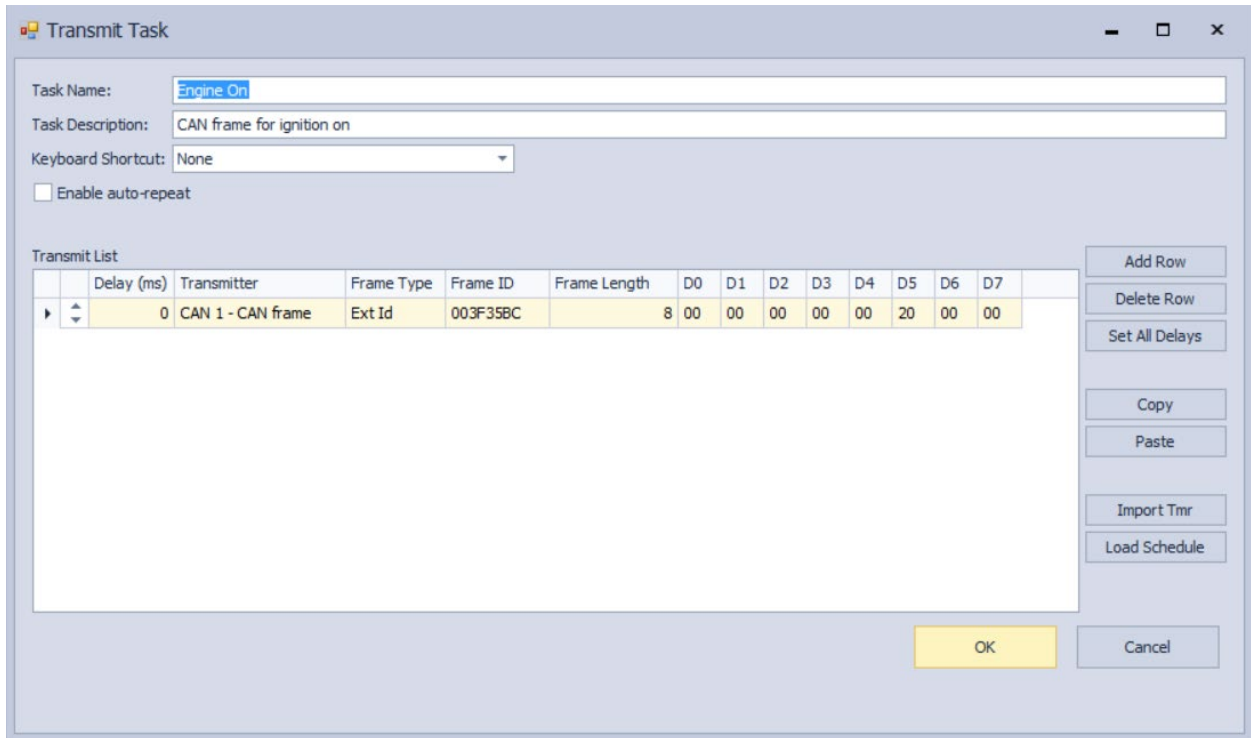
- Right clicking in the Object Transmitter window.



This screen should appear:

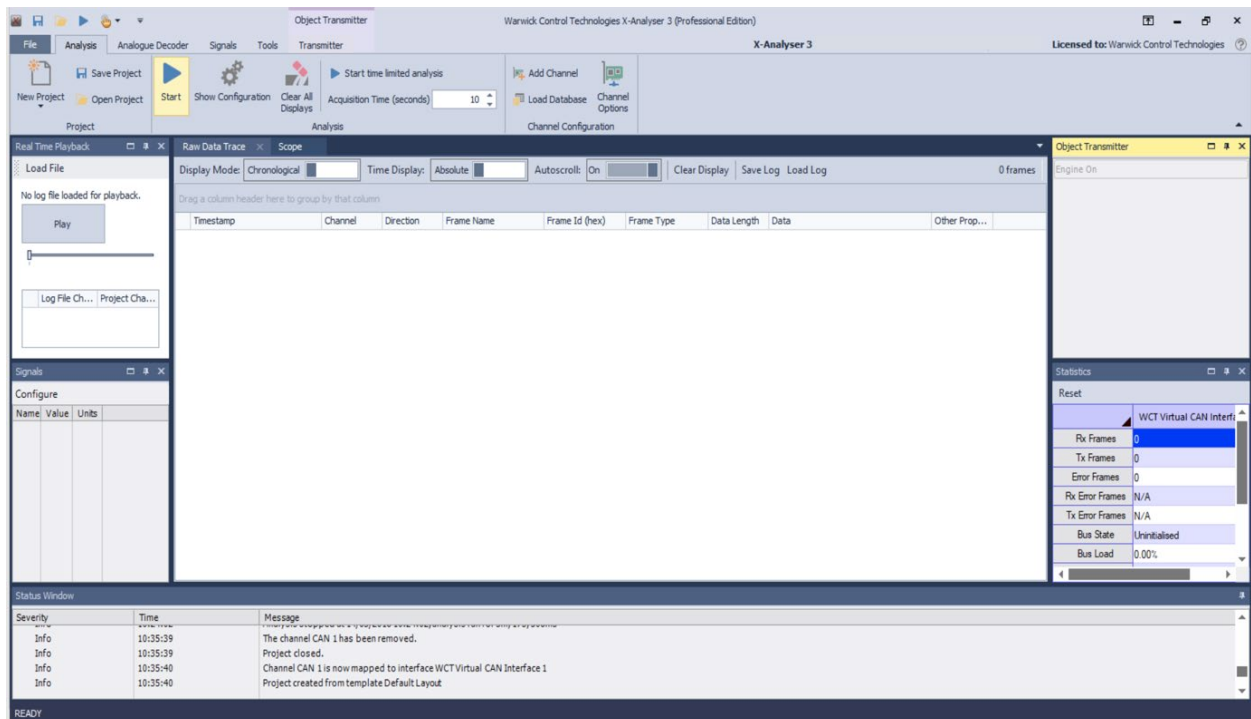


Change the task name and insert your data by clicking **Add Row** and selecting the frame options i.e. **Frame Type**, then click **OK**.



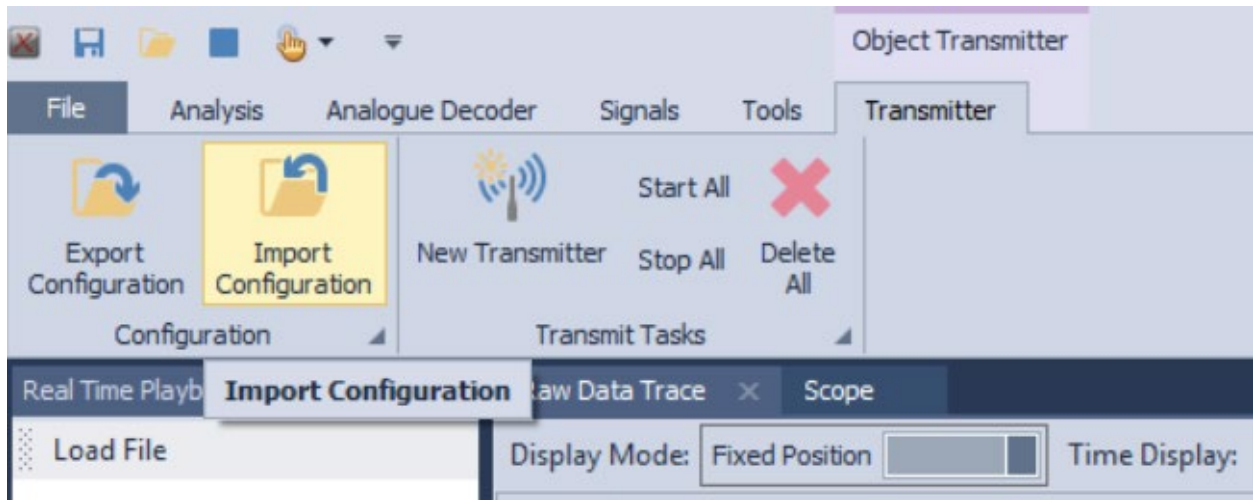
You can also import a transmitter by clicking **Import Configuration** in the object transmitter taskbar. You will then be prompted to select the file.

It will then appear in the Object Transmitters list (top right), pictured below:



This button can then be pressed to transmit the signal at any point while running X-Analyzer

## Object Transmitter Taskbar



- **Export Configuration** - Exports the transmitters created in the Object Transmitter window to a file in a .xatx format.
- **Import Configuration** - Imports .xatx, .otr, and .tmr files into the Object Transmitter window.
- **New Transmitter** - Creates a new transmitter.
- **Start All** - Starts all transmitters at the same time.
- **Stop All** - Stops all transmitters at the same time.
- **Delete All** - Deletes all transmitters.

## LIN Object Transmitter

For LIN object Transmitters only one schedule can be loaded into one transmitter. Then only one transmitter can be sent at any one time. For example;

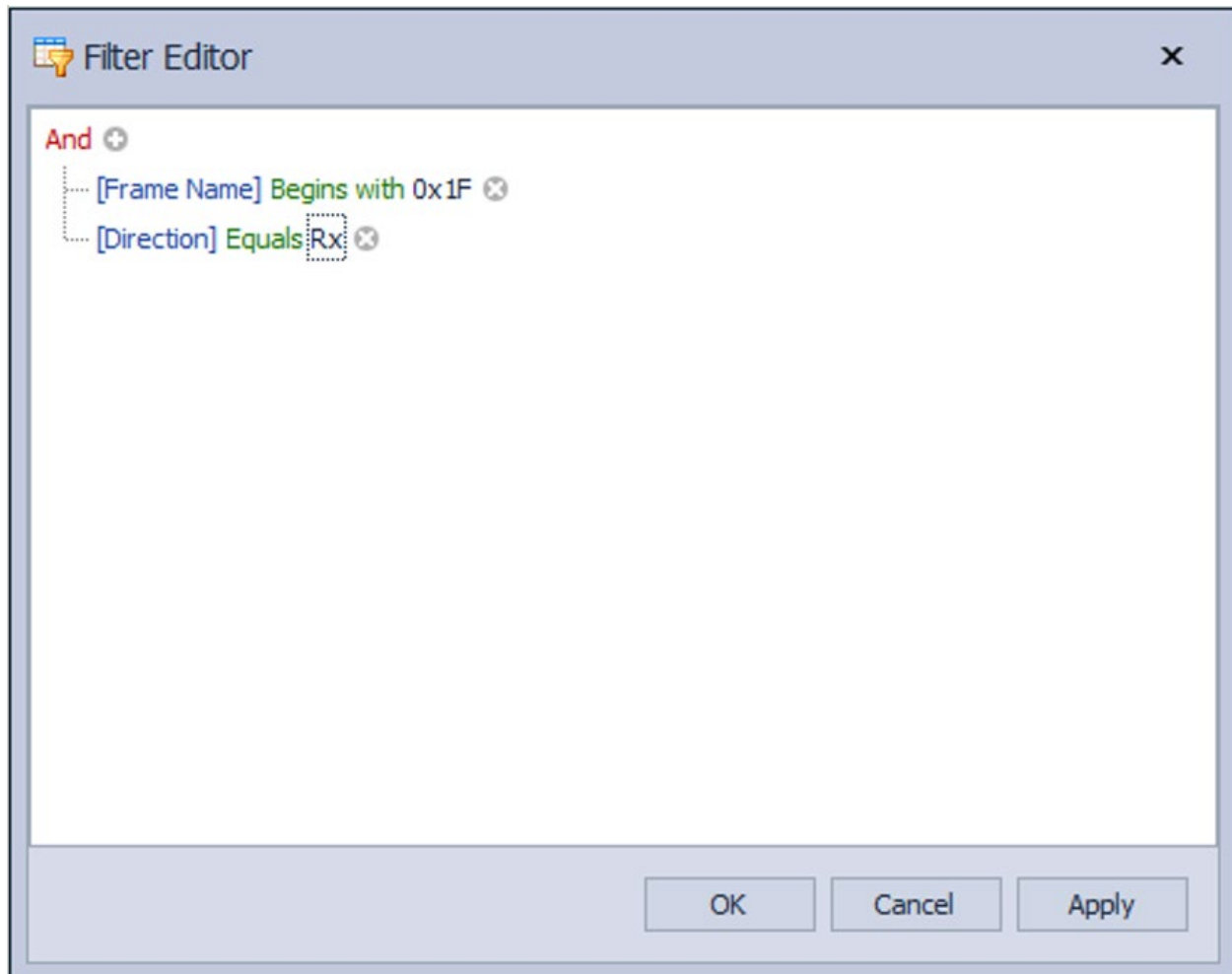


Each transmitter contains a LIN schedule so either **Schedule 1** can be transmitted or **Schedule 2** can be transmitted. They cannot be transmitted at the same time.

## Display Filters

Main Features > Display Filters

This section describes the capabilities provided by the **Filter Editor**, which allows you to visually build filters:



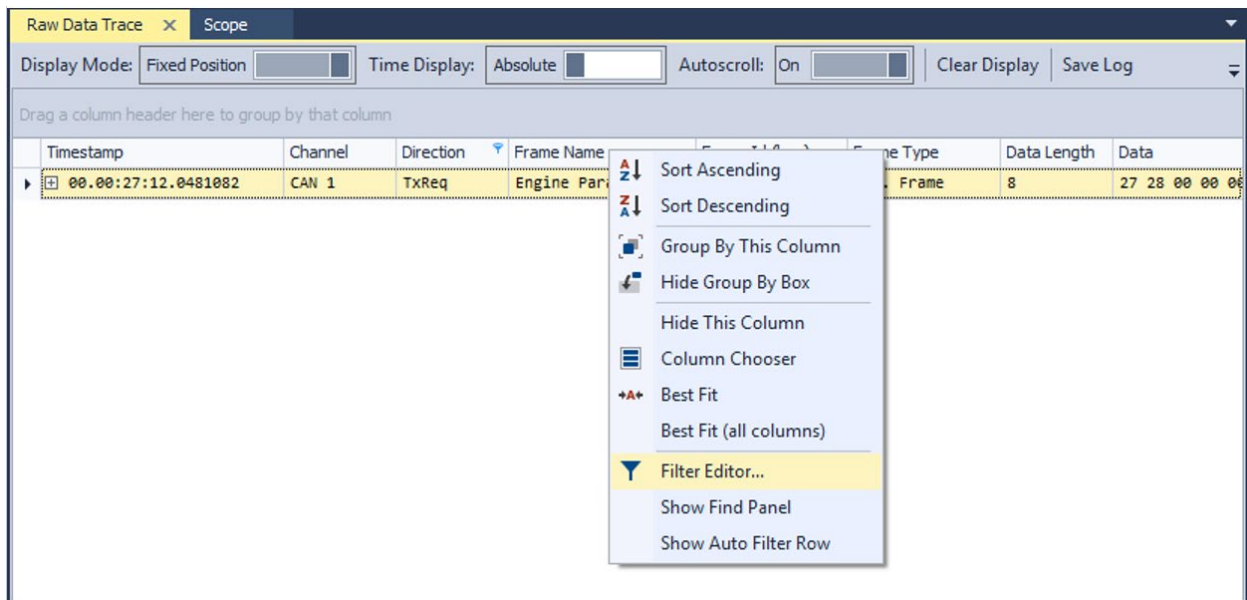
### How to Construct a Simple Filter Condition

Basically, filter conditions specify what data to select from a data source and display in a data-bound control. A typical simple filter condition consists of three parts: the column/field name, operator and a value(s). For instance, '[Frame Id] = 0x1FAA00F0' is a simple filter condition, where '[Frame Id]' is a field name, '=' is an operator and '0x1FAA00F0' is a value.

This condition when applied to a data-aware control will display records that have values in the Frame Id column equal to 0x1FAA00F0. Here is how to create this condition via the Filter Editor.

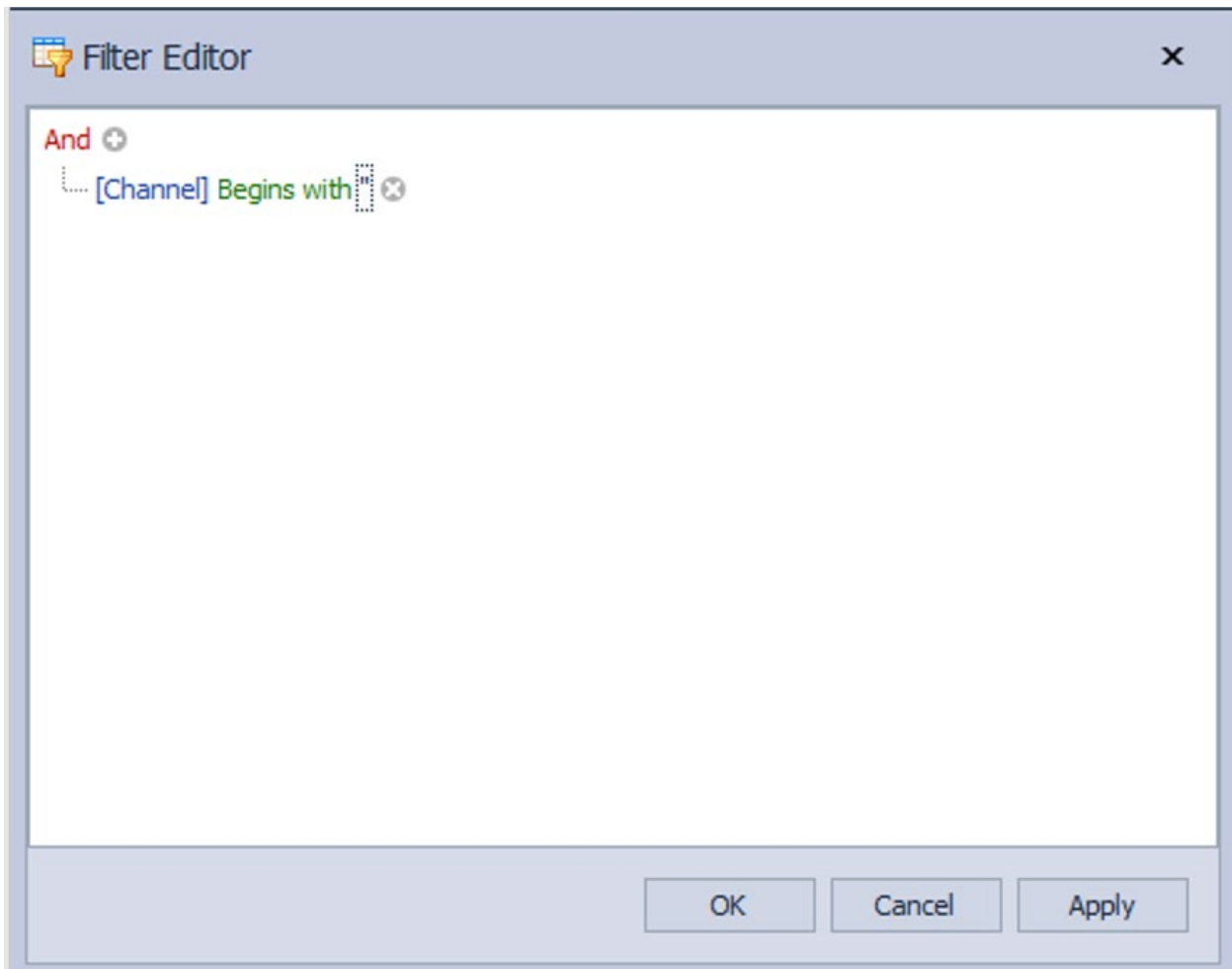
Invoke the Filter Editor.

To invoke the **Filter Editor** in a grid control, right-click any grid column's header and select the **Filter Editor** option.



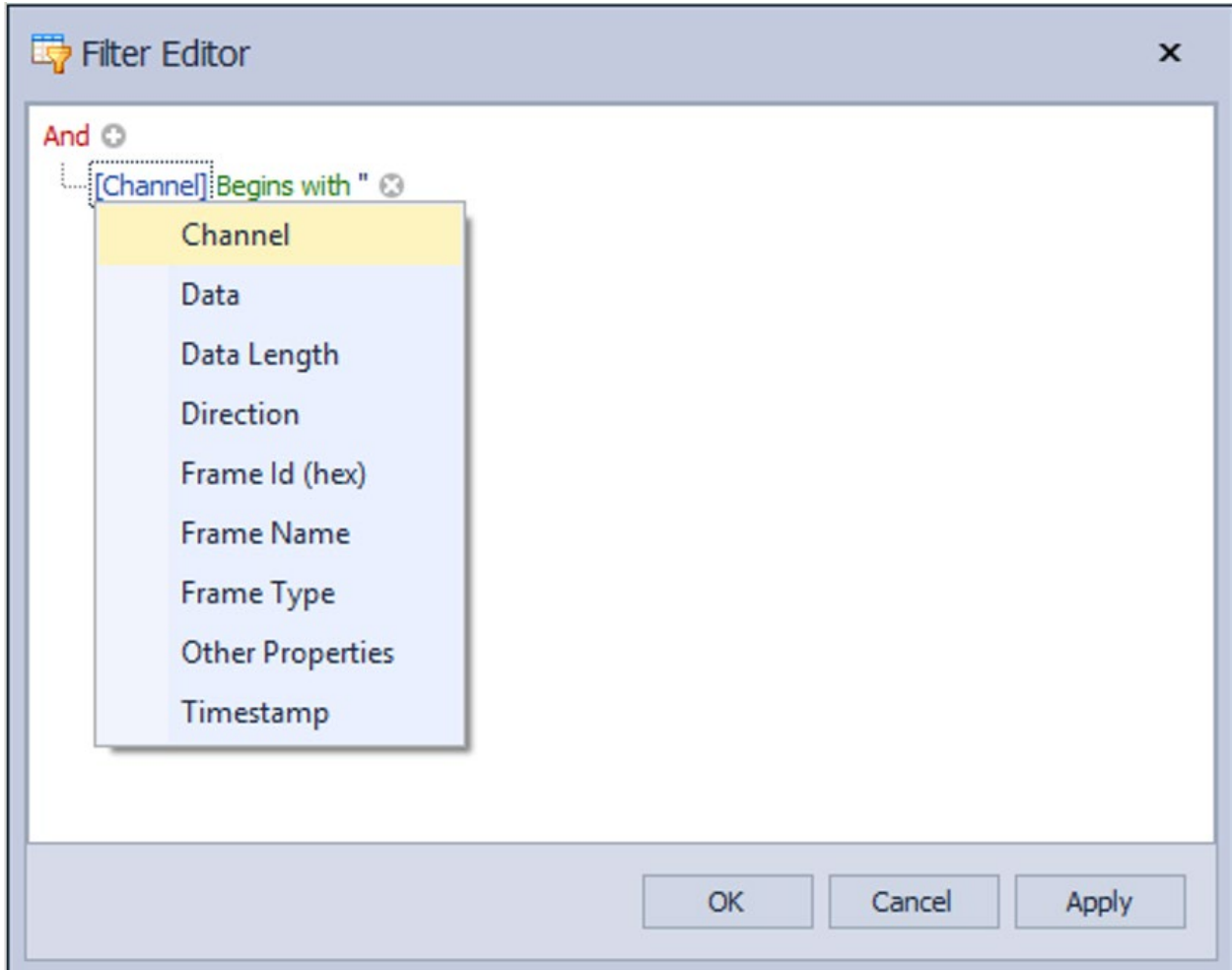
1. The **Filter Editor** will pop up.

When invoking the **Filter Editor** for a grid control, if no filtering has yet been applied, the **Filter Editor** will contain a new filter condition referring to the clicked column. If, for example, the **Filter Editor** has been opened by right-clicking the Channel column, it will look like the image below:

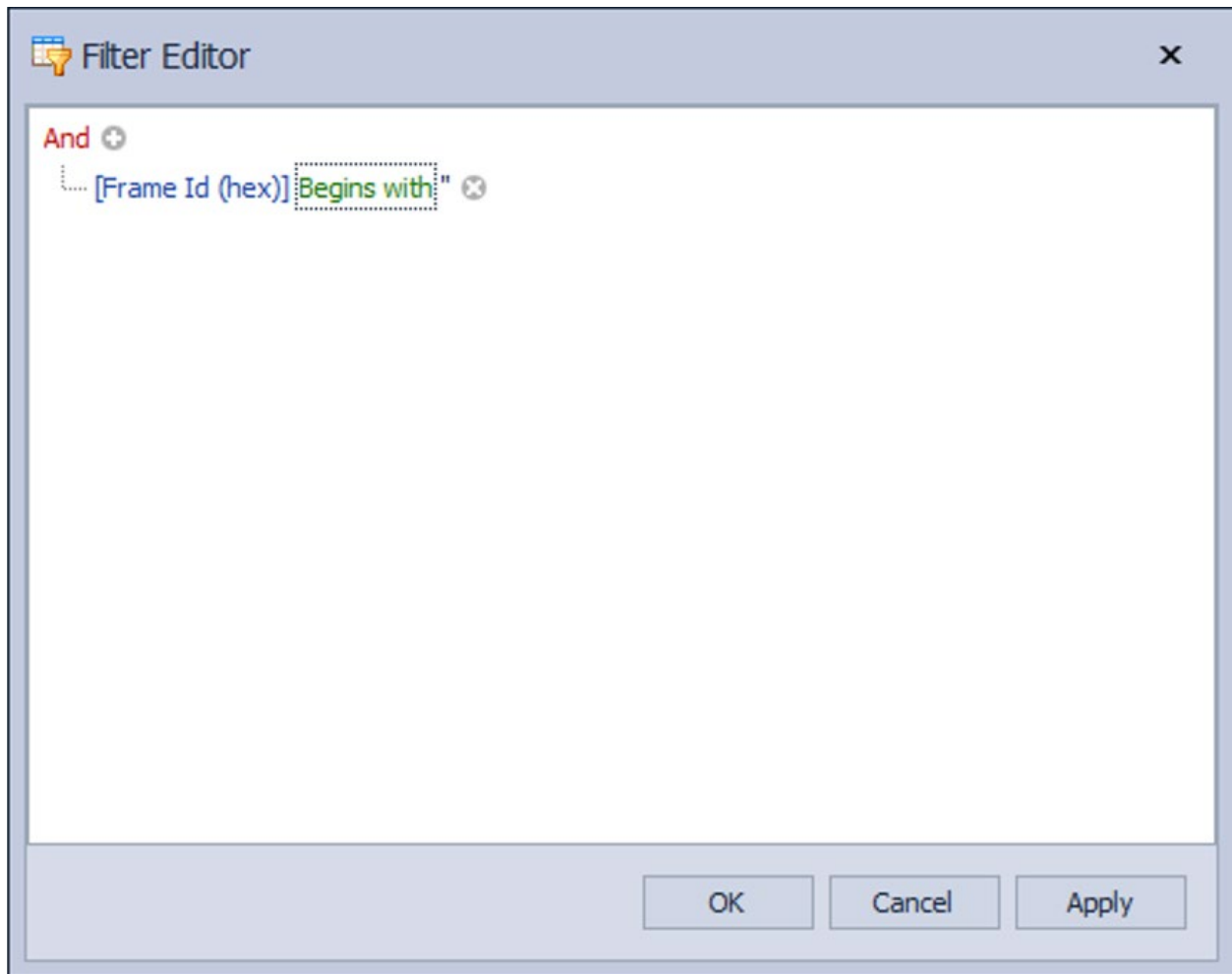


2. Select a column.

Now, to filter against the **Frame Id**, click the condition's link displaying a column name ('Channel'). This will display the list of available columns. Select the **Frame Id** column in this list:

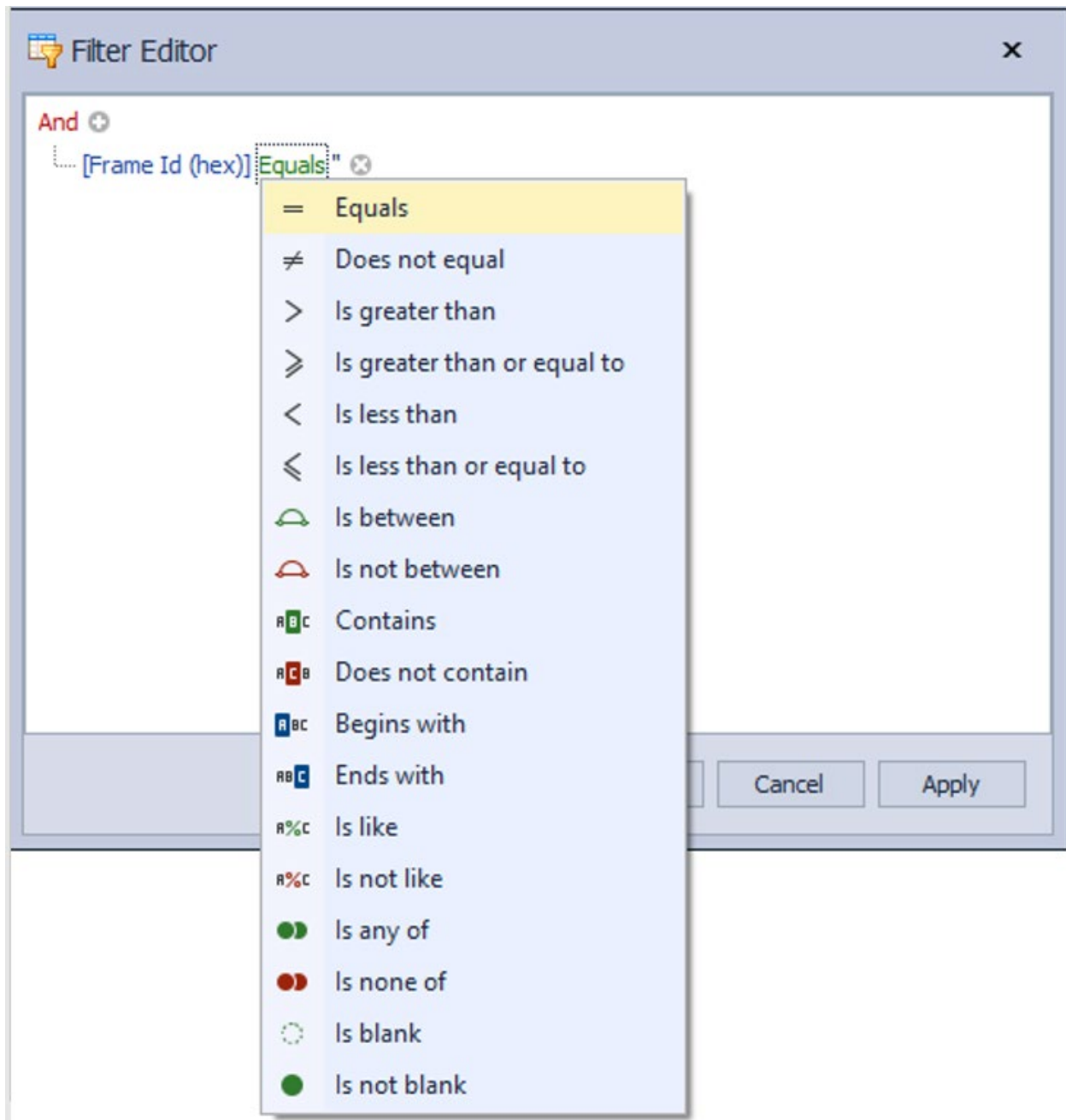






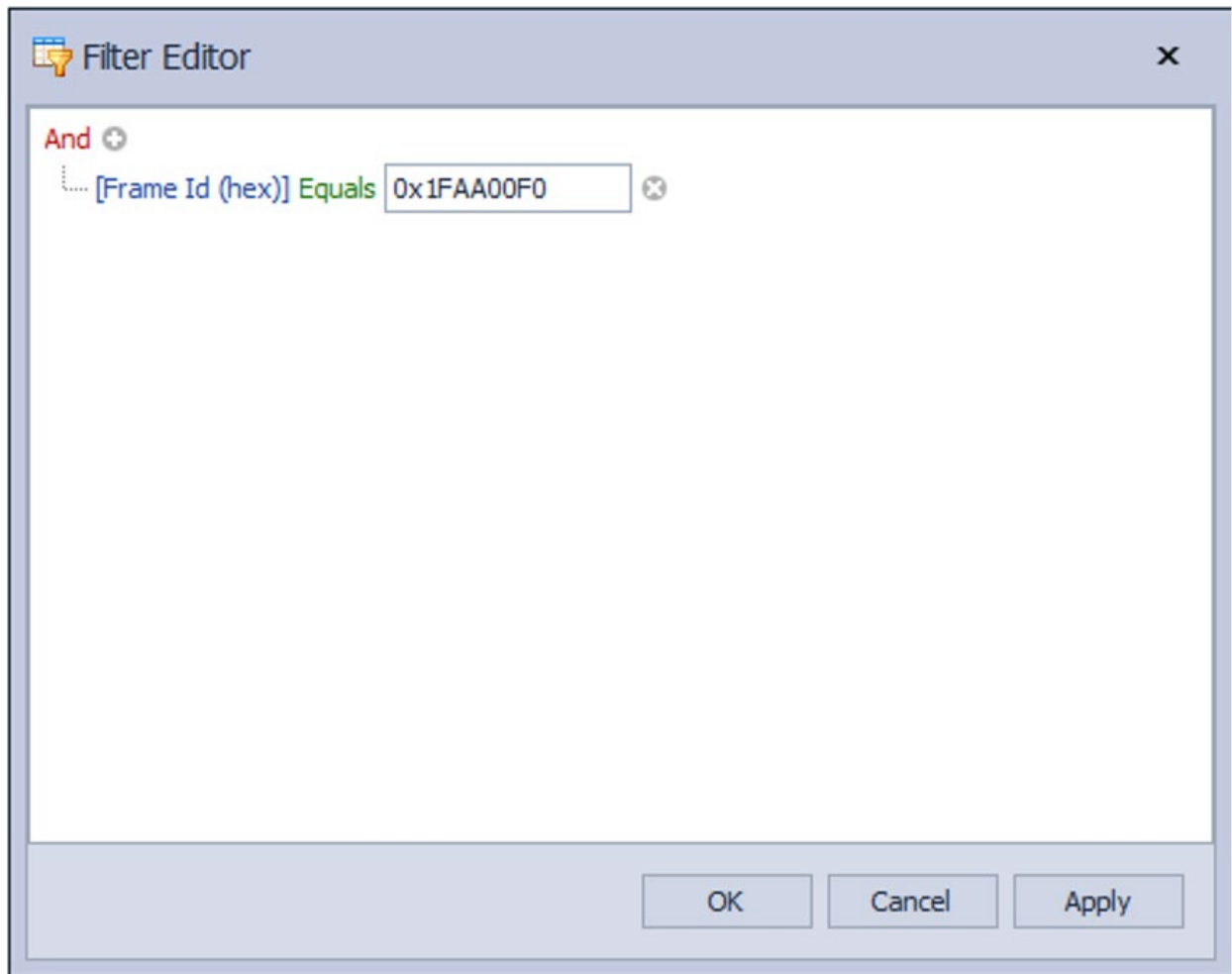
3. Select a comparison operator.

To select the '=' comparison operator, click the condition's operator link ('Begins with') to display the list of supported comparison operators and select the required operator:



4. Enter a value.

Now, click the value box and enter a comparison value ('0x1FAA00F0');



5. Save changes.

Click **OK** or **Apply**, to filter data using the created filter condition. The grid will show the filter panel displaying the current filter criteria:

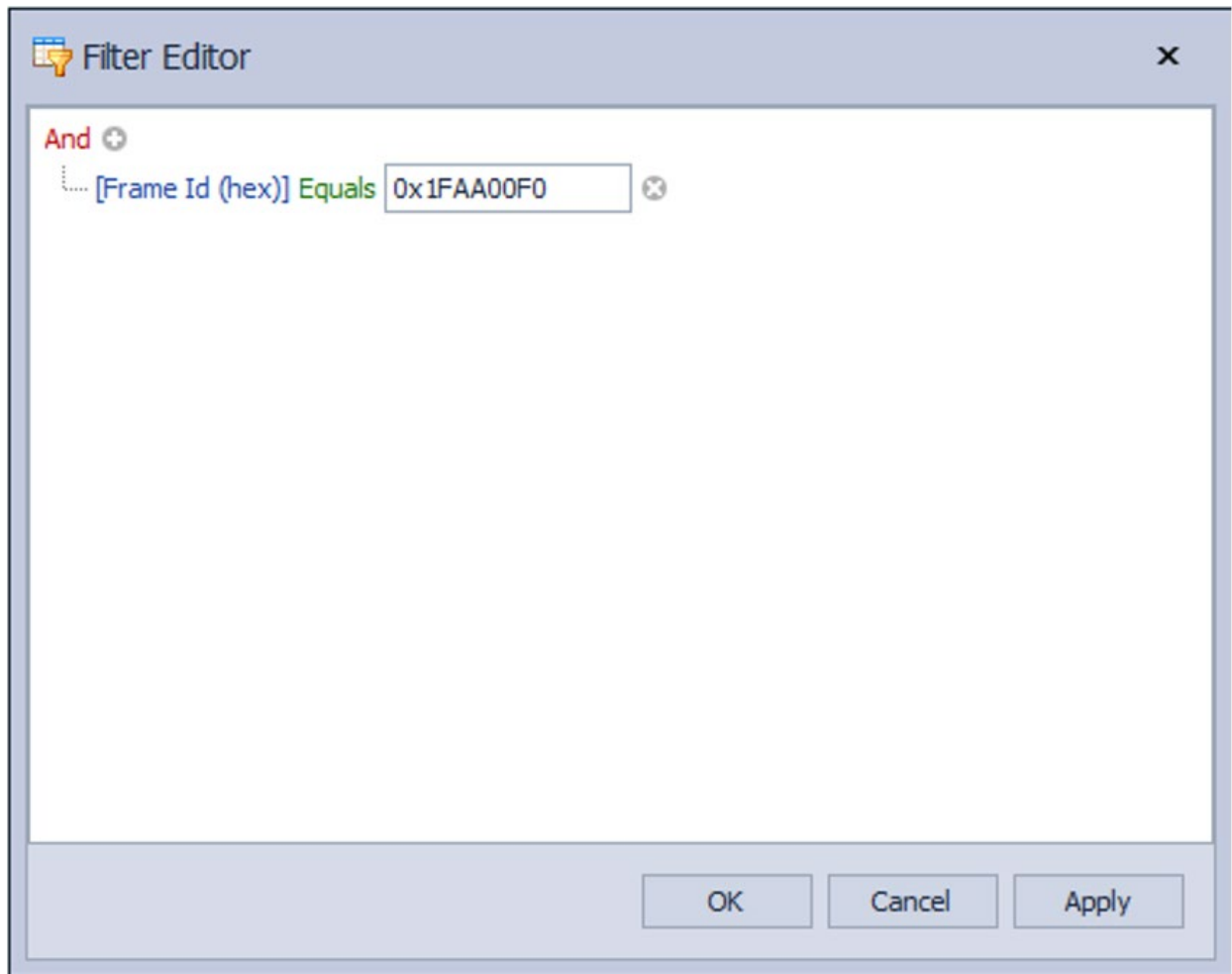
Timestamp	Channel	Direction	Frame Name	Frame Id (hex)	Frame Type	Data Length	Data	Other Prop...
00.00:37:00.3945523	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:00.4951007	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:00.5955567	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:00.6960932	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:00.7967688	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:00.8973205	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:00.9980598	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.0982399	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.1983942	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.2989163	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.3995924	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.5001182	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.6051602	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.7049623	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.8050205	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:01.9060044	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	
00.00:37:02.0065935	CAN 1	TxReq		0x1FAA00F0	Ext. Frame	8	00 00 00 00 01 86 86 86	

The filter panel will contain the **Edit Filter** button, which also allows you to invoke the Filter Editor.

### How to Construct Filter Criteria with Multiple Conditions Joined by One Logical Operator

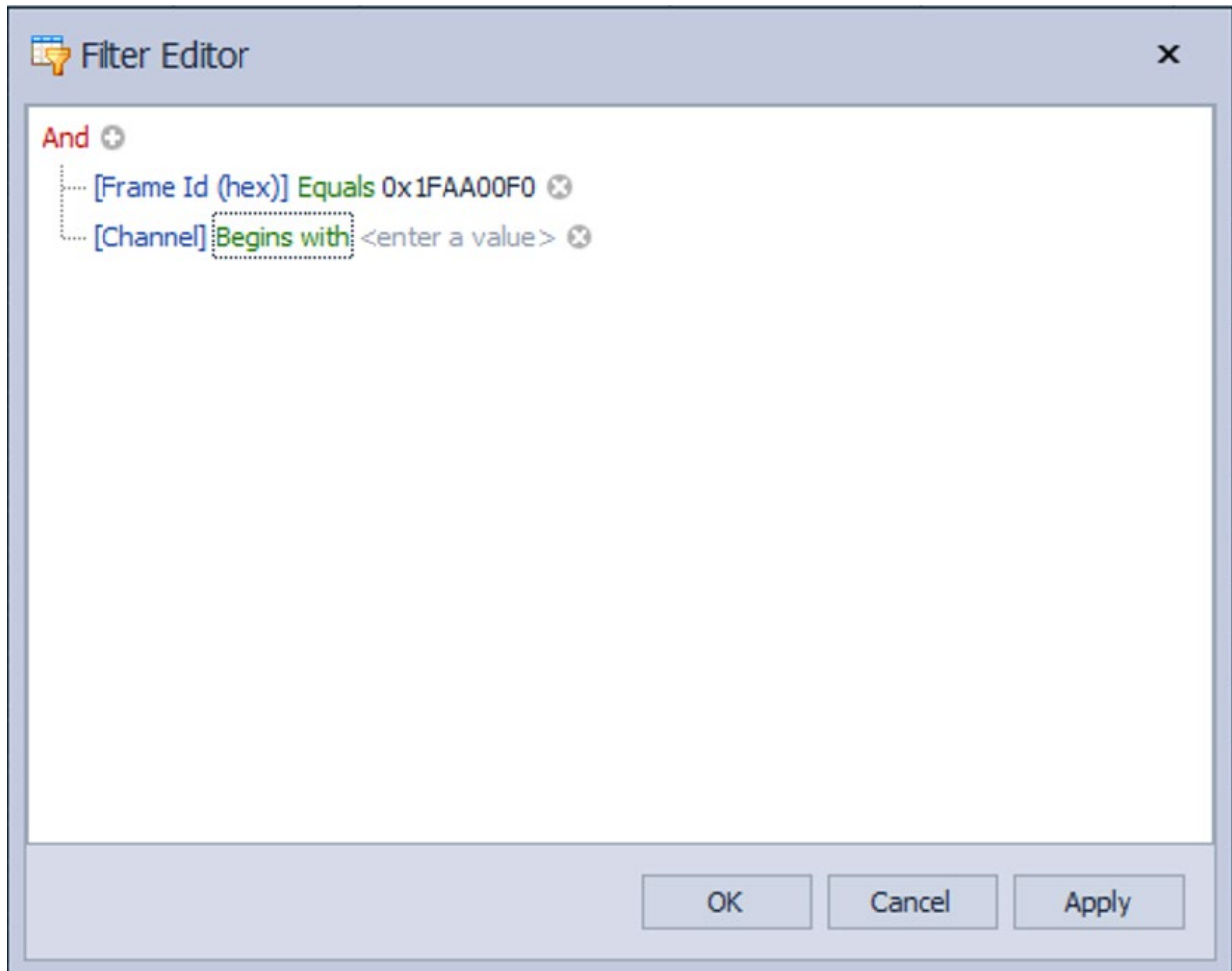
Filter criteria typically consist of two or more simple filter conditions combined by logical operators (AND, OR, NOT AND, NOR OR). The following example shows how to construct filter criteria in the Filter Editor that consist of multiple conditions combined by one logical operator. The "[**Channel**] = 'CAN 1' AND [**Frame Id**] = 0x1FAA00F0 AND [**Direction**] = Rx" filter expression contains three simple filter conditions combined by the **AND** operator. To construct it, do the following:

1. Invoke the **Filter Editor**. When the Filter Editor is invoked for a grid control, the Filter Editor may display an unfinished new filter condition.
2. Set the condition's operator to **Equals** and operand value to '0x1FAA00F0' (as described in the previous section):

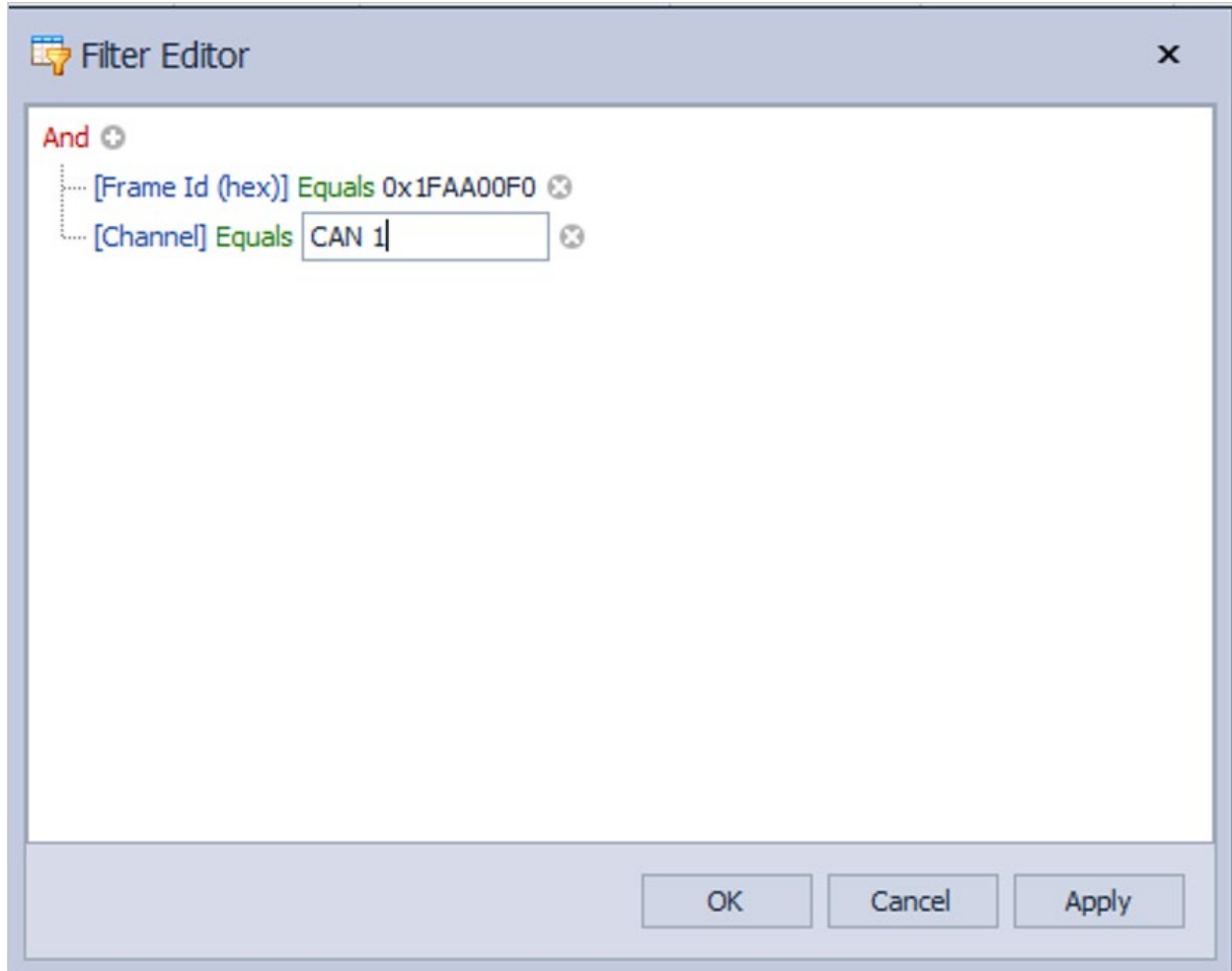


3. To add one more condition, press the + button next to the group's **AND** operator.

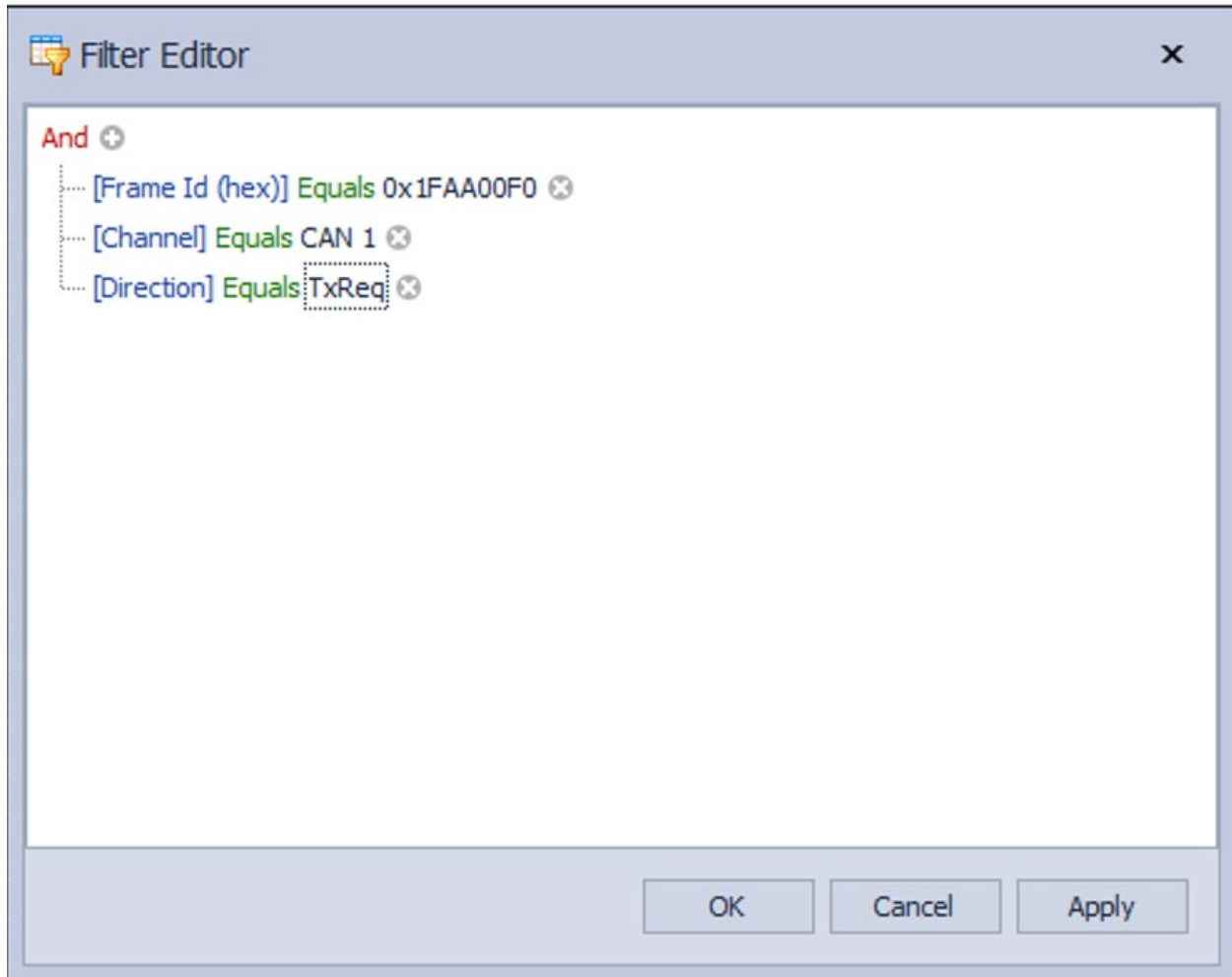
This will create a new condition under the current one:



4. For the second condition, set the column to **Channel**, operator to '=' and operand value to 'CAN 1':



5. To add a third condition to the same group, click the + button again. Set the condition's column to **Direction**, operator to '=' and operand value to **TxReq**. Below is the result:



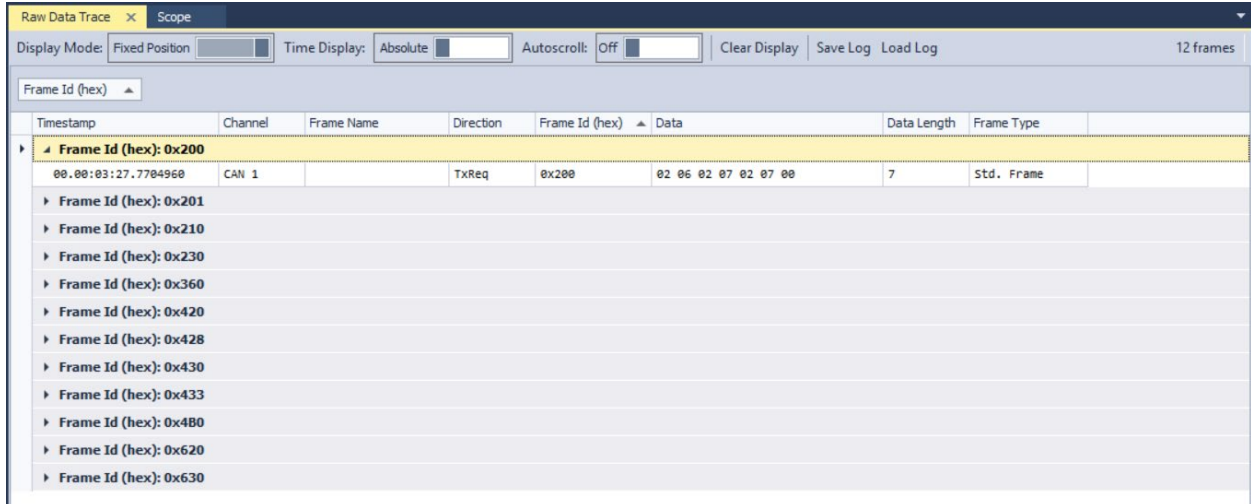
Click **OK** or **Apply**, to apply the created filter criteria.

## Column Options

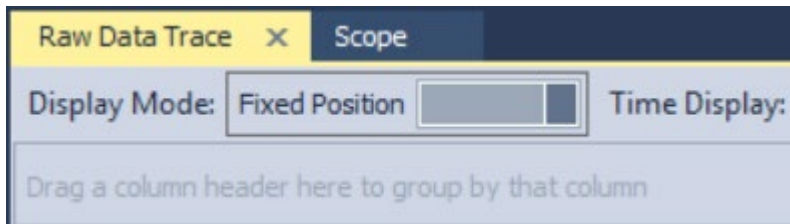
There are also other options for the column displays on the Raw Trace these are described below;

- Sort Ascending - Sorts the column from A to Z or 1 to 10+.
- Sort Descending - Sorts the column from Z to A or +10 to 1.
- Clear sorting - Clears the sorting just performed.
- Group By This Column - Will group the data within the Raw trace as per the selected column i.e grouping Frame Id would look like this.





- Hide Group By Box - Hides the box where column titles can be dragged to be grouped by, shown below.



- Hide This Column - Hides the selected column.
- Column Chooser - Customizes columns.
- Best Fit - Gives the column best fit.
- Best Fit (all columns) - Gives all columns the best fit.
- Filter Editor - Described above.
- Show Find Panel - Gives a search bar to find entries in the raw trace.
- Show Auto Filter Row - Gives a filter option for each column.

They allow you to use X-Analyser when you don't have a physical interface to hand.

The virtual interface mimics a CAN-bus inside your PC which allows you to use real time playback

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Real Time Playback

Main Features > Real Time Playback

Click 'Load File' on the Real Time Playback panel.

Select the file that you would like to replay.

Click play and the data should be playing.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## ISO15765 Support

Main Features > ISO15765 Support

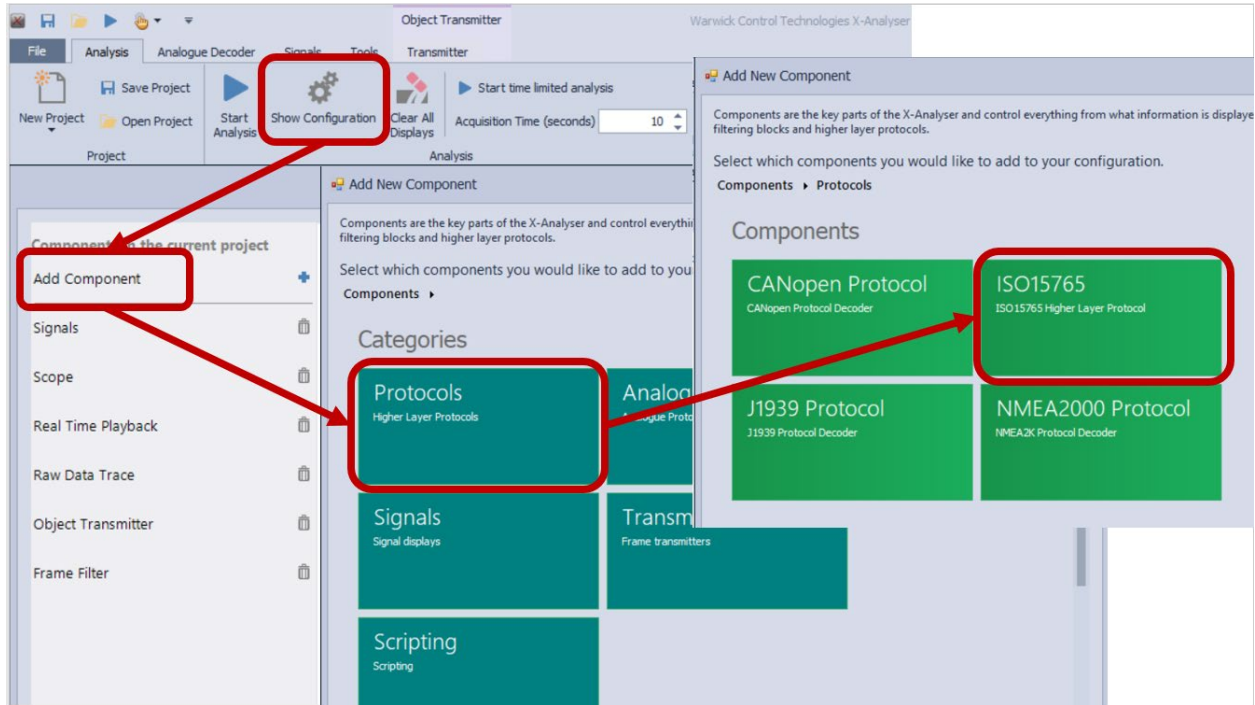
X-Analyser supports the ISO15765 transport protocol over CAN that is used in the automotive diagnostics such as Unified Diagnostics Services (UDS). The ISO15765 transport protocol primarily provides a way of combining messages that cannot be carried in a single CAN frame. With the X-Analyser ISO15765 support you can carry out two things:

1. Setup an ISO15765 interpreter for an ECU that supports ISO15765 and view the UDS messages that are built up from numerous CAN frames.
2. Use diagnostic communications with ECUs that supports ISO15765 and setup Object Transmitters for UDS messages that can be transmitted to the ECU. With this you can carry out actions such as reading part numbers, run diagnostic routines and read/clear Diagnostic Trouble Codes (DTCs).

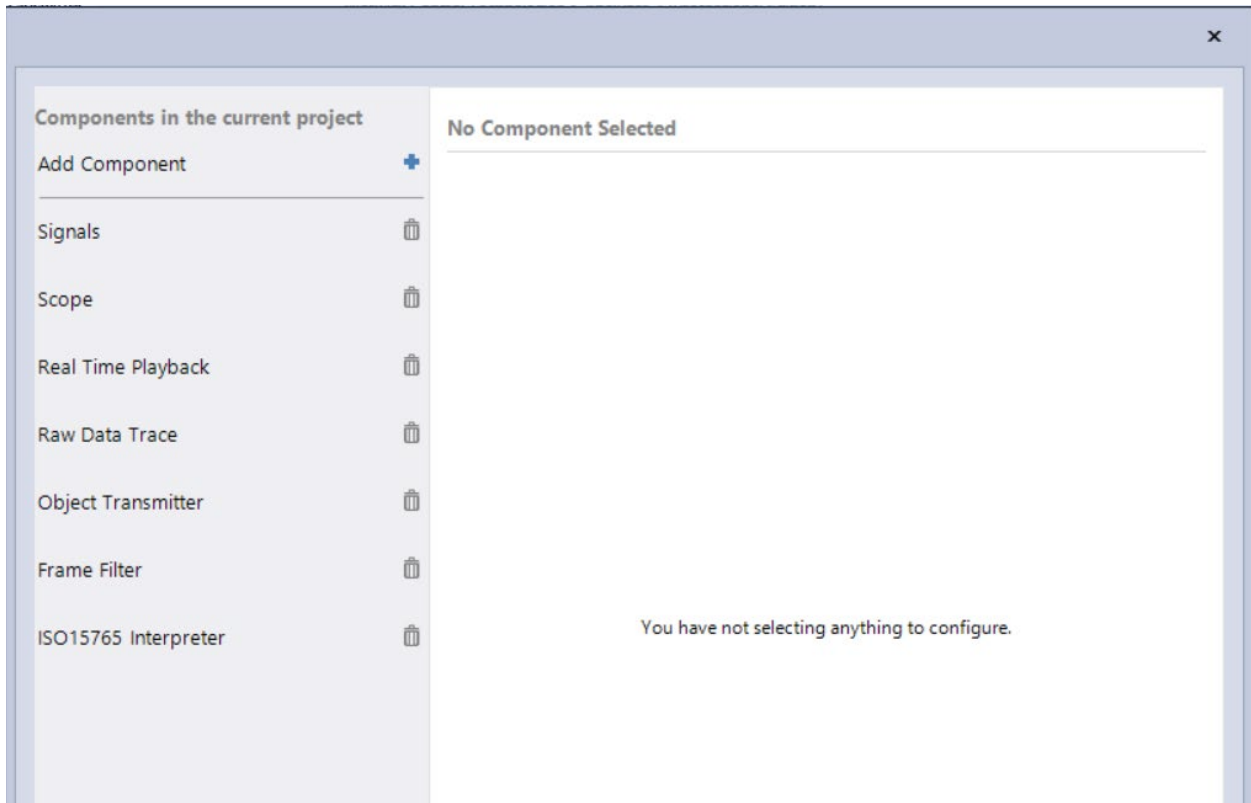
This feature is primarily aimed at ECU developers who are comfortable with viewing ISO15765 /UDS messages in hexadecimal format. The particular advantage of this feature is that an Object Transmitter can be quickly set up to get ECU diagnostic data via ISO15765 when is most tools a full diagnostic database must first be setup and imported.

## Adding ISO15765 Component for an ECU

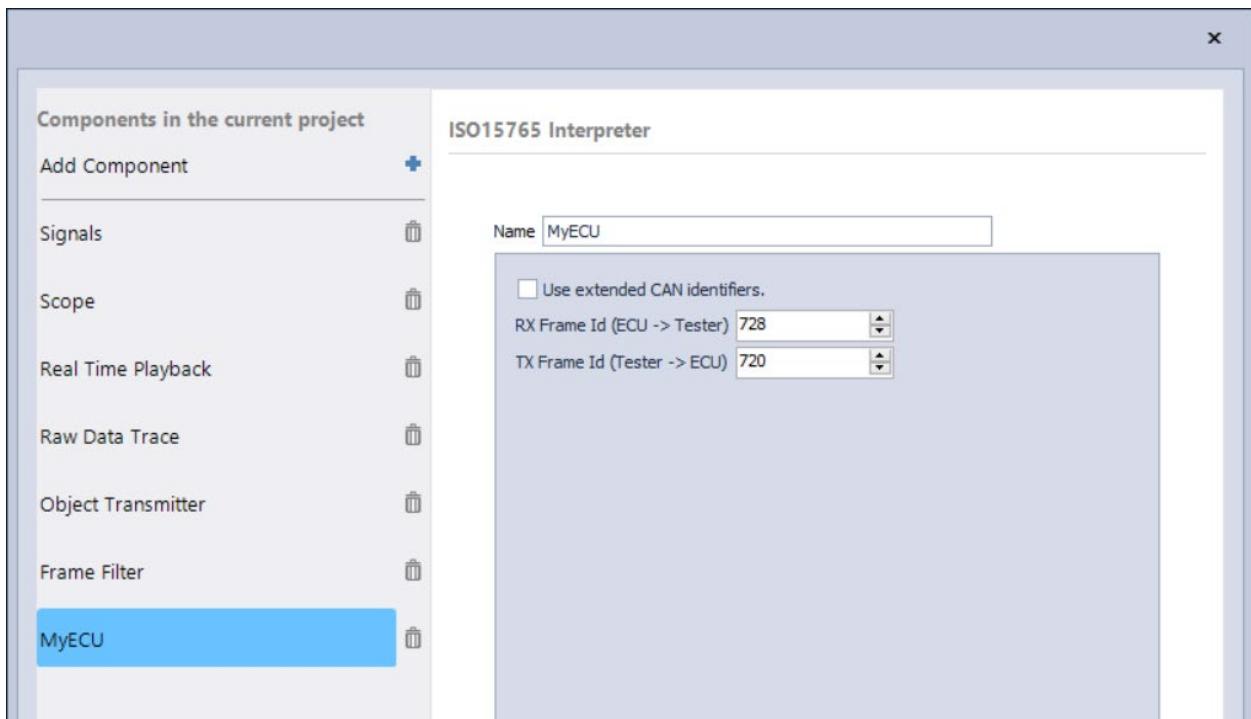
An ISO15765 component must be set up for each that you wish to talk to. To set one up you need to go to **Show Configuration - Add Component -> Protocols -> ISO15765**



You then need to set up the CAN identifiers used by the ECU for diagnostics by clicking on the newly created **ISO15765 Interpreter**.



You can then rename the ISO15765 Interpreter (e.g. MyECU) and set up the CAN IDs that are used for diagnostic communication (e.g. Physical CAN IDs 0x728 and 0x720 in this example).



## Setting Up an ISO15765 Object Transmitter

Go to the Object Transmitter panel and select **Add New Transmitter**.

Then enter the data that you want. At a bare minimum you need to select:

- **Transmitter** – set this to **MyECU – ISO15765 frame**
- **Frame Id** – do not change this
- **Frame Length** – this is the diagnostic message length which can therefore be greater than 8 bytes and NOT the same as Data Length Code (DLC)
- In the Data **D0, D1 etc.** you can put in your UDS diagnostic request data that you require. In the example shown below Service Identifier 0x22 is used (which is ReadDataByIdentifier) and Parameter Identifier (PID) 0xF18C (which is a popular one for reading ECU serial number in the automotive industry).

Transmit Task

Task Name:

Task Description:

Keyboard Shortcut:

Enable auto-repeat

Transmit List

	Delay (ms)	Transmitter	Frame Type	Frame ID	Frame Length	D0	D1	D2
▶	0	MyECU - ISO15765 frame	ISO15765 Frame	0	3	22	F1	8C

Buttons: Add Row, Delete Row, Set All Delays, Copy, Paste, Import Trm, Load Schedule, OK, Cancel

Once this is entered press **OK** to save the Object Transmitter.

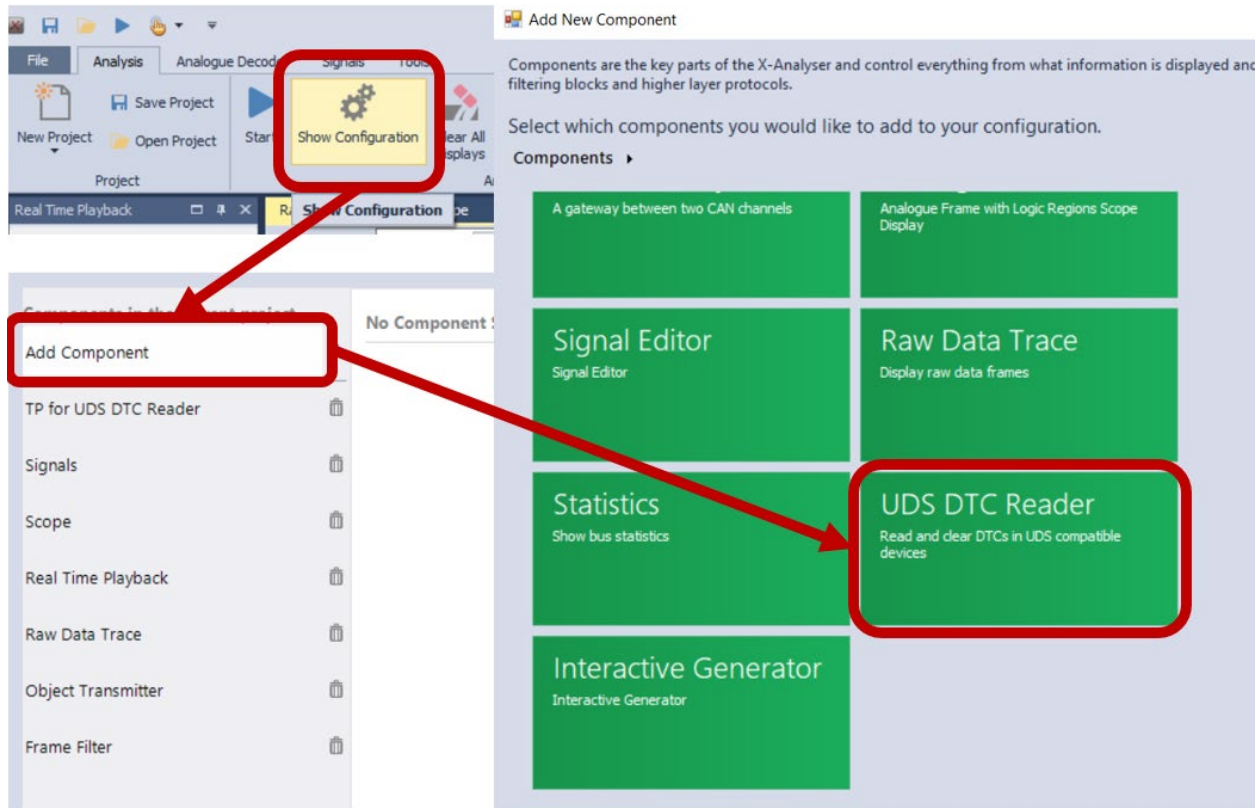
## Interpreting ISO15765 Messages

Once you have setup the ISO15765 Interpreter and Object Transmitter as described previously you can transmit to your ECU using the Object Transmitter and see the data interpreted as an ISO15765 message. The example screenshot below shows message response from the ECU for PID 0xF18C which has a data length of 23 bytes. The response to a Service ID 0x22 is on UDS Frame ID 0x62 because as per the UDS specification the response is Service ID + 0x40.

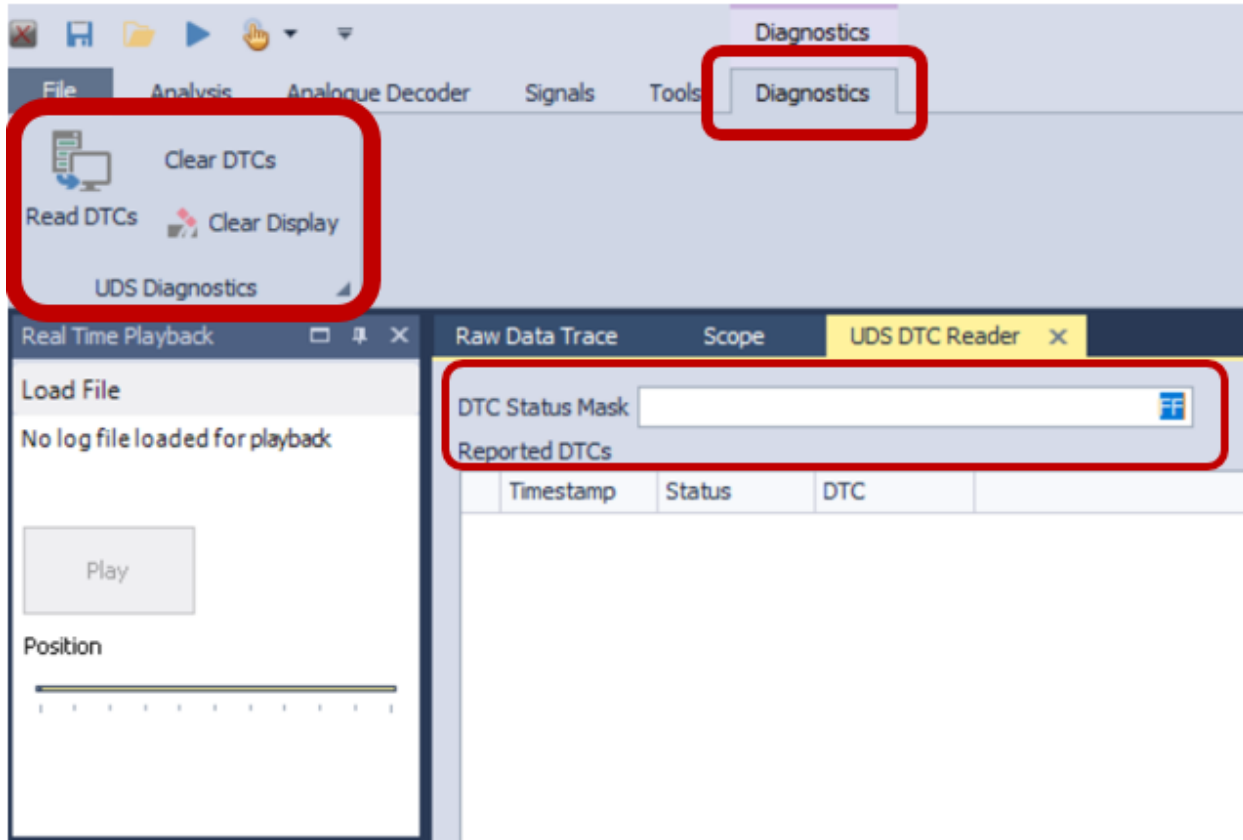
Timestamp	Channel	Direction	Frame Id (hex)	Frame Type	Data Length	Data
00:00:09.0505342	CAN 1	Rx	0x00	UDS Frame	3	7F 22 78
00:00:09.0670000	CAN 1	TxReq	0x720	Std. Frame	8	30 0A 00 00 00 00 00 00
00:00:09.1513112	CAN 1	Rx	0x62	UDS Frame	23	62 F1 8C 32 30 30 30 30 30 30 30 36 31 39 31 38 34 32 20 20 20
00:00:09.1530000	CAN 1	Rx	0x728	Std. Frame	8	23 20 20 20 38 34 32 20
00:00:12.0470000	CAN 1	Rx	0x401	Std. Frame	8	01 04 00 00 00 00 00 00

## UDS DTC Reader

With the **ISO 15765 Interpreter** configured for ECU->Tester and Tester -> ECU ID's. The **UDS DTC Reader** can be configured to read and clear DTC's this is done by selecting **Show Configuration -> Add Component -> UDS DTC Reader** this is shown below;



Once the component has been added and the **DTC Status Mask** set the **Diagnostics** tab can then be used to Read and Clear DTC's, this is shown below.



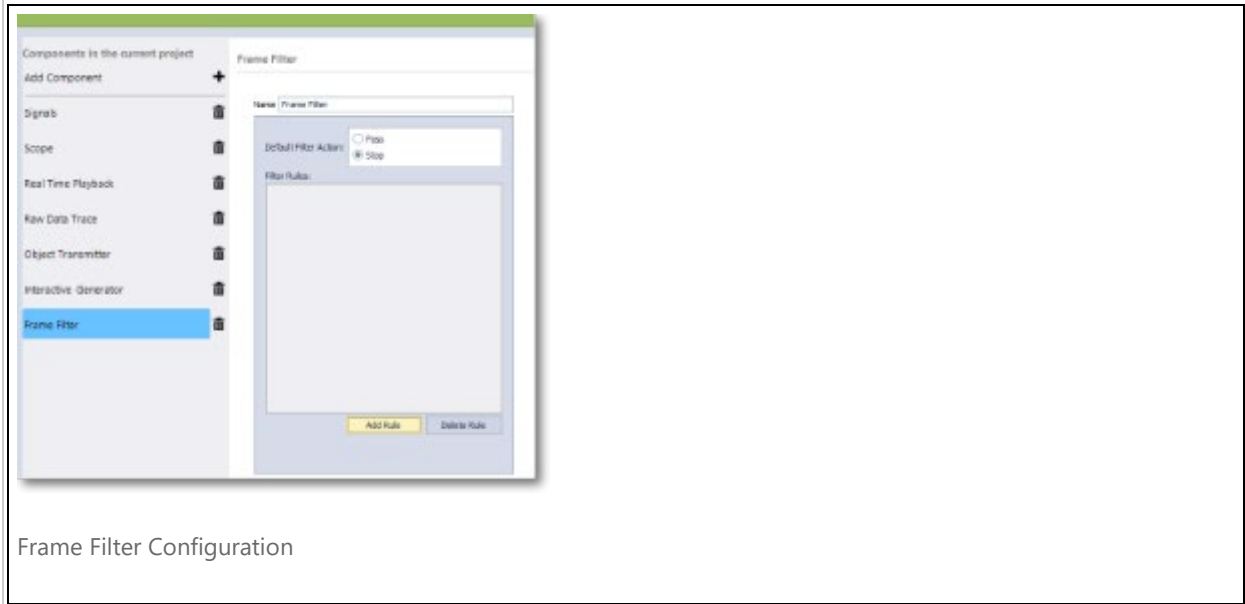
Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Frame Filtering

Main Features > Frame Filtering

In X-Analyser, it is possible to filter CAN messages by their Identifier, or range of Identifiers. It is possible to set up **Pass** or **Stop** rules. This means that you can select a IDs to be blocked or IDs to be displayed.

To set up a frame filter in X-Analyser, you need to stop capturing with X-Analyser and go to Show Configuration. You will be presented with the following screen if you then click on **Frame Filter**.



You must then set up a rule by selecting whether you want a **Pass** or **Stop** filter and then **Add Rule**. You then are presented with the following screen in which the **Mask** and **Match** filter is set up for the rule.



You can see that the default is to allow all frames through, or stop none of the frames.

Here we will use an example of a real situation to illustrate the Stop/Pass capabilities.



## Example Setup

The display below shows data collected from a vehicle.

Vehicle Data Collection

Here you can see that there are 12 IDs displayed. In our example, we will show how you can Pass a range of IDs, or Stop a range of IDs.

We will use all the IDs in the 0x200 range, i.e. 0x200 to 0x299.

### Stop Filter Example

Select the Standard (11 bit) Identifier at the top. This gives a range of 0x00 to 0x7FF.

Filter Setup

In this example you want to stop a frame whose 3 bits of its ID is 0x200 i.e. the ID begins with 2, so the range 0x200-0x299. You need to do the following:

- Select **Stop** before **Add Rule**
- Set **Mask** = 0x700
  - So this is saying that we only care about the upper 3 bits of the CAN ID. So we are looking at the area of the 1st digit of the CAN ID 0x7XX, where "X" means we don't care.
- **Match** = 0x200
  - So this saying that we are looking for 010xxxxxxx, where "x" means we don't care. In the area we have defined to look at in the Mask rule, we want this to Match 2, again the range 0x200-0x299 will be stopped.

The display will appear as below:



If you go back and do a data collection, you will notice that all the IDs in the 200 to 299 range are missing from the display, i.e. they are blocked.

Pass Filter

In this example you want to pass a frame whose 3 bits of its ID is 0x200 i.e. the ID begins with 2, so the range 0x200-0x299. you need to do the following:

- Select **Pass** before **Add Rule**
- Set **Mask** = 0x700
  - So this is saying that we only care about the upper 3 bits of the CAN ID. So we are looking at the area of the 1st digit of the CAN ID 0x7XX, where "X" means we don't care.
- **Match** = 0x200
  - So this saying that we are looking for 010xxxxxxx, where "x" means we don't care. In the area we have defined to look at in the Mask rule, we want this to Match 2, again the range 0x200-0x299 will be passed.

The display will appear the same as above. The only difference is that you have selected Pass instead of Stop.

If you go back and do a data collection, you will notice that all the IDs in the 200 to 299 range are the only ones shown on the display, i.e. they are passed, all others are blocked.

### J1939 Source Address Pass Rule

In the case of J1939 and NMEA2000 nodes have a defined source address. If you want to view frames from just one source address the following Pass Rule would be setup.

- Select **Extended Identifier**
- Select **Pass** before **Add Rule**
- Set **Mask** = 0x000000FF - We are looking at the "FF" section of the CAN ID the lower byte
- Set **Match** (To Source Address i.e. 05) = 0x00000005 - We are looking for the lower byte to match 05

This rule would pass all CAN frames from Source Address 05

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## J1939 Higher Layer Protocol

Main Features > J1939 Higher Layer Protocol

X-Analyser has support for the J1939 higher layer protocol which is used for in-vehicle networking in truck and bus applications. There are two main areas of support for which you must have the X-Analyser Professional edition; message/signal interpretation and message transmission.

## Starting a J1939 Project

To quickly set up the X-Analyser for J1939 work, you need to set up a new project. This is done by **New Project** -> **J1939 Layout**.

## J1939 Message & Signal Interpretation

X-Analyser has the application layer database of Parameter Group Numbers (PGNs) and SPN (Suspect Parameter Number) signals in-built with X-Analyser Professional Edition. Therefore their values can be viewed in the J1939 Messages document when they are received by X-Analyser.

The screenshot displays the X-Analyser software interface. At the top, there are tabs for 'Raw Data Trace', 'Scope', and 'J1939 Messages'. Below the tabs, there are controls for 'Display Mode' (Chronological), 'Time Display' (Absolute), and 'Autoscroll' (Off). The main area shows a table of J1939 messages with columns for Timestamp, Frame Name, Frame Id (hex), Data Length, Data, and Other Prop... The table contains several rows of message data. Below the table, there is a 'Signals' section with a search bar and a table of signal values. The signal table has columns for Signal Name, Value, and Units. The signals listed include Engine Torque Mode, Actual Engine - Percent Torque (Fractional), Driver's Demand Engine - Percent Torque, Actual Engine - Percent Torque, Engine Speed, Source Address of Controlling Device for Engine Control, Engine Starter Mode, and Engine Demand - Percent Torque.

Timestamp	Frame Name	Frame Id (hex)	Data Length	Data	Other Prop...
00.00:01:20.1618284	Auxiliary Input/Output Status 1 (PGN: 65241)	PGN: FED9, Src: 00, DP: 0, Pr: 6, PS: D9	8	0C 0C FF FF 00 00 0F 67	
00.00:01:20.1618682	Electronic Engine Controller 2 (PGN: 61443)	PGN: F003, Src: 00, DP: 0, Pr: 3, PS: 03	8	FF 62 FF 62 FF FF FF FF	
00.00:01:20.1618838	Electronic Transmission Controller 1 (PGN: 61442)	PGN: F002, Src: 03, DP: 0, Pr: 3, PS: 02	8	05 00 0E 00 FF F0 32 FF	
00.00:01:20.1618982	Vehicle Dynamic Stability Control 2 (PGN: 61449)	PGN: F009, Src: 08, DP: 0, Pr: 6, PS: 09	8	37 A0 7F 97 87 59 80 7C	
00.00:01:20.1619130	Electronic Engine Controller 3 (PGN: 65247)	PGN: FEDF, Src: 00, DP: 0, Pr: 6, PS: DF	8	87 FF FF FF FF FF FF FF	
00.00:01:20.1698711	Electronic Transmission Controller 2 (PGN: 61445)	PGN: F005, Src: 03, DP: 0, Pr: 6, PS: 05	8	7F F0 FF 7F 20 44 00 F7	
00.00:01:20.1748192	Electronic Engine Controller 1 (PGN: 61444)	PGN: F004, Src: 00, DP: 0, Pr: 3, PS: 04	8	F1 A8 A8 54 33 0C FF FF	

Signal Name	Value	Units
Engine Torque Mode	1	bit
Actual Engine - Percent Torque (Fractional)	1.875	%
Driver's Demand Engine - Percent Torque	43	%
Actual Engine - Percent Torque	43	%
Engine Speed	1642.5	rpm
Source Address of Controlling Device for Engine Control	12	SA
Engine Starter Mode	15	bit
Engine Demand - Percent Torque	130	%

## J1939 Message Transmission with Interactive Generator

Note: To use J1939 signals in the **Interactive Generator** must use a J1939 CAN database. One can be obtained from Warwick Control upon request free of charge.

The various CAN identifier fields and signal values can be modified in the Interactive Generator.

## J1939 Signal Panels, Scope and Gauges

Note: To use J1939 signals in the Signals, Scope or Gauge panels you must use a J1939 CAN database. One can be obtained from Warwick Control upon request free of charge.

It is important that the CAN database is modified so that the Priority and Source Address is correct for each message (this is the least significant 8-bits of the CAN identifier).

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## J1939 Higher Layer Protocol

Main Features > J1939 Higher Layer Protocol

X-Analyser has support for the J1939 higher layer protocol which is used for in-vehicle networking in truck and bus applications. There are two main areas of support for which you must have the X-Analyser Professional edition; message/signal interpretation and message transmission.

### Starting a J1939 Project

To quickly set up the X-Analyser for J1939 work, you need to set up a new project. This is done by **New Project** -> **J1939 Layout**.

### J1939 Message & Signal Interpretation

X-Analyser has the application layer database of Parameter Group Numbers (PGNs) and SPN (Suspect Parameter Number) signals in-built with X-Analyser Professional Edition. Therefore their values can be viewed in the J1939 Messages document when they are received by X-Analyser.

The screenshot displays the X-Analyser software interface. At the top, there are tabs for 'Raw Data Trace', 'Scope', and 'J1939 Messages'. Below the tabs, there are controls for 'Display Mode' (Chronological), 'Time Display' (Absolute), and 'Autoscroll' (Off). The main area shows a table of J1939 messages with columns for Timestamp, Frame Name, Frame Id (hex), Data Length, Data, and Other Prop... The table contains several rows of message data. Below the message table, there is a 'Signals' section with a search bar and a table of signal names, values, and units. The signal table includes entries like 'Engine Torque Mode', 'Actual Engine - Percent Torque (Fractional)', 'Driver's Demand Engine - Percent Torque', 'Actual Engine - Percent Torque', 'Engine Speed', 'Source Address of Controlling Device for Engine Control', 'Engine Starter Mode', and 'Engine Demand - Percent Torque'. At the bottom of the interface, there is a status bar showing the current message being viewed: '00.00:01:20.1758131 Vehicle Dynamic Stability Control 2 (PGN: 61449) PGN: F009, Src: 00, DP: 0, Pr: 6, PS: 09 8 34 A0 7F DD 87 B5 8A 81'.

Timestamp	Frame Name	Frame Id (hex)	Data Length	Data	Other Prop...
00.00:01:20.1618284	Auxiliary Input/output Status 1 (PGN: 65241)	PGN: FED9, Src: 00, DP: 0, Pr: 6, PS: D9	8	0C 0C FF FF 00 00 0F 67	
00.00:01:20.1618682	Electronic Engine Controller 2 (PGN: 61443)	PGN: F003, Src: 00, DP: 0, Pr: 3, PS: 03	8	FF 62 FF 62 FF FF FF FF	
00.00:01:20.1618838	Electronic Transmission Controller 1 (PGN: 61442)	PGN: F002, Src: 03, DP: 0, Pr: 3, PS: 02	8	05 00 0E 00 FF F0 32 FF	
00.00:01:20.1618982	Vehicle Dynamic Stability Control 2 (PGN: 61449)	PGN: F009, Src: 00, DP: 0, Pr: 6, PS: 09	8	37 A0 7F 97 87 59 8D 7C	
00.00:01:20.1619130	Electronic Engine Controller 3 (PGN: 65247)	PGN: FEDF, Src: 00, DP: 0, Pr: 6, PS: DF	8	87 FF FF FF FF FF FF FF	
00.00:01:20.1698711	Electronic Transmission Controller 2 (PGN: 61445)	PGN: F005, Src: 03, DP: 0, Pr: 6, PS: 05	8	7F F0 FF 7F 20 44 00 F7	
00.00:01:20.1748192	Electronic Engine Controller 1 (PGN: 61444)	PGN: F004, Src: 00, DP: 0, Pr: 3, PS: 04	8	F1 A8 A8 54 33 0C FF FF	

Signal Name	Value	Units
Engine Torque Mode	1	bit
Actual Engine - Percent Torque (Fractional)	1.875	%
Driver's Demand Engine - Percent Torque	43	%
Actual Engine - Percent Torque	43	%
Engine Speed	1642.5	rpm
Source Address of Controlling Device for Engine Control	12	SA
Engine Starter Mode	15	bit
Engine Demand - Percent Torque	130	%

### J1939 Message Transmission with Interactive Generator

Note: To use J1939 signals in the **Interactive Generator** must use a J1939 CAN database. One can be obtained from Warwick Control upon request free of charge.

The various CAN identifier fields and signal values can be modified in the Interactive Generator.

## J1939 Signal Panels, Scope and Gauges

Note: To use J1939 signals in the Signals, Scope or Gauge panels you must use a J1939 CAN database. One can be obtained from Warwick Control upon request free of charge.

It is important that the CAN database is modified so that the Priority and Source Address is correct for each message (this is the least significant 8-bits of the CAN identifier).

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## CAN Gateway

Collapse All Expand All

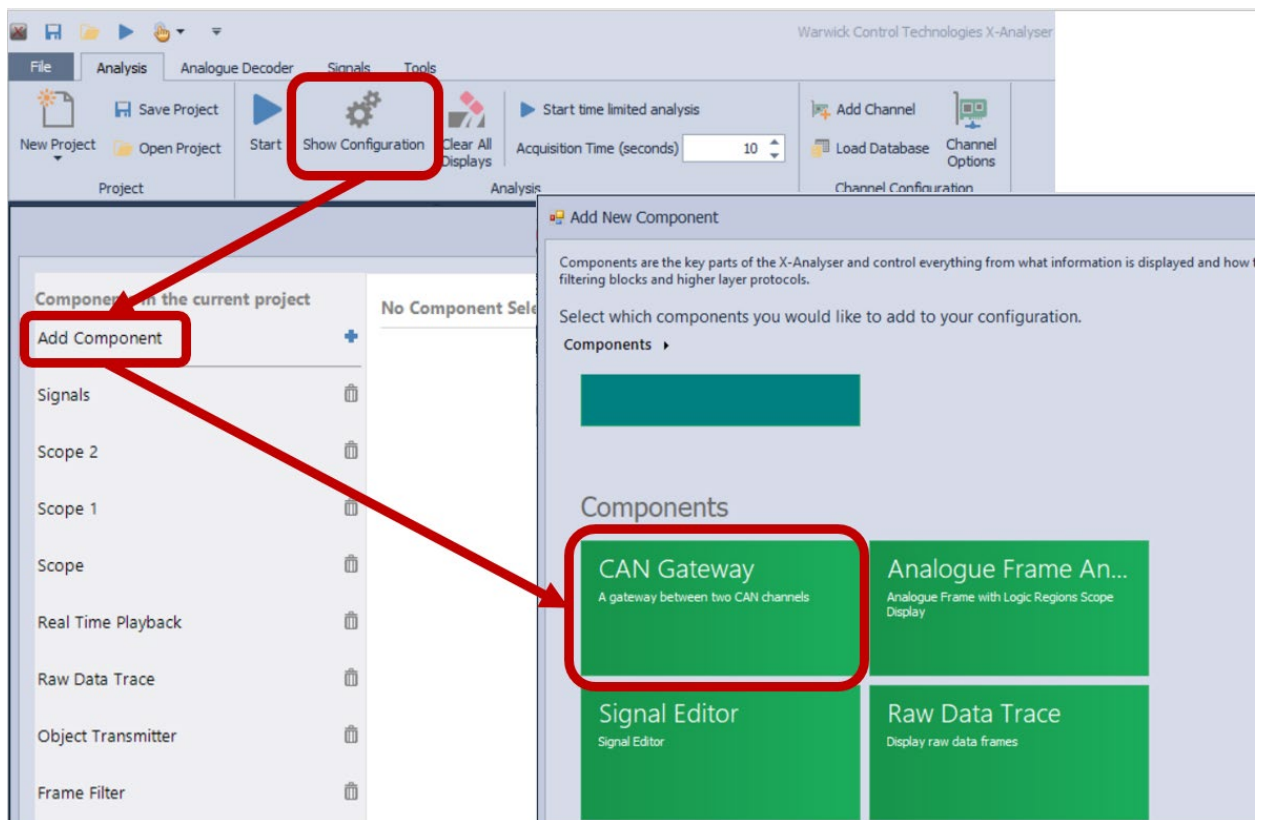
Main Features > CAN Gateway

X-Analyser includes a gateway component that can be used to duplicate frames from one CAN bus on to another.

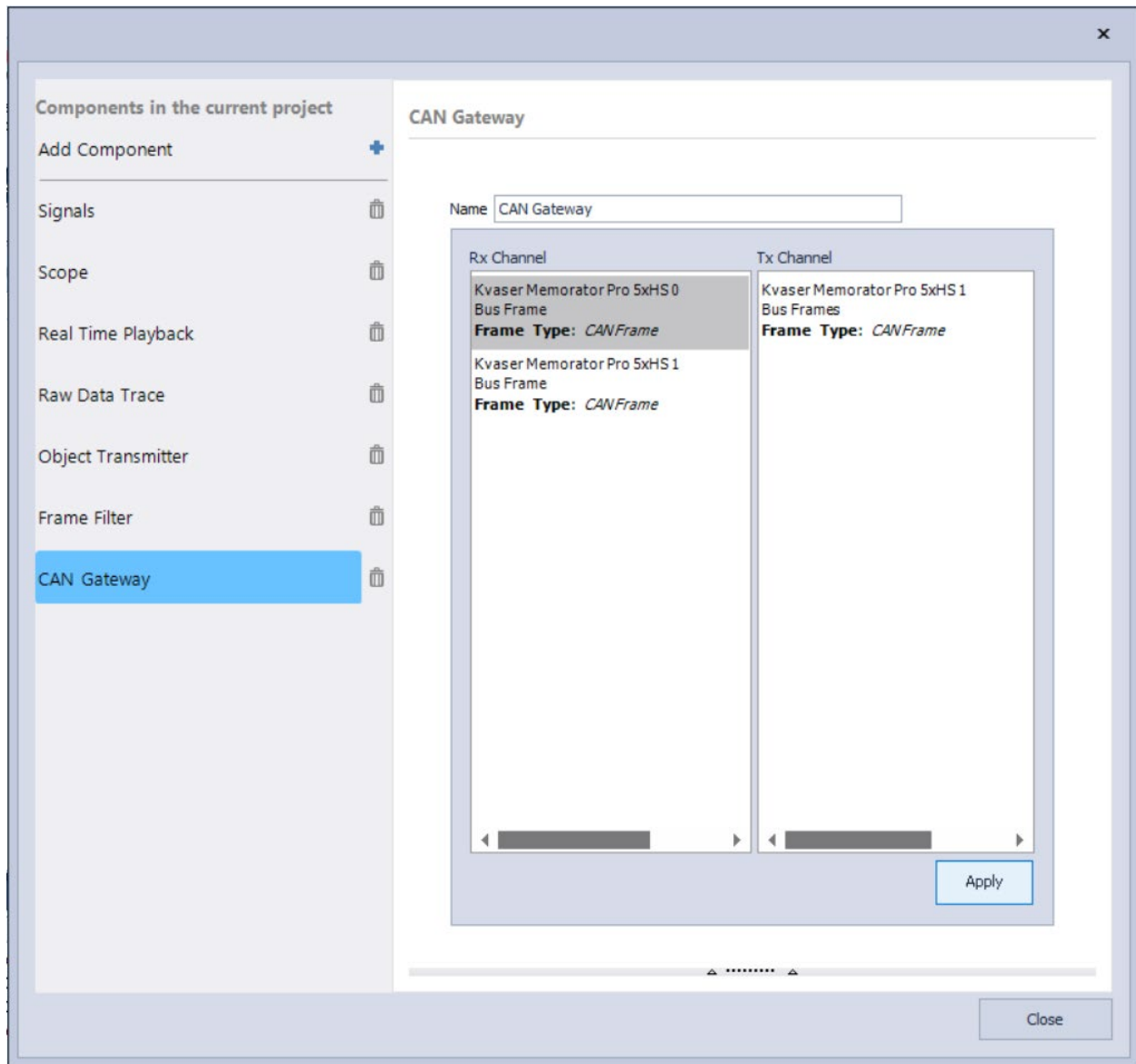
This feature requires you to have two CAN channels configured in your project, you can then configure which channel to receive frames on and which channel to transmit them back out on.

### Adding and Configuring a Gateway

1. Start a **New Project** and configure two CAN channels (see the [Quick Start for CAN](#) for details)
2. Add a new **CAN Gateway** component by selecting **Show Configuration -> Add Component -> CAN Gateway**



3. Once the component has been added, it must be configured by selecting **CAN Gateway** in the **Show Configuration** window. Select the **source** interface that will receive CAN frames, this goes in the RX list.
4. Then select the TX interface, this will be the interface that re-transmits the CAN frames on to a second network.



See Also

[Quick Start Guide](#)

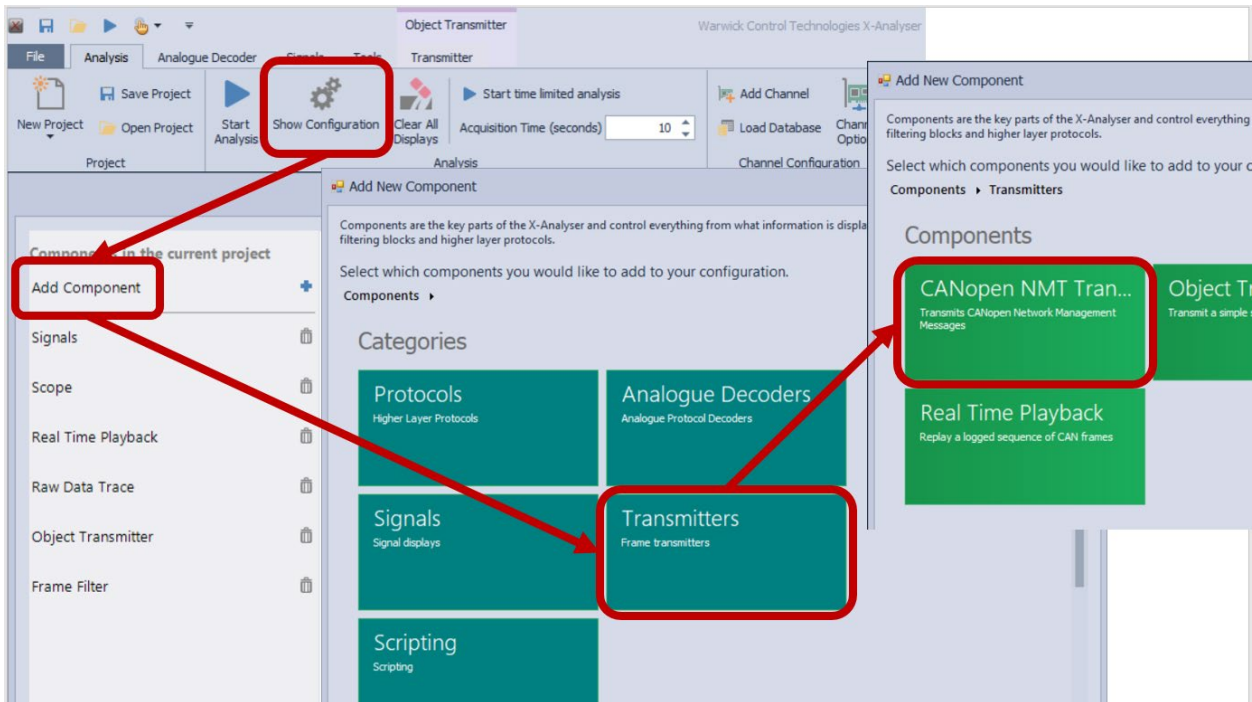
[Quick Start for CAN](#)



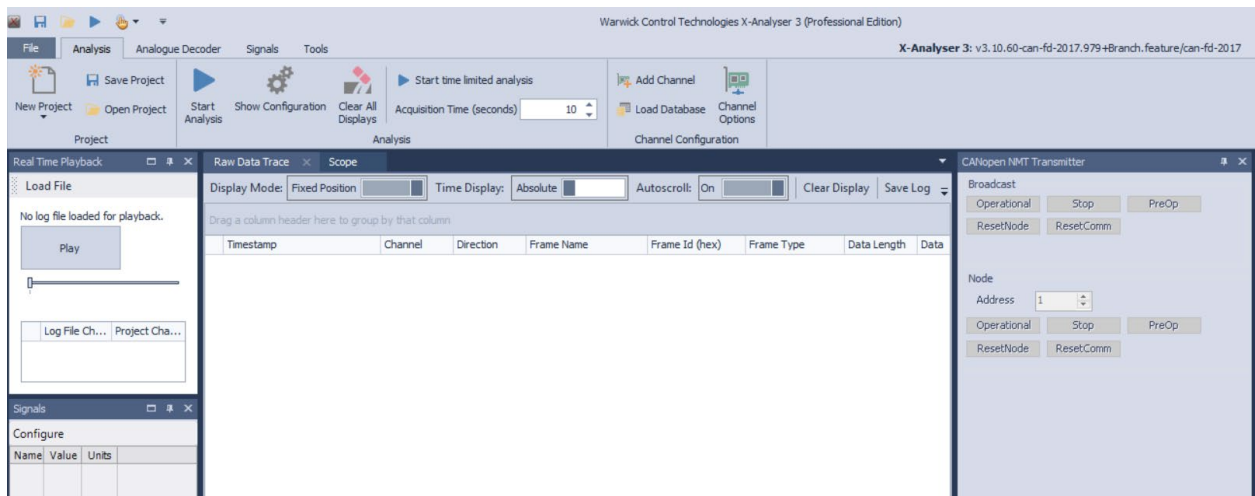
## CANopen NMT Transmitters

Main Features > CANopen NMT Transmitters

X-analyser 3 features a CANopen Network Management Transmitter component. This saves the hassle of creating network management transmitters in the object transmitter for CANopen networks. The component can be used to broadcast network management messages or address messages to a particular node. To add this component go to **Show Configuration** then **Add Component** then **Transmitters** then **CANopen NMT Transmitter**



Once the component is added the display will look as per below



The NMT transmitter is greyed out and will be available for selection once analysis has been started. The NMT buttons cause the following transmissions;

- **Operational** – Changes NMT state to operational.
- **Stop** – Changes NMT state to stopped
- **PreOp** – Changes NMT state to Pre -Operational
- **ResetNode** – Resets node and changes NMT state to sub-state reset application
- **ResetComm** – Resets communication and changes NMT state to sub-state reset communication.

As you can see from the display these options are available to broadcast or send to an individual node.

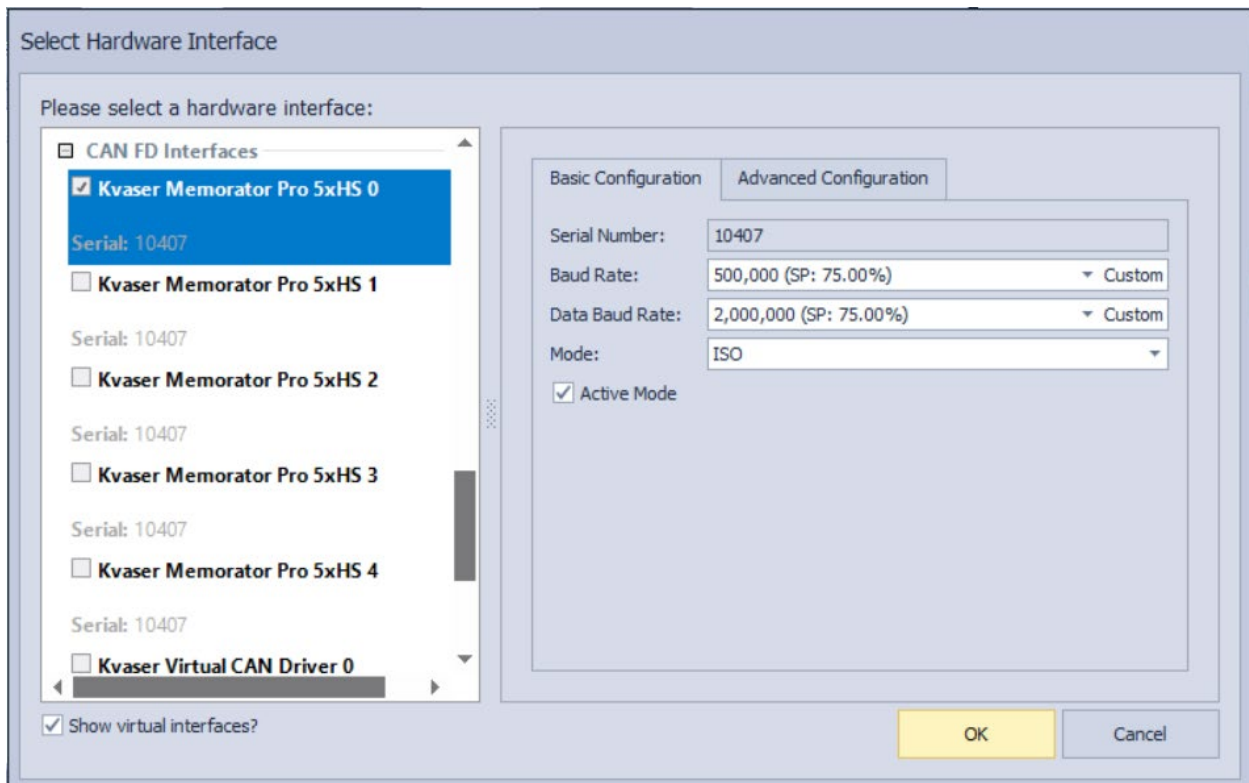
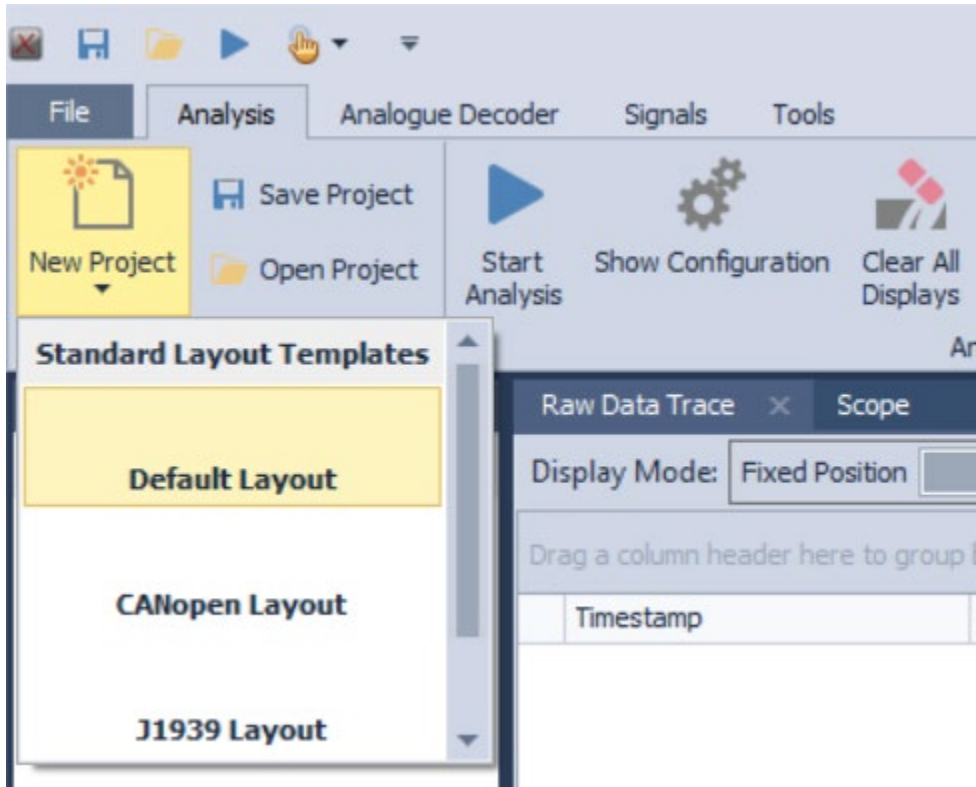
---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## CANFD Configuration

Main Features > CANFD Configuration

To select a CANFD configuration select **New Project** and then **Default Layout** and scroll to a CANFD interface, as shown below.

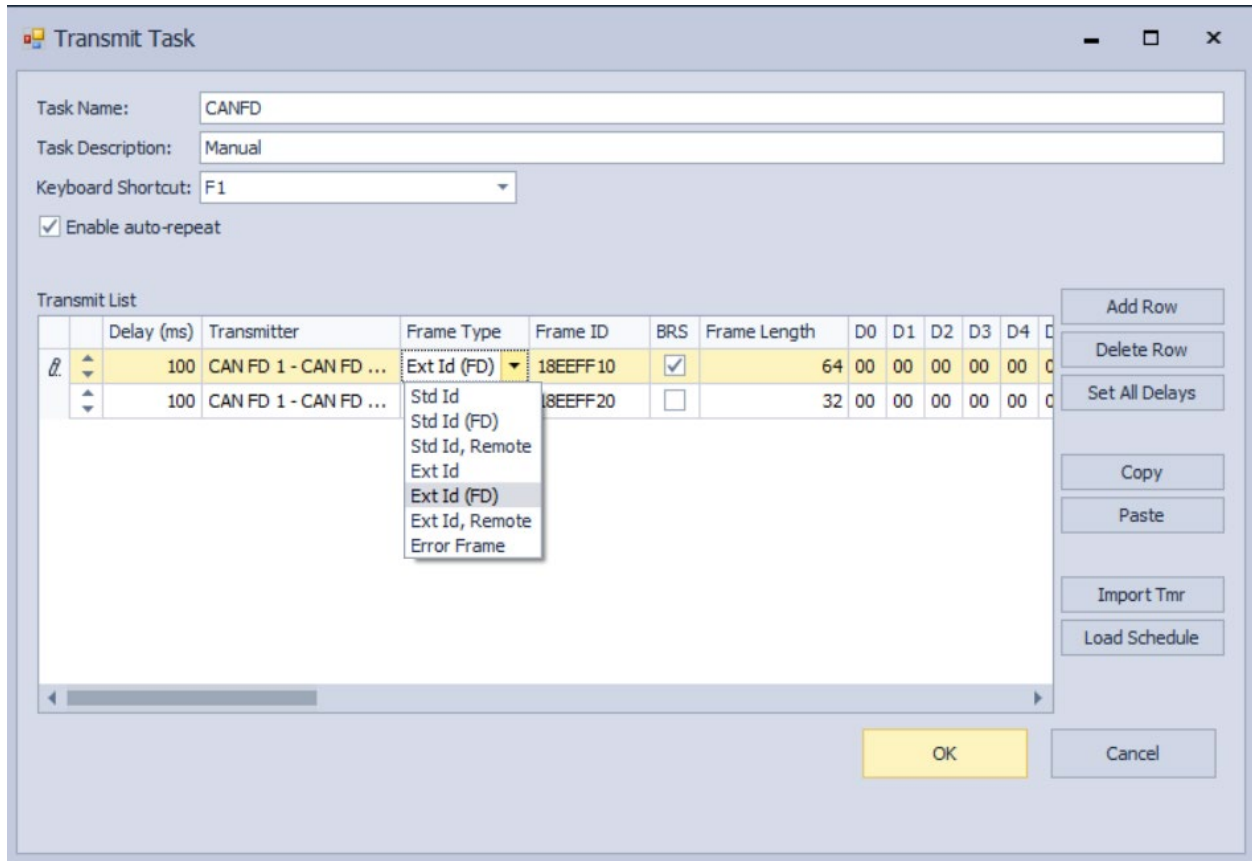


Then select the interface and Baud rates for CANFD. This will give CANFD functionality with the project. The other feature in X-analyser 3 for CANFD is CANFD transmitters. Advantage can be taken of using

500kbps Baud Rate and 500kbps Data Baud Rate and using the 64 byte DLC by selecting **Custom** next to the Data Baud Rate and choosing 500kbps and timing parameters. Also the **BRS** box should be unchecked for this setup.

## CANFD Transmitters

To create a CANFD transmitter select **Add New Transmitter** within the object transmitter window, the same as a CAN transmitter. Then the window below shall appear, to select CANFD transmitter, the **Frame Type** should be **Std Id (FD)** and **Ext Id (FD)**. As shown below.



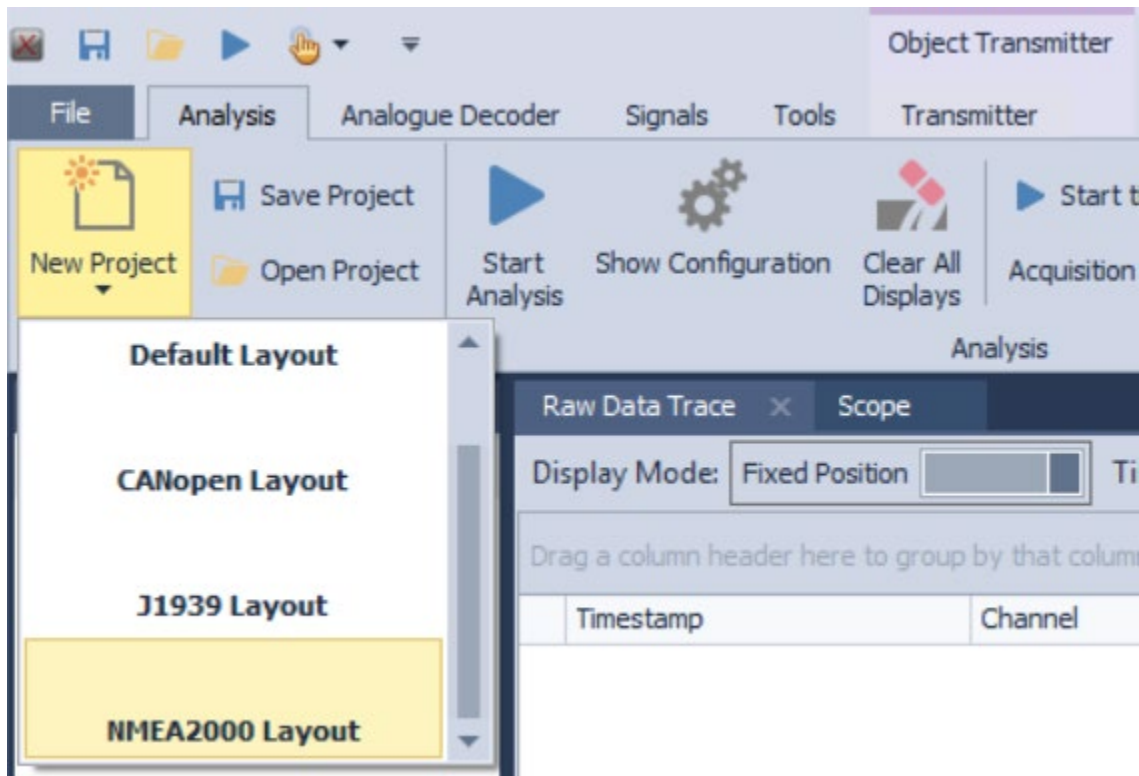
As long as the **(FD)** option is selected X-analyser will produce CANFD frames. As you can see above the **Frame ID** and **Frame Length** have been populated with **BRS** selected on ID 0x18EEFF10. Data bytes can then be filled accordingly.

## NMEA2000 Higher Layer Protocol

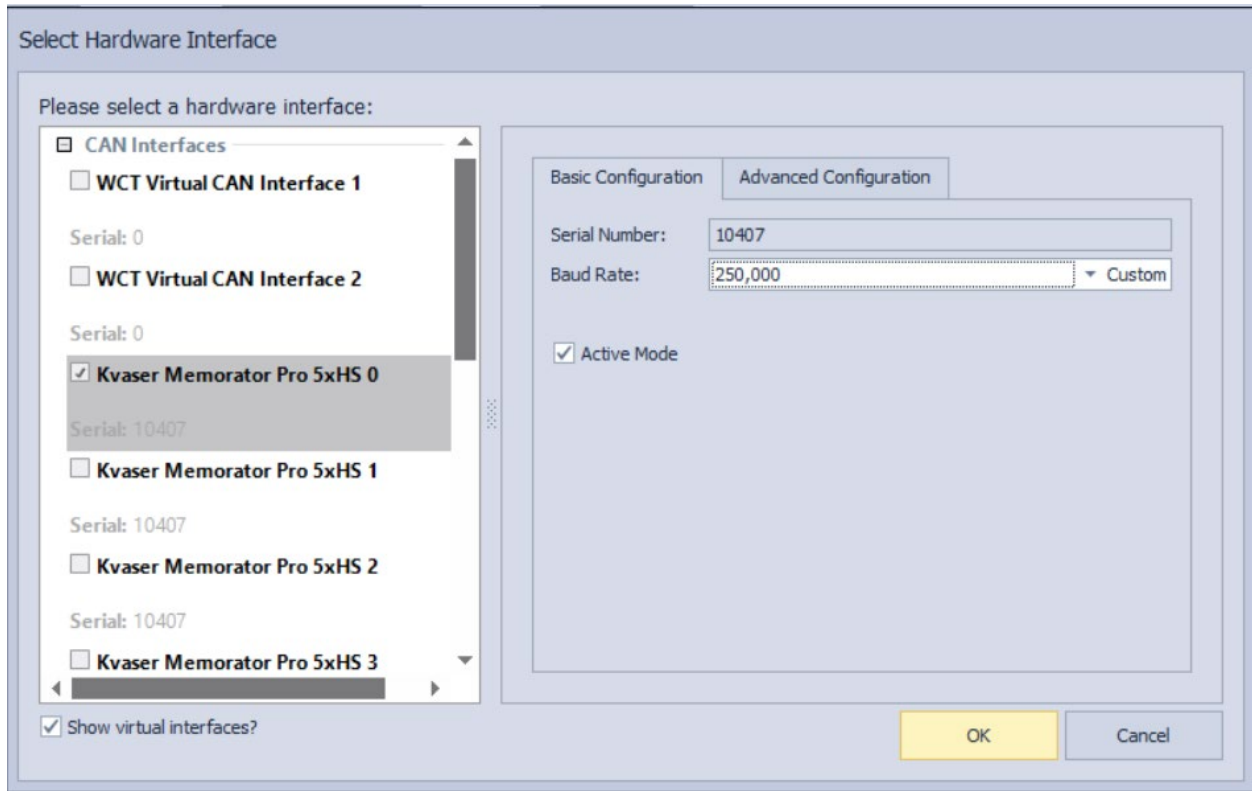
Main Features > NMEA2000 Higher Layer Protocol

### NMEA2000 Layout Setup

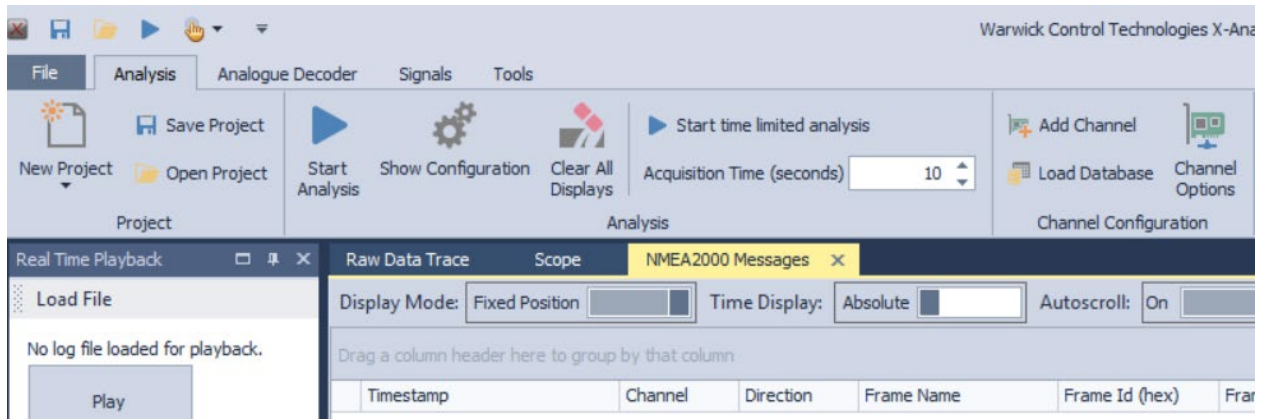
To create an **NMEA2000 Layout** select **New Project** and then **NMEA2000 Layout**.



You will then need to select your connected interface and select the **Baud Rate** which is 250kbps on NMEA2000 networks.



After selecting **OK** the **NMEA2000 Messages** tab will appear on the X-analyser screen.



Then to collect data just select **Start Analysis** and the NMEA2000 CAN messages will appear in the **NMEA2000 Messages** tab.

Below is a screen shot of a NMEA2000 data collection.

Timestamp	Frame Name	Frame Id (hex)	Frame Type	Data Length	Data	Other Prop...
00.00:00:08.1620000	Wind Data	1FD02 Src/Dest: 01/FF	Single Frame	8	06 05 00 09 7A FA FF FF	
00.00:00:08.1730000	Engine Parameters, Rapid Update	1F200 Src/Dest: A0/FF	Single Frame	8	00 00 00 00 00 7F FF FF	
00.00:00:08.1840000	Transmission Parameters, Dynamic	1F205 Src/Dest: A0/FF	Single Frame	8	00 FF FF FF FF FF 00 FF	
00.00:00:08.1950000	Position, Rapid Update	1F801 Src/Dest: 04/FF	Single Frame	8	47 1A 2B 1F 02 7B 00 FF	
00.00:00:08.2060000	Vessel Heading	1F112 Src/Dest: 00/FF	Single Frame	8	08 5C 52 FF 7F FF 7F FD	
00.00:00:08.2180000	Wind Data	1FD02 Src/Dest: 01/FF	Single Frame	8	07 05 00 09 7A FA FF FF	

Signal Name	Value	Units
Sequence ID	7	
Wind Speed	0.05	
Wind Direction	3.1241	
Wind Reference	2	
NMEA Reserved	0	

### NMEA2000 Protocol Layer Transmitters

For the ease of creating NMEA2000 frames within the **Object Transmitter** X-analyser has the option to choose the NMEA2000 protocol layer to create **Single Frame** and **Fast Packet** messages. How to do so shall now be explained.

After selecting an **NMEA2000 Layout** select the **Object Transmitter** and right click this window to get the option to **Add New Transmitter** then the **Transmit Task** window shall then appear.

In the **Transmitter** column select **NMEA2000 Protocol Layer**. Then the **Frame Type** can be selected as **Single Frame** or **Fast Packet**. **Frame ID** will need entering (excluding the source address). The **Frame Length** can be set from 0-8 bytes for **Single Frame** and 9-223 bytes for **Fast Packet** frame type.

**Transmit Task**

Task Name: NMEA Transmitter Instructions

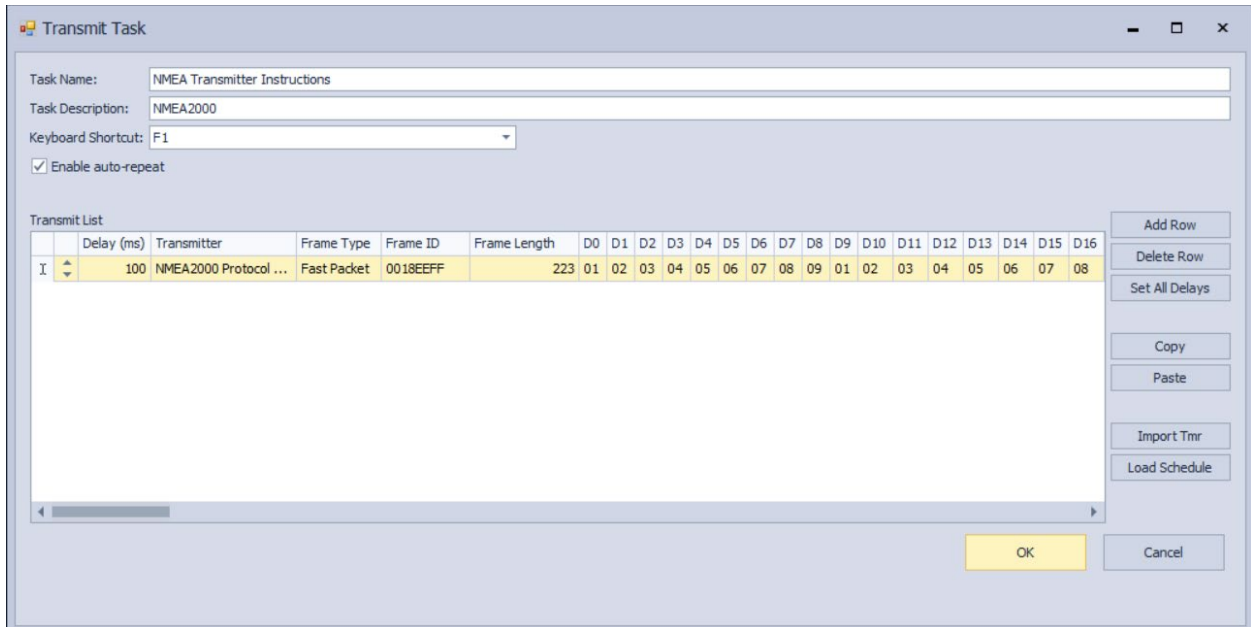
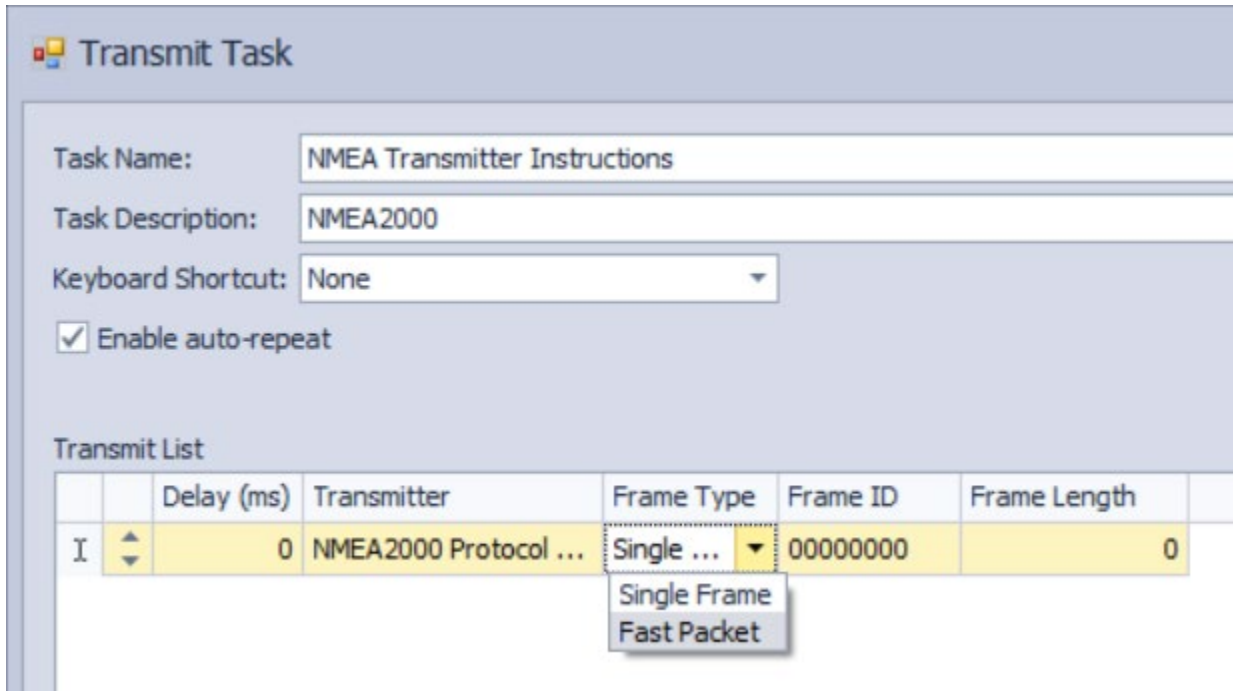
Task Description: NMEA2000

Keyboard Shortcut: None

Enable auto-repeat

Delay (ms)	Transmitter	Frame Type	Frame ID	Frame Length
0	CAN 1 - CAN frame	Ext Id	00000000	0

Transmitter dropdown options:  
 CAN 1 - CAN frame  
 NMEA2000 Protocol Layer - NMEA2000 frame



Once analysis has been started this NMEA2000 Frame can be sent by pressing F1 or starting the transmitter through X-analyser buttons.



Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## CANopen Higher Layer Protocol

Main Features > CANopen Higher Layer Protocol

X-Analyser has support for the CANopen higher layer protocol which is used for industrial automation and other off-highway applications. There are two main areas of support for which you must have the X-Analyser Professional edition; message/signal interpretation and message transmission.

### Starting a CANopen Project

To quickly set up the X-Analyser for CANopen work, you need to set up a new project. This is done by **New Project -> CANopen Layout**.

### CANopen Message Interpretation

The new project will have a new CANopen Messages window that will display received CANopen messages.

---

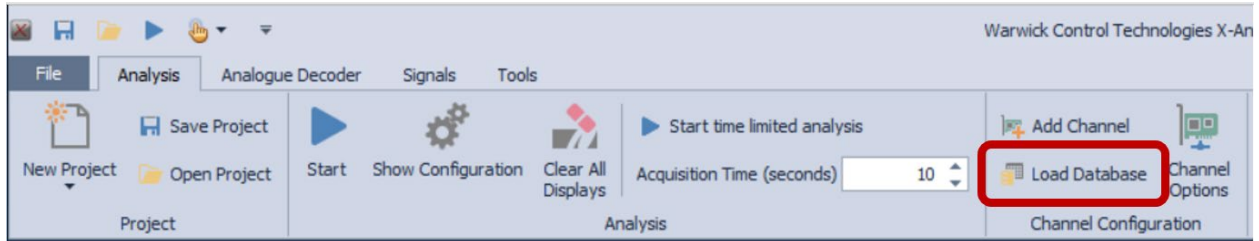
Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

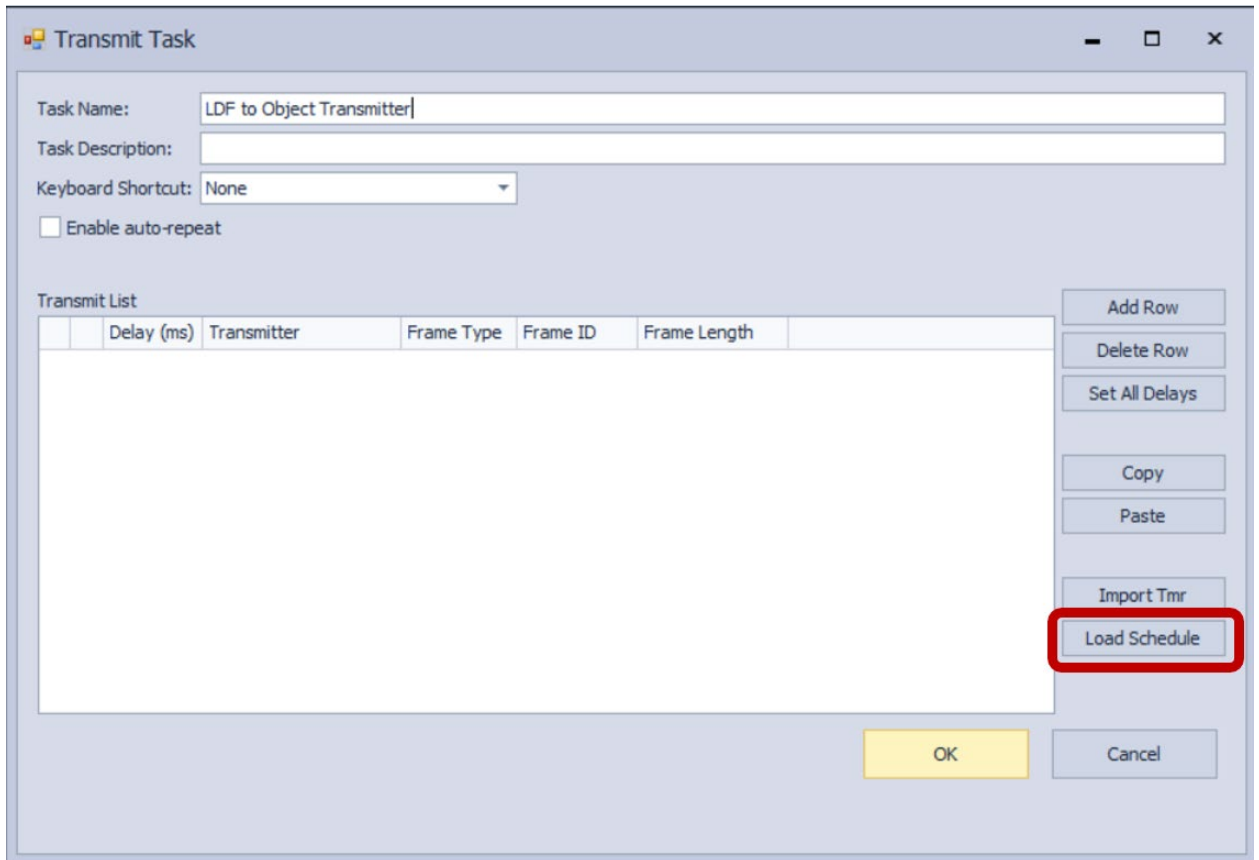
## Load LDF Schedule in Object Transmitter

Main Features > Load LDF Schedule in Object Transmitter

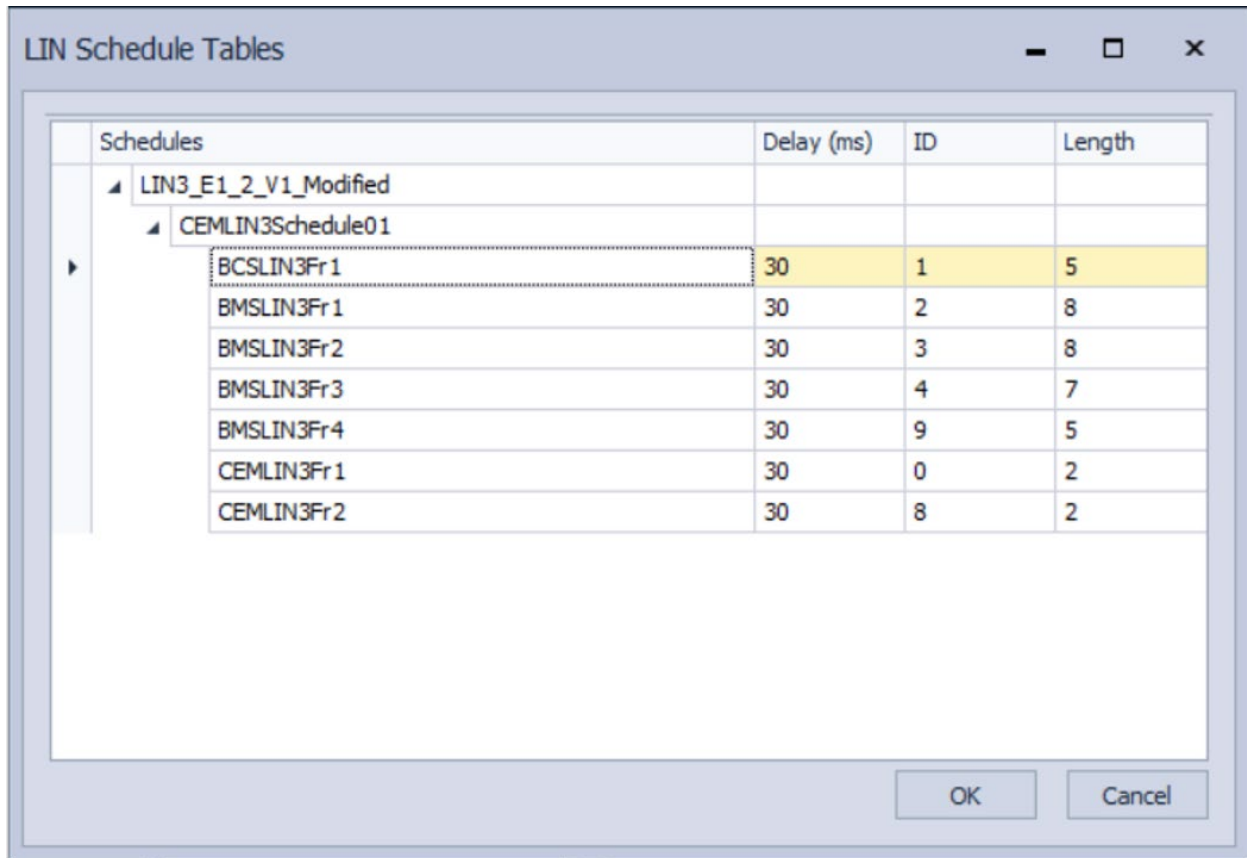
With a LIN interface selected **.ldf** files can be loaded by selecting **Load Database**



With an **.ldf** file loaded as well as LIN signals being able to be configured, you can also load the **.ldf** schedule via the object transmitter. To do so select **New Transmitter/Add New Transmitter** then the below window shall appear. To load an **.ldf** schedule select **Load Schedule**.



This shall then bring up the window below where LIN messages can be selected to load into the Object Transmitter. Select the LIN frame or group of frames from the above trees then press **OK** to create the associated LIN transmitters.



Schedules	Delay (ms)	ID	Length
<ul style="list-style-type: none"> <li>▲ LIN3_E1_2_V1_Modified           <ul style="list-style-type: none"> <li>▲ CEMLIN3Schedule01               <ul style="list-style-type: none"> <li>▶ BCSLIN3Fr1</li> <li>BMSLIN3Fr 1</li> <li>BMSLIN3Fr2</li> <li>BMSLIN3Fr3</li> <li>BMSLIN3Fr4</li> <li>CEMLIN3Fr1</li> <li>CEMLIN3Fr2</li> </ul> </li> </ul> </li> </ul>			
	30	1	5
	30	2	8
	30	3	8
	30	4	7
	30	9	5
	30	0	2
	30	8	2

The LIN frames will then be loaded into the Object Transmitter and then can be transmitted.

---

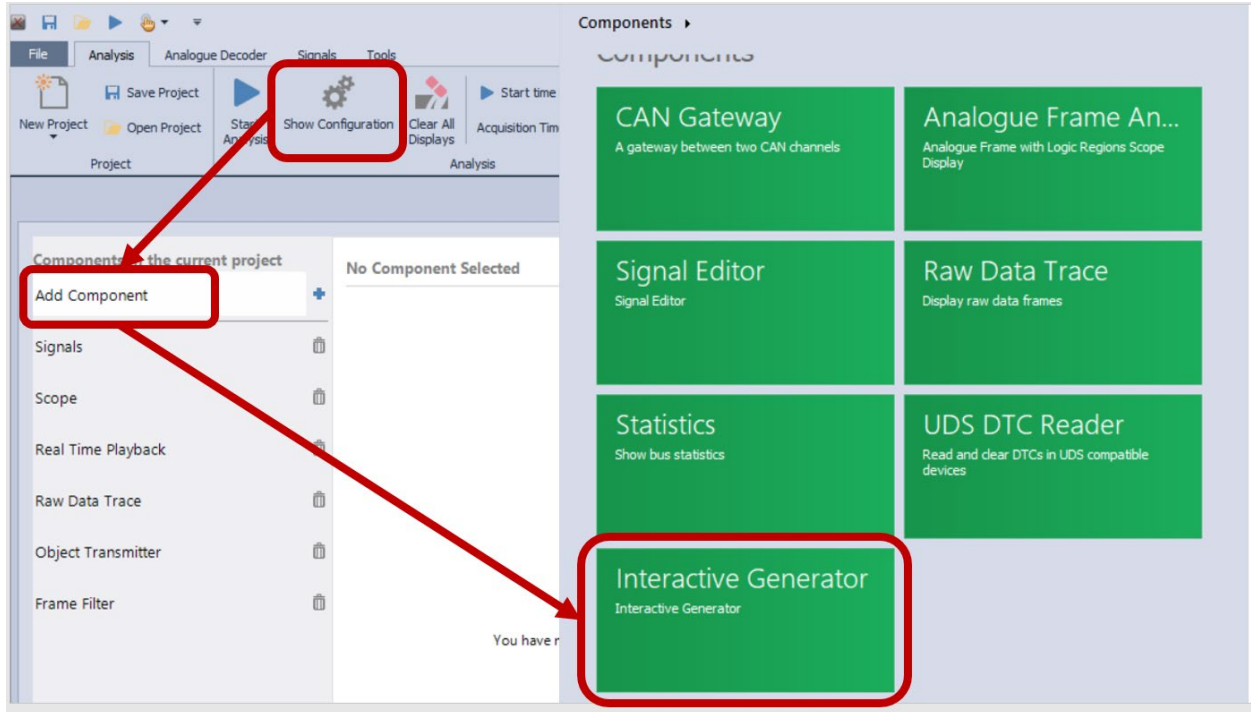
Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

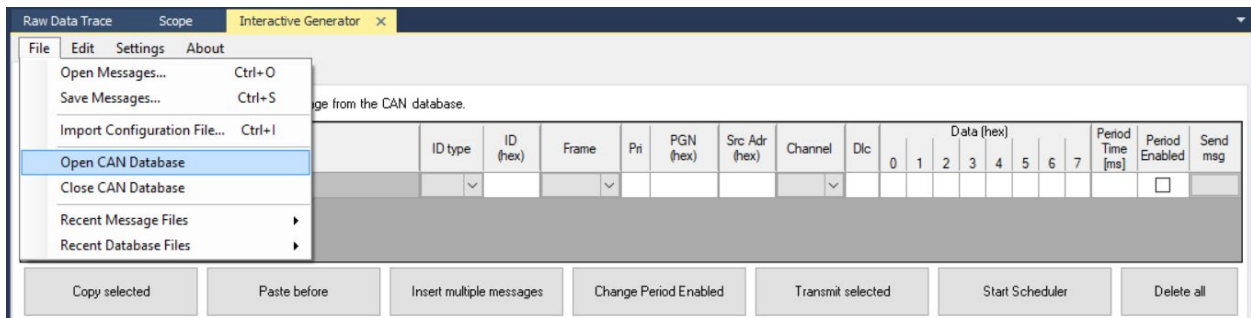
## Interactive Generator

Main Features > Interactive Generator

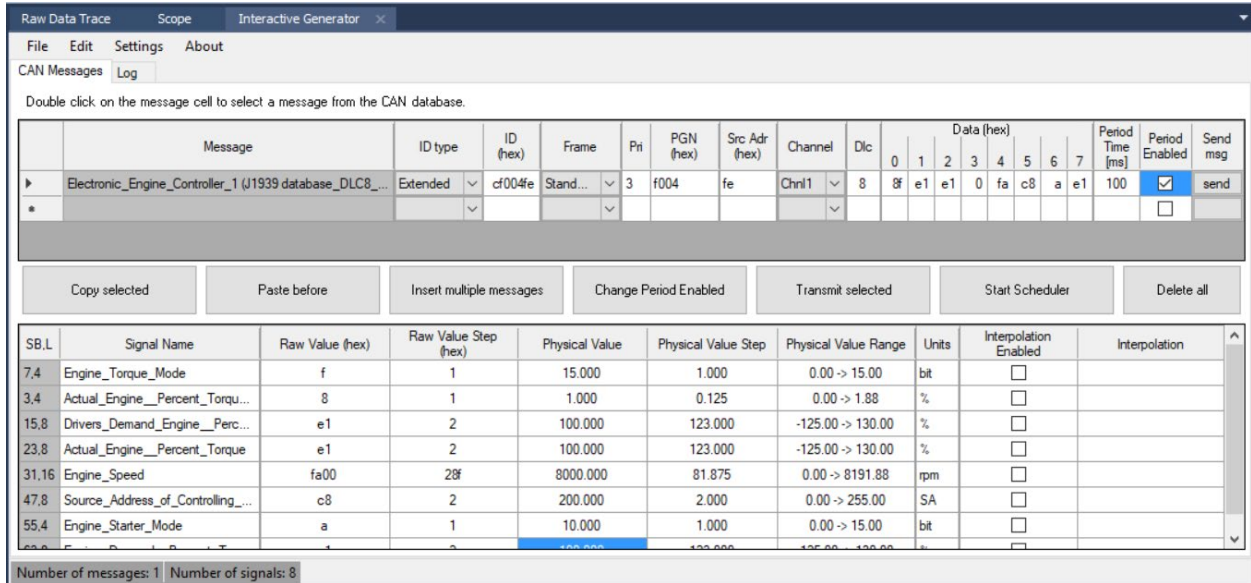
X-Analyser 3 comes with an **Interactive Generator** component. Within this component a **.dbc** file can be loaded and then CAN message, signal values changed on the fly. The component will transmit the frames within the **.dbc** and adjust the signal values according to scaling of the **.dbc**. To add the component select **Show Configuration -> Add Component -> Interactive Generator** as shown below.



With the **IG (Interactive Generator)** added you can then open a CAN database within the IG by selecting **File -> Open CAN Database** as shown.



Then as the IG states "Double Click on the message cell to select a message from the CAN database" and select a message from the database. I have chosen to use the J1939 database that comes with X-Analyser 3 and can be found under **Documents -> X-Analyser 3 -> J1939** and selected message **Electronic Engine Controller 1**.



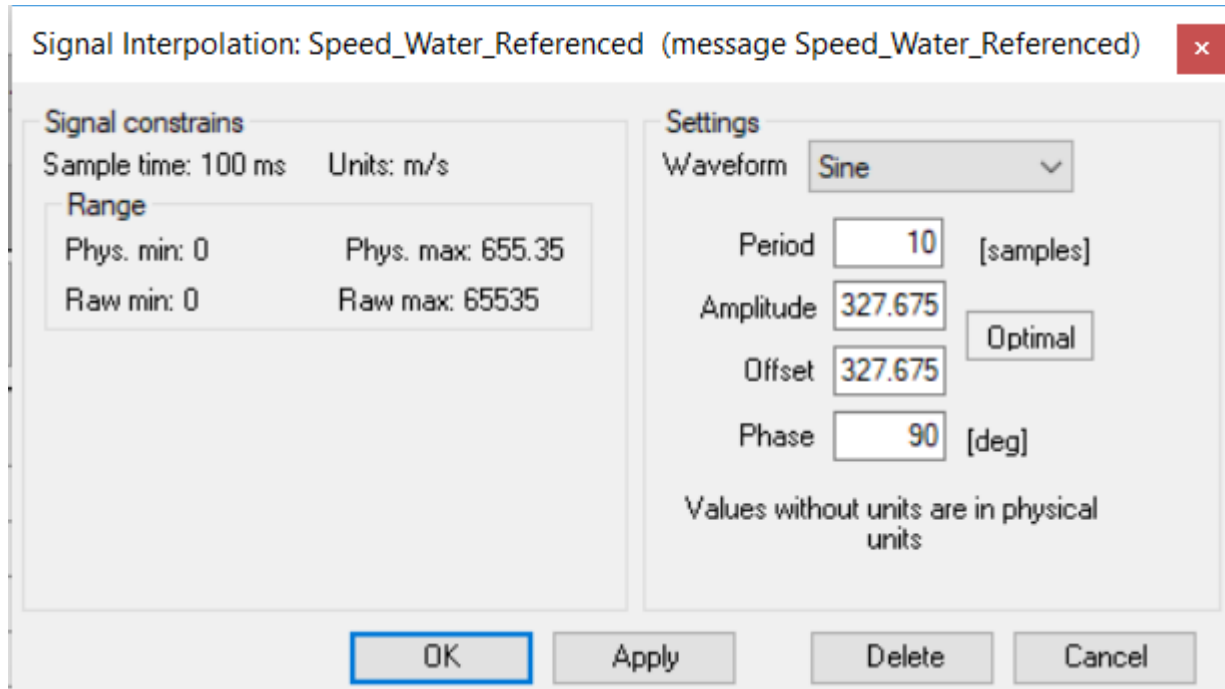
You can then enter the **Physical Value** for the signal you require and the IG will automate this into a Hexidecimal value into the CAN frame. It can then be sent onto a network with the press of the **Send** button or transmitted at a regular interval when analysis starts by selecting **Period Enabled** and selecting the **Period Time (ms)**.

There is also the option for **Interpolation** of a signal to select **Interpolation** type double click the **Interpolation** box next to the signal, for interpolation there are 3 options;

- Sine - Produces a sine waveform with signal values.
- Random - Moves through random values by setting a min and max for signal values.
- Toggle Switch - Switches between 2 values in the signal.

Below shows the configuration of each type of interpolation.

### Sine



Period - Increasing the number of samples will show less rate of change throughout the sine wave. Giving a longer smoother sine wave.

Amplitude - Height of the sine wave from vertical zero point.

Offset - Offset from zero vertically.

(NB - Amplitude and Offset **Optimal** settings will give a sine wave that moves throughout the signal range just below max and just above min.)

Phase - Offset in time base horizontally.

### Toggle Switch

Signal Interpolation: Speed\_Water\_Referenced (message Speed\_Water\_Referenced) ✕

Signal constrains		Settings	
Sample time: 100 ms	Units: m/s	Waveform	Toggle switch ▾
Range		Value 1	<input type="text" value="0"/> Min
Phys. min: 0	Phys. max: 655.35	Value 2	<input type="text" value="655.35"/> Max
Raw min: 0	Raw max: 65535	Values without units are in physical units	

Value 1 - 1st defined signal value

Value 2 - 2nd defined signal value

**Random**

Signal Interpolation: Speed\_Water\_Referenced (message Speed\_Water\_Referenced) ✕

Signal constrains		Settings	
Sample time: 100 ms	Units: m/s	Waveform	Random ▾
Range		Minimum	<input type="text" value="0"/> Min
Phys. min: 0	Phys. max: 655.35	Maximum	<input type="text" value="655.35"/> Max
Raw min: 0	Raw max: 65535	Values without units are in physical units	

Minimum - Minimum value of random values.

Maximum - Maximum value of random values.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## Introduction to CAN

The Controller Area Network was originally developed during the late 1980's by the German company Robert Bosch GmbH, for the automotive industry. Their motivation for the development of CAN was to provide a solution to the problem of the enormous and constantly-growing wiring harness required for inter-ECU communication in modern vehicles - customers were demanding more and more functional features of their vehicle, most of which operated electronically and required some form of communication with another on-board system, which meant more and more wiring. Their answer was to design a single network bus to which all the on-board peripherals could be attached.

In 1993 CAN became the standards ISO 11898 (for high-speed applications) and ISO 11519 (for lower-speed applications). It is a multi-master serial communications bus whose basic design specification called for high speed, high noise-immunity and error-detection features. As a result of these features the CAN bus has also become widely used within the manufacturing and aerospace industries.

The CAN bus is designed for communication between microcontrollers – in an automotive environment it is used to exchange information between on-board Electronic Control Units (ECUs) such as the Engine Management System, gearbox controls, instrument packs, and body electronics.

Theoretically up to 2032 devices (nodes) can be connected on a CAN bus (assuming one node with one ID number) on a single network. However due to the practical limitation of the hardware (i.e. the transceivers), it realistically allows up to 110 nodes (using the Philips 82C250 CAN controller) on a single network. SAE J2284 recommends a maximum of 16 nodes at 500 Kbit/sec, and 32 nodes at 250 Kbit/sec and 125 Kbit/sec.

CAN offers data communication up to 1 Mbit/sec, thus facilitating real-time control. In addition, the Error Confinement and the Error Detection features make it more reliable in noise-critical environments. In the automotive industries, the common data rate for powertrain controls is 500 Kbit/sec, and 125 Kbit/sec for body control systems. SAE J1939 (Truck and Bus Standard) recommends 250 Kbit/sec.

CAN's most attractive features include:

1. Low cost
2. Extreme robustness
3. High data transmission speeds (up to 1 MBit/sec).
4. Reliability. Excellent error handling and Error Confinement abilities.
5. Automatic re-transmission of faulty messages.



6. Automatic bus disconnection of nodes that are suspected to be physically faulty.
7. Functional addressing – data messages do not contain source or destination addresses, only identifiers relating to their function and/or priority.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

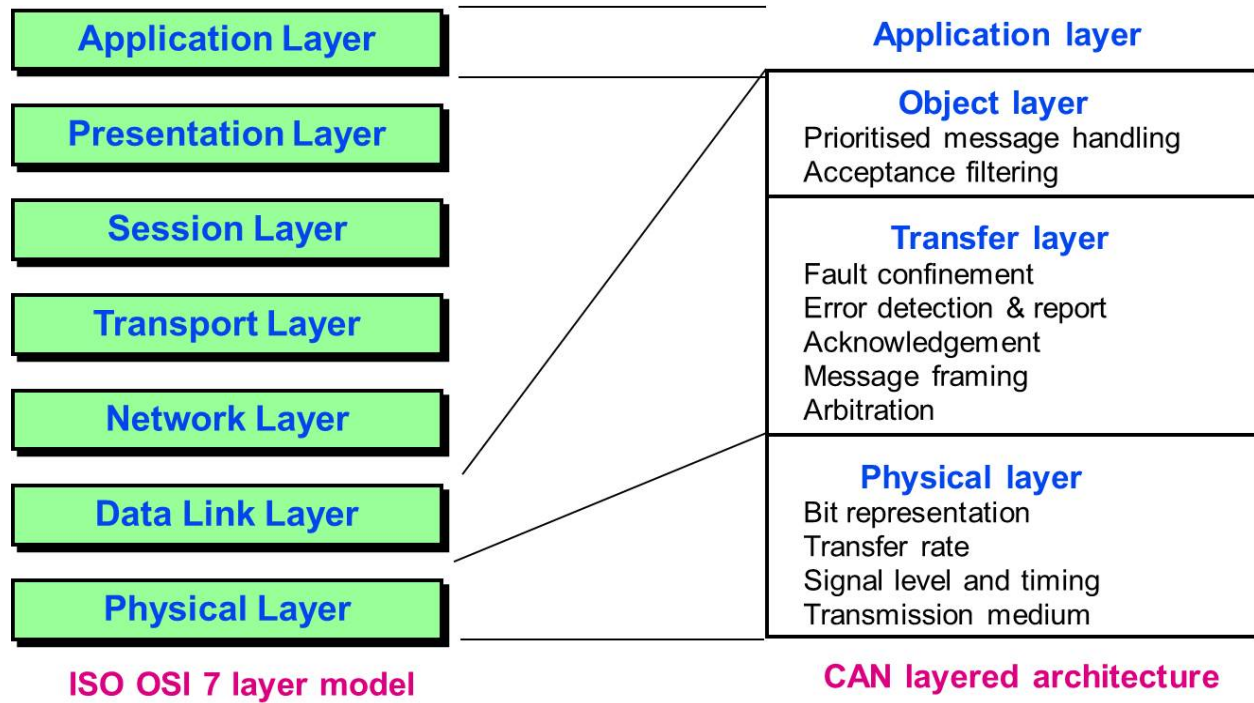
CAN Layered Architecture

[Introduction to CAN](#) > CAN Layered Architecture

The CAN communications protocol specifies the method by which data is passed between communicating devices on a CAN bus. It conforms to ISO's Open System Interconnection (OSI) model, which is a seven-layer description of a telecommunications network standard.

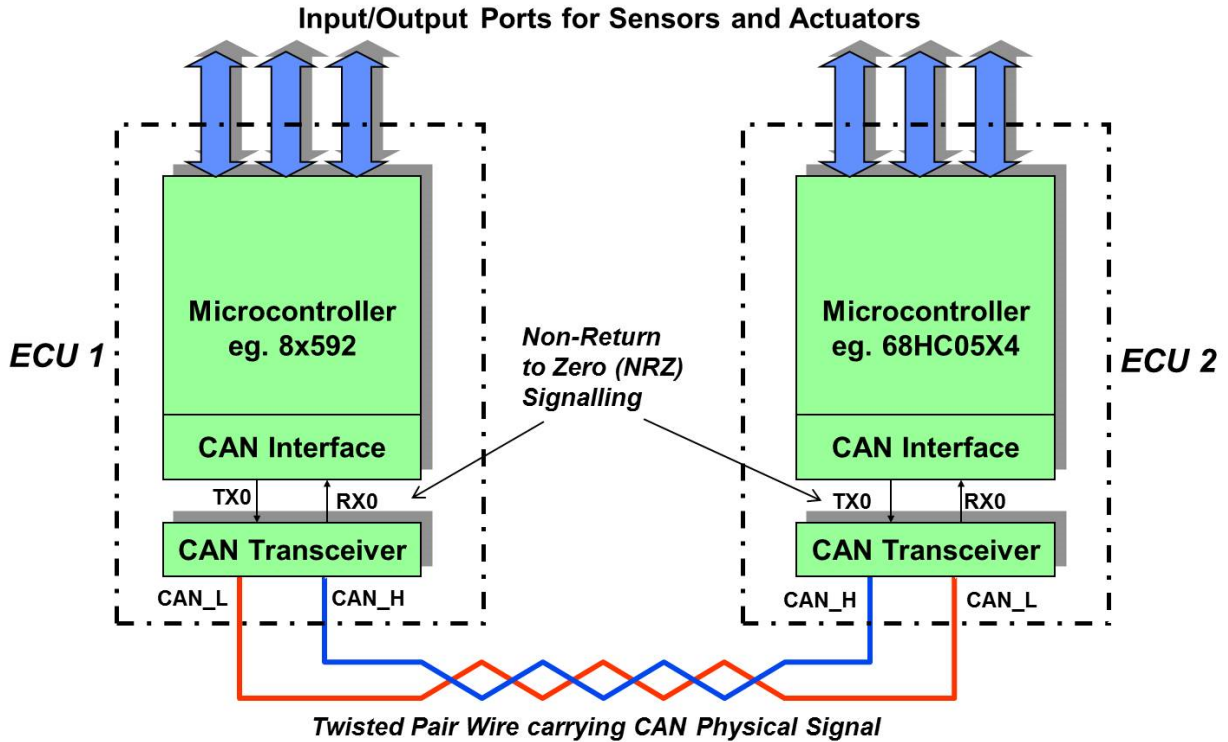
As with office networks, the modern vehicle utilises computer networks to interface the numerous ECUs distributed around the vehicle. Here we consider the model for all networks that was introduced by the ISO in very early days of computer networks. The model is a product of the Open Systems Interconnection (OSI) project at the International Organization for Standardization (ISO), and it is laid out in seven layers. Each of these layers are explained in ISO/IEC 7498-1 specification. It is the basis of all computer networks today. Here we simply show it as a comparison with the simplified CAN layered architecture. The layers of the OSI model are shown in the figure below, where it is compared with the simplified CAN layered model.

In fact the CAN protocol can be described by the lowest two layers of the OSI model – the Data Link Layer and the Physical Layer. The Application layer protocols can be proprietary schemes developed by individual CAN users, or one of the emerging standards used within particular industries. Common application layer standards used in the process-control/ manufacturing field are CANopen and DeviceNet, which are especially suited to the networking of PLCs and intelligent sensors and actuators. The SAE have introduced a standard for the Truck and Bus industries known as J1939. In the automotive industry most manufacturers use their own proprietary standard.



***The OSI 7- layer model compared with the simplified CAN layered model***

To relate this layered model to an ECU configuration in a vehicle, Figure 2 shows the basic configuration of an ECU. Here it can be seen that the Microcontroller contains the Application Layer. The CAN Interface contains the Data Link Layer and part of the Physical Layer. The CAN Transceiver and Twisted Pair wire form the rest of the Physical Layer.



The Table below briefly describes the 7 layers of the OSI model

7	Application Layer	The top layer. This is the layer that describes the software with which the user interacts with the network. For example Device Net.
6	Presentation Layer	Describes the syntax of data being transferred – for instance the transfer of floating point numbers between two systems using different math's formats.
5	Session Layer	Describes the handling of data sequences larger than the packets handled by lower layers.
4	Transport Layer	Describes the quality and nature of the data delivery between two communicating nodes. Handles issues like retransmission and error recovery.
3	Network Layer	Describes how a series of exchanges over various data links can deliver data between any two nodes in a network. Handles issues like routing and addressing.
2	Data Link Layer	Describes the arrangement and organisation of the data bits transmitted over a particular medium. For instance, this layer handles checksums and framing.
1	Physical Layer	Describes the physical characteristics of the communications media, as well as the electrical properties and interpretation of exchanged signals.

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## CAN Physical Layer

[Introduction to CAN](#) > CAN Physical Layer

Covered in this section will be the physical aspect of the CAN data transmission. These areas include:

- Wiring (Twisted Pair)
- Digital data format (NRZ)
- Digital signaling (differential voltage sensing)
- Bit Stuffing

### CAN Wiring

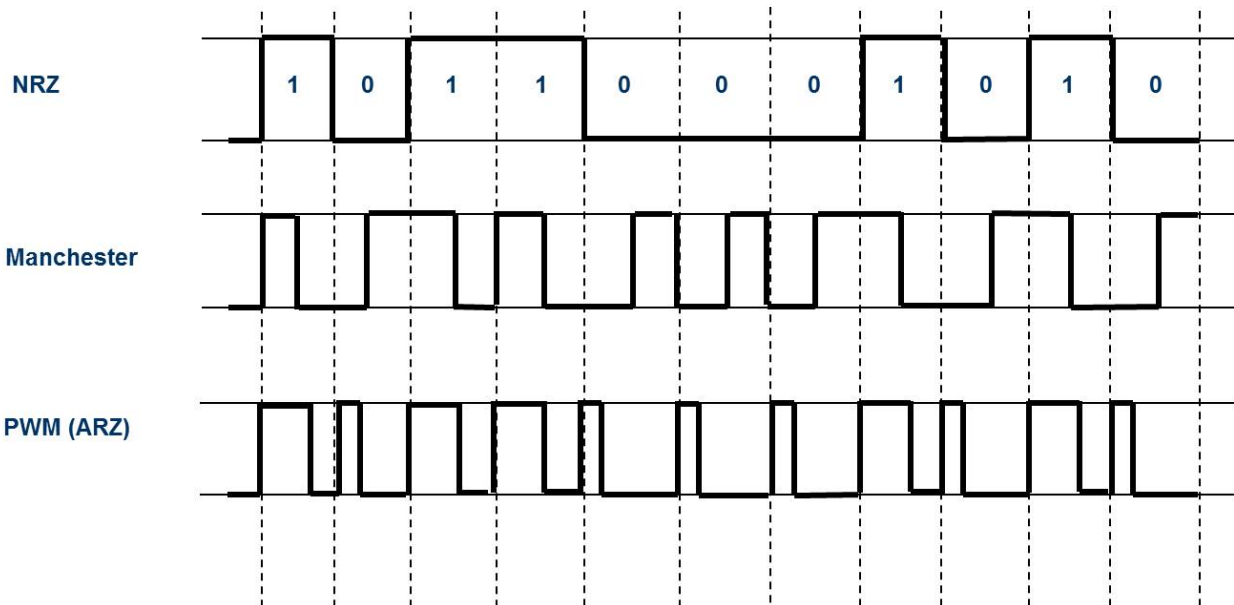
In most applications of CAN, the wiring used is an Unshielded Twisted Pair. In some exceptional cases, you may find a Shielded Twisted Pair. As the most common use is Unshielded Twisted Pair, expect to find this, particularly in automotive applications. The use of the twisted pair allows for use of differential voltage sensing. This will be covered below. There is a specification for single wire CAN, but this is only used for less time critical application, as it is data rate limited.

The CAN high-speed applications utilising the twisted pair allows the data rate to go up to 1Mbps. Most automotive powertrain systems operate at 500 Kbps. The single wire CAN systems are limited to 100 Kbps or below, depending on the CAN Transceiver manufacturer.

### CAN Data Format

The data format designated by the Bosch designers is what is known as NRZ – Non Return to Zero. As the name suggests, there is no transition between same polarity bits. Other data formats such as Manchester Encoding (used typically in Ethernet) or Pulse Width Modulation (used in lower speed control applications) have a transition at every bit. Figure 3 shows a comparison of these data formats. In most cases, the voltage level of bit level 1 is 5 volts, and bit level 0 is 0 volts.

## Digital Bit Representations



### **NRZ Data format compared to Manchester Encoding and PWM**

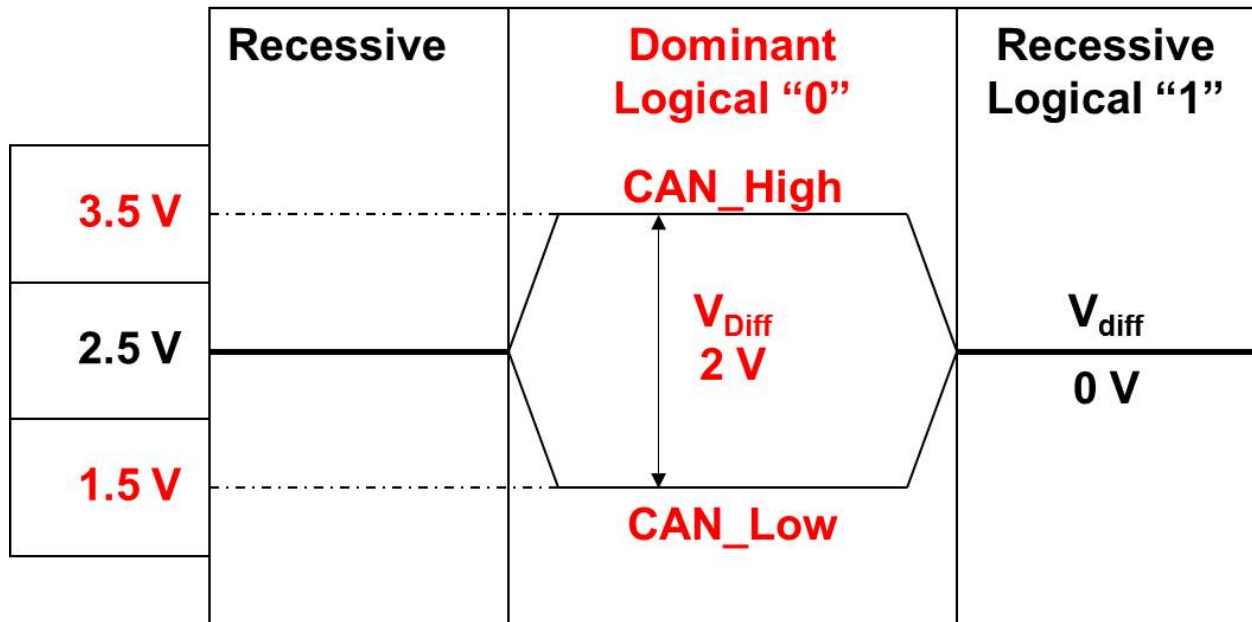
As seen in Manchester encoding and PWM, there is a transition for every bit. In Manchester encoding, a bit 1 is represented by a transition from 1 to 0, a bit 0 is represented by a transition from 0 to 1. In PWM (Always Return to Zero – ARZ) bit 1 is a wide pulse, and bit 0 is a narrow pulse. These formats use the transitions for each bit as a means for synchronisation.

NRZ does not necessarily have transitions for each bit. When there are successive bits of the same polarity, the level remains the same. There are two advantages using the NRZ format. One is that the less transitions, the less noise. Digital communications is inherently noisy as each transition generates noisy emissions. The other advantage is that the arbitration method of the CAN data access can only be accomplished with NRZ. The access process will be covered in a later chapter.

### CAN-bus Digital Signaling

After the CAN packet is passed from the CAN interface, the CAN transceiver (transmitter/receiver) converts it into a differential signal for transmission over the twisted pair. Most automotive and vehicle industries utilise the physical differential signaling specified in ISO 11898-2 – High speed CAN up to 1Mbps. Referring to Figure 2, the digital data will be a standard 0 to 5 volt transition (as shown in Figure 3) from the CAN controller to the CAN transceiver. The CAN transceiver converts this data into a differential pair to be transmitted on the twisted pair wire. The signals transmitted on the twisted pair are designated as CAN High (CAN\_H) and CAN Low (CAN\_L). They both represent the same signal, but are used in a differential sensing manner for use in an electrically noisy environment.

## ISO 11898-2 CAN High Speed

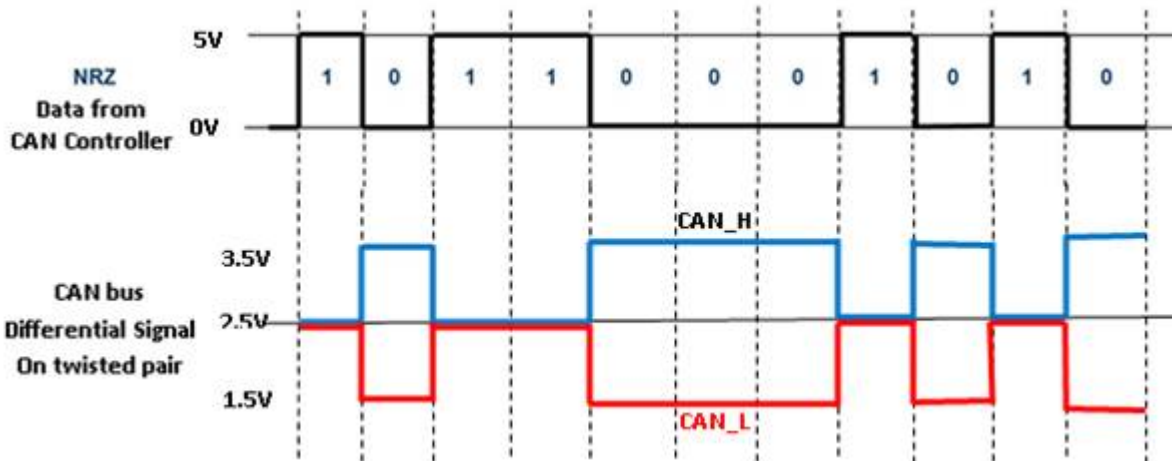


### ***CAN Output signal from ISO 11898-2 CAN High Speed Transceiver***

In CAN, the terminology of the bit value is Dominant for a bit 0 and Recessive for a bit 1. A Dominant will always over-ride a Recessive. This is relevant when we discuss arbitration later. Bus Idle is Recessive.

The signal representation in Figure 4 shows that during Data 1 (Recessive) both CAN\_H and CAN\_L are driven at 2.5 volts, causing a V difference of 0 volts. During Data 0 (Dominant) CAN\_H increases to 3.5 volts while CAN\_L decreases to 1.5 volts, causing a V difference of 2 volts. This is considered a very simple but effective means of operating in an electrically noisy environment. The differential voltage sensing allows the receiver just sense a voltage differential even there is a lot of electrical noise on the signal.

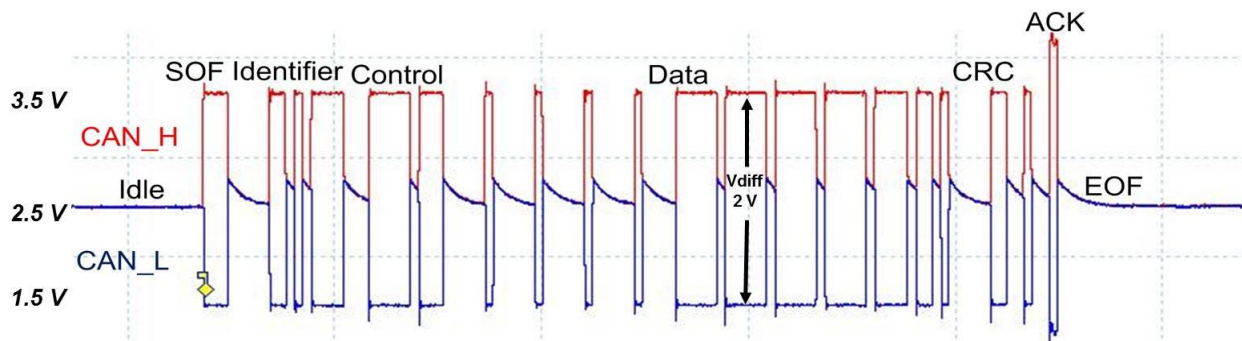
Comparing the data stream in Figure 3, the resulting output from the CAN Transceiver will appear as shown in Figure 5.



### CAN Differential Physical Signal

Therefore, at the receiver side of the Transceiver, it is interpreting a V diff of 0V as a logical 1 (Recessive), and a V diff of 2V as a logical 0 (Dominant).

A typical display of a CAN frame is show in Figure 6. This is a zoomed in view on a 2 channel scope (Channel 1 is CAN\_H, and Channel 2 is CAN\_L). The particulars of the sections of the CAN frame will be explained in a later section.



### CAN Waveform on Oscilloscope

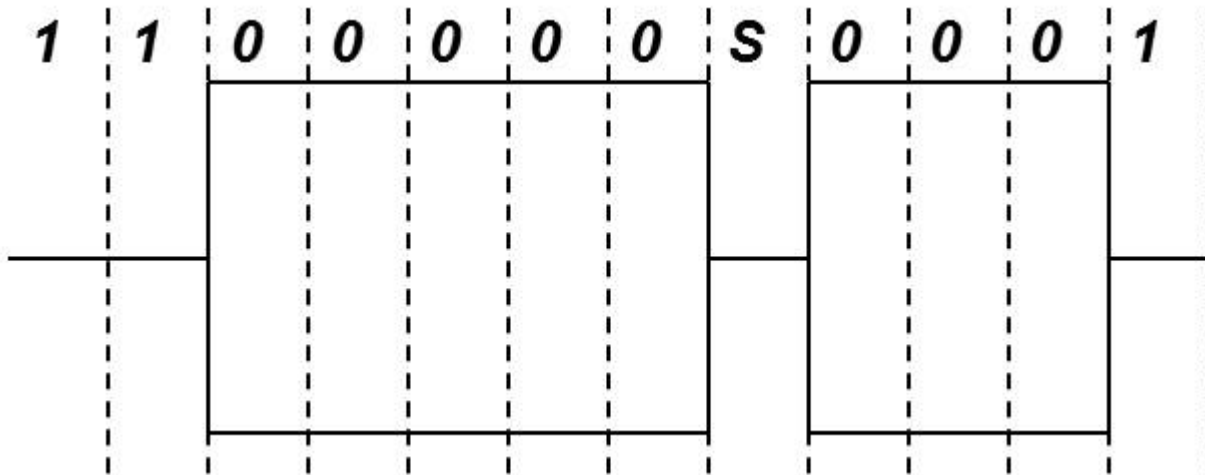
#### Bit Stuffing

NRZ the ideal data format for two reasons:

- Less transition means less noise - Digital communication is inherently noisy with hard transitions between values (usually 0 and 5 volts). These transitions emit noisy harmonics and can cause interference to other electronic components.
- Suitable for Bitwise Arbitration - The Bitwise Arbitration utilised by CAN can only be accomplished using NRZ. This is covered more in a later section.

The main disadvantage of NRZ is its non-determinant synchronisation. If there is a long string of bits of the same polarity, it becomes difficult to maintain accurate bit sampling. Because there is no transition between bits, synchronisation becomes an issue. It is possible to incorrectly interpret a bit polarity if the data stream is too long. The solution to this is Bit Stuffing.

In CAN, Bit Stuffing occurs when there is 5 or more bits of the same polarity. After 5 bits of the same polarity, the CAN protocol automatically inserts an opposite bit to allow synchronisation. At the receiver, the node automatically strips out the stuff bit. An example of this is shown in Figure 7 below.



### **Bit Stuffing Example**

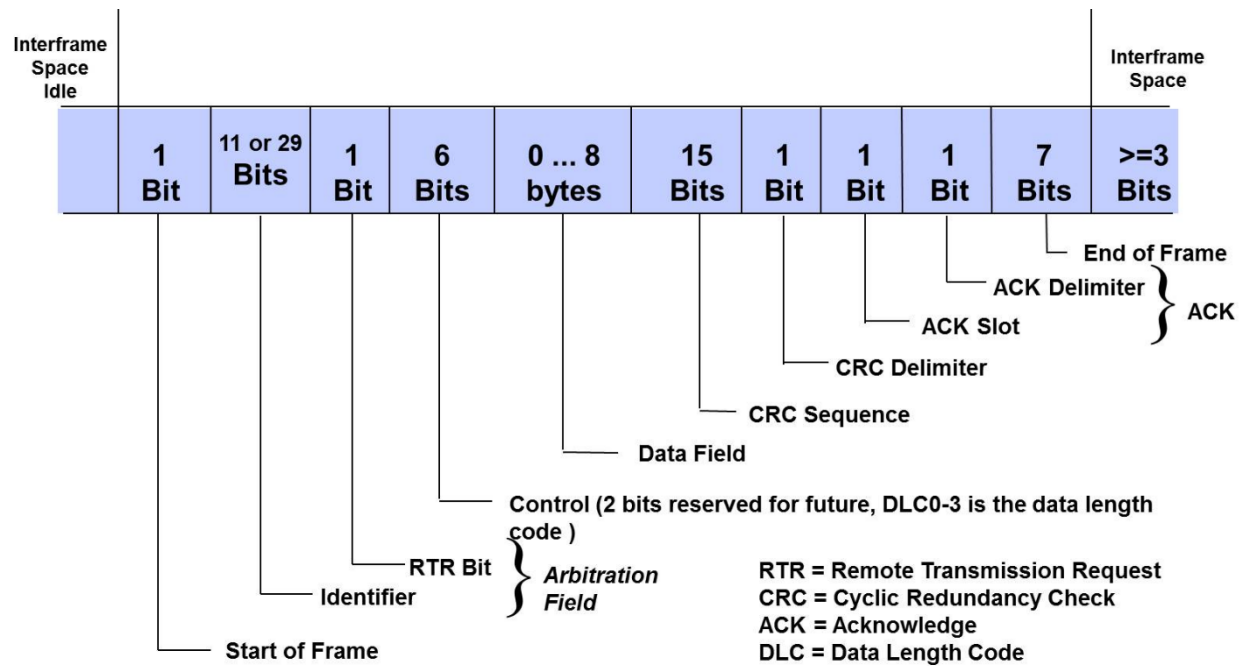
In this example, the data stream is 1100000001. There are 8 Dominants (0s) in a row. The CAN transmitter will automatically insert an opposite polarity Recessive (1) after the 5th 0 to allow for synchronisation. The receiver will automatically extract the Stuff bit at the CAN controller.

This process is suspended after the ACK bit and during idle.

When a microcontroller formats information for sharing with other ECUs, it will pass the data to the CAN chip for framing the relevant information to send onto the other ECUs. Here we will look at how the CAN interface puts the information into a frame to transmit over the CAN bus.



The CAN message format simplistically appears as the block diagram in the figure below:



### **CAN Message frame format**

The **Start of Frame (SOF)** is a dominant 0 to tell all the other ECUs that a message is on the way.

Following this is the **Identifier field (ID)**. This is usually a functional address (e.g. Engine parameters, Wheel speeds, etc.). In some applications, the ID can contain source and destination addresses. The value of the Identifier determines the message priority. The lower the value, the higher the priority.

The Identifier value can be 11 bits or 29 bits. This is known as Standard CAN and Extended CAN. The original CAN specification designated 11 bit identifiers. In 1991, the CAN specification was updated to include this extended CAN format. Looking at the possible number of identifiers, 2 to the power of 11 = 2048 possible IDs. 2 to the power of 29 allows for more than 500 million possible IDs. The reason for the introduction of the extended frame format was not so much to increase the number of IDs, but to adapt CAN to the requirements of the American car-manufacturers. With the 29 bit identifier a message strategy can be realised, which is similar to the J1850-protocol published by the American Society of Automotive Engineers (SAE).

Many applications implement either. The SAE J1939 standard for Trucks and Buses dictates that the extended format is to be used. This is not so much to allow for more Identifiers, but to allow for versatile types of messaging. This will be covered in a later section.

The **Remote Transfer Request (RTR)** bit allows an ECU to request an ID from another ECU that has not been transferred within a specified time period. When an RTR message is sent, this bit is set to Recessive (Data 1). During normal messages, the RTR bit is always set to Dominant (Data 0).

The ID and RTR are considered to be known as the Arbitration Field, as this determines the message priority during a message arbitration. This occurs when two or more messages collide during bus access.

The **Control Field** contains the **Data Length Code (DLC)**. The DLC simply informs of the length of the data field. This can be a value between 0 and 8 Bytes. It is always in Bytes. Even if the node wants to send just one bit, a Byte must be assigned. There can be 0 Byte frames. The RTR message is a 0 Byte message. In some protocols, a 0 Byte message can be used as health/heartbeat message.

The **Data Field** contains the information to be relayed to other ECUs, e.g. the Engine Parameters message may contain info such as Engine speed, Water & Oil Temp, etc. The format of the data and how it is interpreted is covered in a later section.

The **Cyclic Redundancy Check (CRC)** is a 15 bit sequence used for checking the integrity of the data. Sometimes known as Check Sum, it is a common error checking method in digital communication. The way it works is that the transmitting node will generate a 15 bit digital sequence based on the prior data (ID, Control, and Data fields). Once the frame is brought into the receiver node, that node will generate its own CRC sequence based on the received info (ID, Control, and Data fields).

The receiver's CRC is then compared with the transmitted CRC. If there is a match, the data is deemed OK, and the Acknowledge is generated. If they do not match, there has been corruption of the data, and the receiver node will generate an Error Frame (to be covered later).

The **Acknowledge (ACK)** slot is a single bit generated by all receiving CAN node to indicate that their CRC processes are OK. The transmitter node leaves this bit recessive (Data 1), and the receiver nodes all insert a dominant bit (Data 0) in this slot. This indicates that all receivers are OK with the integrity of the CAN frame data. If there was a problem within the CRC process, a node will generate an Error Frame, negating any ACK process (covered later).

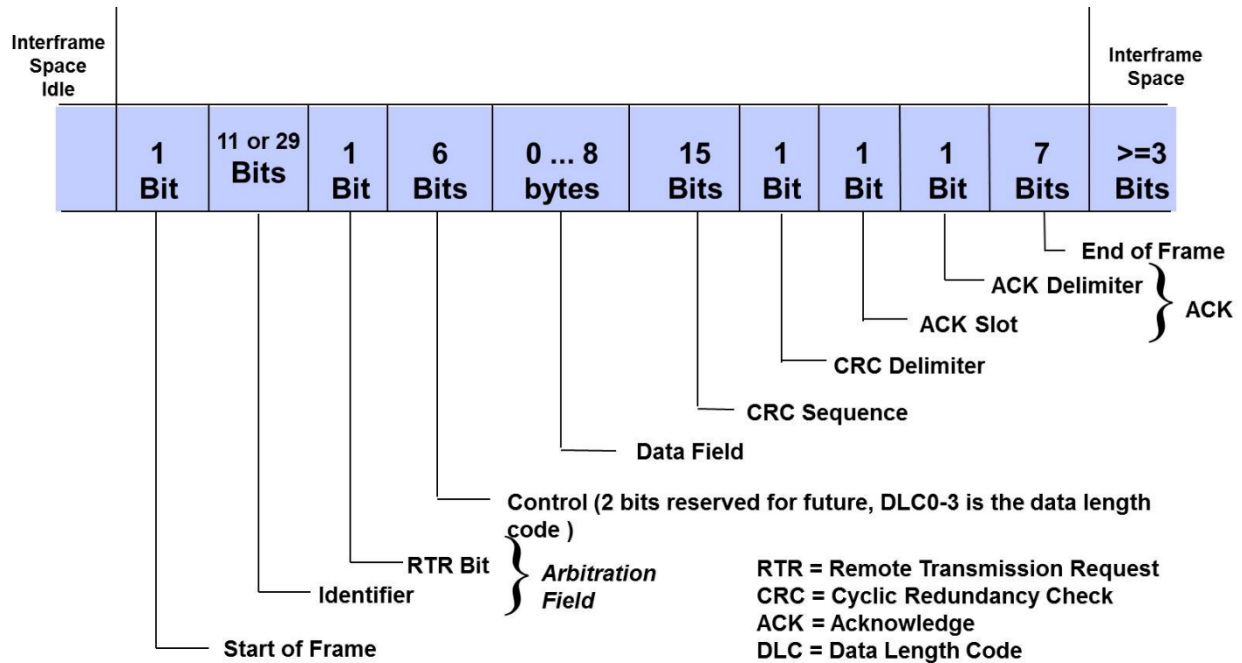
**Delimiters.** Note that the CRC and the ACK fields have a one bit Delimiters. These are just to allow a space for the next process.

**End-of-Frame (EOF).** EOF is a seven bit recessive sequence at the end of the CAN frame. This, along with the minimum of 3 bits of Interframe Space, give enough time delay to allow for the next access of CAN messages. If there are back-to-back CAN messages, you will observe a minimum of 11 bits of Recessive before the next CAN frame. These 11 bits is comprised of ACK delimiter, EOF and 3 bits Interframe space.

## Introduction to CAN &gt; CAN Frame Message Format

When a microcontroller formats information for sharing with other ECUs, it will pass the data to the CAN chip for framing the relevant information to send onto the other ECUs. Here we will look at how the CAN interface puts the information into a frame to transmit over the CAN bus.

The CAN message format simplistically appears as the block diagram in the figure below:

**CAN Message frame format**

The **Start of Frame (SOF)** is a dominant 0 to tell all the other ECUs that a message is on the way.

Following this is the **Identifier field (ID)**. This is usually a functional address (e.g. Engine parameters, Wheel speeds, etc.). In some applications, the ID can contain source and destination addresses. The value of the Identifier determines the message priority. The lower the value, the higher the priority.

The Identifier value can be 11 bits or 29 bits. This is known as Standard CAN and Extended CAN. The original CAN specification designated 11 bit identifiers. In 1991, the CAN specification was updated to include this extended CAN format. Looking at the possible number of identifiers, 2 to the power of 11 = 2048 possible IDs. 2 to the power of 29 allows for more than 500 million possible IDs. The reason for the introduction of the extended frame format was not so much to increase the number of IDs, but to adapt CAN to the requirements of the American car-manufacturers. With the 29 bit identifier a message strategy can be realised, which is similar to the J1850-protocol published by the American Society of Automotive Engineers (SAE).

Many applications implement either. The SAE J1939 standard for Trucks and Buses dictates that the extended format is to be used. This is not so much to allow for more Identifiers, but to allow for versatile types of messaging. This will be covered in a later section.

The **Remote Transfer Request (RTR)** bit allows an ECU to request an ID from another ECU that has not been transferred within a specified time period. When an RTR message is sent, this bit is set to Recessive (Data 1). During normal messages, the RTR bit is always set to Dominant (Data 0).

The ID and RTR are considered to be known as the Arbitration Field, as this determines the message priority during a message arbitration. This occurs when two or more messages collide during bus access.

The **Control Field** contains the **Data Length Code (DLC)**. The DLC simply informs of the length of the data field. This can be a value between 0 and 8 Bytes. It is always in Bytes. Even if the node wants to send just one bit, a Byte must be assigned. There can be 0 Byte frames. The RTR message is a 0 Byte message. In some protocols, a 0 Byte message can be used as health/heartbeat message.

The **Data Field** contains the information to be relayed to other ECUs, e.g. the Engine Parameters message may contain info such as Engine speed, Water & Oil Temp, etc. The format of the data and how it is interpreted is covered in a later section.

The **Cyclic Redundancy Check (CRC)** is a 15 bit sequence used for checking the integrity of the data. Sometimes known as Check Sum, it is a common error checking method in digital communication. The way it works is that the transmitting node will generate a 15 bit digital sequence based on the prior data (ID, Control, and Data fields). Once the frame is brought into the receiver node, that node will generate its own CRC sequence based on the received info (ID, Control, and Data fields).

The receiver's CRC is then compared with the transmitted CRC. If there is a match, the data is deemed OK, and the Acknowledge is generated. If they do not match, there has been corruption of the data, and the receiver node will generate an Error Frame (to be covered later).

The **Acknowledge (ACK)** slot is a single bit generated by all receiving CAN node to indicate that their CRC processes are OK. The transmitter node leaves this bit recessive (Data 1), and the receiver nodes all insert a dominant bit (Data 0) in this slot. This indicates that all receivers are OK with the integrity of the CAN frame data. If there was a problem within the CRC process, a node will generate an Error Frame, negating any ACK process (covered later).

**Delimiters.** Note that the CRC and the ACK fields have a one bit Delimiters. These are just to allow a space for the next process.

**End-of-Frame (EOF).** EOF is a seven bit recessive sequence at the end of the CAN frame. This, along with the minimum of 3 bits of Interframe Space, give enough time delay to allow for the next access of CAN messages. If there are back-to-back CAN messages, you will observe a minimum of 11 bits of Recessive before the next CAN frame. These 11 bits is comprised of ACK delimiter, EOF and 3 bits Interframe space.

## CAN Higher Layer Protocols

### [Introduction to CAN](#) > CAN Higher Layer Protocols

A CAN higher level protocol (also known as the Application layer protocol) is a protocol implemented 'on top' of the existing lower-level CAN layers (physical layer and data-link layer). The higher level protocol uses CAN's physical and data link layers as a base for the developed application layer. Many systems, (e.g. in the automotive industry) use a propriety application layer, but in many industries, this approach is not cost-effective. Several organisations have developed standardised open application layers to ensure ease of system integration.

Some available CAN higher level protocols are:

- DeviceNet by the Open DeviceNet Vendors Association.
- Smart Distributed System (SDS) by Honeywell.
- CANOpen (subset of CAL) by CiA
- SAE J1939

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Introduction to LIN

LIN (Local Interconnect Network) is a low cost, serial communication system for distributed electronic systems in vehicles.

It is aimed to complement existing automotive multiplex networks, such as CAN. The specification covers in addition to the definition of the protocol and the physical layer also the definition of interfaces for development tools and application software. LIN enables a cost-effective communication for smart sensors and actuators where the bandwidth and versatility of CAN is not required.

The communication is based on the SCI (UART) data format, a single-master/multiple-slave concept, a single-wire 12V bus, and a clock synchronisation for nodes without stabilised time base.

The LIN Consortium has been developed to standardise a concept of a serial low cost communication concept in conjunction with a development environment, that enables the car manufacturers and their suppliers to create, implement, and handle complex hierarchical multiplex systems in a very cost competitive way.

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## The LIN Communication Concept

[Introduction to LIN](#) > The LIN Communication Concept

A LIN network comprises one master node and one or more slave nodes.

All nodes include a slave communication task that is split in a transmit and a receive task, while the master node includes an additional master transmit task.

The communication in an active LIN network is always initiated by the master task. The master sends out a message header which comprises of the synchronisation break, the synchronisation byte, and the message identifier.

Exactly one slave task is activated upon reception and filtering of the identifier and starts the transmission of the message response.

The response comprises of up to eight data bytes and one checksum byte. The header and the response part form one message frame.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

X-Analyser User Manual

## Key Features of LIN

[Introduction to LIN](#) > Key Features of LIN

- Low cost single-wire implementation based on enhanced ISO 9141
- Speed up to 20Kbit/s (limited for EMI-reasons)
- Single Master / Multiple Slave Concept therefore no arbitration necessary
- Low cost silicon implementation based on common UART interface hardware which means that almost any microcontroller has necessary hardware on chip

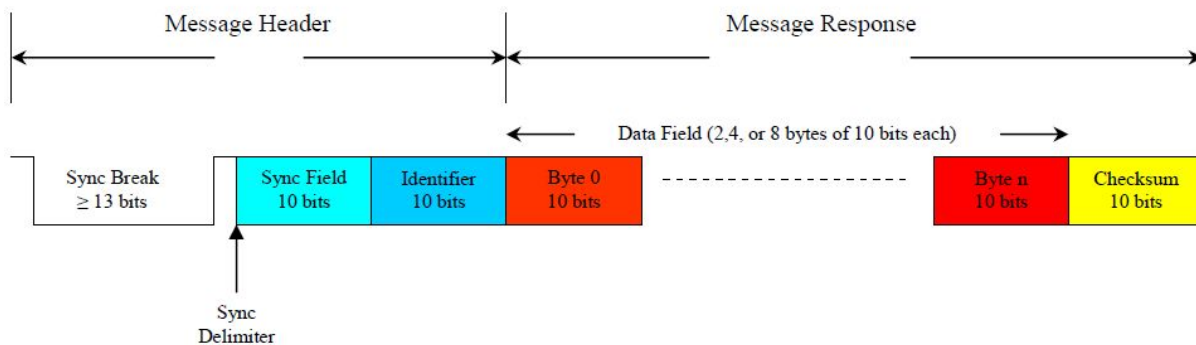
- Self synchronisation in the slave nodes without crystal or ceramics resonator which leads to significant cost reduction of slave hardware
- Guaranteed latency times for signal transmission therefore predictable systems are possible
- Nodes can be added to the LIN network without requiring hardware or software changes in other slave nodes.
- The size of a LIN network is typically under 12 nodes (though not restricted to this), resulting from the small number of 64 identifier and the relatively low transmission speed.

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Inside the LIN Frame

[Introduction to LIN](#) > Inside the LIN Frame

The contents of the LIN message frame are shown below:



The identifier of a message denotes the content of a message but not the destination. This communication concept enables the exchange of data in various ways: from the master node (using its slave task) to one or more slave nodes, and from one slave node to the master node and/or other slave nodes. It is possible to communicate signals directly from slave to slave without the need for routing through the master node, or broadcasting messages from the master to all nodes in a network. The sequence of message frames is controlled by the master and may form cycles.

### Sync Break

The Sync Break signifies the beginning of a LIN frame and is always generated by the Master Task. The Sync Break tells the Slave Task to prepare for the Sync Field. The Sync Break consists of two different

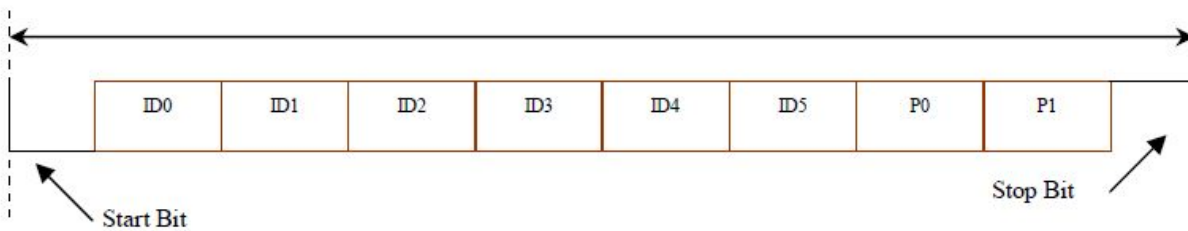
parts; a dominant (low) level that should be a minimum of 13 bit-times (Tbit) which is followed by a recessive (high) period that should be in the range one to four Tbit (Sync Delimiter). The length of the Sync Field has been chosen to be at least 13 Tbit so that LIN Slave Tasks can distinguish between a valid Sync Break and the maximum possible allowed sequence of dominant bit within a data frame (i.e. 9 Tbit).

### Sync Field

The Sync Field contains the signalling required for the slaves to synchronise with the Master's clock. The Sync Field is a byte containing the data "0x55" giving a waveform with five falling edges and five rising edges. These edges can be used during synchronisation to tune the slave node transmission and reception speed to match the Master node. Therefore the bit time is found by measuring the time of either 5 rising edges or 5 falling edges and dividing by eight (logical shift by 3 bits).

### Identifier Field

The Identifier Field contains information about the contents and length of a message. This field is divided into three sections as shown below; identifier bits (four bits), length control bits (two bits) and parity bits (two bits).




The Identifier bits (ID0 to ID3) represent the identifier of the node who is to respond in the Message Response part of the frame. Thus there can be up to 16 nodes on a LIN subnet; one master and up to 15 slaves. The Identifier Field therefore describes the content of the message and not the destination.

ID5	ID4	NData (no. of bytes)
0	0	2
0	1	2
1	0	4
1	1	8

Number Bytes in Data Field as Dictated by Length Control.



The final two bytes in the Identifier Field are parity bits which are defined by a mixed parity algorithm. This ensures that the Identifier Field will never consist of an all Dominant or all Recessive bit pattern. It must be noted that the parity check only detects errors, it does not correct them.

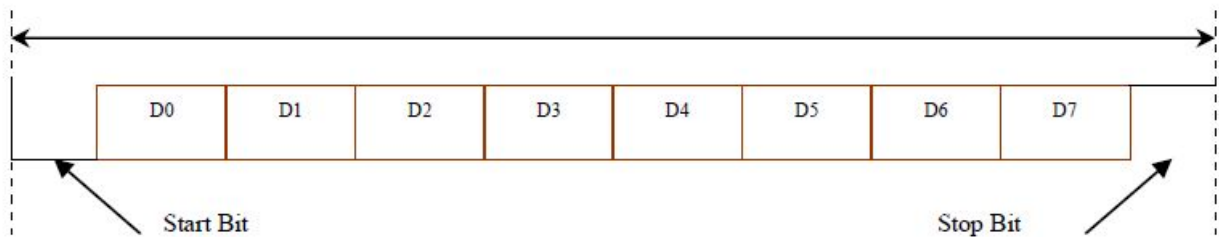
 The parity check bits are calculated by the following mixed parity algorithm:

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$$

$$P1 = ID1 \oplus ID3 \oplus ID4 \oplus ID5$$

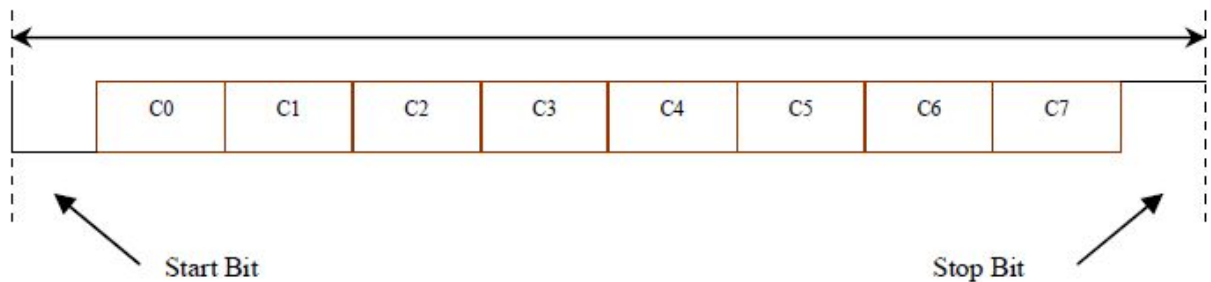
### Data Field

The Data Field contains two, four or eight bytes of data, each consisting of one stop bit, eight data bits and one stop bit. Transmission is done with the LSB first. The Data Field is written by the responding slave task. Since there is no bus arbitration, only one slave task should be allowed to respond to each identifier. All other slave tasks are limited to reading the response and act accordingly. The format of a single byte within the Data Field is shown below.



### Checksum Field

The last field in the Message Frame is the Checksum Field. This byte contains the inverted modulo-256 sum of all data bytes within the Data Field. The sum is calculated by doing an "Add with carry" on all data bytes and then inverting the answer. The properties of the inverted modulo-256 sum are such that if this number is added to the sum of all data bytes, the result will be "0xFF". The format of the Checksum Field is shown below.



## Applications for LIN

[Introduction to LIN](#) > Applications for LIN

Typical applications for the LIN bus are assembly units such as doors, steering wheel, seats, climate regulation, lighting, rain sensor, or alternator. In these units the cost sensitive nature of LIN enables the introduction of mechatronic elements such as smart sensors, actuators, or illumination. They can be easily connected to the car network and become accessible to all types of diagnostics and services.

Under LIN implementations, commonly used analogue coding of signals will be replaced by digital signals, leading to an optimised wiring harness.

In automotive application the following are ideal for LIN implementation:

### **Vehicle Roof**

- Rain Sensor
- Light Sensor
- Light Control
- Sun Roof

### **Vehicle Doors**

- Mirror
- Central Locking
- Mirror Switch
- Window Lift

### **Engine**

- Sensors
- Small Motors

### **Steering Wheel**

- Cruise Control Switches
- Wiper
- Turn Signal
- Radio
- Climate Control

## Seat

- Seat Position Motors
- Occupancy Sensor

Although LIN was originally designed for automotive applications, it has also received interest as a sensor bus for industrial automation and even consumer electronics.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019

## Frequently Asked Questions

X-Analyser User Manual

FAQ: What is the format of the timestamp?

Frequently Asked Questions > FAQ: What is the format of the timestamp?

An example of the timestamp reading from X-Analyser is shown below:

Timestamp
00:04:56.0106331
00:04:56.5109652

The format of this timestamp reading is:

Hours: Minutes: Seconds

Therefore the top reading in the image above is saying 0 Hours, 4 Minutes, 56 seconds, 10633.1 microseconds.

The resolution of this timestamp reading is 0.1 microseconds (or 100 nanoseconds). The accuracy of the timestamp reading is dependent upon the stated accuracy of the CAN or LIN interface manufacturer.

---

Copyright Warwick Control Technologies, © 2019. Last Updated: 20 July 2019