IOM6 CAN/LIN Use cases

Publish Date: 09/02/2021

Table of Contents

1 Introduction

Having a dedicated Network Analyzer synchronized with the program trace can be very useful, especially when trying to debug a large number of interconnected devices (like ECUs in a car).

This application note describes a few different examples of how an IOM6 CAN/LIN can be used to ease the development and the debugging of connected devices that are using either a CAN or a LIN network:

- Network traffic sniffing
- Timing Analysis
- Message injection
- CAN/LIN message as trigger for trace recording
- isystem.connect SDKs

IOM6 CAN/LIN module can be used to:

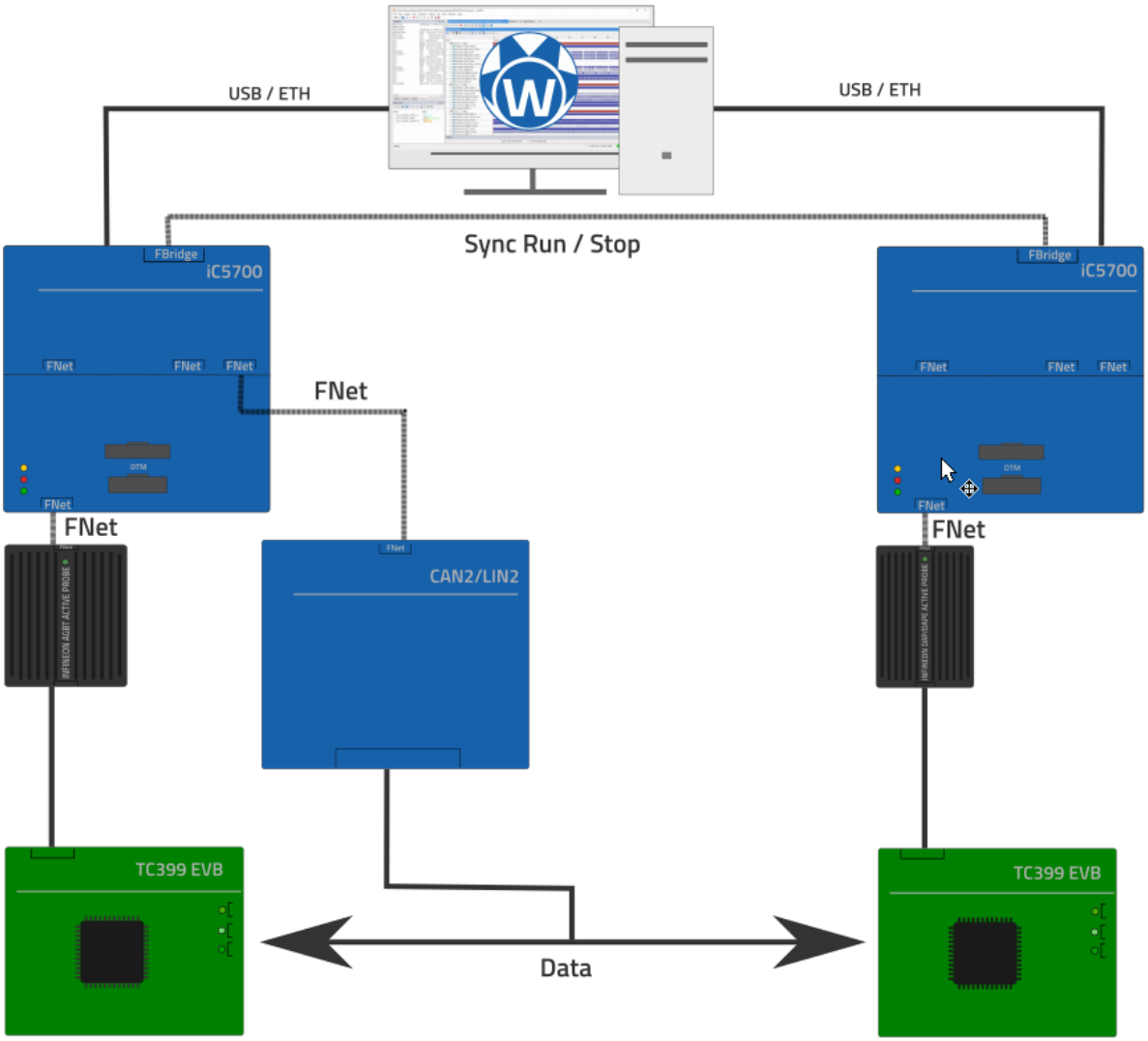
- Track system bus activity
- Measure worst case time analysis
- Measure system end-to-end response times
- Inject CAN messages
- Trigger trace recording

A block diagram of hardware that will be used when working on practical examples is shown below.

Your configuration will depend on your needs, but note that at least one BlueBox with IOM6 CAN/LIN module connected to its FNet will be needed to replicate all of the examples described below.

Below setup consists of:

- PC with winIDEA
- Two iC5700 BlueBox with IOM Hub
- IOM CAN/LIN module
- Infineon AGBT Active Probe
- Infineon DAP/DAPE Active Probe
- Two TC399 Evaluation Boards (EVB)



Note that IOM6 CAN/LIN module doesn’t send an ACK (Acknowledgement) after detecting a message. This means that you should use an already *working* bus (two or more nodes connected with ACK response) otherwise an ACK error will be detected and the message will be retransmitted indefinitely by the sender.

Do note as well that the CAN bus can be traced as a separate entity even if the microcontroller does not have any trace capabilities but in this case a synchronous trace won't be possible.

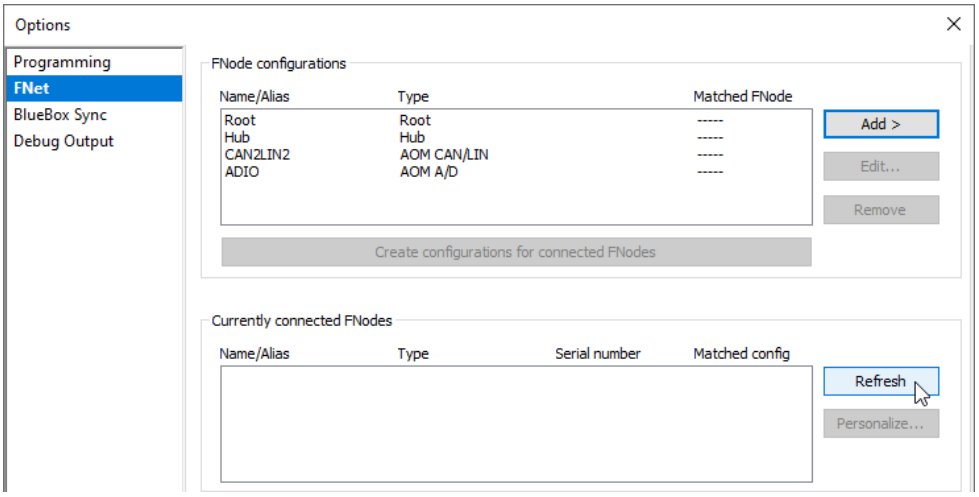
2 IOM6 CAN/LIN Configuration

Before you can start using the IOM6 CAN/LIN module to record network data its global configuration has to be set.

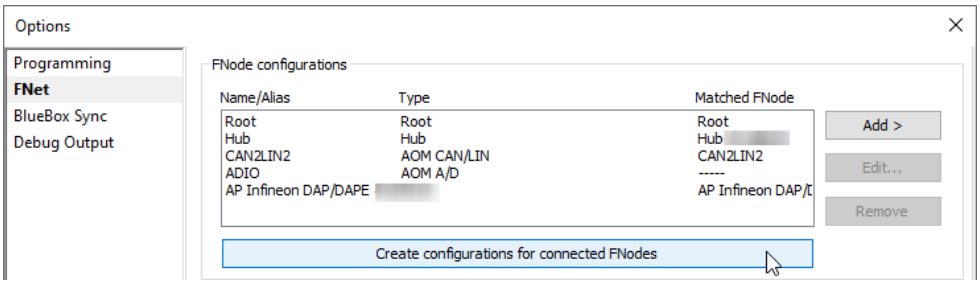
1 Create a CAN/LIN FNode.

i. Open *Hardware menu / Options / FNet*.

ii. Press *Refresh*, if your CAN/LIN device is not listed in *Currently connected FNodes*, and wait for winIDEA to list it.



iii. Press *Create configurations for connected FNodes* to create FNode configurations.

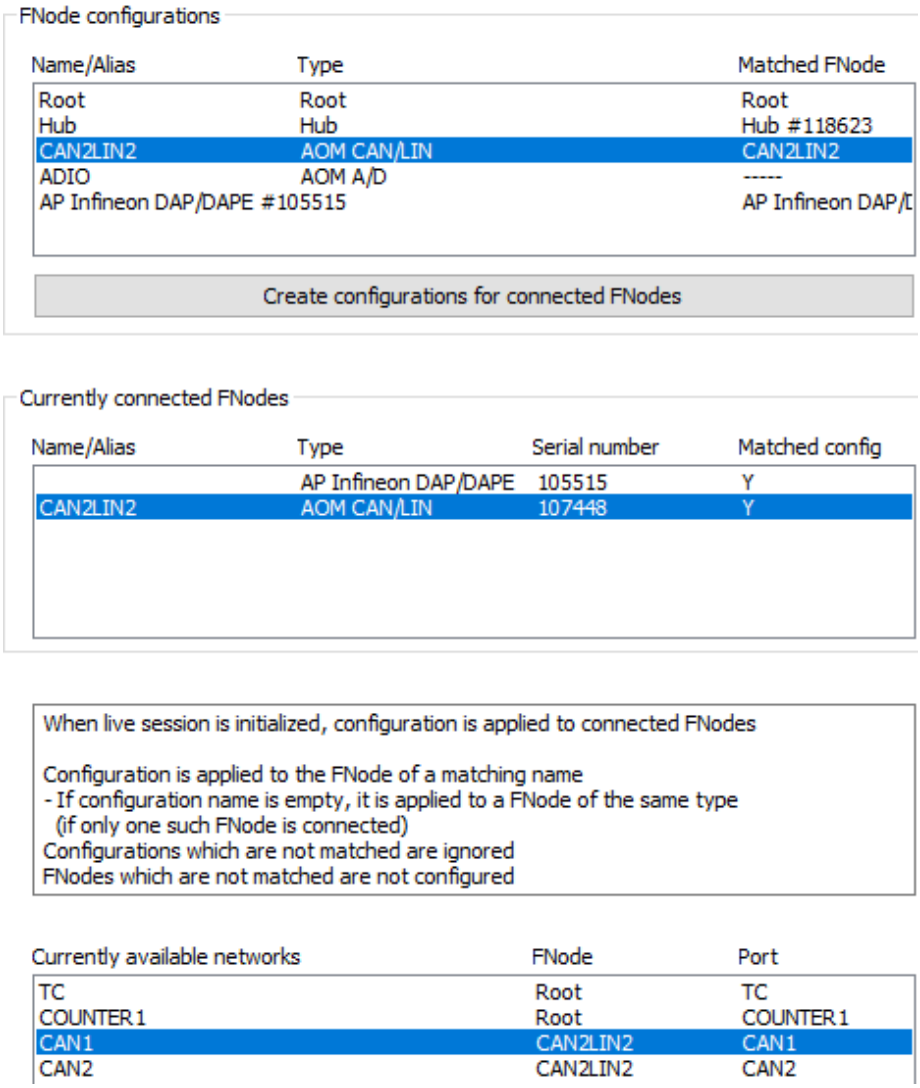


iv. Make sure that CAN/LIN device is found. What to look for:

a. Listed CAN/LIN device with under *Matched FNode*.

b. Y (for yes) under *Matched config.*

c. List of *Currently available networks.*

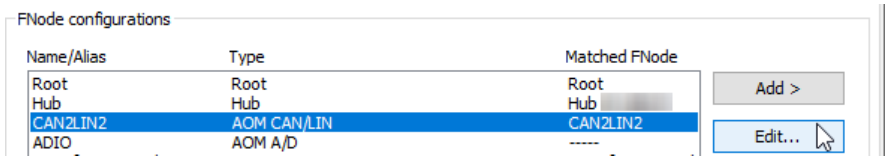


➢ Having issues? Please check chapter Network description in winIDEA Help.

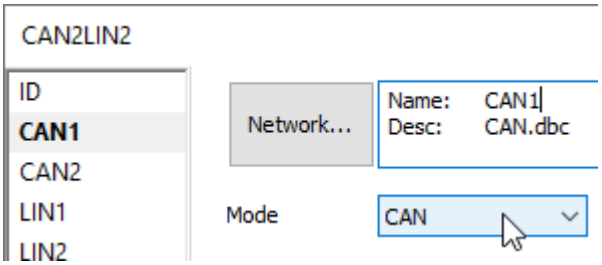
2 Edit the settings of the CAN/LIN module.

In this example a CAN1 channel with classic CAN protocol and a baudrate of 500 kbps will be observed.

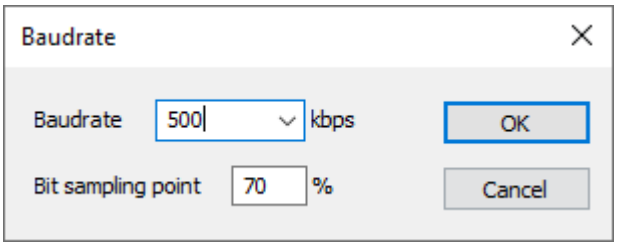
i. Select *AOM CAN/LIN* and press *Edit* to open the *CAN/LIN* dialog.



ii. Open CAN1 page and select *CAN* from the *Mode* drop-down.



iii. Press *Arbitration Phase* to adjust the *Baudrate* to 500 kbs.



iv. Check option *Allow message injection*, if you wish to inject messages on the bus.

Now the general configuration is set and you can continue with the following use cases.

3 Network traffic sniffing

One of the simplest use cases for an IOM6 CAN/LIN module is a so-called network traffic sniffing.

The IOM6 CAN/LIN is connected to the network and it simply listens and records all of the ongoing traffic, i.e., network data on the CAN1 channel of the IOM6 CAN/LIN as soon as a recording session starts. As such it doesn't provide a great insight into the working of the device, but can be a great starting point to check if everything is working as expected.

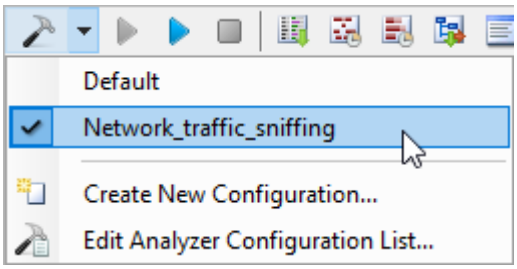
Basic usage of the CAN/LIN module typically consists of the following steps.

1 Create new (or modify the existing) Analyzer Configuration.

i. Open *View menu / Analyzer*.

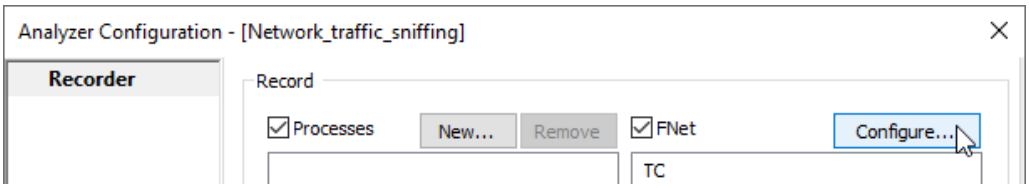
➢ How to create different Analyzer configurations?

ii. Press *Analyzer Configuration* button and select your configuration.

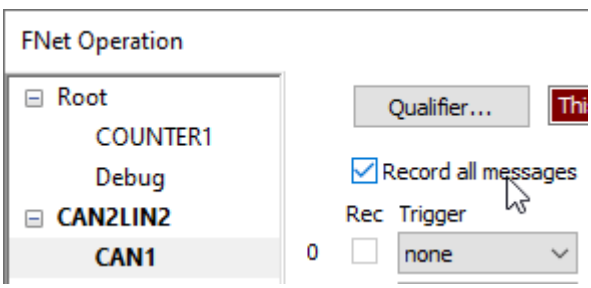


2 Configure desired bus.

i. Enable *FNet* check box.
ii. Press *Configure* to open the *FNet Operation* dialog.

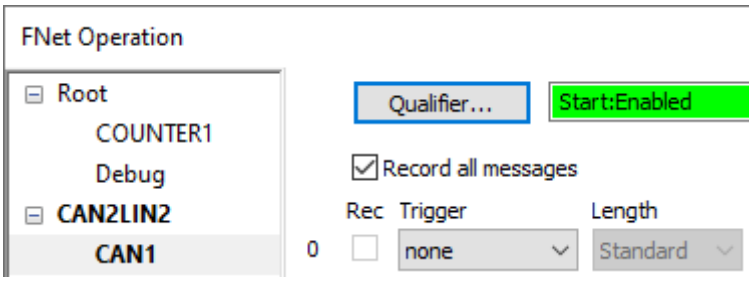


iii. Open *CAN1* page and make sure option *Record all messages* is checked.



iv. Press *Qualifier* and enable option *Recording is enabled from the start*.

CAN1 page is now set.

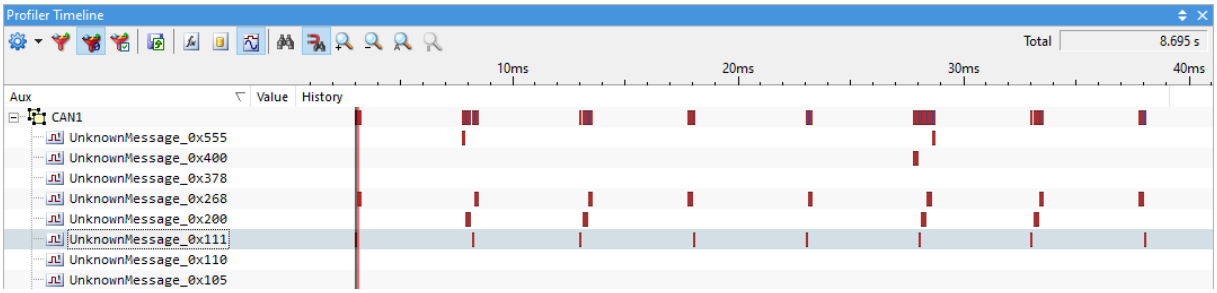


3 Start the Analyzer Session.

➢ How to start a Trace recording?

3.1 Results of the recording

After starting and stopping the Analyzer Session a list of recorded messages will be displayed under the Aux tab in the Profiler Timeline.



Information about messages is displayed similar to the Code and Data trace, with which you might already be familiar, including the more detailed properties and statistics of each message (e.g., its average period) in Profiler Statistics window which offers different types of statistics information: Interval Statistics, Area Statistics, Session Statistics.

- How to operate with Profiler Statistics windows?

3.2 Further configuration

Similar to this, other channels, like CAN2 or LIN1/2 can be configured. Furthermore, filtering messages and description files are two of the most commonly used features.

Note that there are a lot of additional settings that can be used, but are out of scope for this simple configuration. For example, you can start the recording on a specific trigger or even filter out the unwanted messages by their ID or Data. All of them are available under the configuration dialog of the CAN/LIN module.

3.2.1 Filtering messages

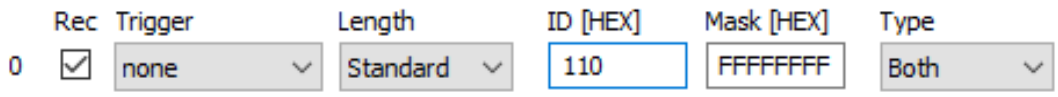
This feature is used when debugging a heavy traffic network. Below example shows how to configure a trigger with one message with a particular ID on the whole network.

To enable message filtering navigate to the *FNet Operation* dialog, select the appropriate channel and configure the filtering options. An example, where only the message with `ID 0x110` will be recorded, is shown below.

1. Open FNet Operation / CAN1 page.

2. Configure the Trigger 0.

   i. Disable *Record all messages* check box if enabled. The *Rec* check box is grayed disabled if *Record all* is checked.
   ii. Enter the ID.



3.2.2 Description files

winIDEA also supports usage of description files for decoding raw CAN/LIN bus data to physical values or human-readable form.

Today one of the most used description files and the standard for storing CAN bus decoding rules is a so-called DBC (Database Container) file.

When working on a large number of interconnected devices, the bus descriptions might be divided into multiple files for each individual device, so more than one file can be added to sufficiently describe all of the network frames. For LIN channels the most common description file is a LDF (Log Database) file, which is also supported in winIDEA.

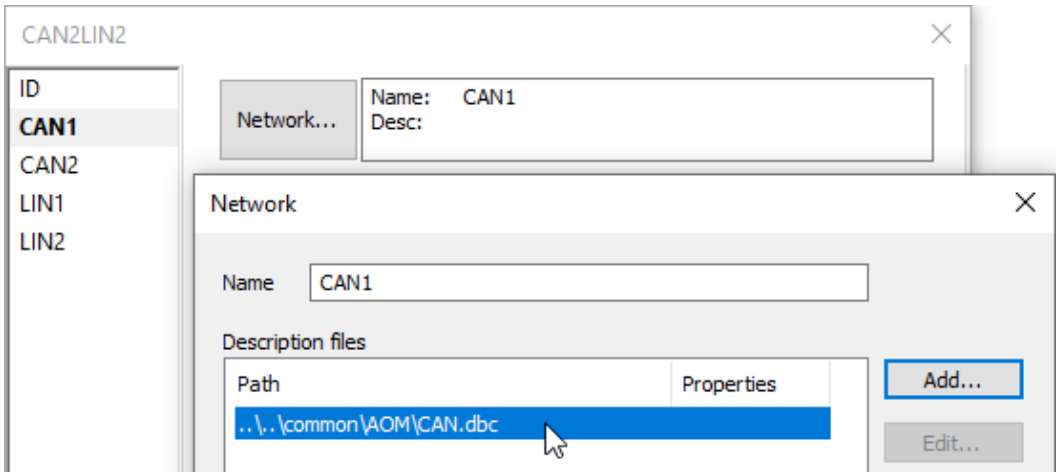
To include a description file in winIDEA follow these steps:

1. Open the CAN/LIN dialog via Hardware / Options / FNet.

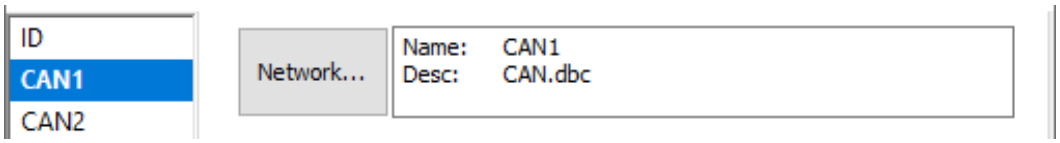
2. Select the appropriate channel.

3. Add the description file to the list.

   i. Press *Network*.
   ii. Press *Add* and select your description file.



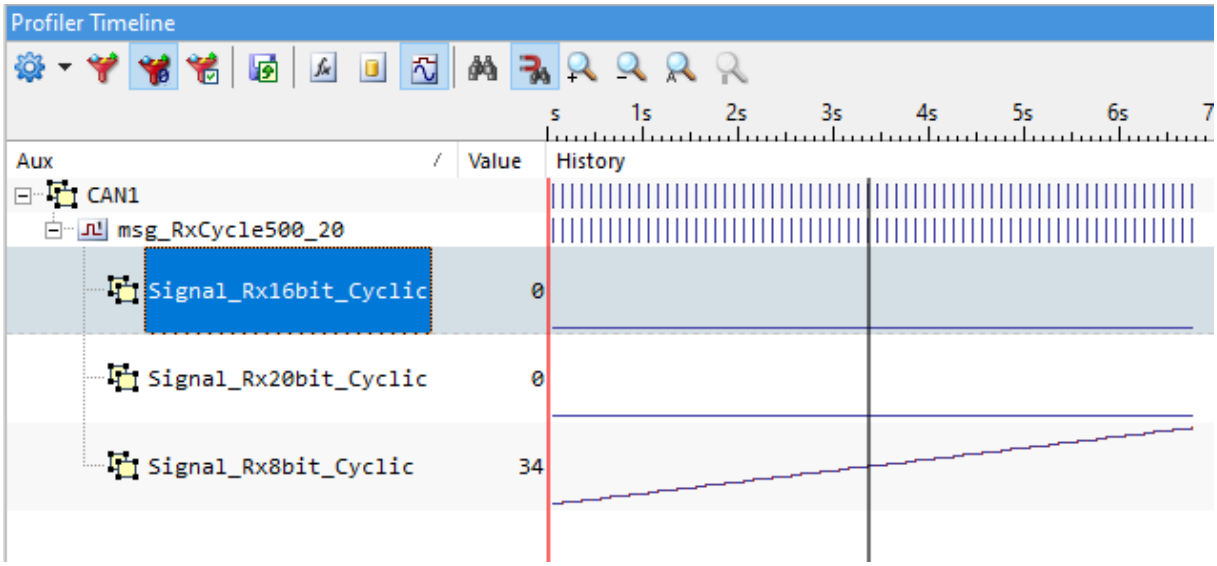
The description file is now added and listed in dialog.



3.2.3 Results of the recording

Using these two features, Filtering messages and Description files results in the view as shown in the picture below.

Only frames from the message with configured ID are recorded, and their raw data analyzed and shown in a more user-friendly fashion. Placing a marker (black vertical line) on a timeline will show the value of each signal at that time. Some additional statistical properties are also available for the message and each one of the signals.



4 Timing Analysis

After requiring the knowledge how to configure and use the basic functionality of the CAN/LIN module, you can continue with more advanced use cases.

The added value of using proprietary iSYSTEM tools is reflected in their seamless integration among them, which is also true for the IOM6 CAN/LIN module. The most important element when using the trace-capable debugging tool and a Network Analyzer is their synchronization. Without it, all of the collected data can only be viewed as a separate entity, and not relative to each other.

Using the iSYSTEM's BlueBox trace capabilities coupled with the IOM6 CAN/LIN Network Analyzer can therefore produce some helpful results that would otherwise be very hard to achieve.

Example below shows how to measure the time that the system uses from the data send request to actually sending it.

A synchronous trace and CAN/LIN recording can be done on all the microcontrollers that use a BlueBox timestamp. This means that the microcontroller must either use a dedicated trace port (e.g., Aurora, Nexus) or a Software Trace. Using synchronous trace with an on-chip buffer is not possible at the moment.

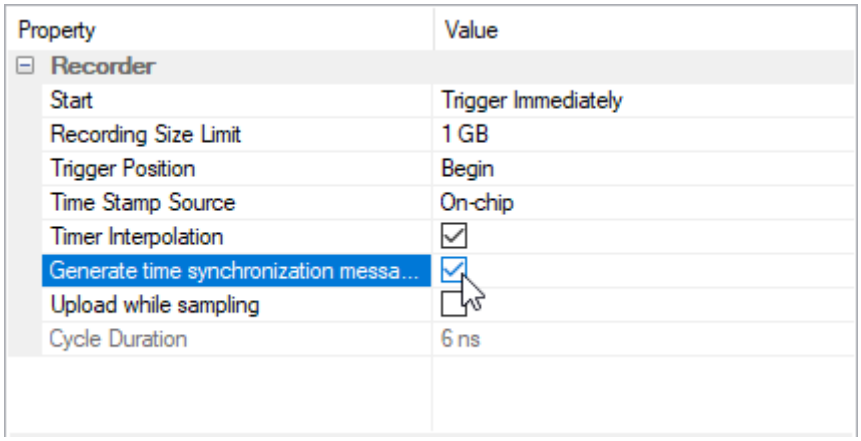
- More information on winIDEA Help Analyzer Configuration.

4.1 System delay

Most of the more complicated applications today run on top of some sort of an operating system. This can either be a simple internally developed RTOS or an entire stack of different services provided by AUTOSAR.

The key difference compared to the pure bare-metal application is that the data transmission is usually managed by the operating system. Therefore, it is very difficult to determine the exact amount of time the system will need to actually output the data on the network bus and might vary with time.

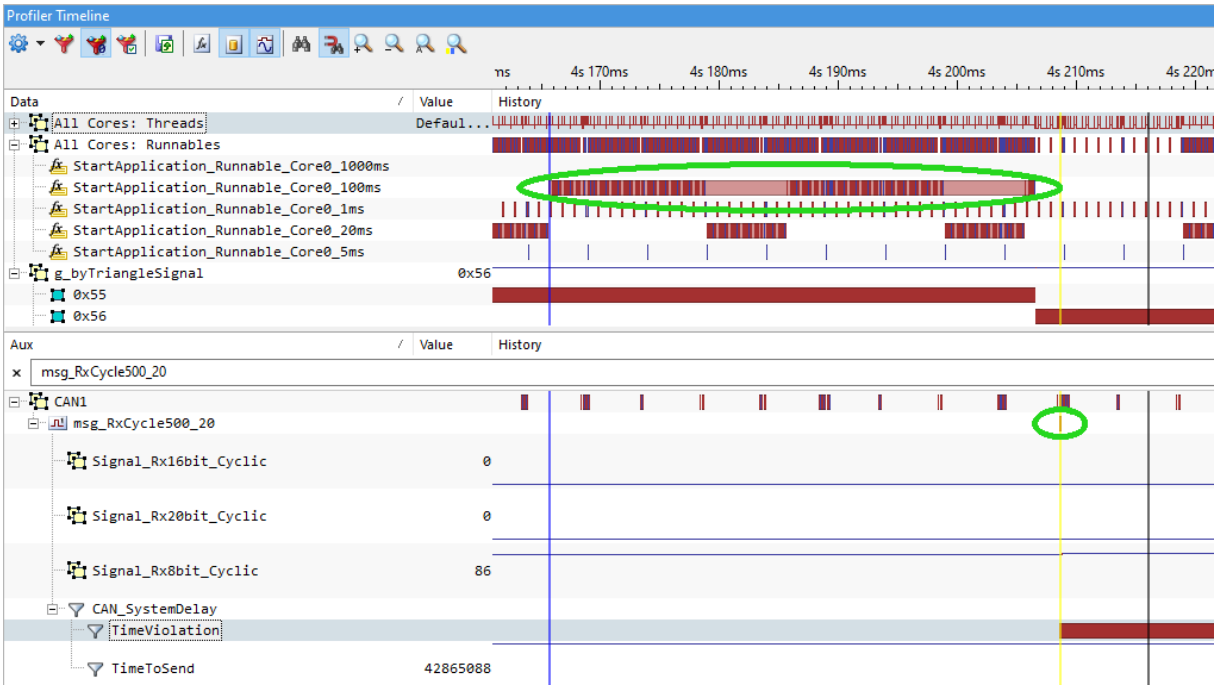
The only configuration needed to synchronize the IOM6 CAN/LIN module with on-chip trace is option *Generate time synchronization messages* in the *Analyzer Configuration* window.



Imagine an example with an operating system running a task that reads some data from a sensor, performs some calculations on it and forwards the formatted data to another device for evaluation via the CAN bus. Putting this task in a time-critical functionality you get some time constraints that have to be met. Using only the program trace you can easily determine whether the specified task performed its operations in a dedicated time slot or not. However, without the Network Analyzer you can't be sure if the system then actually sent the message via the bus.

Using a synchronized program and network trace capabilities, you can easily determine the system time needed to execute the specified functionality.

In the picture below, task execution and CAN message detection are marked with a green color. You can use markers (blue and yellow) to manually determine the elapsed time from the start of task execution to CAN message detection. For automatic calculation of this time winIDEA offers *Profiler Inspectors* functionality and raises a `TimeViolation` flag when the elapsed time is out of bounds.



- More information about Profiler Inspectors in winIDEA Help.

5 Message injection

5.1 Manual message injection

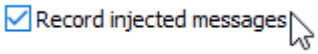
A manual message injection can be useful to trigger a specific functionality of the device. This means that for validation purposes of a device's functionality, you do not need to replicate the exact physical conditions or set up the whole network of interconnected devices.

Message injection can be configured via *FNet Operation* as shown below. Example below shows how to inject a standard-length CAN message with ID 0x123 and 32-bit all-zero data. A TC4 is set as an action trigger on when to inject this message.

1. Open FNet Operation / CAN1 page.

2. Configure the Action 0.

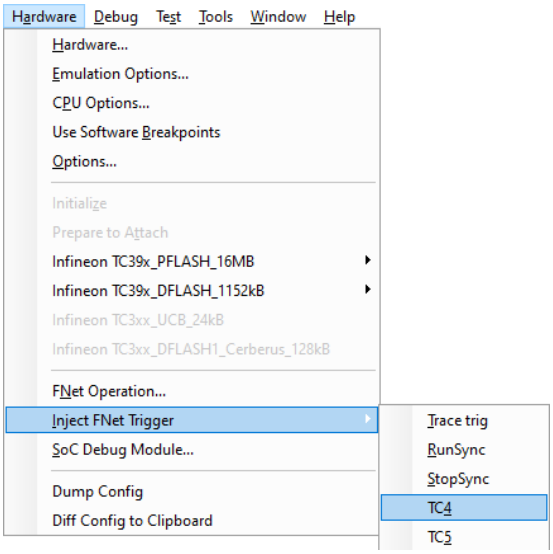
   iii. Check option *Record injected messages*.



   iv. Select *TC4* from the *Action* drop-down.
   v. Enter ID and Data.

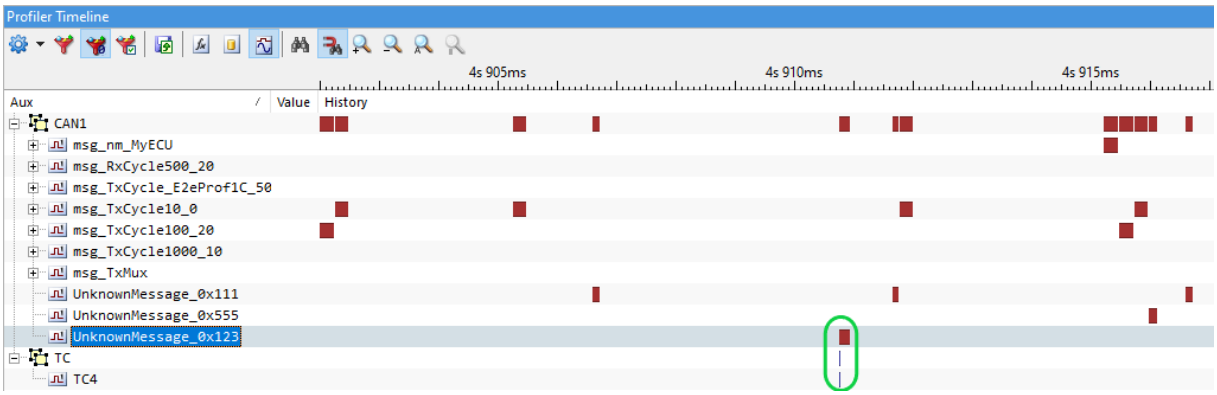
3. Start the Analyzer Session.

4. Inject trigger via Hardware / Inject FNet Trigger / TC4.



5.1.1 Result of the recording

After manually triggering an injection, a trigger and the injected message will be shown on the Profiler Timeline as can be seen below, together with all of the messages that were already recorded on the bus.



5.2 Injection on message reception

There is a possibility that a message is injected on the reception of another specific message. This way a response-like functionality can be simulated by the IOM6 CAN/LIN, avoiding the need for a second device on the network and therefore decreasing the price of a required hardware.

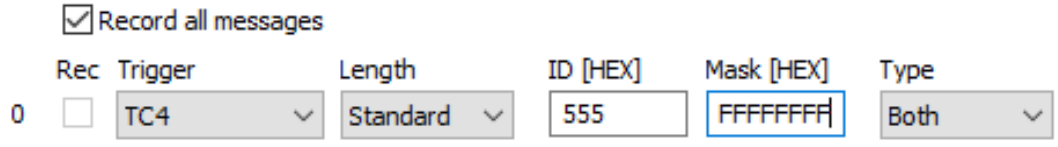
Injecting the response message on the bus follows the same principle as a manual message injection. The additional step needed here is detecting the incoming message and assigning it to the respective trigger.

As an example, you will be looking for the message with an ID 0x555, assigning detection of it to TC4 and triggering the output message with ID 0x123. A picture below shows a complete configuration for this kind of behavior.

1 Open FNet Operation / CAN1 page.

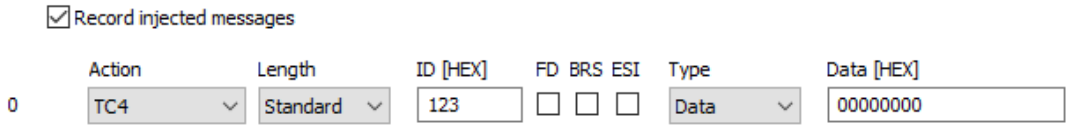
2 Configure the Trigger 0.

i. Check option *Record all messages*.
ii. Select *TC4* from the *Action* drop-down.
iii. Enter ID and Mask.



3 Configure Action 0.

i. Check option *Record injected messages*.
ii. Select *TC4* from the *Trigger* drop-down.
iii. Enter ID and Data.



5.2.1 Result of the recording

As soon as a message with the correct ID will be detected a response message will be injected on the bus. A trigger will also be recorded to distinguish between injected and incoming messages (in case a message with the same ID is already present on the bus).

6 CAN/LIN message as trigger for trace recording

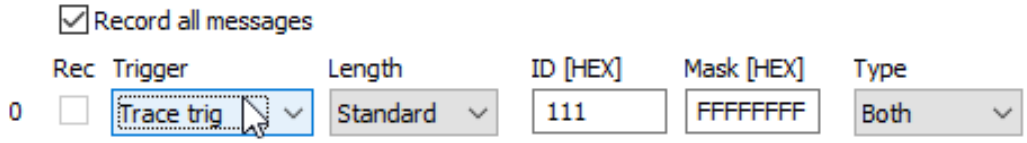
A CAN or a LIN message can also be used to start the trace recording. This is especially useful when a user is for example trying to debug a program flow of a microcontroller with limited trace capabilities (e.g., small trace storage RAM). A detection of a specific message has to be connected to a *Trace trig* trigger that will consequently start the trace recording.

As an example, you will be looking for a CAN message with an ID `0x111` and triggering a program flow trace. The required configuration is in *FNet Operation* dialog.

1 Open FNet Operation / CAN1 page.

2 Configure the Trigger 0.

i. Check option *Record all messages*.
ii. Select *Trace trig* from the *Trigger* drop-down.
iii. Enter ID and Mask.



3 Start the Analyzer Session manually.

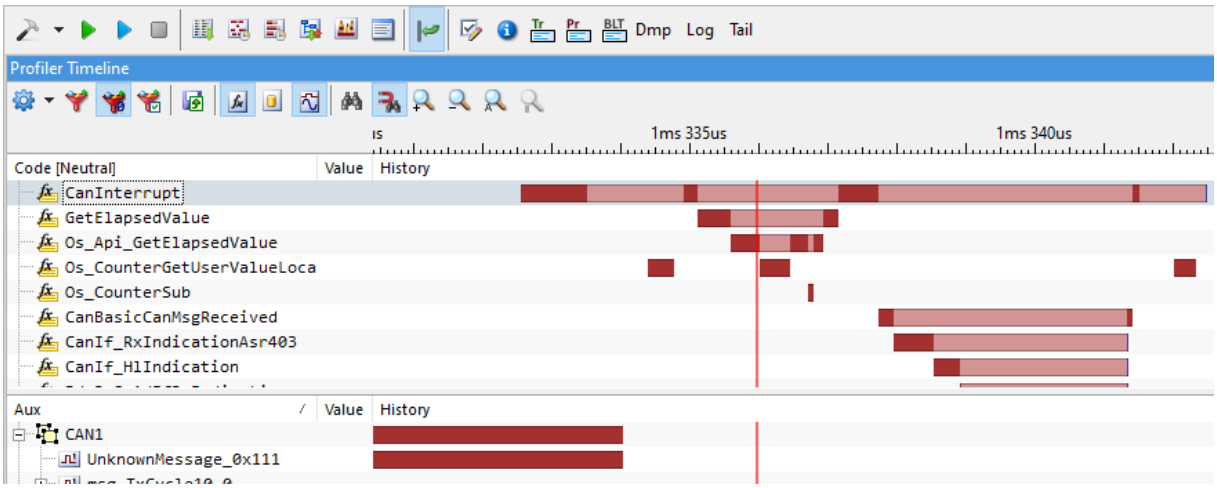
Note that a *New Session* must still be started manually via  in the Analyzer window.

After that an IOM6 CAN/LIN will be inspecting the network data and on detection of a specified message start the trace recording. This can be either an entire program flow or snippets of it, data or network trace or anything else according to your device's capabilities and must be configured separately. The displayed configuration will only start the trace recording but not set what to record.

➢ How to start a Trace recording?

6.1.1 Result of the recording

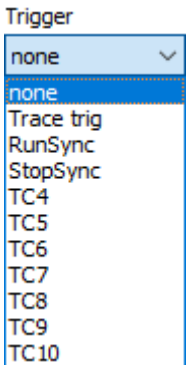
In our example the whole program trace was recorded as can be seen in the picture below.



6.2 Further trigger configuration

Other triggers can be raised on detection of a specific message, not just trace trigger. Current possibilities are shown in the picture below:

- *Trace trig* operation is described above.
- *RunSync* and *StopSync* can be used when working with interconnected BlueBoxes and synchronously debugging multiple targets.
- *TC4* - *TC10* can be used as manual triggers or as a means of interconnecting the various operations available on IOM6 CAN/LIN and other FNet devices.



7 isystem.connect SDKs

All of the described use cases and examples can be replicated and automated using our isystem.connect SDKs for a variety of different programming languages.

- For more information about different possibilities and supported languages visit or contact our technical support.

isystem.connect SDK allows you to automate every step of the configuration process and manipulate the target the same way as using the interactive GUI of winIDEA. This is especially useful for the purposes of Continuous Integration (CI) which is becoming increasingly popular even in the embedded world. Additional information can be found on our company's website, including a webinar for beginners and professionals using Jenkins and iSYSTEM Tools.

7.1 Message injection

The Python script example, which is attached to this document, shows how to:

- Configure a response message injection (the same way as in the chapter Injection on message reception);
- Start a new profiler session;
- Run a target and manually injecting a message via SDK;
- Set a breakpoint;
- Run the target until the breakpoint is hit;
- Inject a CAN message after it;

using isystem.connect Python SDK.

Note that the messages can be injected as long as there is a valid connection between winIDEA and BlueBox, it does not matter if the target is running or stopped or if the analyzer session is started.

7.2 Exporting and analyzing data

Once the desired data has been captured, you can continue with its analysis. You can export the data into a number of different external files (XML, BIN or BTF) for use with external analysis tools, or use isystem.connect SDK to analyze the captured events.

The Python script example of exporting data into an XML format and analyzing is attached to this document.

During the analysis, let's assume that you're only interested in CAN messages that contain data for `Signal_Rx8bit_Cyclic`. A capture of these messages can be seen in the picture under chapter Filtering messages and description files. Each one of these messages is for analysis purposes represented as a time event, with its handle, value and timestamp.

Output example for one of the `timeEvents - CProfilerTimeEvent`:

```
handle: 1073741836
eventType: 1
value: 37
valueUnit:
floatValue: 37
time: 3667651152
eventSource: -1
```

7.3 Examples

7.3.1 Custom example

This Python script example shows how to perform message injection and export data via isystem.connect SDK.

Open this document in Acrobat Reader. Right-click on the  icon to save the file. Before saving it delete the file name extension .txt in order to create a Python file format.

CANLIN_Usecase.py.txt	

7.3.2 isystem.connect for Python

isystem.connect Python is supported by a comprehensive *User's Guide* and a wide range of Python Examples scripts to get you started which are regularly updated.

- fnetCAN.py – Basic FNet Root Counter controller initialization and usage.
- profiler2Example.py – Profiler and export.
- Python Examples by Groups – Python examples are available for various actions, e.g., Debug Control, Trace, Profiler, FNet, winIDEA control and options etc.

8 Technical support

8.1 More resources

Online Help ▸ winIDEA and testIDEA help	Knowledge Base ▸ Tips & tricks categorized by issue type and architecture	Tutorials ▸ From beginner to expert
Technical Notes ▸ How-tos for winIDEA functionalities with scripts	Application Notes ▸ How-to notes on advanced use cases	Webinars ▸ Technical webinars about iSYSTEM tools with use cases

8.2 Contact

Please visit https://www.isystem.com/contact.html for contact details.