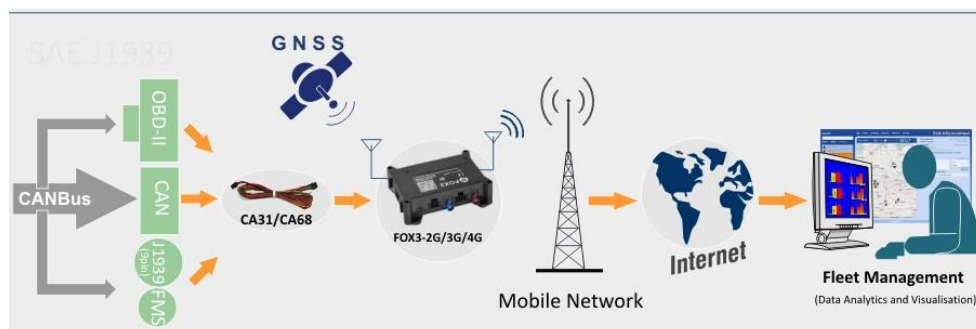


## Application Note

*How to collect CAN FMS/J1939/OBD-II data with FOX3-2G/3G/4G Series*



## Intellectual Property

© 2019 Lantronix, Inc. All rights reserved. No part of the contents of this publication may be transmitted or reproduced in any form or by any means without the written permission of Lantronix.

*Lantronix* is a registered trademark of Lantronix, Inc. in the United States and other countries.

Patented: [www.lantronix.com/legal/patents/](http://www.lantronix.com/legal/patents/); additional patents pending.

All trademarks and trade names are the property of their respective holders.

## Contacts

### **Lantronix, Inc.**

7535 Irvine Center Drive, Suite 100

Irvine, CA 92618, USA

Toll Free: 800-526-8766

Phone: 949-453-3990

Fax: 949-453-3995

### **Technical Support**

Online: [www.lantronix.com/support](http://www.lantronix.com/support)

### **Sales Offices**

For a current list of our domestic and international sales offices, go to the Lantronix web site at [www.lantronix.com/about/contact](http://www.lantronix.com/about/contact)

## Disclaimer

All information contained herein is provided "AS IS." Lantronix undertakes no obligation to update the information in this publication. Lantronix does not make, and specifically disclaims, all warranties of any kind (express, implied or otherwise) regarding title, non-infringement, fitness, quality, accuracy, completeness, usefulness, suitability or performance of the information provided herein. Lantronix shall have no liability whatsoever to any user for any damages, losses and causes of action (whether in contract or in tort or otherwise) in connection with the user's access or usage of any of the information or content contained herein. The information and specifications contained in this document are subject to change without notice.

## Revision History

Date	Rev.	Comments
March 2012	1.0.0	Initial version.
August 2012	1.0.1	Extended chapter 1 and added chapter 2
October 2014	1.0.2	Added new FMS variable (dynamic entries)
August 2016	1.0.3	Added new FMS variable (dynamic entries) – FMS_HR_TOTAL_FUEL_USED
March 2018	1.0.4	Updated section 2.3.1, removed section 2.4.2.
October 2019	A	Initial Lantronix document. Added Lantronix document part number, logo, contact information, and links.

For the latest revision of this product document, please check our online documentation at [www.lantronix.com/support/documentation](http://www.lantronix.com/support/documentation).

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
<b>2</b>	<b>Requirements for reading data from CAN-OBD-II/FMS/J1939 .....</b>	<b>6</b>
2.1	Install the SIM card .....	6
2.2	Connect the device to the vehicle bus .....	6
2.3	Configuring devices for reading vehicle information (examples).....	8
2.4	Find out OBD-II PIDs supported by a vehicle .....	10
<b>3</b>	<b>SUPPORTED Commands and Configuration Syntax for CAN .....</b>	<b>11</b>
3.1	Activate and configure the OBD-II, FMS or J1939 interface for reading .....	11
3.2	Device configuration settings .....	12
3.3	Device generated events used for OBDII, FMS and J1939 messages.....	14
<b>4</b>	<b>Dynamic variables used for reporting .....</b>	<b>16</b>
4.1	Dynamic variables for FMS/J1939 .....	16
4.2	Dynamic variables for OBD-II messages.....	21

# 1 INTRODUCTION

This document describes how to collect CAN-Bus information from a vehicle using FOX3-2G/3G/4G series devices with CAN-Bus premium-feature.

F FOX3-2G/3G/4G series currently support three major vehicle bus protocols: OBD-II, J1939 and FMS. They use two lines (CAN-High and CAN-Low) to connect to an OBD-II, J1939 or FMS interface or to the CAN bus either using CANCrocodile (recommended method), or by direct contact and access the data available on the vehicle CAN bus. The data can be transferred in real-time to a remote server for further analysis and visualizations.

A vehicle CAN bus protocol is a specialized internal communications network that interconnects components inside a vehicle transmitting information like vehicle speed, engine loading, engine R.P.M, fuel level and much more. For example, OBD-II supports more than 79 kinds of vehicle information. For a list of basic OBD-II information (called PIDs), their definitions, and the formula to convert raw OBD-II output to meaningful diagnostic units, refer to this external link: [OBD-II PIDs](#).

The On-Board Diagnostics, Second Generation (OBD-II) is mostly for small car information while the SAE J1939 is implemented for off-road vehicles with diesel engines and the FMS is for commercial trucks or buses. For more information and message structure about the FMS messages, see <http://www.fms-standard.com>.

The OBD-II uses a female 16-pin (2x8) SAE J1962 connector on the vehicle while J1939 uses a 9-pin round connector and FMS gateway a 4-pin round connector. The counterparts for these connectors are not provided by Lantronix but they can be ordered from external suppliers.

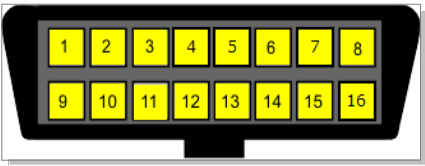

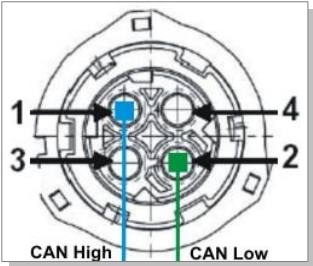
Interface	Interface connector	Connector Pinout
OBD-II		16-pin (2x8) connector interface. Pin Nr: Description 4 Common ground vehicle 6 CAN HIGH 14 CAN LOW 16 Battery power (+) (Not available on all cars)
J1939		9-pin round connector interface. Pin Nr: Description A Common ground vehicle C J1939 + D J1939 - B Battery power (+) (Not available on all cars)
FMS		Standard 4-pin round connector interface. Pin Nr: Description 1 CAN HIGH 2 CAN LOW 3 Option CAN ground 4 Not used by Bus-FMS-Standard

Table 1: CAN-Bus Vehicle interface connectors

## 2 REQUIREMENTS FOR READING DATA FROM CAN-OBD-II/FMS/J1939

To read data from a vehicle bus you'll need:

1. FOX3-2G/3G/4G series device with activated CAN-INTERFACE (Premium-Feature),
2. **SIM card** supporting 2G/3G/4G data for transmitting vehicle data to a server,
3. **Device configuration**, the settings in the device to read information from the vehicle bus,
4. **CA123**, the OBD-II cable to connect the device with the OBD-II connector in your car,
5. **CA31b**, **CA39b** or **CA68** – one vehicle installation cable to power the device and connect it to a FMS gateway or CAN in your truck,
6. A vehicle supporting SAE J1939, CAN, CAN-OBD-II or CAN-FMS,
7. **Power supply**, permanent voltage (12 - 32 vdc) from the fuse box in your car.

The CAN-Bus interface in the **FOX3-2G/3G/4G series** belongs to the hardware Premium-features. To use it you have first to activate it. Please refer to the AppNote "AppNotes\_HowToActivatePremiumFeatures\_vxx.pdf" available on our website: <https://www.lantronix.com/resources/app-notes/>.

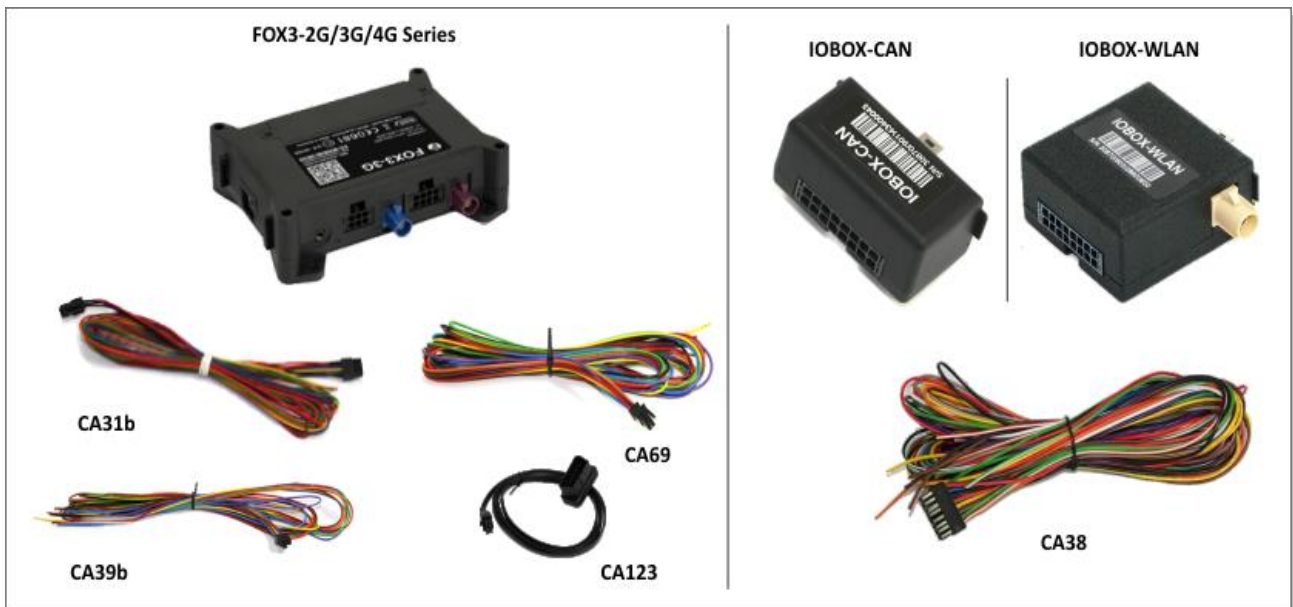


Figure 1: FOX3-2G/3G/4G series or with IOBOX-CAN/WLAN (accessory devices) support CAN-INTERFACE

### 2.1 Install the SIM card

Please refer to the hardware manual of the device you are using to install the SIM card.

### 2.2 Connect the device to the vehicle bus

To connect a FOX3-2G/3G/4G series to the vehicle bus, you need:

- Either activate the premium feature "CAN-INTERFACE" on the FOX3-2G/3G/4G series or use the accessory device IOBOX-CAN or IOBOX-WLAN.
- Use CA123 cable and connect the device an OBD-II connector on your car
- Use CA31b, CA39b or CA68 cable and connect the device either usingCANCrocodile (recommended method), or by direct contact to the CAN bus on your vehicle.
- Use CA31 or CA68 cable and connect the device to the FMS interface in your truck

- Identify the CAN-High, CAN-Low, Ground and V+ lines from the FOX3-2G/3G/4G series to connect to and power up the device. Refer to the AppNote “AppNotes\_AVL\_Installation\_Guide\_vx.x.x.pdf” for more details.

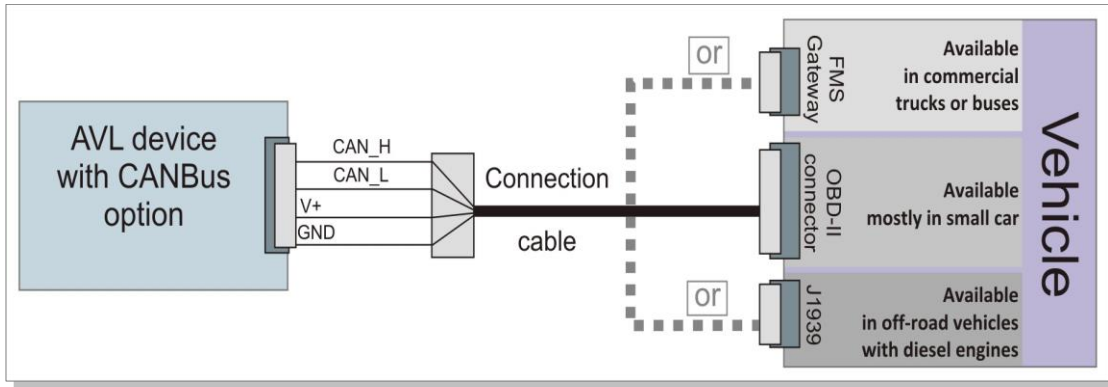


Figure 2: Connecting AVL devices to vehicle CAN Bus


FOX3-2G/3G/4G series support two CAN interfaces. 1<sup>st</sup> CAN interface is built-in in the device and the signals CAN-H and CAN-L are available on the main port (8-pin connector). The 2<sup>nd</sup> CAN interface is available on the IOBOX-CAN or IOBOX-WLAN and the signals CAN-H and CAN-L are available on the 16-pin connector. The CA123 cable can be used to connect the 1<sup>st</sup> CAN interface directly to the OBDII connector on the vehicle while other cables CA31b, CA39b or CA68 can be used to connect the device to a FMS gateway or to the CAB bus either using CANCrocodile (recommended method), or by direct contact. The polarity of the CAN\_High and CAN\_Low lines must be observed when connecting your AVL device to the vehicle bus and both wires (CAN\_High and CAN\_Low) should be twisted.

The following table lists the CAN interface lines of the FOX3-2G/3G/4G series and IOBOX-CAN/WLAN devices:

FOX3-2G/3G/4G Series CA31b, CA39b,CA68				
	Color	Standard	CAN-Option	Meaning / Note
	Green	DI2	CAN_H	Dominant HIGH. Twisted with CAN_L (yellow wire).
	Yellow	DI3	CAN_L	Dominant LOW. Twisted with CAN_H (green wire).
	Brown	GND	GND	Ground
	Red	V+	V+	Input voltage (+10.8...32.0V DC)
IOBOX-CAN or IOBOX-WLAN with CA38				
	Color	Standard	CAN-Option	Meaning / Note
	Brown	IO2	CAN_H	Dominant HIGH. Twisted with CAN_L (black wire).
	Black	IO3	CAN_L	Dominant LOW. Twisted with CAN_L (brown wire).
	Orange	GND	GND	Ground

Table 2: CAN interface lines for FOX3-2G/3G/4G series and IOBOX-CAN/WLAN devices

Locating the connection points on the vehicle:

	<p>- Locating your OBD-II connector can be a difficult task as vehicle manufacturers tend to hide away the socket. Usually OBD-II connector is located on the driver's side of the passenger compartment near the center console. Sometimes it's located in the driver's foot well, under the steering wheel, behind panels in the dashboard fascia and the central area between the driver's seat and the passenger seat. Some connectors have been located behind ashtrays, under the passenger seat and even over by the passengers door.</p>
<p>- NO PICTURE</p>	<p>- The FMS interface gateway and the connectors are located behind the dashboard. Contact your vehicle manufacture for more details.</p>

### 2.3 Configuring devices for reading vehicle information (examples)

To read messages on the vehicle bus, you have to connect first the FOX3-2G/3G/4G series to the vehicle bus as shown above and then activate the CAN interface and enable the FMS or OBD-II interface on the FOX3-2G/3G/4G series. **Do not activate/enable both OBD-II and FMS interfaces at the same time.** The examples below show how to request the vehicle Engine Speed and Vehicle Speed from the vehicle bus and send them to a TCP server for further analysis. For more details about the PFAL command description, refer to the chapter [Error! Reference source not found.](#)

<p><b>a) When connecting to the vehicle OBD-II connector use the configuration below:</b></p>	
<p>Activate CAN bus functionality and set the correct communication baud rate on FOX3-2G/3G/4G series.</p>	
<pre>\$PFAL, Sys.CAN.Enable, 500K, RW</pre>	
<p>Enable OBD-II functionality on AVL device.</p>	
<pre>\$PFAL, Sys.CAN.OBDII.Enable</pre>	
<p>Giving alerts when Engine Speed enters specific ranges first enter the PID of the Engine Speed (0C) and specify the speed ranges when the OBD-II events should be generated. Example below shows that <i>Sys.eOBDII=0,0</i> and <i>Sys.eOBDII=0,1</i> events will be generated when the vehicle Engine Speed is between 2500 and 3500 R.P.M. or 3520 and 6000 R.P.M. respectively.</p>	
<pre>\$PFAL, Cnf.Set, DEVICE.CAN.EVENT_0=OBDII.0C, 2, V0, 26E8, 2710, 36B0, 36D8, V0, 36D8, 3700, 5DC0, 5DE8</pre>	
<p>See alarm configuration below to know how to report via TCP the actual Engine Speed when one of the predefined speed ranges is entered and the corresponding event is occurred.</p>	
<pre>\$PFAL, CNF.Set, AL25=Sys.eOBDII=0, 0:TCP.Client.Send, 8, "R.P.M. &amp; (OBDII0C) &lt; 3500" \$PFAL, CNF.Set, AL26=Sys.eOBDII=0, 1:TCP.Client.Send, 8, "R.P.M. &amp; (OBDII0C) &gt; 3500"</pre>	
<p>To give alerts when Vehicle Speed enters specific ranges, first define the Vehicle Speed PID and specify the speed ranges in which the OBD-II events should be generated.</p>	
<pre>\$PFAL, Cnf.Set, DEVICE.CAN.EVENT_1=OBDII.0D, 2, V0, 0, 1, 4F, 50, V0, 50, 51, 5A, 5B</pre>	
<p>See alarm configuration below to know how to report the actual Vehicle Speed via TCP when one of the predefined speed ranges is entered and the corresponding event is occurred.</p>	
<pre>\$PFAL, CNF.Set, AL27=Sys.eOBDII=1, 0:TCP.Client.Send, 8, "Speed &amp; (OBDII0D) &lt; 80 kmph" \$PFAL, CNF.Set, AL28=Sys.eOBDII=1, 1:TCP.Client.Send, 8, "Speed &amp; (OBDII0D) &gt; 80 kmph"</pre>	



**b) When connecting to the vehicle FMS-Gateway or J1939 connector use the configuration below:**

Activate CAN Bus and set the communication baud rate on AVL device.

```
$PFAL, Sys.CAN.Enable, 250K, RO
```

Enable FMS functionality on AVL device and restart the AVL device.

```
- - $PFAL, Sys.CAN.FMS.Enable
```

Giving alerts when Engine Speed enters specific ranges, first enter the predefined FMS parameter of the Engine Speed (FMS.ENGINE\_SPEED) and specify the speed ranges in which the FMS events should be generated. Example below shows that *Sys.eFMS=0,0* and *Sys.eFMS=0,1* events will be generated when the vehicle Engine Speed is between 2500 and 3500 R.P.M. or 3520 and 6000 R.P.M. respectively.

```
- - $PFAL, Cnf.Set, DEVICE.CAN.EVENT_0=FMS.ENGINE_SPEED, 2, V0, 26E8, 2710, 36B0, 36D8, V0, 36D8, 3700, 5DC0, 5DE8
```

See alarm configuration below to know how to report via TCP the actual Engine Speed when one of the predefined speed ranges is entered and the corresponding event is occurred.

```
$PFAL, CNF.Set, AL25=Sys.eFMS=0, 0:TCP.Client.Send, 8, "R.P.M.  
&(FMS.ENGINE_SPEED) < 3500"  
$PFAL, CNF.Set, AL26=Sys.eFMS=0, 1:TCP.Client.Send, 8, "R.P.M.  
&(FMS.ENGINE_SPEED) > 3520"
```

To give alerts when Vehicle Speed enters specific ranges, first define the FMS parameter of the Vehicle Speed and specify the speed ranges in which the FMS events should be generated.

```
- - $PFAL, Cnf.Set, DEVICE.CAN.EVENT_1=FMS.SPEED_WB_KMPH, 2, V0, 0, 1, 4F, 50, V0, 50, 51, 5A, 5B
```

See alarm configuration below to know how to report the actual Vehicle Speed via TCP when one of the predefined speed ranges is entered and the corresponding event is occurred.

```
$PFAL, CNF.Set, AL27=Sys.eFMS=1, 0:TCP.Client.Send, 8, "Speed &(FMS.SPEED_WB_KMPH) < 80 kmph"  
$PFAL, CNF.Set, AL28=Sys.eFMS=1, 1:TCP.Client.Send, 8, "Speed &(FMS.SPEED_WB_KMPH) > 80 kmph"
```

To give alerts when Brake Pedal is pressed or released, first define the FMS parameter of Brake Switch and specify the states when the FMS Brake events should be generated.

```
- $PFAL, Cnf.Set, DEVICE.CAN.EVENT_2=FMS.BRAKE_SWITCH, 2, BH0, BLO
```

See alarm configuration below to know how to report via TCP the state of Brake Switch when it is pressed or released.

```
$PFAL, CNF.Set, AL29=Sys.eFMS=2, 0:TCP.Client.Send, 8, "BRAKE: &(FMS.BRAKE_SWITCH)"  
$PFAL, CNF.Set, AL30=Sys.eFMS=2, 1:TCP.Client.Send, 8, "BRAKE: &(FMS.BRAKE_SWITCH)"
```

## 2.4 Find out OBD-II PIDs supported by a vehicle

This external link "[http://en.wikipedia.org/wiki/OBD-II\\_PIDs](http://en.wikipedia.org/wiki/OBD-II_PIDs)" shows a list of parameter IDs that any OBD-II compatible vehicle must support. Beyond these parameters every vehicle manufacturer can have their own proprietary parameters, which are not defined in the OBD-II standard.

To find out which PIDs are supported in your vehicle, use the command below and enter within the `<pid_hex>` the PID (hex) of the OBD-II message you are interested in. Finally, send the command with the defined `<pid_hex>` to the FOX3-2G/3G/4G series.

PFAL Syntax	<code>\$PFAL, Sys.Can.OBDII.Request,&lt;pid_hex&gt;</code>
Send to the device	<code>\$PFAL, Sys.Can.OBDII.Request,0C // Requests if Engine RPM is supported</code>
Send to the device	<code>\$PFAL, Sys.Can.OBDII.Request,0D // Requests if Vehicle speed is supported</code>
Send to the device	<code>\$PFAL, Sys.Can.OBDII.Request,1F // Requests if Run time since engine start is supported</code>

after that execute the command below to see the report:

Send to the device	<code>\$PFAL,MSG.Send.Serial0,0,"&amp;(OBDII0C); &amp;(OBDII0D); &amp;(OBDII1F)"</code>
--------------------	---

If the AVL device is properly connected to the vehicle OBD-II interface, it will output on the 1<sup>st</sup> serial port (8pin connector) the current value of the requested PIDs.

If the device outputs **&(ERR)** or **&(n/a)**, then either the FOX3-2G/3G/4G series is not connected properly to the OBD-II port or the vehicle does not support that OBD-IIs at all.

The formula how to translate the response into meaningful data can be found in same table within the standard PIDs from the external link above.

## 3 SUPPORTED COMMANDS AND CONFIGURATION SYNTAX FOR CAN

### 3.1 Activate and configure the OBD-II, FMS or J1939 interface for reading

To activate the CAN bus on the FOX3-2G/3G/4G series and set the communication baud-rate use one of the following commands listed in table below. Following modes are supported by FOX3-2G/3G/4G series which indicate if the message will be a Read, ReadWrite or Test.

#### 3.1.1 Activate CAN interface

CAN Modes	PFAL Command	Description
ReadOnly	\$PFAL,Sys.CAN.Enable,250K,RO	This mode should be enabled for FMS and J1939 messages which do not need external request. Some FMS messages need external request so they required the <b>ReadWrite (RW)</b> option. <b>Hint:</b> This setting is NOT usable for OBD-II. It works for most FMS/J1939 messages (except some specific messages like: vehicle weight etc.).
ReadWrite	\$PFAL,Sys.CAN.Enable,500K,RW	This mode should be enabled for OBD-II messages that need external request. That means, the AVL device has to send a request to the vehicle computer (ECU) in order to the get an response from the vehicle computer. <b>Hint:</b> This setting is usable for OBDII and some of FMS/J1939 messages. Be aware that the device sends data to the vehicle CAN-Bus. Therefore, use this setting at your own risk and with special caution (i.e. use specific CAN gateways, never connect to Motor CAN directly).
Test (silent loop-back)	\$PFAL,Sys.CAN.Enable,100K,LBS	This mode is to simulate messages without having access to vehicle CAN-Bus. <b>Hint:</b> This setting is usable for testing OBD-II and most FMS/J1939 messages when no connection to the CAN-Bus is available. Detailed protocol knowledge of the used messages is required in order to send CAN messages with correct format.

#### 3.1.2 Activate FMS or OBD-II interface

To enable/disable the FMS interface use the following PFAL command:		
	\$PFAL,Sys.CAN.FMS.Enable	To activate the FMS execute this command just one time. The FMS is then enabled automatically at each system start.
	\$PFAL,Sys.CAN.FMS.Disable	To disable FMS interface execute this command just one time.
To enable/disable the OBD-II interface use the following PFAL Command:		
	\$PFAL,Sys.CAN.OBDII.Enable	To activate the OBD-II execute this command just one time. The OBD-II is then enabled automatically at each system start.
	\$PFAL,Sys.CAN.OBDII.Disable	To disable OBD-II interface execute this command just one time.

## 3.2 Device configuration settings

These settings can be used to generate events out of the changed CAN variables without checking them periodically for changes. FOX3-2G/3G/4G series provide the possibility to analyze 10 slots allowing 10 different CAN messages to be analyzed at the same time.

CAN Messages will be checked for changes approx. every 128 ms (8 times per second). Here below is an example to show how to configure an AVL device to get a single event when the vehicle engine R.P.M. is between 0 and 1387 (hex):

```
$PFAL,Cnf.Set,DEVICE.CAN.EVENT_0=OBDII.0C,1,V0,-1,0,1387,1388
```

- Single event if R.P.M. is between 0 and 1387 (4 x R.P.M.)

```
$PFAL,Cnf.Set,DEVICE.CAN.EVENT_0=FMS.BRAKE_SWITCH,2,BH0,BL0
```

- 2 events if brake switch state changes (Pressed & Released)

Syntax	\$PFAL,Cnf.Set,DEVICE.CAN.EVENT_<slot>=<standard>.<msg_id>,<event_cnt>[,<eventX>]
--------	---

<slot>

It specifies the index of the CAN event slot, in the range from **0** to **9**, for storing the value received from the PID.

<standard>

It specifies one of the CAN bus protocols supported by Lantronix AVL devices.

Value	Meaning
<b>OBDII</b>	Self-diagnostic and reporting capability of a vehicle used in cars and light trucks.
<b>FMS</b>	It is a standard interface to vehicle data of commercial vehicles. The amount of data is dependent on the manufacturer and model of the vehicle and might be different.
<b>J1939</b>	It is the vehicle bus standard used for communication and diagnostics among vehicle components, originally by the car and heavy duty truck industry.

<msg\_id>

Separated by a dot ".", it defines the OBD-II PID or FMS/J1939 parameter representing the data you want to read. Format depends on the chosen <standard> protocol:

- If OBDII protocol is specified in the <standard> entry, then specify here a PID (in hex) that represents the data you want to read (i.e. "0C" reads the engine rotations per minute R.P.M. ), see also [OBD-II PIDs](#).
- If FMS or J1939 protocol is specified in the <standard> entry, then specify here an FMS or J1939 parameter that represents the data you want to read (i.e. "BRAKE\_SWITCH" reads the brake switch state). The supported FMS/J1939 parameters are listed in chapter [Error! Reference source not found.](#), [Error! Reference source not found.](#).
- ♦ Note, that J1939 is only partially supported:
  - J1939 is supported for all periodic messages.
  - Depending on the used CAN interface, some J1939 messages might not be periodically sent on the vehicle CAN bus, but they have to be requested from the AVL device. These request packets are currently not sent automatically (but can be sent by configuring an alarm). Due to CAN message licensing restrictions, no support can be given for command examples or command syntax lists).

### <event\_cnt>

It specifies the number of events that will be configured in the <eventX> entry for the specified CAN message in the <msg\_id> entry (several events can be configured for one CAN message - i.e. various engine speed ranges, vehicle speed etc.).

### {<eventX>}

Optional. It defines the list of events (the list of events specified in the <eventX> entry must match to the number defined in the <event\_cnt> entry).

Configuration of the specific event. Following syntaxes can be used:

<eventX>=<s\_type><event\_edge><s\_info\_id>

or

<eventX>=<s\_type><s\_info\_id>,<rst\_Lmax>,<minval>,<maxval>,<rst\_Hmin>

### <s\_type>

Sub-index type. Variable type and possible values:

**B** Bit (value 0 or 1)

**V** Value

Value	Sub-entry	Sub-Value and description
If <s_type>= <b>B</b> then:	1st Syntax: <eventX>=<s_type><event_edge><s_info_id>	
	<event_edge>	<b>H</b> Event occurs when Bit changes from 0 -> 1 <b>L</b> Event occurs when Bit changes from 1 -> 0
	<s_info_id>	Sub-index ID (ID of info within CAN message). If a single message contains just one piece of information, this index is 0. However, a message may contain also several information. Within the FMS/OBDII/J1939 specification they are listed. <s_info_id> specifies a decimal number corresponding to the index of information within a PID response that you want to use to create the event. Example: <b>BH2</b> Event occurs when Bit changes from 0 -> 1 within 3rd information inside configured message. <b>B10</b> Event occurs when Bit changes from 1 -> 0 within 1st information inside configured message.
Example:	<eventX>=BH2	
If <s_type>= <b>V</b> then:	2nd Syntax: <eventX>=<s_type><s_info_id>,<rst_Lmax>,<minval>,<maxval>,<rst_Hmin>	
	<s_info_id>	Sub-index ID (ID of info within CAN message). A request for some OBD-II messages returns data, which may contain more than one piece of information. If a single message contains just one piece of information, this index is 0. This information is listed within the FMS/OBDII/J1939 specification. <s_info_id> - Specifies a decimal number corresponding to the index of information within a PID response that you want to use to create the event. The index start with 0. <b>For example:</b> A request for PID "4F" (see also OBD-II PIDs) returns 4 pieces of information: <b>Maximum value for equivalence ratio, oxygen sensor voltage, oxygen sensor current, and intake manifold absolute pressure.</b> So, if you want to generate an event for <b>oxygen sensor voltage</b> then specify here <b>1</b> , for <b>Maximum value for equivalence ratio</b> specify <b>0</b> and for <b>oxygen sensor current</b> specify <b>2</b> . (i.e. Event-Ranges of the specified CAN information: <rst_Lmax>,<minval>,<maxval>,<rst_Hmin>).

<rst_Lmax>	Specifies in hexadecimal the maximum value which resets the event condition (-> after reaching this value or a lower value, the event can be triggered again).
<minval>	Specifies in hexadecimal the minimum value which triggers the event.
<maxval>	Specifies in hexadecimal the maximum value which can still trigger the event.
<rst_Hmin>	Specifies in hexadecimal the minimum value which resets the event condition (-> after reaching this value or a higher value, the event can be triggered again).
Example:	<eventX>=V0,26E8,2710,36B0,36D8
<p><b>Info:</b> Using these parameters it is possible to define a kind of hysteresis.</p> <p><b>For example engine R.P.M.:</b> an event is defined to be launched between 2500 R.P.M. (2500*4=2710-hex) and 3500 R.P.M. (3500*4=36B0-hex). In order to prevent arising of events very often when the engine R.P.M. changes around 3495 and 3505 R.P.M., it is possible to set the excepted ranges (minimum and maximum levels) which must be exceeded first in order to cause another event of the same type. For this scenario ranges could be defined as:</p> <p>&lt;rst_Lmax&gt;: 26E8 (hex) = 9960 dec. / 4 = 2490 R.P.M.          &lt;minval&gt;: 2710 (hex) = 10000 dec. / 4 = 2500 R.P.M.          &lt;maxval&gt;: 36B0 (hex) = 14000 dec. / 4 = 3500 R.P.M.          &lt;rst_Hmin&gt;: 36D8 (hex) = 14040 dec. / 4 = 3510 R.P.M.</p> <p>The range in R.P.M. from &lt;rst_Lmax&gt;=2490 to &lt;minval&gt;=2500 and &lt;maxval&gt;=3500 to &lt;rst_Hmin&gt;=3510 will generate no event, this range are defined as save areas and this range must be exceeded first to allow the AVL device to generate the next new event.</p> <p>The example corresponding to the diagram above:          \$PFAL,Cnf.Set,DEVICE.CAN.EVENT_0=OBDII.0C,1,V0,26E8,2710,36B0,36D8</p>	

### 3.3 Device generated events used for OBDII, FMS and J1939 messages

#### 3.3.1 Events for OBDII messages

<b>Syntax</b>	Sys.eOBDII=<event_slot>,<sub_event>
<b>Example</b>	Sys.eOBDII=0,0 // event for the slot 0, index 0
<b>Functionality</b>	This event is generated whenever a configured OBD-II event occurs. See chapter <b>Error! Reference source not found.</b> , "Error! Reference source not found." for more details.
<b>Parameters</b>	<p>&lt;event_slot&gt;,&lt;sub_event&gt;</p> <p>&lt;event_slot&gt; It specifies the index of the event slot that generates the desired event. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details</p> <p>&lt;sub_event&gt; It specifies the event number (index of &lt;eventX&gt;) configured within this event slot. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details</p>

#### 3.3.2 Events for FMS messages

<b>Syntax</b>	Sys.eFMS=<event_slot>,<sub_event>
<b>Example</b>	Sys.eFMS=0,0 // event for the slot 0, index 0
<b>Functionality</b>	This event is generated whenever a configured FMS event happens. See chapter <b>Error! Reference source not found.</b> , "Error! Reference source not found." for more details.

<b>Parameters</b>	<event_slot>,<sub_event>
	<a href="#">&lt;event_slot&gt;</a> It specifies the index of the event slot that generates the desired event. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details
	<a href="#">&lt;sub_event&gt;</a> It specifies the event number (index of <eventX>) configured within this event slot. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details

### 3.3.3 Events for J1939 messages

<b>Syntax</b>	Sys.eJ1939=<event_slot>,<sub_event>
<b>Example</b>	Sys.eJ1939=0,0 // event for the slot 0, index 0
<b>Functionality</b>	<p>This event is generated whenever a configured FMS event happens. See chapter <b>Error! Reference source not found.</b>, "<b>Error! Reference source not found.</b>" for more details.</p> <p>Note that, J1939 is supported partially only:</p> <ul style="list-style-type: none"> <li>- J1939 is supported for all periodic messages.</li> <li>- Depending on used CAN interface, some J1939 messages might not be sent periodically but have to be requested. These request packets are currently not sent automatically (but can be sent using alarm config). Due to CAN message licensing restrictions, no support can be given for command examples or command syntax lists).</li> </ul>
<b>Parameters</b>	<event_slot>,<sub_event>
	<a href="#">&lt;event_slot&gt;</a> It specifies the index of the event slot that generates the desired event. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details
	<a href="#">&lt;sub_event&gt;</a> It specifies the event number (index of <eventX>) configured within this event slot. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details

## 4 DYNAMIC VARIABLES USED FOR REPORTING

### 4.1 Dynamic variables for FMS/J1939

These dynamic variables (PID) names can be used in to report their FMS values to the remote server. Please refer to FMS specification for detailed information and message formats.

To transmit them to a remote server, use this command:

```
$PFAL,MSG.Send.TCP,8,"identifier = &(FMS Variable) identifier2= &(FMS Variable2) ...."
```

e.g. to report the Accel and the Break switch use this command:

```
$PFAL,MSG.Send.TCP,8," ACCEL: &(FMS.ACCEL) BRAKE:&(FMS.BRAKE_SWITCH)"
```

FMS Variables	Names	Responses and descriptions
FMS.ACCEL	Accelerator pedal position	Reports the current accelerator pedal position in percent (%).
FMS.ACCEL1_PEDAL_LOW	//	Accelerator pedal 1 low idle switch
FMS.ACCEL2_PEDAL	//	Accelerator pedal position 2
FMS.ACCEL2_PEDAL_LOW	//	Accelerator pedal 2 low idle switch
FMS.ACCEL_KICKDOWN	//	Accelerator pedal kick down switch
FMS.ACCEL_LIMIT	//	Vehicle acceleration rate limit status
FMS.ACCEL_REM_PDL	//	Remote accelerator pedal position
FMS.AMB_TEMP	//	Temperature of air surrounding vehicle
FMS.BRAKE_SWITCH	Brake switch	Reports the current status of the brake switch: 0 - Brake switch is off 1 - Brake switch is on err - Device error n/a - Value not available
FMS.CLUTCH_SWITCH	Clutch switch	Reports the current status of the clutch switch 0 – Clutch switch is off 1 – Clutch switch is on err – Device error n/a – Value not available
FMS.CRUISE_CONTROL	Cruise control status	Reports the current status of the cruise control 0 – Cruise control is off 1 – Cruise control is on err – Device error n/a – Value not available
FMS.DELAY_CALENDER	//	Delay calender time based - 1 week/bit
FMS.DELAY_TIME	//	Delay time operational time based - 1 h/bit
FMS.DRIVER_ID	//	Drivers identification
FMS.ENG_CTRL_ADDR	//	Source address of controlling device for engine control - 8 bit addr
FMS.ENG_FUEL_TEMP1	//	Engine fuel temperature 1 - 1°C/bit (-40° Offset)
FMS.ENG_HR	//	Total engine revolutions



FMS Variables	Names	Responses and descriptions
FMS.ENG_LOAD	//	Engine percent load at current speed
FMS.ENG_MAX_TORQUE	//	Actual maximum available Engine percent torque
FMS.ENG_OIL_TEMP1	//	Engine oil temperature - 1/32 °C/bit (-273° offset)
FMS.ENG_REV	//	Total engine hours
FMS.ENG_START_MODE	//	Engine starter mode - 16 modes
FMS.ENG_TORQUE	//	Actual engine percent torque - 1%/bit offset -125%
FMS.ENG_TORQUE_HR	//	Actual engine percent percent torque high resolution part - 0,125%/bit bit3==0->n/a
FMS.ENG_TORQUE_MODE	//	Engine torque mode - 16 states
FMS.ENGINE_SPEED	Engine speed	Reports the engine speed in rotations per minute (R.P.M.).
FMS.ENGINE_TEMP	Engine coolant temperature	Reports the current engine coolant temperature in degree Celsius (°C).
FMS.FUEL%	Fuel used	Displays the fuel level in % (percent).
FMS.FUEL2%	//	Fuel level 2 - 0.4%/bit
FMS.FUEL_FLTR_PRE	//	Engine fuel filter differential pressure - 2 kPa/bit
FMS.FUEL_RATE	//	Fuel_rate - 0.05 l/h/bit
FMS.FUEL_TOTAL	Total fuel used	Reports the total used fuel in litre (l).
FMS_HR_TOTAL_FUEL_USED	High resolution total fuel used	Used to report the high resolution total fuel used in l (litre).
FMS.INTERCOOL_TEMP	//	Intercooler temperature - 1°C/bit (-40° Offset)
FMS.INTERCOOL_TOPEN	//	Intercooler thermostat opening - 0-100% 0,4%/bit
FMS.MAINTANCE	Next regular maintenance	Reports the remaining distance to the next regular maintenance in kilometres (km).
FMS.OIL_FLTR_PRES	//	Engine oil filter differential pressure - 0,5 kPa/bit

FMS Variables	Names	Responses and descriptions
FMS.PTO	PTO status	<p>Reports the current status of the power take off governor.</p> <ul style="list-style-type: none"> <li>00 – Off / disabled</li> <li>01 – Hold</li> <li>02 – Remote hold</li> <li>03 – Standby</li> <li>04 – Remote standby</li> <li>05 – Set</li> <li>06 – Decelerate/Coast</li> <li>07 – Resume</li> <li>08 – Accelerate</li> <li>09 – Accelerator override</li> <li>10 – Preprogrammed set speed 1</li> <li>11 – Preprogrammed set speed 2</li> <li>12 – Preprogrammed set speed 3</li> <li>13 – Preprogrammed set speed 4</li> <li>14 – Preprogrammed set speed 5</li> <li>15 – Preprogrammed set speed 6</li> <li>16 – Preprogrammed set speed 7</li> <li>17 – Preprogrammed set speed 8</li> <li>18 – PTO set speed memory 1</li> <li>19 – PTO set speed memory 2</li> <li>31 – Value not available</li> </ul>
FMS.PTO_ENG	//	At least one PTO is engaged
FMS.SERV_COMP1	//	Service component identification - component ID (Table SPN911_A)
FMS.SERV_COMP2	//	Service component identification - component ID (Table SPN911_A)
FMS.SERV_COMP3	//	Service component identification - component ID (Table SPN911_A)
FMS.SPD_LIMIT	//	Road speed limit status
FMS.SPEED_WB	Vehicle speed (wheel based)	Displays the wheel based vehicle speed in cm/s
FMS.SPEED_WB_KMPH	Vehicle speed (wheel based)	Reports the wheel based vehicle speed in km/h
FMS.TC_DIR	Direction indicator	<p>Reports the motion direction:</p> <ul style="list-style-type: none"> <li>0 – Forward</li> <li>1 – Reverse</li> <li>err – Device error</li> <li>n/a – Value not available</li> </ul>
FMS.TC_DRV1_CARD	Driver 1 card	<p>Reports the current driver 1 card status:</p> <ul style="list-style-type: none"> <li>0 – Not present</li> <li>1 – Present</li> <li>err – Device error</li> <li>n/a – Value not available</li> </ul>
FMS.TC_DRV1_STATE	Driver 1 working state	<p>Reports the current Driver 1 working state:</p> <ul style="list-style-type: none"> <li>0 - Rest</li> <li>1 - Available</li> <li>2 - Work</li> <li>3 - Drive</li> <li>6 - ERROR</li> <li>7 - n/a</li> </ul>

FMS Variables	Names	Responses and descriptions
FMS.TC_DRV1_TIME	Driver 1 time related states	Reports the current driver 1 time related state: 00 - Normal 01 - 15 min bef. 4½ h 02 - 4½ h reached 03 - 15 min bef. 9 h 04 - 9 h reached 05 - 15 min bef, 16 h 06 - 16 h reached 09 - Other 14 - ERROR 15 - n/a
FMS.TC_DRV2_CARD	Driver 2 card	Reports the current driver 2 card status: 0 – Not present 1 – Present err – Device error n/a – Value not available
FMS.TC_DRV2_STATE	Driver 2 working state	Reports the current Driver 2 working state: 0 - Rest 1 - Available 2 - Work 3 - Drive 6 - ERROR 7 - n/a
FMS.TC_DRV2_TIME	Driver 2 time related states	Reports the current driver 2 time related state: 00 - Normal 01 - 15 min bef. 4½ h 02 - 4½ h reached 03 - 15 min bef. 9 h 04 - 9 h reached 05 - 15 min bef, 16 h 06 - 16 h reached 09 - Other 14 - ERROR 15 - n/a
FMS.TC_EVENTS	System events	0 – No system events 1 – System events err – Device error n/a – Value not available
FMS.TC_HANDLING	Handling information	0 – No handling information present 1 – Handling information present err – Device error n/a – Value not available
FMS.TC_MOTION	Vehicle motion	Reports the current vehicle motion status: 0 – Not moving 1 – Moving err – Device error n/a – Value not available

FMS Variables	Names	Responses and descriptions
FMS.TC_OVERSPEED	Vehicle overspeed	Reports whether or not the vehicle speed exceeds the speed limit registered in digital tachograph memory: 0 – Not exceeding 1 – Exceeding err – Device error n/a – Value not available
FMS.TC_PERF	Tachograph performance	Reports the current tachograph performance status: 0 – Normal performance 1 – Performance analysis err – Device error n/a – Value not available
FMS.TC_SHAFT_SPEED	Tachograph shaft speed output	Reports the calculated output shaft speed in rotations per minute (R.P.M.).
FMS.TC_SPEED	Tachograph speed vehicle	Reports the calculated output shaft speed in rotations per minute (R.P.M.).
FMS.TC_SPEED_KMPH	Tachograph speed vehicle	Reports the calculated output shaft speed in rotations per minute (R.P.M.).
FMS.TC_STATE	//	Consolidated driver state (first 4 bytes of PGN FE6C )
FMS.THROTTLE	//	Throttle position - 0.4%/bit
FMS.THROTTLE2	//	Throttle position2 - 0.4%/bit
FMS.TORQUE_DRV_DEMAND	//	Driver's demand engine percent torque - 1%/bit offset -125%
FMS.TORQUE_ENG_DEMAND	//	Engine demand percent torque - 1%/bit offset -125%
FMS.TOTAL_FUEL_USED	//	High resolution engine total fuel used
FMS.TURBO_OIL_TEMP	//	Turbo oil temperature - 1/32 °C/bit (-273° offset)
FMS.VEHICLE_DIST	High resolution total vehicle distance	Reports the high resolution total vehicle distance in meters (m).
FMS.VEHICLE_ID	Vehicle identification number	Reports the vehicle identification number.
FMS.VER_DIAG_SUPP	Diagnostics supported	Reports the diagnostics state. 0 – Diagnostics not supported 1 – Diagnostics supported err – Device error n/a – Value not available
FMS.VER_REQU_SUPP	Requests supported	Reports the requests state. 0 – Requests not supported 1 – Requests supported err – Device error n/a – Value not available
FMS.VERSION	Software version	Reports the FMS software version as a string.
FMS.WASHER_LVL	//	Washer fluid level - 0.4%/bit

FMS Variables	Names	Responses and descriptions
FMS.WEIGHT	Axle weight	Reports the vehicle weight based on axis(axle). Format: <axis>:<weight>{,<axis>:<weight>}... <axis>: hexadecimal index of axis (might depend on car type and number of axis available) <weight>: decimal weight in kg (rounded down to full kg) Note: if working with simulated values, messages have to be resent within 20 seconds before reading this protocol, else output will be reset. This allows to i.e. add /remove a trailer (hanger) to/from a lorry and readout updated values.
J1939.FUEL_ECO_AVRG	Average fuel eco	Used to report the average fuel consumption in l / 1000 km (litre/1000 kilometres). Not in FMS-standard.
J1939.FUEL_ECO_INST	Instant fuel eco	Used to report the instant fuel consumption in l / 1000 km (litre/1000 kilometres). Not in FMS-standard.
J1939.FUEL_TRIP	Trip fuel used	Used to report the used fuel in l (litre). Trip based not in FMS-standard
J1939.PARK_BRAKE_SWITCH	Parking brake switch	Used to report the current status of the parking brake switch ( <i>additional - it is not in FMS-standard</i> ) 0 – Parking brake switch is off 1 – Parking brake switch is on err – Device error n/a – Value not available
J1939.VEHICLE_DIST_TRIP	High resolution trip vehicle distance	Used to report the high resolution total vehicle distance in m (meters). Not in FMS-standard.

## 4.2 Dynamic variables for OBD-II messages

To transmit them to a remote server, use this command:

```
$PFAL,MSG.Send.TCP,8,"identifier = &(OBDII<PID_hex>) identifier2= &(OBDII<PID_hex>) ...."
```

e.g. to report the vehicle speed and the engine RPM use this command:

```
$PFAL,MSG.Send.TCP,8," SPEED: &(OBDII0D) BRAKE:&(OBDII0C)"
```

OBD-II Parameters	Responses/Description
OBDII<PID_hex>	Reports the state of the already requested PID on the vehicle bus. For more details about the OBD-II PIDs, see <a href="https://en.wikipedia.org/wiki/OBD-II_PIDs">OBD-II PIDs</a> on the wikipedia.org.