FALCOM

Application Notes

CAN FMS/J1939

CAN OBD-II

Version: 1.0.4; Created: Thursday 15 March 2018

Version history:

This table provides a summary of the document revisions.

Version	Author	Changes	Modified
1.0.4	F. Beqiri	- Updated section 2.4.1 and removed section 2.4.2.	15/03/2018
1.0.3	F. Beqiri	- Added new FMS variable (dynamic entries) - FMS_HR_TOTAL_FUEL_USED	29/08/2016
1.0.2	F. Beqiri	- Added new FMS variable (dynamic entries)	29/10/2014
1.0.1	F. Beqiri	- Extended chapter 1 and added chapter 2	07/08/2012
1.0.0	F. Beqiri	- Initial version	01/03/2012

Table of contents

1 INTRODUCTION 5

2 REQUIREMENTS FOR READING DATA ON THE VEHICLE BUS VIA OBD-II/FMS/J1939 . 6

2.1 FALCOM AVL devices with CAN-Bus option 6

2.2 Installing the SIM card 7

2.3 Connecting AVL device to the vehicle bus 7

2.4 Configuring devices for reading vehicle information (examples) 8

2.4.1 How to find out, which OBD-II PIDs are supported by a vehicle? 10

3 COMMANDS AND CONFIGURATION SYNTAX 11

3.1 Configure and activate OBD-II, FMS or J1939 interface 11

3.2 Firmware configuration 12

3.3 OBDII, FMS, J1939 Events 14

3.3.1 OBDII event 14

3.3.2 FMS event 14

3.3.3 J1939 event 15

4 DYNAMIC PROTOCOL 16

4.1 Dynamic protocols for FMS/J1939 16

4.2 Dynamic protocol for OBD-II 21

Cautions

Information furnished herein by FALCOM is believed to be accurate and reliable. However, no responsibility is assumed for its use. Please, read carefully the safety precautions.
If you have any technical questions regarding this document or the product described in it, please contact your vendor.
General information about FALCOM and its range of products are available at the following Internet address: http://www.falcom.de/

Trademarks

Some mentioned products are registered trademarks of their respective companies.

Copyright

FALCOM GmbH.

The manufacture assumes no liability for any errors or discrepancies that may have occurred in preparation of this document.

Note

Specifications and information given in this document are subject to change by FALCOM without notice.

1 INTRODUCTION

This document describes how to get CAN-Bus information from a vehicle using FALCOM AVL devices with CAN-Bus option.

FALCOM AVL devices currently support three major vehicle bus protocols: OBD-II, J1939 and FMS. FALCOM AVL devices with CAN-Bus option use two lines (CAN-High and CAN-Low) to connect to an OBD-II, J1939 or FMS interface which are connected to a vehicle and access the data available on the **E**lectronic **C**ontrol **U**nits (ECU) of the vehicle. The data can be transferred in real-time to a remote server for further analysis.

A vehicle bus protocol is a specialized internal communications network that interconnects components inside a vehicle. It also provides interface for users to acquire vehicle data for travel information such as 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 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 FALCOM but they can be ordered from external suppliers.

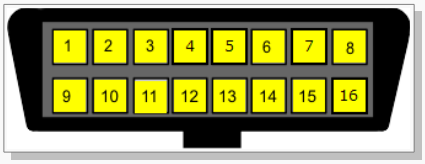
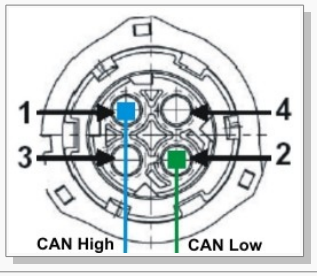
Interface	Interface connector	Connector Pinout
OBD-II		16-pin (2x8) connector interface. **Pin Nr: Description** 4 Common ground vehicle 6 CAN HIGH 14 CAN LOW 16 Battery power (+) (Not available on all cars)
J1939		9-pin round connector interface. **Pin Nr: Description** A Common ground vehicle C J1939 + D J1939 - B Battery power (+) (Not available on all cars) ! Vehicles that have a 6pin connector, instead of a 9pin connector, only support the J1708 protocol.
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 ! Vehicle manufacturers are providing a different connector other than the standard.

Table 1: CAN-Bus Vehicle interface connectors

2 REQUIREMENTS FOR READING DATA ON THE VEHICLE BUS VIA OBD-II/FMS/J1939

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

1. A Falcom AVL device with CAN-Bus opton (e.g. STEPPIII-UX-CH-B1),
2. A SIM card supportng GPRS for transmitng vehicle data over GPRS to a server,
3. Device confguraton setngs for getng informaton from the vehicle bus,
4. Connectng AVL devices to the vehicle bus with the help of vehicle installaton cable,
5. Vehicle supportng SAE J1939, OBD-II or FMS,
6. Power supply from vehicle battery.

2.1 FALCOM AVL devices with CAN-Bus option

The following FALCOM AVL devices can be supplied on request with CAN-Bus.

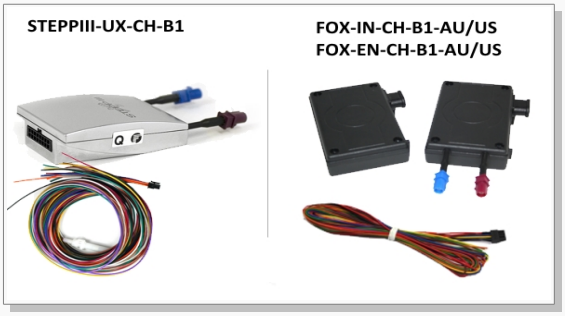


Figure 1: AVL devices with CAN Bus option

The CAN-Bus interface in the FALCOM FOX3 series belongs to the hardware PAID-Features. To use it with FOX3 you have to activate it. Please refer to the AppNote "AppNotes_HowToActivatePremiumFeatures_vxx.pdf" available on our website: http://www.falcom.de/support/documentation/application-notes/.

2.2 Installing the SIM card

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

2.3 Connecting AVL device to the vehicle bus

To connect a Falcom AVL device with CANBus option to the vehicle bus, you need:

- A 16pin (2x8) male connector that plugs in to the vehicle's OBD-II female diagnostic connector or an FMS connector that plugs in to the FMS - Gateway connector in the vehicle.
- CAN-High, CAN-Low, Ground and V+ lines from the Falcom AVL device.

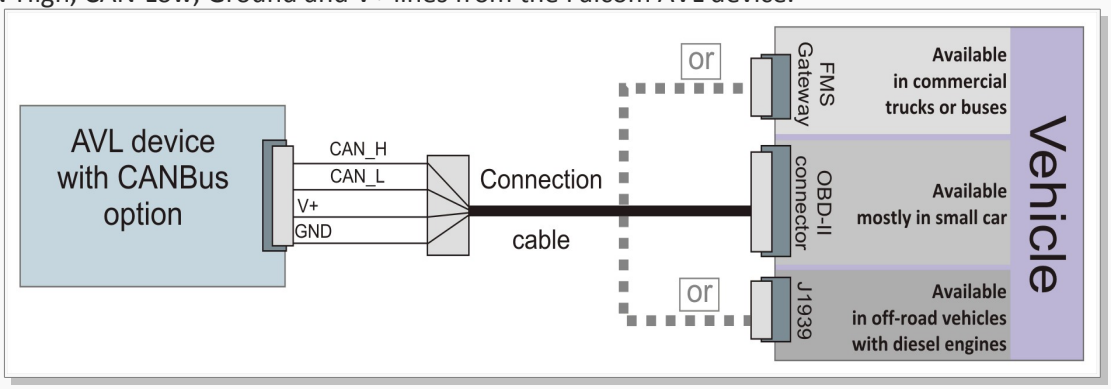


Figure 2: Connecting AVL devices to vehicle CAN Bus

Falcom AVL devices support two CAN bus signal pins CAN_High and CAN_Low. A cable called "Vehicle installation cable" is shipped with the device and used to connect your device to the vehicle. The polarity of the CAN_High and CAN_Low lines must be observed when connecting your AVL device to the vehicle bus.

The following table lists the CAN interface lines of the STEPPIII-UX and FOX-IN/EN vehicle installation cable:

***S*TEPPIII-UX-CH-B1**		
Color	**IO**	**Meaning**
Grey	DI0	CAN_H (dominant HIGH)
White	DI1	CAN_L (dominant LOW)
Brown	GND	Ground
Red	V+	Input voltage (+10.8...32.0V DC)
FOX3/FOX-IN/EN-CH-B1-AU/US		
Color	**IO**	**Meaning**
Green	IO2	CAN_H (dominant HIGH)
Yellow	IO3	CAN_L (dominant LOW)
Brown	GND	Ground
Red	V+	Input voltage (+10.8...32.0V DC)

Table 2: CAN interface lines on the AVL devices with CAN Bus option

Locating the connection points on the vehicle:

	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.
NO PICTURE	The FMS interface gateway and the connectors are located behind the dashboard. Contact your vehicle manufacture for more details.

2.4 Configuring devices for reading vehicle information (examples)

To read messages on the vehicle bus, you have to connect first the AVL device to the vehicle bus as shown above and then activate the CAN interface and enable the FMS or OBD-II interface on the AVL device. **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 3.2.

a) When connecting to the vehicle OBD-II connector use the configuration below:

Activate CAN bus functionality and set the correct communication baudrate on AVL device.
$PFAL,Sys.CAN.Enable,500K,RW
Enable OBD-II functionality on AVL device.
$PFAL,Sys.CAN.OBDII.Enable
To give alerts when Engine Speed enters specific ranges, first enter the PID of the Engine Speed (0C) and specify the speed ranges in which the OBD-II events should be generated. Example below shows that *Sys.eOBDII=0,0* and *Sys.eOBDII=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=OBDII.0C,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.eOBDII=0,0:TCP.Client.Send,8,"R.P.M. &(OBDII0C) < 3500" $PFAL,CNF.Set,AL26=Sys.eOBDII=0,1:TCP.Client.Send,8,"R.P.M. &(OBDII0C) > 3500"
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.
$PFAL,Cnf.Set,DEVICE.CAN.EVENT_1=OBDII.0D,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.eOBDII=1,0:TCP.Client.Send,8,"Speed &(OBDII0D) < 80 kmph" $PFAL,CNF.Set,AL28=Sys.eOBDII=1,1:TCP.Client.Send,8,"Speed &(OBDII0D) > 80 kmph"

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

Activate CAN Bus and set the communication baudrate 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
To give 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,BL0
See alarm configuration below to know how to report via TCP the sate 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)"

2.4.1 How to find out, which OBD-II PIDs are supported by a vehicle?

This external link "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 your are interested in. Finally, send the command with the defined <pid_hex> to the FALCOM device.

PFAL Syntax: $PFAL,Sys.Can.OBDII.Request,<pid_hex>
Execute: $PFAL,Sys.Can.OBDII.Request,0C // Requests if **Engine RPM** is supported
Execute: $PFAL,Sys.Can.OBDII.Request,0D // Requests if **Vehicle speed** is supported
Execute: $PFAL,Sys.Can.OBDII.Request,1F // Requests if **Run time since engine start** is supported

after that execute the command below to see the report:

Execute: $PFAL,MSG.Send.Serial0,0,"&(OBDIIOC); &(OBDII0D); &(OBDII1F)"

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

If the device outputs **&(ERR)** or **&(n/a)**, then eith the AVL device 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 COMMANDS AND CONFIGURATION SYNTAX

3.1 Configure and activate OBD-II, FMS or J1939 interface

To activate the CAN bus on the AVL device and set the communication baud-rate use one of the following commands listed in table below. Following modes are supported by AVL devices which indicate if the message will be a Read, ReadWrite or Test:

CAN Modes	PFAL Command	Description
ReadOnly	**$PFAL,Sys.CAN.Enable,250K,RO**	This mode should be enabled for FMS and J1939 messages which do not need external request. Some FMS messages need external request so they required the **ReadWrite (RW)** option. **Hint:** This setting is NOT usable for OBD-II. It works for most FMS/J1939 messages (except some specific messages like: vehicle weight etc.).
ReadWrite	**$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. **Hint:** 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. **Hint**: 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.

2) Enable 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 Firmware configuration

These settings can be used to generate events out of the changed CAN variables without checking them periodically for changes. AVL devices provide the possibility to analyse 10 slots allowing 10 different CAN messages to be analysed 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 OBDII-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 thee CAN bus protocols supported by FALCOM AVL devices.

Value	*Meaning*
OBDII	Self-diagnostic and reporting capability of a vehicle used in cars and light trucks.
FMS	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.
J1939	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 4, Dynamic protocol .
  - 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** then:	1st Syntax: <eventX>=<s_type><event_edge><s_info_id>	
	<event_edge>	**H** Event occurs when Bit changes from 0 -> 1 **L** 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:** **BH2** Event occurs when Bit changes from 0 -> 1 within 3rd information inside configured message. **BL0** Event occurs when Bit changes from 1 -> 0 within 1st information inside configured message.
	Example:	<eventX>=BH2
If **<s_type>=V** 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. **For example:** A request for PID "4F" (see also OBD-II PIDs) returns 4 pieces of information: **Maximum value for equivalence ratio**, **oxygen sensor voltage**, **oxygen sensor current**, and **intake manifold absolute pressure**. So, if you want to generate an event for **oxygen sensor voltage** then specify here **1,** for **Maximum value for equivalence ratio** specify **0** and for **oxygen sensor current** specify **2**. (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

Info: Using these parameters it is possible to define a kind of hysteresis.

For example engine R.P.M.: 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:

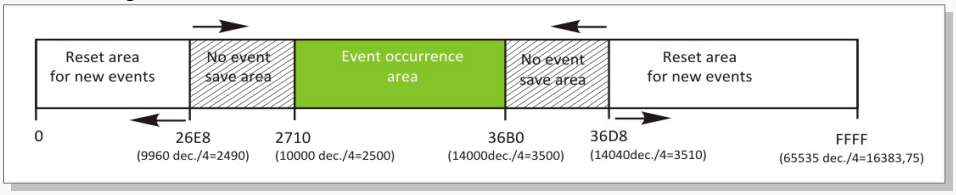
<rst_Lmax>: 26E8 (hex) = 9960 dec. / 4 = 2490 R.P.M.

<minval>: 2710 (hex) = 10000 dec. / 4 = 2500 R.P.M.

<maxval>: 36B0 (hex) = 14000 dec. / 4 = 3500 R.P.M.

<rst_Hmin>: 36D8 (hex) = 14040 dec. / 4 = 3510 R.P.M.

The range in R.P.M. from <rst_Lmax>=2490 to <minval>=2499 and <maxval>=3501 to <rst_Hmin>=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.



The example corresponding to the diagram above:

$PFAL,Cnf.Set,DEVICE.CAN.EVENT_0=OBDII.0C,1,V0,26E8,2710,36B0,36D8

3.3 OBDII, FMS, J1939 Events

3.3.1 OBDII event

Syntax: Sys.eOBDII=<event_slot>,<sub_event>

Example: Sys.eOBDII=0,0 *// event for the slot 0, index 0*

Functionality: This event is generated whenever a configured OBD-II event occurs. See chapter 3, "Commands and Configuration Syntax" for more details.

Parameters: <event_slot>,<sub_event>

<event_slot> It specifies the index of the event slot that generates the desired event. See manual "*AVL_PFAL_Configuration_Command_Set.pdf*" for more details.

<sub_event> It specifies the event number (index of <eventX>) configured within this event slot. See manual "*AVL_PFAL_Configuration_Command_Set.pdf*" for more details.

3.3.2 FMS event

Syntax: Sys.eFMS=<event_slot>,<sub_event>

Example: Sys.eFMS=0,0 *// event for the slot 0, index 0*

Functionality: This event is generated whenever a configured FMS event happens.See chapter 3, "Commands and Configuration Syntax" for more details.

Parameters: <event_slot>,<sub_event>

<event_slot> It specifies the index of the event slot that generates the desired event. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details.

<sub_event> 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 J1939 event

Syntax: Sys.eJ1939=<event_slot>,<sub_event>

Example: Sys.eJ1939=0,0 *// event for the slot 0, index 0*

Functionality: This event is generated whenever a configured J1939 event happens. See chapter 3, "Commands and Configuration Syntax" for more details.

Note that, J1939 is supported partially only:

1. J1939 is supported for all periodic messages.
2. 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).

Parameters: <event_slot>,<sub_event>

<event_slot> It specifies the index of the event slot that generates the desired event. See manual "AVL_PFAL_Configuration_Command_Set.pdf" for more details.

<sub_event> 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 PROTOCOL

4.1 Dynamic protocols for FMS/J1939

These dynamic protocol (PID) names can be used in messages to display values of FMS variables. Please refer to FMS specification for detailed information and message formats:

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 Variables	Names	Responses and descriptions
FMS.ENG_HR	//	Total engine revolutions
FMS.ENG_LOAD	//	Engine percent load at current speed
FMS.ENG_MAX_TORQUE	//	Actual maximum available Engine percent torqe
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	Reports the current status of the power take off governor. **00** – Off / disabled **01** – Hold **02** – Remote hold **03** – Standby **04** – Remote standby **05** – Set **06** – Decelerate/Coast **07** – Resume **08** – Accelerate **09** – Accelerator override **10** – Preprogrammed set speed 1 **11** – Preprogrammed set speed 2 **12** – Preprogrammed set speed 3 **13** – Preprogrammed set speed 4 **14** – Preprogrammed set speed 5 **15** – Preprogrammed set speed 6 **16** – Preprogrammed set speed 7 **17** – Preprogrammed set speed 8 **18** – PTO set speed memory 1 **19** – PTO set speed memory 2 **31** – Value not available
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	Reports the motion direction: **0** – Forward **1** – Reverse **err** – Device error **n/a** – Value not available
FMS.TC_DRV1_CARD	Driver 1 card	Reports the current driver 1 card status: **0** – Not present **1** – Present **err** – Device error **n/a** – Value not available

FMS Variables	Names	Responses and descriptions
FMS.TC_DRV1_STATE	Driver 1 working state	Reports the current Driver 1 working state: **0** - Rest **1** - Available **2** - Work **3** - Drive **6** - ERROR **7** - n/a
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 Variables	Names	Responses and descriptions
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.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 output shaft speed	Reports the calculated output shaft speed in rotations per minute *(R.P.M.).*
FMS.TC_SPEED	Tachograph vehicle speed	Reports the calculated output shaft speed in rotations per minute *(R.P.M.).*
FMS.TC_SPEED_KMPH	Tachograph vehicle speed	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 Variables	Names	Responses and descriptions
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.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 (*additional - it is not in FMS-standard*) **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.

Table 3: Supported FMS parameters

4.2 Dynamic protocol for OBD-II

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 OBD-II PIDs on the wikipedia.org.