FMS OEM CHIP V7
Remote firmware update and differences from previous version V6

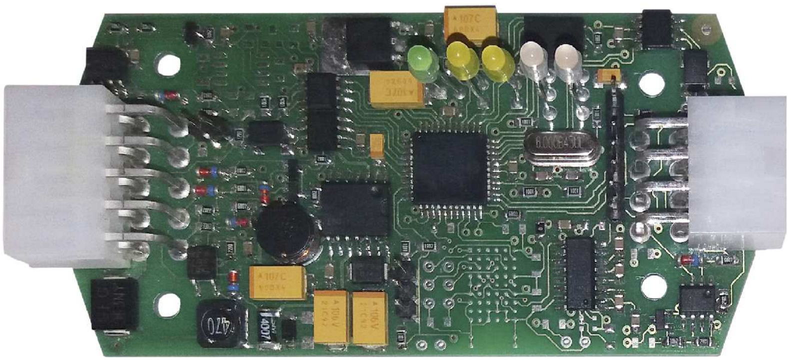
Ing. David Španěl
Mgr. Vítězslav Rejda
CANLAB s.r.o.

Content:

Basic description

FMS OEM CHIP V6 is preprogrammed microcontroller designed for integration into system for car monitoring, systems for fleet management, etc. Chip preprocesses data from car CAN bus, from digital tachograph and J1708/J1587 bus (VOLVO), IBIS bus and others, it provides decoded data via serial link.

Chip is supplied separately, or FW can be uploaded into MCU in client´s device or installed in CAR2COM device (classic serial link or TTL), CAR2USB (FTDI cable), CAR2BT (bluetooth).



Firmware Update

Firmware update is made by serial link from superior system. Update can be made during normal operation of FMS OEM CHIP and its functions are not significantly limited in this moment. For update, external FLASH memory is used, where new firmware image is downloaded during operation. When new firmware is completely sent, flag is entered into the memory that memory contains new firmware. Firmware is installed from FLASH memory to processor during start of processor, within switch on or after software reset.

Update Commands

Firmware version

Firmware version and identification string are sent automatically from chip after its start or after request. Request has following format:

*$PCAN,C,VER,*32<enter>*

Response can have e.g. following form:
*$PCAN,C,VER,1.01,FMS_CHIP_V7,CANLABsro,????????????,*6A<enter>*

FW version is 1.01, it is FMS OEM CHIP version 7. Identification chain was not set.

Preparation of FLASH memory

It is necessary to delete FLASH memory before update initiation. It is made by following command:

*$PCAN,H,ERASE,*38<enter>*

Chip confirms acceptation by following response:

*$PCAN,H,ERASE,OK,*10<enter>*

In case of using of little configuration FLASH memory update is not supported and chip returns following response:
*$PCAN,H,ERASE,ERROR,*4C<enter>*

Instruction sequence record

Firmware is saved in file with HEX suffix. This file has to be read and decoded.

Example of part of HEX file:

```
:10FFE00065B4220084C0B300844A78000B00370057
:10FFF000EE42900004CEB30084CF520004003E00D5
:020000040001F9
:1000000065B4220094C0B300844A7800030037002E
:1000100065B42200A4C0B300844A7800A4992200E9
        --------
           I1    --------
                    I2    --------
                             I3    --------
                                     I4
```

Each line of HEX file starts with colon sign. Information about number of data in bytes follows – just data are calculated into this information – information about address and type of record is not included. All data is stated in hexadecimal form.
Information about address follows – it has just two bytes, therefore it is possible to address only in the range of 0..65535. Data record is marked by type of record 00. 04 record type is used to address bigger space – address expansion. Data of this record means number of page with size of 65536 bytes. Real address is calculated: A = address on data line + number of page*65536.

10 – length of data
FFE0 – address
00 – type of data record
04 – type of address expansion record
0001 – address expansion data, offset 1*65535 is added to addresses from this position
65B42200A4C0B300844A7800A4992200 – data - instruction
E9 – control sum – sum of byte values of line modulo 256 must be zero.

For used processor, instruction is always represented as 32bit number. Bytes are stated in inverse sequence, therefore it is necessary to change endian. For example, instruction D0280000 has to be changed to 000028D0.

Records with different type of record than 00 or 04 can be ignored.

It is also necessary to secure that lines with 0 address or address higher than 0xAA000 are ignored.

Example of HEX line of file 1:
```
:10000800D0280000DA28000000290000E2280000BB
```
Command:
*$PCAN,H,HEX,A00000008,I000028d0,I000028da,I00002900,I000028e2,*4F*<enter>
Response:
*$PCAN,H,HEX,OK,A00000008,*60*<enter>

Example of HEX line of file 2:
```
:10001800EC280000F6280000002900000029000054
```
Command:
*$PCAN,H,HEX,A00000018,I000028ec,I000028f6,I00002900,I00002900,*1F*<enter>
Response:
*$PCAN,H,HEX,OK,A00000018,*61*<enter>

Example of HEX line of file 3:
```
:10002800002900000029000000290000D2880000F3
```
Command:
*$PCAN,H,HEX,A00000028,I00002900,I00002900,I00002900,I000088d2,*17*<enter>
Response:
*$PCAN,H,HEX,OK,A00000028,*62*<enter>

If error of record to FLASH occurs – different value is read than was recorded, error is signalized by following message:
*$PCAN,H,HEX,ERROR,WRITE,*28*<enter>

If an error of data format is detected it is signalized by following message:
*$PCAN,H,HEX,ERROR,CMD,*3F*<enter>

From FW 1.60 version, for FW record it is also possible to use a variant, when more lines of HEX file are sent in one packet. Length of such packet is limited by maximum length of packet, which corresponds to 580 characters. Identifier HEX in command is replaced by identifier HE2. Confirming command uses identifier HE2, as well. Furthermore, confirming command doesn't contain address of the first instruction.

*$PCAN,H,HE2,A00038ed0,I003a000c,I00242694,I00784214,I00fb8204,A00038ee0,I00780104,I0023d8c1,I0023dcc0,I000233ba,*1F*<enter>

Behind each address, there can be 1 up to 4 instructions.

*$PCAN,H,HE2,OK,*6F*<enter>

Record of flag of firmware loading completion

By this command, flag that new FW is saved in FLASH is recorded into FLASH memory and during chip restart it shall be loaded into memory of processor.
*$PCAN,H,REBOOT,*79*<enter>

Chip confirms reception by following response:

*$PCAN,H,REBOOT,OK,*51*<enter>

For setting reset into original state you can use following command:

*$PCAN,C,RESET,*26*<enter>

This command sets FMS OEM CHIP into original state (setting of data, gauge, CAN and so on) and both CANs are in Listen only regime.

IBIS

IBIS is bus used for communication of driver´s check-in device with information panels in buses. From this bus, it is possible to learn information about connection, routeline, stop, and so on.

*$PCAN,U,SET,MODE,I,*6D<enter>*
Setting of UART2 line to IBIS regime.

*$PCAN,U,GET,MODE,*1C<enter>*
Request for mode.

*$PCAN,I,GET,I@014@840105@036@M@,*4F<enter>*
Number of line is 840105, inter-urban.

*$PCAN,I,GET,I@019@101@001@S@,*60<enter>*
Character S – detected sign on board "Service ride".

For implementation of reading of IBIS bus it is necessary to consult setting of devices in buses with carrier.

RFID

If SECAR RFID reader is used, it is possible to connect it to UART2. To this regime, link is set by following command:
*$PCAN,U,SET,MODE,R,*76<enter>*

If data is received from reader, this data is sent via serial link to superordinate system in following form:
$PCAN,R,FID,<data>,<checksum><enter>*

Reading of DM1 error messages at cars with J1939 protocol

In case of cars that use J1939 protocol (trucks), chip saves DM1 error codes that are detected at CAN bus. Error codes can be requested any time by sending a request. After reading, errors are deleted in a table and table is gradually filled from the beginning.

DM1 error codes request:
$PCAN,C,DM1,*4B

It there are no error codes detected, chip responses by following message:

$PCAN,C,DM1,EMPTY,*32

In the opposite case the response is messages sequence, when each message corresponds to one error code:

$PCAN,C,DM1,E0,37,70,9,9,*11
Data in decimal form:
37 – ECU address
70 – SPN error
9 – FMI
9 – OC

$PCAN,C,DM1,E1,47,62,9,1,*1C

$PCAN,C,DM1,E2,37,190,9,8,*2D

Alternatively, method of reading of error codes one by one can be used. Number of items can be determined by following command:

$PCAN,C,DM1,SIZE,*62

Chip responses with number of error codes:

$PCAN,C,DM1,SIZE,5,*7B

Subsequently, it is possible to read messages using POP request:

$PCAN,C,DM1,POP,*28

Unit responses with one message with error code:

$PCAN,C,DM1,POP,11,84503,31,129,*2A

We repeat POP request, until all codes are read. Receiving of new DM1 codes is interrupted, until all saved codes are read. If there is no error code saved, it responds:

$PCAN,C,DM1,EMPTY,*32
FMS OEM CHIP version 7 supports possibility to set data reading regime by POP command, when message is marked as read after its reading, however, chip remembers it till switch off (or deleting by command) and ignores other occurrences of this error. Therefore the same error is not repeatedly read by POP command, if chip detects it again.
Classic regime deletes message after its reading by POP regime, it deletes error from its memory and in case of another occurrence it includes it repeatedly between errors read via POP.

Regime is set by following command:
$PCAN,C,DM1,FLG,1,*37
It is possible to cancel it and set classic regime via following command:
$PCAN,C,DM1,FLG,0,*36
Request for setting:
$PCAN,C,DM1,FLG,?,*39

Value 0/1 is bit flag. It is possible that in future it will be necessary to combine it with flags for other functions.

DM2 error messages reading at cars with J1939 protocol

From FW version 1.67

DM2 error messages were formerly stated as active error codes (saved error codes). Reading of DM2 error codes is the same as in case of DM1, there are just few differences:

- DM2 can be deleted
- For each error code, there is a CAN port from which the error code was received
- Error codes are requested by sending a request to relevant ECU

Tank value calibration

Tank calibration 0:
Value = measured_value*(fuel_level_scale/100.0) - fuel_level_offset[0];

*$PCAN,C,FLS,0,O:50,S:200,*1D<enter>*

Tank calibration 1:
*$PCAN,C,FLS,1,O:100,S:2000,*18<enter>*

Calibration reset
*$PCAN,C,FLS,RSS,*54<enter>*

Request for tank calibration 0

*$PCAN,C,FLS,0,O:0,S:100,*2B<enter>*

Example:
Full tank without set calibration (calibration reset) gives value 255.
Empty tank gives value 0.
*Therefore Offset (*fuel_level_offset*) is 0.*

We request full tank in percentage:
fuel_level_scale = requested value *100/indicated value of full tank
fuel_level_scale = 100*100/255
fuel_level_scale = 39.2156, rounded to 39
$PCAN,C,FLS,0,O:0,S:39,*10
Command:
$PCAN,C,FLS,0,O:0,S:18,*13

We request full tank in liters, full tank is 45 liters.
fuel_level_scale = requested value *100/indicated value of full tank
fuel_level_scale = 45*100/255
fuel_level_scale = 17,64, rounded to 18
Command:
$PCAN,C,FLS,0,O:0,S:18,*13

Calibration of calculated fuel rate per ride

fuel_rate_scale= measured_value (trip_fuel_scale/10000000.0);

*$PCAN,C,FTS,0,S:1000,*6A<enter>*

*$PCAN,C,FTS,0,?,*3D<enter>*

*$PCAN,C,FTS,RSS,*4C<enter>*

Calibration of passed distance per ride

If there is in the car any form of information about passed distance per ride (value D), it is possible to set scale for this information if the information is not provided correctly by FMS chip.
Data is scaled according to following formula:
Value = measured_value*(trip_distance_scale/ 1000000.0)

*$PCAN,C,DTS,0,S:1000,*68<enter>*

*$PCAN,C,DTS,0,?,*3F<enter>*

*$PCAN,C,DTS,RSS,*4E<enter>*

Secondary CAN

In V7 version, secondary CAN is realized as native, therefore without using of CAN switch as in case of V6 version. Therefore different command is used for setting of this CAN:

*$PCAN,2,SET,C9,LISO,*03<enter>*
Letter C is replaced by number 2 (second CAN), speed 500k, standard identifiers, listen only regime.

*$PCAN,2,SET,C8,EXT,FMS,*06<enter>*
Speed 250k, extended identifiers, FMS output regime.

FMS CAN output

Version V7 allows to specify FMS command in $PCAN,2,SET command. In this regime, secondary CAN is used as FMS output for device with FMS input.

*$PCAN,2,SET,FMS,C9,*62*

CAN 2 is set to FMS output regime, speed of output is 500 kb/s.

Fuel level from secondary CAN

This setting can be used in case of connection to car, where additional fuel probe (float) was installed with CAN bus according to SAE J1939. Following command sets reading of secondary tank from CAN 2.

*$PCAN,2,SET,C8,EXT,FLS1,*36*

Therefore it is possible to combine any car connected to CAN 1 with fuel probe at CAN 2.

Switching off this setting:

*$PCAN,2,SET,C8,EXT,FLS0,*37*

During next setting primary and secondary fuel tank is read from CAN 2. Information at CAN 1 (if available) is ignored:

*$PCAN,2,SET,C8,EXT,FLS2,*35*

Signalizing by digital output

From FW version 1.65

Using digital output that is designed for awakening of digital tachograph some states can be signalized in case of FW without this function. For now, signalization of active PTO was added, if such information is available at CAN.

Request for setting:
*$PCAN,C,SIG,?,*3D*

Function switch off:
*$PCAN,C,SIG,0,*32*

Signalization PTO permit:
*$PCAN,C,SIG,P,*52*

Zeroing data after key switch off

From FW version 1.65

In case of some cars, after switching off a key, car still sends information about fuel tank state and power of float is off, therefore incorrect information is sent after switch off.
In case of key signal switch off due to this setting tank state is indicated as unknown.

*$PCAN,C,POX,L,?,*47*
*$PCAN,C,POX,L,0,*48*
*$PCAN,C,POX,L,1,*49*

Temporary setting of listen only regime

From FW version 1.65

If listen only regime is not active, however, it is necessary to activate it due to any reason, it is possible via this command. Change of listen only is not saved as change of configuration, after unit reset or restore, the original state is set.
Setting of LISO regime at primary CAN, switching on with value 1 or without value (this case).
*$PCAN,C,LISO,*6A*

LISO switch off on primary CAN
*$PCAN,C,LISO,0,*76*

LISO regime setting at secondary CAN, switching on with value 1 (this case) or without value.
*$PCAN,2,LISO,1,*06*

LISO switch off at secondary CAN
*$PCAN,2,LISO,0,*07*

From FW version 1.66

LISO regime is active also after reset of unit till restore.

Request for change of any configuration during setting of LISO regime changes state permanently – it is non-restorable! It applies also for separate setting of LISO regime at both CANs. If it is necessary to activate LISO at both CANs, it is necessary to use following variant:

*$PCAN,C,LISO,A,0,*1B*
*$PCAN,C,LISO,A,1,*1A*

These variants deactivate or activate LISO regime at both CAN buses simultaneously.

Setting of LISO regime also deactivates any requests for data sent to CAN.

Decoding of CAN data by user

From FW version 1.80

If it is necessary to decode any information about car differently than it is decoded standardly by FMS OEM CHIP, it is possible to change decoding by user. However, it is necessary that user himself identifies and decodes necessary information at CAN.

Decoding by user can be set for following data:
- Primary and secondary tank
- Total fuel
- Fuel per ride
- Total passed distance
- Distance per ride
- Moto-hours
- Revolutions
- Speed

Example:
$PCAN,R,L,1,ID,200,EXT,BIG,FI,8,LE,8,MU,1.0,OF,0,*01
L- tank
1- primary
ID – identifier of CAN message
EXT – 29bit identifier, if it is standard, it is not stated
BIG – big endian data format
FI- first bit (LSB)
LE – number of data bits
MU – multiplier
OF- offset, it is subtracted

If big endian is used, BIG flag is entered. LSB (the lowest bit) is always entered. Therefore, if 16bit value in DB0 and DB1 is saved in big endian format, value 8 is entered.
In case of little endian format, no flag is needed.
In case of message with 29bit identifier, EXT flag is added. In case of 11bit identifier nothing is stated.

Following identification is used for individual data:

Primary tank	L,1
Secondary tank	L,2
Fuel per ride 1	Y,1
Alternative fuel per ride	Y,2
Total fuel	F,1
Total passed distance	T
Distance per ride	D
Moto-hours	H
Engine revolutions	R
Car speed	S

If any of these quantities is set, for set type of car data is decoded using this principle. Result of decoding by internal algorithm for set type of car is ignored.
Data can be set, deleted, or verified if it is set. It is not possible to read setting again due to security of know-how of a customer, who uses FMS CHIP and enters this setting of decoding by user into the unit.

Reset of whole setting of CAN data decoding defined by user.
$PCAN,R,RST,*37

Banning of decoding defined by user. However, setting of items is still saved.
$PCAN,R,DIS,*3C

Allowance of decoding defined by user.
$PCAN,R,ENB,*2B

Request if decoding of primary fuel tank is set.
$PCAN,R,STA,L,1,*59

Response that decoding of primary fuel tank is set.
$PCAN,R,STA,L,1,1,*44

$PCAN,R,ERR,*27
Error message of FMS CHIP, if setting command is not successfully decoded.

Request for state if revolution decoding defined by user is set.
$PCAN,R,STA,R,*5A

Response, decoding is not set.
$PCAN,R,STA,R,0,*46

Setting of revolutions decoding. Revolutions in message with 11bit identifier, second and third data byte, little endian, multiplier 4.0, offset 0.
*$PCAN,R,R,ID,100,FI,8,LE,16,MU,4,OF,0,*20*

Power control

In recommended scheme, power source is controlled from two sources. First is external signal key (15) and second is signal from FMS OEM CHIP. When the power is activated by key, after transmission from bootloader to application FMS OEM CHIP starts to keep power source active, as well. After key switch off, FMS OEM CHIP can determine moment of switch off.

*$PCAN,C,PWR,10,*0B*
Command of setting of switch off delay for 10 seconds. Maximum value is 60,000 seconds.

*$PCAN,C,PWR,?,*35*
Request for actual setting of switch off delay.

*$PCAN,C,PWR,DWN,????????????,*7B*
Message sent by chip shortly before switch off. In this case, chip ID is not set and it is replaced by question marks.

*$PCAN,C,PWR,DWN,*57*
Command speeds up chip switch off after signal 15 disconnecting.

*$PCAN,C,PWR,RST,*5F*
Command resets processor with software.

From FW version 1.76

*$PCAN,C,PWR,KON,1,*5D*
Command activates regime, when switch off counter is reset and timeout countdown for switch off runs again from 0 at reception of message via command serial line.

*$PCAN,C,PWR,KON,0,*5C*
This regime switch off.

*$PCAN,C,PWR,KON,?,*53*
Request for set regime.

Accessible types of cars

From FW version 1.76

Request if car_type 16 is available in FW version in FMS OEM CHIP.
*$PCAN,C,AVA,16,*0E*

Response 1: car_type 16 is available.
*$PCAN,C,AVA,16,1,*13*

Request, if car_type 17 is available in FW version in FMS OEM CHIP.
*$PCAN,C,AVA,17,*0F*

Response 0: car_type 17 is not available.

*$PCAN,C,AVA,17,0,*13*

Tachograph – info interface

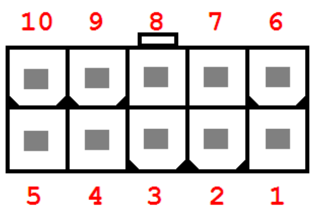
Setting of type of digital tachograph at connection of info interface at D8 pin.

VDO:
*$PCAN,T,SET,T1,P0+0,###,*37*

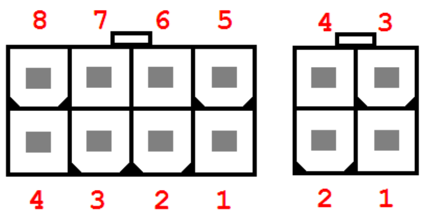
Stoneridge:
*$PCAN,T,SET,T2,P0+0,###,*34*
Tachograph Stoneridge requires activation of info interface and setting of correct regime:
RecordDataidentifier FD12 = 0x01 : Enable D8, Standard SRE output

After change of tachograph type it is necessary to restart unit/processor by key or command.

Connector wiring

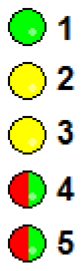


Pin	Description
1	Power 8-36V
2	Digital output, power switched on
3	CAN H
4	J1708 A
5	Tachograph A – signal
6	Signal 15 (startup-shutdown)
7	GND
8	CAN L
9	J1708 B
10	Tachograph B – GND



Pin	Description – CAR2COM	Description – CAR2BT
1	RX UART (incoming data)	CAN 2, high
2	CAN 2, high	GND
3	GND	CAN 2, low
4	UART 2 RX	GND
5	TX UART (outgoing data)	
6	CAN 2, low	
7	GND	
8	UART 2 GND	

Signal LED

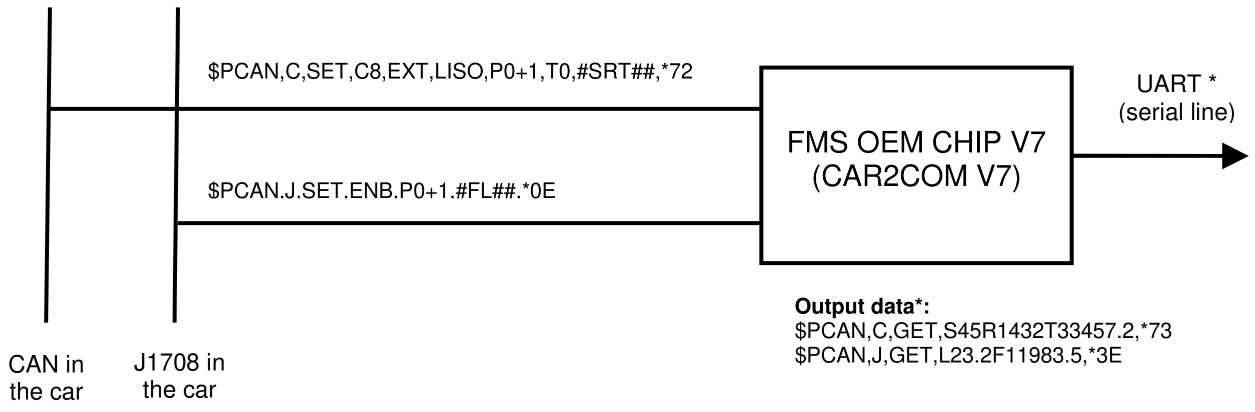


LED	Description
1	Power signalizing.
2	Data UART
3	Auxiliary UART (IBIS, Secar RFID)
4 red	J1708
4 green	CAN 2
5 red	Tachograph D8
5 green	CAN 1

Examples of less common usage

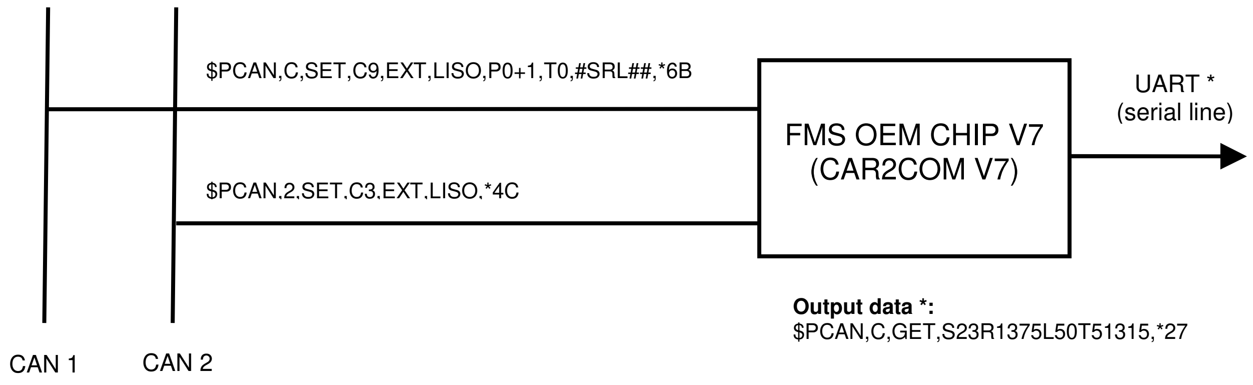
Car with CAN and J1708

Older Volvo and Renault trucks. At CAN, data about fuel are not available, it is read from J1708.


$PCAN,C,SET,C8,EXT,LISO,P0+1,T0,#SRT##,*72
$PCAN.J.SET.ENB.P0+1.#FL##.*0E
FMS OEM CHIP V7
(CAR2COM V7)
UART *
(serial line)
Output data*:
$PCAN,C,GET,S45R1432T33457.2,*73
$PCAN,J,GET,L23.2F11983.5,*3E
CAN in the car
J1708 in the car

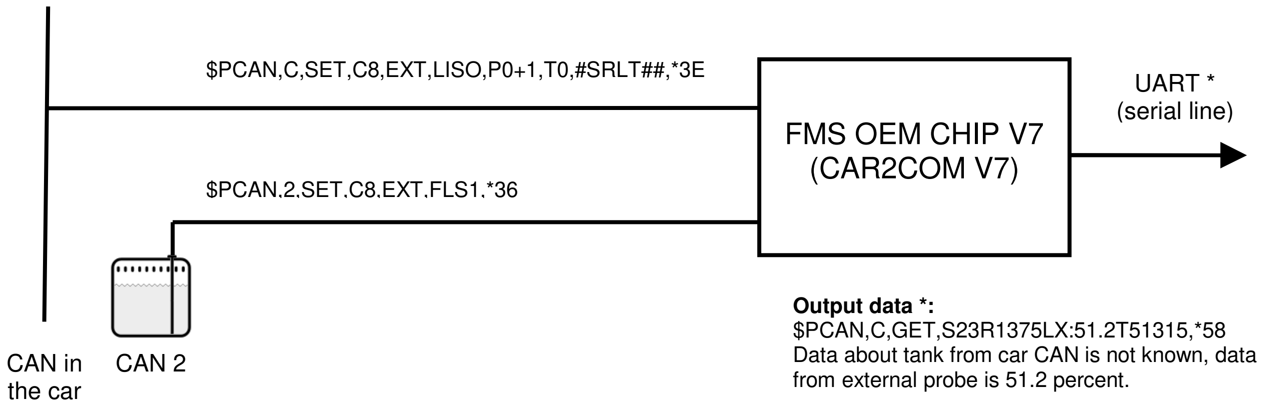
Car with 2 x CAN connection

In case of cars, where it is necessary to read data from two CANs. For example, at Fiat Ducato data about total kilometers is read from second CAN.


$PCAN,C,SET,C9,EXT,LISO,P0+1,T0,#SRL##,*6B
$PCAN.2.SET.C3.EXT.LISO.*4C
FMS OEM CHIP V7
(CAR2COM V7)
UART *
(serial line)
Output data *:
$PCAN,C,GET,S23R1375L50T51315,*27
CAN 1
CAN 2

Car CAN and external fuel probe at CAN

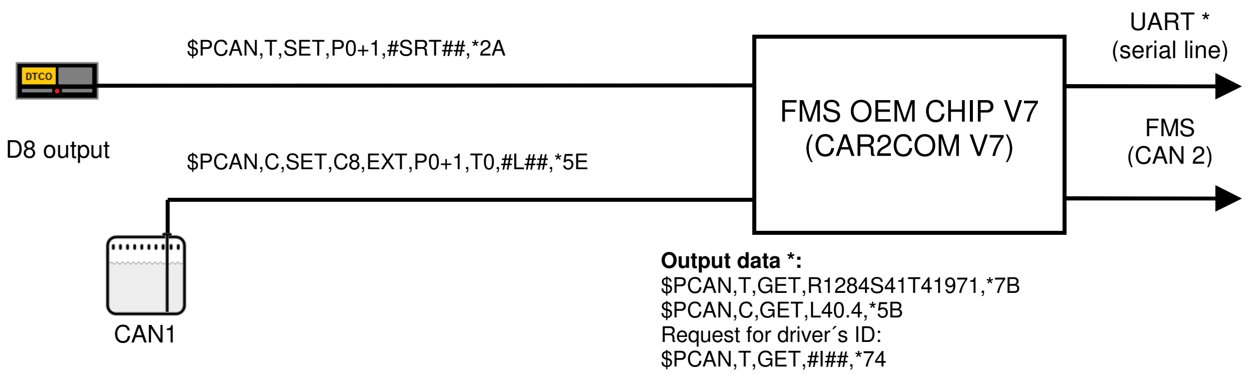
In situations, when there is not data about tank at car CAN and another fuel probe with CAN output is installed, or probe is installed due to accuracy or additional tank is added without possibility of measurement and probe is added.


$PCAN,C,SET,C8,EXT,LISO,P0+1,T0,#SRLT##,*3E
$PCAN,2,SET,C8,EXT,FLS1,*36
FMS OEM CHIP V7
(CAR2COM V7)
UART *
(serial line)
CAN in the car
CAN 2

Output data *:
$PCAN,C,GET,S23R1375LX:51.2T51315,*58
Data about tank from car CAN is not known, data from external probe is 51.2 percent.

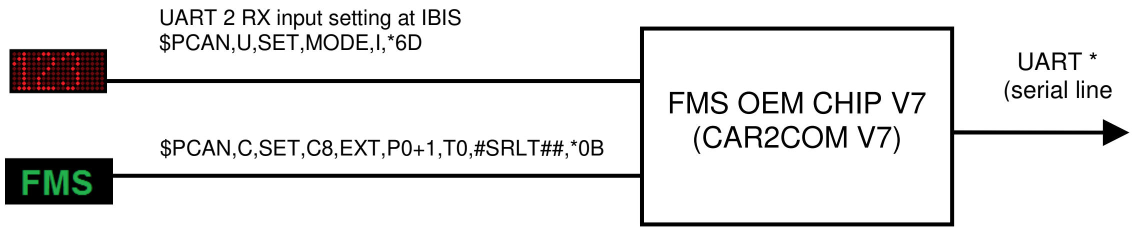
Car with digital tachograph and external fuel probe at CAN and emulated FMS output

Car without CAN bus, but with digital tachograph and added probe with CAN output. Speed, revolutions, and total kilometers are read from tachograph, tank is read from fuel probe. Simultaneously FMS output is emulated for another external device at CAN 2.



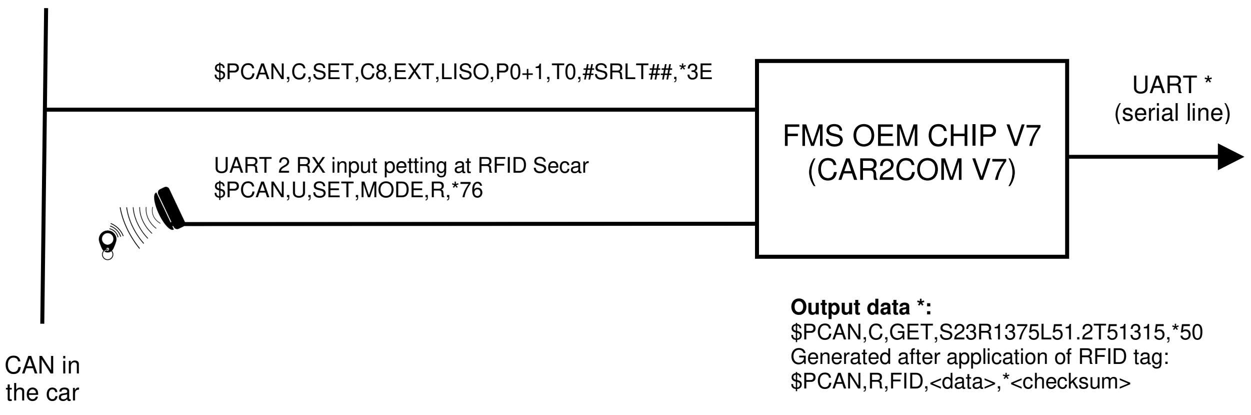
Output data *:
$PCAN,T,GET,R1284S41T41971,*7B
$PCAN,C,GET,L40.4,*5B
Request for driver´s ID:
$PCAN,T,GET,#I##,*74
Response
$PCAN,T,GET,I@0000000000465000@0000000000465700@,*10

Car FMS gate and IBIS



Output data *:
$PCAN,C,GET,S23R1375L40.0T51315,*38
Request for IBIS
$PCAN,I,GET,#I##,*69
Response
$PCAN,I,GET,I@019@101@001@S@,*60

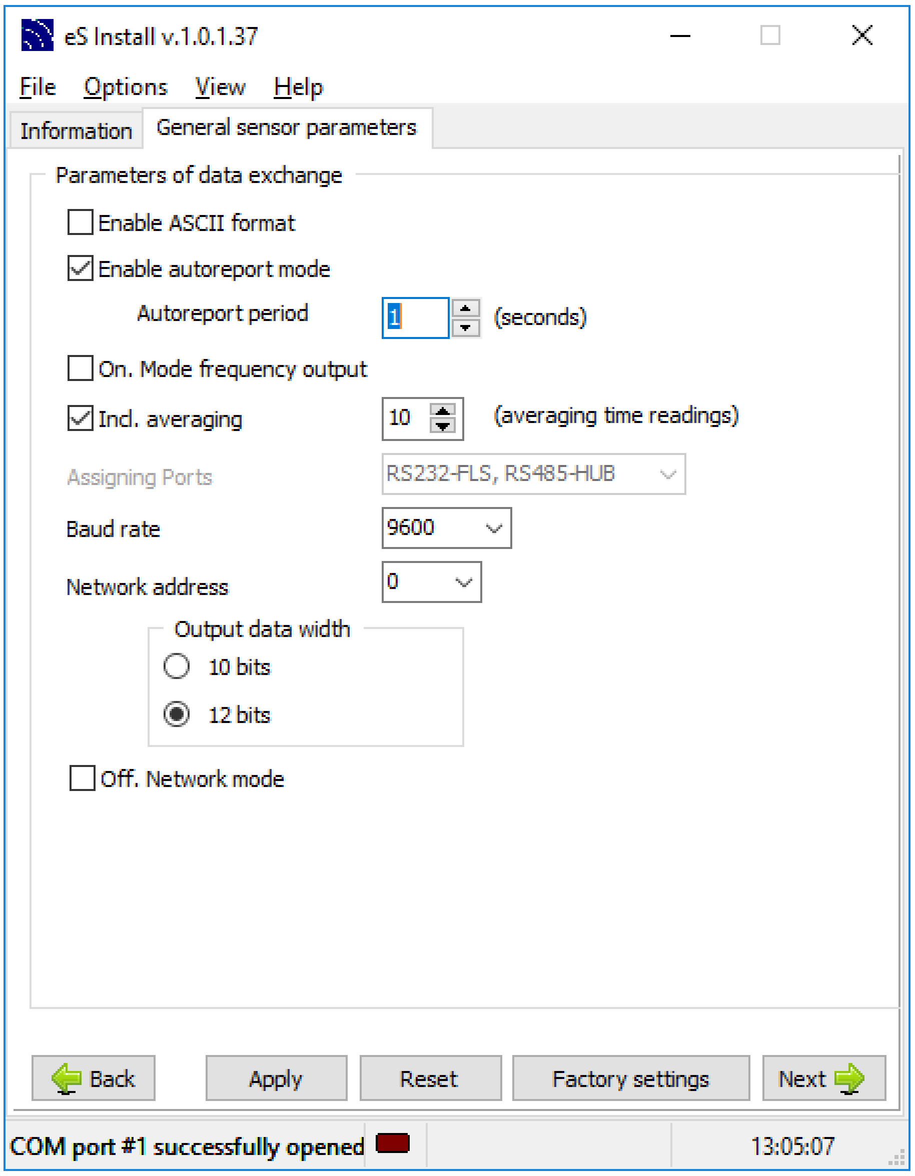
Car CAN and RFID



Output data *:
$PCAN,C,GET,S23R1375L51.2T51315,*50
Generated after application of RFID tag:
$PCAN,R,FID,<data>,*<checksum>

Car with connection to CAN and one or two fuel probes with RS232

Car with CAN bus and fuel probe with RS232 connected to tachograph input or UART2 RX input. If float with RS232 is connected to tachograph output or UART2 input, data from this fuel probe is sent as secondary fuel at CAN. If two fuel probes are connected to both inputs, data from UART2 input is sent as primary tank and data from tachograph input is sent as secondary tank. Now, RCS EPSILON ES2 probes are supported.



eS Install v.1.0.1.37

File Options View Help

Information | General sensor parameters

Parameters of data exchange

- [ ] Enable ASCII format
- [x] Enable autoreport mode

Autoreport period: 1 (seconds)

- [ ] On. Mode frequency output
- [x] Incl. averaging: 10 (averaging time readings)

Assigning Ports: RS232-FLS, RS485-HUB

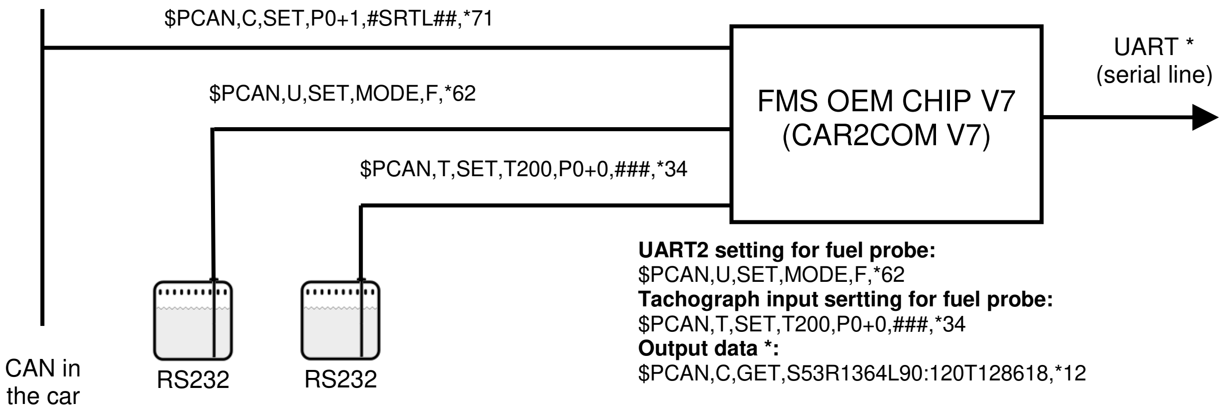
Baud rate: 9600

Network address: 0

Output data width: 10 bits [✓ 12 bits]

- [ ] Off. Network mode

COM port #1 successfully opened 13:05:07

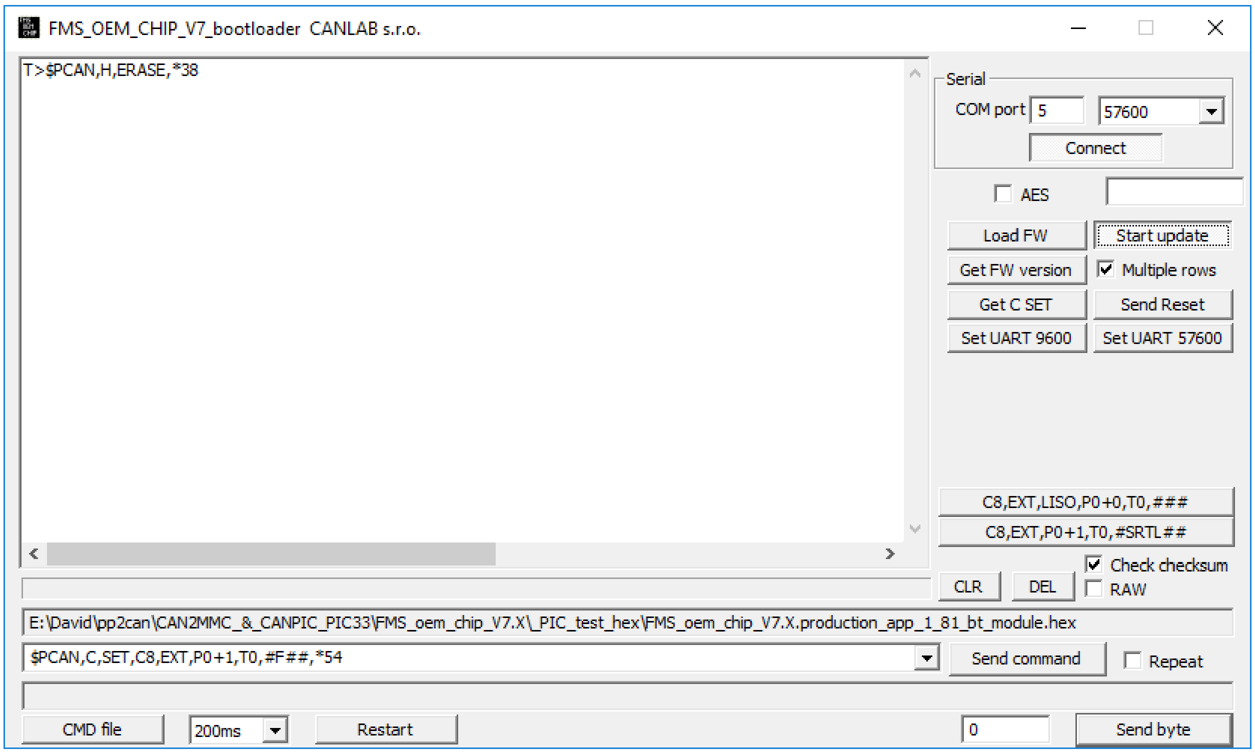


UART2 setting for fuel probe:
$PCAN,U,SET,MODE,F,*62
Tachograph input sertting for fuel probe:
$PCAN,T,SET,T200,P0+0,###,*34
Output data *:
$PCAN,C,GET,S53R1364L90:120T128618,*12

Firmware update from PC

Update is made via serial port. In case of variants with serial port with voltage level of 5V or 3V3 it is necessary to use appropriate transducer, e.g. made by Future Technology Devices International Limited (https://www.ftdichip.com/Products/Cables/USBTTLSerial.htm). Also for classic serial port, for PC without serial port, it is possible to use transducers by the same company (https://www.ftdichip.com/Products/Cables/USBRS232.htm).

Update is made via FMS OEM CHIP V7 bootloader. It can be found on our website (http://www.rs.canlab.cz/?q=cs/fms_oem_chip).


FMS_OEM_CHIP_V7_bootloader CANLAB s.r.o.

```
T>$PCAN,H,ERASE,*38
```

Serial
COM port 5 | 57600

☐ AES
☑ Multiple rows
☑ Check checksum
☐ RAW

E:\David\pp2can\CAN2MMC_&_CANPIC_PIC33\FMS_oem_chip_V7.X_PIC_test_hex\FMS_oem_chip_V7.X.production_app_1_81_bt_module.hex

$PCAN,C,SET,C8,EXT,P0+1,T0,#F##,*54

☐ Repeat
200ms
0

In the right upper corner, set appropriate COM port and press Connect button. Then, verify, if application communicates with FMS OEM CHIP, e.g. via sending a request “Get FW version”. If communication is functional, firmware loads file after pushing “Load FW” button. Update itself can be initiated by pressing “Start update” button. If an option “Multiple rows” is active, messages in HE2 format are used (from FW 1.60) and update is processed more quickly.

Immediately after pressing “Start update” message ERASE occurs in log, after that, communication is interrupted for several seconds – FW is deleted from FLASH memory. After deleting, loading is initiated – huge amount of loaded data occurs in log. When occurrence of this data is finished and progress bar in lower part of application is finished, *REBOOT,OK* message occurs.

In this moment, FW is loaded in FLASH memory, but original FW is still running. Loading to processor is run during restart of transducer power or after its software reset by appropriate command. After restart, loading takes several tens of seconds, individual phases are signalized by LED flashing.

FMS OEM CHIP V7 bootloader application can be used as well as console during setting/testing of transducer/FMS OEM CHIP. It facilitates work, e.g. it remembers last used commands. By using of left button double-click copy command from log to clipboard and it always automatically calculates (controls) checksum. Therefore, in manually written commands it is possible to insert checksum e.g. with value *00 and during sending it is automatically recalculated.

Cars support

Below, there are names of profiles, other cars based on same platform can work with these individual profiles.

Profile name	**CAN speed**	**Note**	**CAR type**
ChryslerVoyager	500k		T256
Citroen jumper	50k		T147 / T148
Fiat Ducato, Fiorino	Primary 500k, secondary50k	Connect secondary CAN	T146
Fiat Ducato, Peugeot boxer	Primary 500k, secondary 50k	Connect secondary CAN	T150
Fiat Ducato	500k	500k, st id	T145
Fiat Doblo 50k LS	20k		T149
Fiat Fullback			T151
Ford Mondeo	500k		T128
Ford Tranzit 2017	500k		T138/T137
Ford Tranzit 2015	500k		T134
Ford Tranzit <2015	500k		T129
Ford Turneo	500k		T133
Ford C-max	500k		T130
Ford S-max	500k		T131
Ford Fusion	500k		T132
Ford Ranger	500k		T135
Ford Turneo Connect	500k		T136
Honda Civic	500k		T480
Hyunday/Kia <2016	500k		T320
Hyunday/Kia >2016	500k		T321
Mazda	500k		T160
Nissan - Note, Micra	500k		T64
Nissan - Primastar, Kubistar	250k/500k		T65
Nissan - Navara	500k		T66

Nissan - X trail	500k		T68
Nissan	500k		T67
Citroen Berlingo	500k		T208
Peugeot 207	500k		T224/T227
Peugeot 308 (Toyota ProAce 2019)	500k		T225/T228
VW	500k		T16
VW MQB	500k		T192
VW MQB CNG	500k		T193
MQB Audi	500k		T194
Mercedes truck E5 and older	500k		T15
Mercedes sprinter/vito	Primary 500k, secondary 83.3k	tot. dis. just older cars or connect also secondary CAN T51/T52	T48/T49/T50/T51
Mercedes Sprinter 2018	500k		T53
VW crafter <2016	Primary 500k, secondary 83,3k	tot. dis. just older cars or connect also secondary CAN	T48
OPEL ASTRA J	500k		T80
OPEL Movano/Vivaro	500k or 250k	250 or 500k	T81/T82/T83
TOYOTA	500k		T96
Toyota Auris	500k		T97
Renault Master	250k		T241/T242
Renault Master od 2010	500k		T240/T243
SUZUKI SX4	500k		T176
Mercedes C180	500k		T272
Renault megane	500k		T288
EscapeTalisman			T290
Megan 2014,Trafic 2015,Scenic 2012			T289
BMW	500k		T352
Both input CANs according to J1939, data accessible on both CANs are combined			T512
Volvo Truck from 2013, primary CAN 250k,			T7

secondary 500k			
J1939, calculation of trip fuel from fuel rate			T10
Fendt 936 vario, primary engine 250k, secondary ISO CAN 250k, trip fuel from fuel rate			T521
Takeuchi	250k		T520
Volvo XC90 2013	125k		T640
JeepGrandCherokee 2017	500k		T672
Volvo XC90 2015	500k		T641
Kubota M7171	250k		T522
Iveco Stralis and Eurocargo	500k or 250k	As SAE1939, but tank is also read	T14

Document version

7.0.0	10.9.2016	First version of the document.
7.0.1	10.4.2017	Added pinout and LED description
7.0.3	8.10.2017	Fuel level, trip fuel/distance scale
7.0.6	11.1.2018	Added IBIS, secondary CAN
7.0.7	9.2.2018	Examples of less common usage
7.0.8	19.2.2018	Fuel probes RS232 from FW 1.36
7.0.9	17.9.2018	POP regime at DM1 reading
7.0.13	5.3.2019	$PCAN,C,PWR,KON command
7.0.14	20.3.2019	Accessible types of cars
7.0.15	15.4.2019	CAN data decoding defined by user.
7.0.16	30.5.2019	Setting of digital tachograph input.
7.0.17	31.5.2019	Added Update FW from PC chapter.