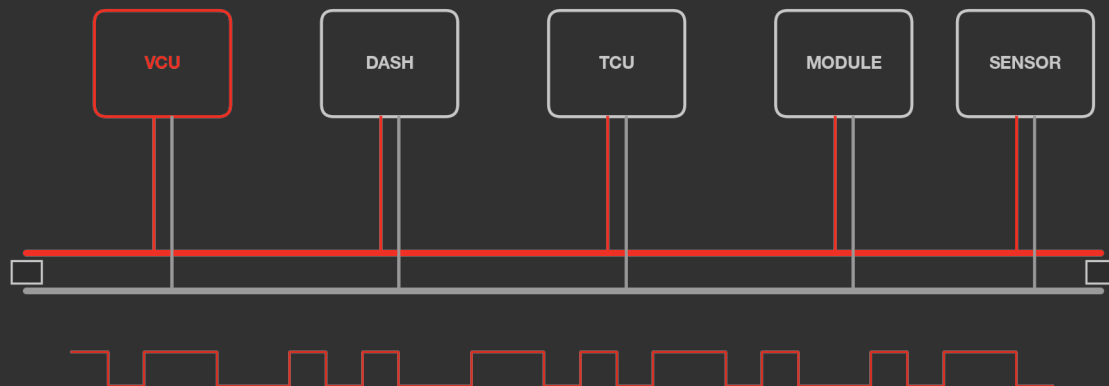




CAN TECHNICAL GUIDE

*CAN network, CAN FD, Customizable CAN, DBC and sniffer
FuelTech Vision 3.0 · FT700 and FT470 VCUs*

CAN Technical Guide · Version 1.2, October 13, 2026

FuelTech Vision 3.0 software and VCU firmware 00.03.00.01 (FT700 and FT470).

The screens and limits described apply to these versions. For later versions, see the release notes.

Contents

1 Presentation.....	5	How the ID works in FTCAN 2.0	22
About this guide	5	Golden rule: one protocol per bus	23
Where to start.....	5		
Who this guide is for.....	6	7 VCU CAN ports	24
Versions covered	6	How many ports and what each one supports	24
Before connecting to the vehicle.....	6	The CAN ports screen	25
2 CAN network basics.....	8	Bit rates and sample point	27
What CAN is.....	8	FTCAN 2.0 realtime data	27
Physical layer: the wires.....	8	How the VCU reacts to port errors.....	28
Topology and termination.....	9		
The CAN frame.....	11	8 FuelTech devices on the network	30
Arbitration: who talks first	12	Connection.....	30
Bit rate and sample point.....	13	CAN Network	30
Error detection.....	14	Diagnostics Panel	31
3 Signals: from physical value to bits	15	Functions that use a CAN port.....	32
Message and signal.....	15		
Raw value and physical value	15	9 CAN custom packages	33
Byte order: Intel and Motorola.....	16	Purpose.....	33
Signed signals (negative values).....	17	The 20 packages.....	33
4 CAN FD	18	Format on the bus	34
What changes	18	Legacy broadcasts	35
Frame sizes in CAN FD.....	18	When to use each feature	37
Compatibility with classic CAN.....	19		
5 DBC files.....	20	10 Customizable CAN	38
What a DBC is.....	20	Purpose.....	38
Anatomy of a message	20	Frames and signals.....	39
Other common lines	21	Frame fields.....	41
6 FTCAN 2.0 and other brands' networks	22	Signal fields	43
Two worlds on the same wire.....	22	Arithmetic channel	44
		How the VCU builds the bits.....	45
		Receiving (RX): important rules	45
		Transmitting (TX): important rules	46
		Errors and how to see them.....	47
		Customizable CAN limits	47

11 Import and export DBC.....	48
Where it is	48
What the import does.....	48
After importing: checklist	49
What is read from the file	50
Export DBC	50
12 CAN Sniffer.....	51
What it is	51
Where to open it.....	51
Controls.....	52
Columns.....	54
Limits.....	54
Method for discovering a signal	54
13 Step-by-step examples.....	56
Example 1: third-party dash (TX).....	56
Example 2: third-party sensor (RX).....	57
Example 3: fixed frame (waking up a module)	57
14 Troubleshooting	58
Diagnostic tools.....	58
Symptoms, causes and solutions	59
15 Glossary	61



1 Presentation

About this guide

This guide explains how VisionFT line VCUs communicate over a CAN network and how to configure that communication in the FuelTech Vision software. VCU (Vehicle Control Unit) is the FuelTech control unit installed in the car: the FT700 and the FT470. The guide starts with the basics (what a CAN network is, how a frame is built, what changes in CAN FD) and moves on to the everyday tools: the VCU's CAN ports, FuelTech devices, Customizable CAN, DBC file import and the sniffer.

If you already know CAN, go straight to the configuration chapters (6 onward). If you are just starting, read chapters 2 to 5 first: almost every communication problem comes from a detail of the theory (termination, bit rate, byte order, signal scale). If you are only connecting FuelTech products, chapters 7 and 8 have what you need. If you are not the person who configures the network (the car's driver, for example), read at least the warning and the "Before connecting to the vehicle" list at the end of this chapter: the list is carried out by whoever installs or configures the network (the tuner or the installer), and for the driver it shows what to ask them for and what to check with them. The technical terms in the list are explained in the Glossary at the end of the guide.

Where to start

I want to...	Chapters
Connect a FuelTech product (O2 sensor, injector driver, ignition, expander)	7 CAN ports and 8 FuelTech devices
Send data to a third-party dash or datalogger, or read a third-party sensor or module	10 Customizable CAN and 13 Examples
Connect a third-party inverter, BMS (battery management system) or charger in an electric vehicle	Outside the scope of this guide. Before any configuration, read the warning in this chapter.
Use the DBC file that came with the equipment	11 Import and export DBC
Feed a dash that already reads the FuelTech format	9 CAN custom packages
Find out what a module is transmitting	12 CAN Sniffer
Troubleshoot a communication problem	14 Troubleshooting

Who this guide is for

- Tuners and installers who need to connect the VCU to a dash, transmission module, injection module, sensor or the vehicle's original (OEM) network.
- Anyone who wants to read data from a third-party module or send data from the VCU to it.
- Teams that receive a DBC file from an equipment manufacturer and need to use it in the VCU.

Versions covered

This guide describes the FuelTech Vision 3.0 software and the VCU 3.0 firmware (version 00.03.00.01), on the FT700 and FT470 VCUs. In earlier versions some fields may have a different name or may not exist. Always update the software and firmware before configuring the network.

WARNING

A wrong CAN configuration can disrupt other modules in the vehicle. Never configure the VCU to transmit on a vehicle's original (OEM) network (brakes, airbag, steering, transmission) or on an electric vehicle network (inverter, battery management, charger) without knowing exactly what each frame does. Test first on the bench or with the vehicle stopped, and save a copy of the map before any change.

Before connecting to the vehicle

1. Save a copy of the map.
2. In **VCU settings > CAN ports**, check each port in use: turned on, **Operation mode** (FTCAN 2.0 for FuelTech products, RAW mode for the vehicle and third-party equipment), **Baudrate** (the bit rate) equal to the network's (in RAW mode the default is 1000 kbps) and **CAN Terminator** (the VCU's 120 Ω resistor for the end of the bus; on by default; leave it on only if the VCU is at one end of the bus). This also applies if you only listen to the network: a port at a different rate generates errors that disturb the other modules.
3. Write the map (**Write VCU** button) and turn the VCU off and back on (power-cycle it): port changes only take effect at power-up (chapter 7).
4. In Customizable CAN (the tool for frames you define yourself), a scale below 1 or a negative offset requires **Arithmetic channel** on, the signal field that makes the VCU calculate with real numbers (chapter 10).

5. After importing a DBC (the file that describes the equipment's messages), disable the frames you will not use before writing: otherwise they transmit zeros on Port A (chapter 11).
6. On the bench, connect at least one other module to the port: alone on the bus the VCU cannot transmit, because no module acknowledges its frames (chapter 10).
7. Before trusting a signal, check a known value in real time (chapter 3).

2 CAN network basics

What CAN is

CAN (Controller Area Network) is a communication network created in the 1980s to connect a vehicle's electronic modules with few wires. Instead of one wire per signal, all modules share a single pair of wires, the bus, and exchange short messages over it. The standard that defines the network is ISO 11898.

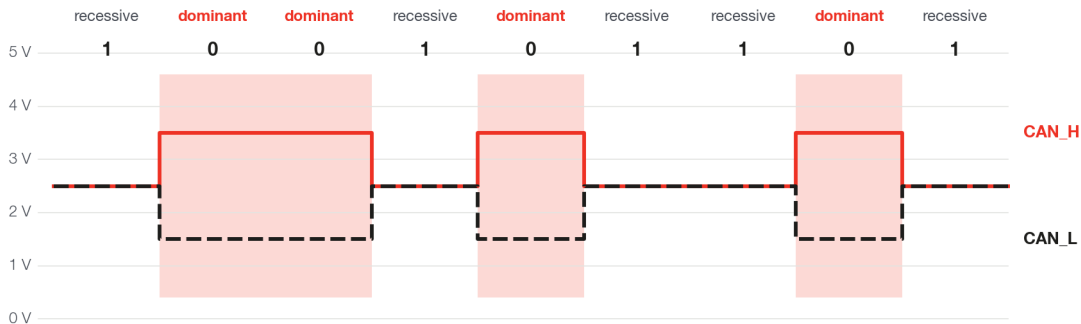
Three ideas explain almost everything about CAN:

- **Everyone hears everything.** When a module transmits, all the others receive the message. Each one decides whether that message matters to it or not.
- **The message has a name, not a recipient.** Each message carries an identifier (ID) that says what it contains, for example “engine speed and temperature”. There is no destination address.
- **There is no boss.** Any module can start transmitting when the bus is free. If two start together, the network itself decides which one continues, without losing time and without corrupting the message (see Arbitration, below).

In practice, the VCU uses CAN to talk to FuelTech devices (O2 sensor, ignition module, injector drivers, dashes), to third-party equipment and to the vehicle's original modules.

Physical layer: the wires

The high-speed CAN bus uses two wires, CAN_H (CAN high) and CAN_L (CAN low), twisted together. The information is in the voltage difference between them, not in the voltage of each one relative to ground. That is why the network tolerates the engine's electrical noise well: interference that enters both wires at the same time does not change the difference.



Recessive: both lines at ~2.5 V (difference ~0 V). Dominant: CAN_H ~3.5 V and CAN_L ~1.5 V (difference ~2 V).

Electrical levels of the CAN bus. The dominant bit (0) always wins over the recessive bit (1).

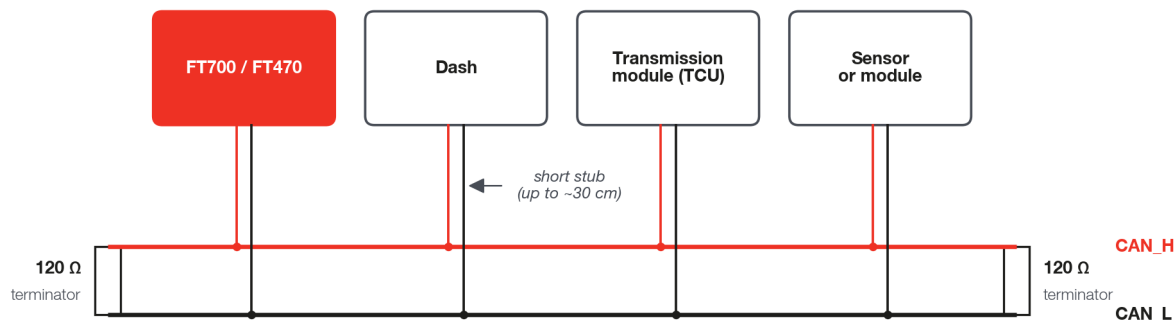
The two possible states are:

- **Recessive (bit 1):** both wires sit at about 2.5 V. Nobody is driving the bus.
- **Dominant (bit 0):** CAN_H rises to about 3.5 V and CAN_L drops to about 1.5 V, a difference of about 2 V.

If one module transmits dominant and another recessive at the same instant, the bus becomes dominant. This simple rule is the basis of arbitration and of acknowledgment.

Topology and termination

The bus is a line, with a beginning and an end. Each module connects to it through a short stub. At the two ends of the line, and only there, a 120 Ω resistor goes between CAN_H and CAN_L, called a terminator. It absorbs the signal that reaches the end of the wire and keeps it from bouncing back and mixing with the next bits.



Measured between CAN_H and CAN_L with everything off: about 60 Ω (two 120 Ω terminators in parallel)

Twisted pair, linear topology: one terminator at each end, never in the middle

Linear bus with a 120 Ω terminator at each end.

Practical installation rules:

- Use twisted pair (about 1 twist every 2 to 3 cm, roughly 1 in). Shielded cable helps in installations with a lot of noise; ground the shield at one end only.
- Build it as a line. Avoid star topology, with several long branches leaving the same point.
- Keep stubs short, preferably up to 30 cm (about 12 in).
- Exactly two terminators per bus. Many modules already have an internal terminator; others, such as the VCU, let you turn the terminator on or off in the software.

NOTES

Quick termination test: with everything off, measure the resistance between CAN_H and CAN_L with a multimeter. About 60 Ω: correct (two 120 Ω terminators in parallel). About 120 Ω: one terminator is missing. About 40 Ω or less: there are too many terminators. Very high or infinite resistance: no terminator or an open wire.

The total length of the network limits the speed, because the signal has to go and return within one bit. The values below

are usual cable references for high-speed CAN:

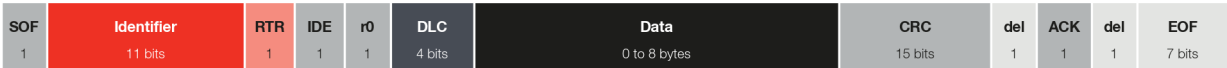
Bit rate	Reference maximum length
1 Mbit/s	40 m (about 130 ft)
500 kbit/s	100 m (about 330 ft)
250 kbit/s	250 m (about 820 ft)
125 kbit/s	500 m (about 1,640 ft)

In a race car the distances are short and any of these rates usually works. What usually fails is wrong termination, a poorly made splice and a long stub.

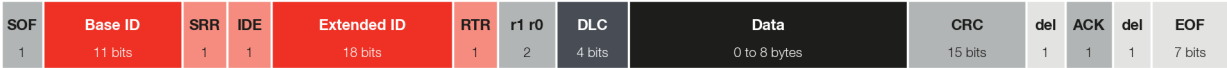
The CAN frame

Each message travels inside a frame. The frame has fixed fields before and after the data: they identify the message, say how many bytes it has and check whether it arrived without error.

Standard frame (11-bit ID, CAN 2.0A)



Extended frame (29-bit ID, CAN 2.0B)



 arbitration (who talks first)

 control (length)

 data (the signals)

 check and acknowledgment

Structure of the classic CAN frame, standard and extended.

Field	What it does
SOF	Start of Frame. A dominant bit that synchronizes all modules.
Identifier (ID)	The name of the message and, at the same time, its priority. It has 11 bits in the standard frame (0x000 to 0x7FF) or 29 bits in the extended frame (up to 0x1FFFFFFF).
RTR, IDE, SRR	Control bits: they say whether the ID is standard or extended and whether the frame is a data frame or a remote request (rare today).
DLC	Data Length Code: the number of data bytes, from 0 to 8 in classic CAN.
Data	From 0 to 8 bytes. This is where the signals (engine speed, temperature, pressure...) are.
CRC	Check code calculated over the frame. The receiver recalculates it and compares.
ACK	Acknowledgment. Any module that received the frame without error forces this bit to dominant.
EOF	End of Frame.

IDs are usually written in hexadecimal, with the 0x prefix. ID 0x640, for example, is 1600 in decimal. DBC files write the ID in decimal; screens and analyzers usually use hexadecimal.

NOTES

*Standard (11-bit) and extended (29-bit) IDs coexist on the same bus, but they are different IDs: standard ID 0x100 and extended ID 0x00000100 are distinct messages. When configuring a frame, always check whether the equipment uses an extended ID (in FuelTech Vision, the field is called **Extended ID**).*

Arbitration: who talks first

When the bus becomes free, any module can start transmitting. If two start at the same time, each one keeps sending its ID bit by bit and, at the same time, reading the bus. Since the dominant bit (0) wins over the recessive bit (1), the module that sends 1 and reads 0 realizes it lost, stops transmitting and starts listening. The winner does not even notice the contest: its frame continues intact.

	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Node A (ID 0x100)	0	0	1	–	–	–	–	–	–	–	–
Node B (ID 0x0F0)	0	0	0	1	1	1	1	0	0	0	0
Bus	0	0	0	1	1	1	1	0	0	0	0

At bit 8 node A sends 1 (recessive) and reads 0 (dominant): it loses arbitration, stops transmitting and becomes a receiver.

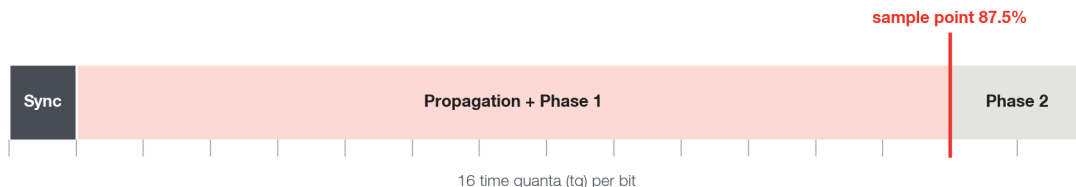
Node B carries on without noticing any contention. Lowest ID = highest priority.

Arbitration between two modules. The lowest ID wins.

The practical consequence is that **a lower ID has higher priority**. Critical messages, such as engine speed and torque, usually have low IDs. A module that transmits a very low ID very often can delay all the others.

Bit rate and sample point

All modules on a bus must use the same bit rate (baud rate; on the **CAN ports** screen, **Baudrate**, in kbps, the same as kbit/s). The most common rates in vehicles are 500 kbit/s and 250 kbit/s; racing equipment and the FuelTech FTCAN network use 1 Mbit/s. A module at a different rate understands nothing and, worse, generates errors that disturb the others.



Example: at 500 kbit/s each bit lasts 2 μ s. The controller reads the bus level at the sample point;
all nodes need the same bit rate and close sample points.

Division of a bit into time quanta and the sample point.

Each bit is divided into small time slices. The sample point is the instant, within each bit, when the controller reads whether the bus is at 0 or 1. It is given as a percentage of the bit duration and normally sits between 75% and 87.5%. Modules with very different sample points may even work on the bench and fail with a long cable or with noise.

Error detection

CAN is a very robust network because all modules watch all frames. The main mechanisms are:

- **CRC:** the receiver recalculates the check code; if it does not match, the frame is discarded.
- **Acknowledgment (ACK):** the transmitter waits for at least one other module to confirm reception. By the standard, a module alone on the bus never receives acknowledgment and keeps repeating the frame (the VCU's reaction is in chapter 7).
- **Bit stuffing:** after five identical bits in a row, the transmitter inserts an inverted bit. Six identical bits in a row indicate an error.
- **Form check and bit monitoring:** fixed fields with a wrong value, or a bit that does not come out as transmitted, are also errors.

When it detects an error, the module destroys the frame with an error signal and the transmitter tries again. Each module keeps error counters. With few errors it works normally; with too many it goes into a restricted (error-passive) state and, at the limit, disconnects from the bus so it does not disturb the others. This state is called **bus-off**. A different bit rate, wrong termination and CAN_H swapped with CAN_L are the most common causes of a module in bus-off.

3 Signals: from physical value to bits

Message and signal

A CAN frame has up to 8 bytes (64 bits) of data. A single value rarely fills the whole frame: the manufacturer divides the bytes into several signals. An engine frame might carry, for example, engine speed in 16 bits, temperature in 8 bits, throttle position in 8 bits and a few 1-bit switches.

To read or write a signal you need to know exactly:

- **Start bit and length:** where the signal starts within the frame and how many bits it occupies.
- **Byte order:** Intel or Motorola (explained next).
- **Signed or unsigned:** whether the value can be negative (in Customizable CAN, the Value type field).
- **Factor and offset:** how to convert the transmitted integer into the physical value.

Raw value and physical value

What travels on the network is always an integer, the raw value. The physical value (rpm, °C, bar, %) comes from a simple calculation:

$$\text{physical value} = \text{raw value} \times \text{factor} + \text{offset}$$



Example: engine temperature in 1 byte, factor 1 and offset −40. On transmit the calculation is inverted: $\text{raw} = (\text{physical} - \text{offset}) / \text{factor}$.

Conversion of the raw value into the physical value.

The factor sets the resolution. A factor of 0.1 on a pressure in bar means each unit of the raw value is worth 0.1 bar. The offset shifts the range. Temperature with an offset of −40 lets you transmit from −40 °C upward using only positive numbers.

To transmit, the calculation is the inverse: $\text{raw value} = (\text{physical value} - \text{offset}) \div \text{factor}$.

❗ IMPORTANT

A wrong factor and offset do not cause a communication error: the network works and the value simply comes out wrong. Always check a known value before trusting the signal, for example the ambient temperature with a cold engine or the idle speed.

Byte order: Intel and Motorola

When a signal occupies more than one byte, you need to know which byte comes first.

- **Intel (little-endian):** the least significant byte comes first. In the DBC it appears as @1.
- **Motorola (big-endian):** the most significant byte comes first. In the DBC it appears as @0.

The start bit also changes meaning. In the convention used by DBC files, in Intel the start bit is the least significant bit of the signal; in Motorola it is the most significant bit, and the counting goes byte by byte.

Intel (little-endian) @1

start = 4, length = 12: the start bit is the LSB

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	7	6	5	4	3	2	1	0
byte 1	15	14	13	12	11	10	9	8
byte 2	23	22	21	20	19	18	17	16
byte 3	31	30	29	28	27	26	25	24

-  most significant bit (MSB)
-  least significant bit (LSB)

Motorola (big-endian) @0

start = 7, length = 12: the start bit is the MSB

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	7	6	5	4	3	2	1	0
byte 1	15	14	13	12	11	10	9	8
byte 2	23	22	21	20	19	18	17	16
byte 3	31	30	29	28	27	26	25	24

-  most significant bit (MSB)
-  least significant bit (LSB)

The numbers are the bit numbering used by the DBC file (byte × 8 + bit). The same 12-bit signal occupies different cells

depending on the byte order: choosing the wrong order raises no error, it just gives an absurd value.

The same 12-bit signal in the two byte orders, in the DBC bit numbering.

Getting the byte order wrong does not cause an error: it gives a nonsensical number. A 16-bit signal with the bytes swapped turns 1,000 rpm into 59,395 rpm. If the value read jumps strangely or becomes huge, the first suspect is the byte order.

Signed signals (negative values)

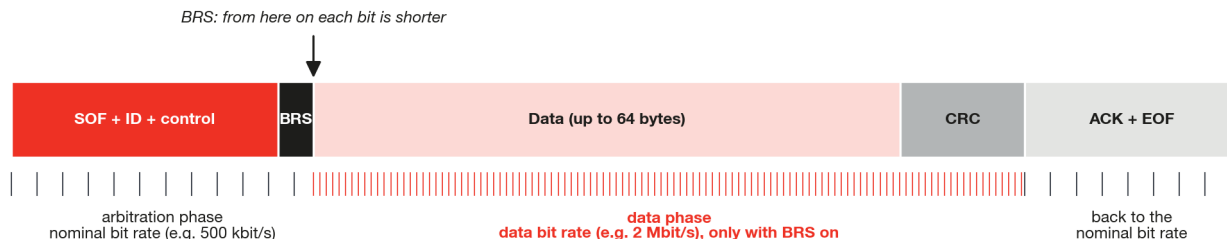
Values that can be negative, such as ignition advance, fuel correction or lateral acceleration, are usually transmitted in two's complement, the standard way to write negative numbers in binary: the upper half of the range of raw values counts as negative. In an 8-bit signed signal, raw values 0 to 127 are worth 0 to 127 and 128 to 255 are worth -128 to -1; raw value 255 (0xFF) means -1. If the signal is configured as unsigned, the same data appears as 255. In the DBC, the + sign indicates unsigned and - indicates signed.

4 CAN FD

What changes

CAN FD (Flexible Data-rate) is the evolution of CAN, standardized in 2015 (ISO 11898-1). It keeps the same bus, the same wires and the same arbitration, and changes two things:

- **More data per frame:** up to 64 bytes, versus 8 in classic CAN.
- **Faster data phase:** after arbitration, the frame can switch to a higher rate only during the data (for example, arbitration at 500 kbit/s and data at 2 Mbit/s). This switch is controlled by the BRS bit (Bit Rate Switch).



Each tick mark is one bit. With BRS the data portion runs faster; without BRS the whole frame stays at the nominal bit rate.

CAN FD frame with the bit rate switch (BRS).

Arbitration stays at the nominal bit rate because all modules must contend for the bus at the same time. After only one module is transmitting, the bit can become shorter.

Frame sizes in CAN FD

In CAN FD the DLC field (the frame size code) still has 4 bits, so not every size is possible. Above 8 bytes the valid sizes are fixed:



DLC	Data bytes
0 to 8	0 to 8 (same as classic CAN)
9	12
10	16
11	20
12	24
13	32
14	48
15	64

A frame configured with 10 bytes, for example, is transmitted with 12 bytes; the remaining bytes are padded.

Compatibility with classic CAN

A classic CAN controller does not understand an FD frame and treats it as an error, destroying the frame. For this reason you do not mix a classic-only module with FD frames on the same bus. A port configured for CAN FD can still transmit classic frames when the frame is marked for that (in Customizable CAN, **Force classic CAN**), which allows talking to a classic piece of equipment on the same bus, as long as no FD frame circulates there.

 **NOTES**

To use CAN FD, all modules on the bus must support CAN FD and be configured with the same nominal bit rate and the same data bit rate.

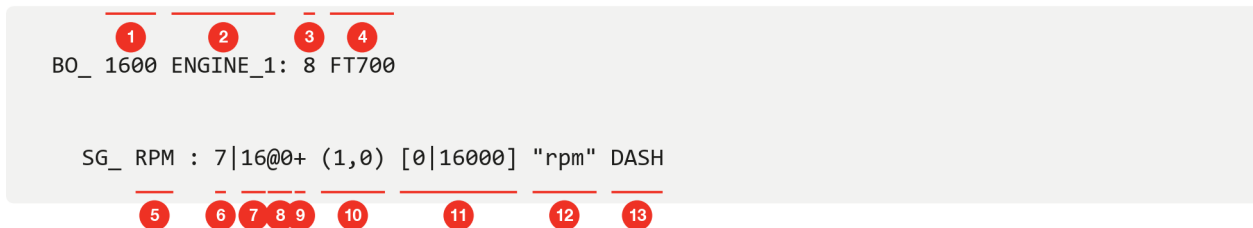
5 DBC files

What a DBC is

DBC (CAN database) is a text file that describes a CAN network: which messages exist, who transmits each one and where each signal is within them. The format became the de facto industry standard. Manufacturers of dashes, transmission modules, sensors and dataloggers usually provide the equipment's DBC, and it is what you use to configure communication without having to decipher it bit by bit.

Anatomy of a message

Each message starts with a BO_ line and is followed by one SG_ line for each signal:



One message and one signal in a DBC file. The numbers are explained in the table below.

No.	Field	Meaning
1	ID	Message identifier, in decimal (1600 = 0x640). An extended ID appears with bit 31 set, that is, with 2147483648 (0x80000000) added.
2	Message name	For human reading only.
3	Size	Number of data bytes (DLC).
4	Transmitter	The module that sends the message.
5	Signal name	For human reading only.

No.	Field	Meaning
6	Start bit	Position of the signal in the frame (convention explained in chapter 3, Signals).
7	Size	Number of bits of the signal.
8	Byte order	@1 = Intel, @0 = Motorola.
9	Sign	+ = unsigned, - = signed.
10	Factor and offset	physical value = raw × factor + offset.
11	Minimum and maximum	Range of the physical value. Informational.
12	Unit	Informational.
13	Receivers	The modules that use the signal.

Other common lines

Besides BO_ and SG_, a DBC usually has comment lines (CM_), attributes (BA_DEF_ and BA_; for example GenMsgCycleTime, the transmit period of each message, in milliseconds) and value tables (VAL_, which give a name to each value of a signal, such as 0 = Off and 1 = On). CAN FD messages are marked by an attribute, normally VFrameFormat, which says whether the frame is classic or FD.

NOTES

To open and check a DBC on the computer there are free viewers and open-source libraries. Opening the file before importing helps count how many messages and signals it has and choose only what matters.

6 FTCAN 2.0 and other brands' networks

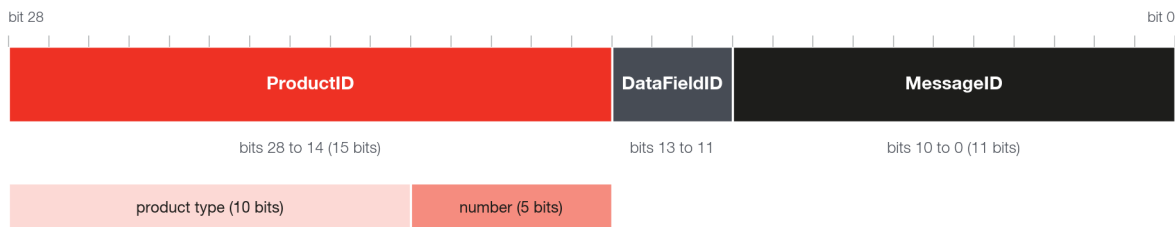
Two worlds on the same wire

The VCU talks over CAN in two different ways, and each port works in one of them:

- **FTCAN 2.0:** FuelTech's protocol, used among FuelTech products (O2 sensors, ignition modules, injector drivers, input expanders, button panels, dataloggers and compatible dashes). Always at 1 Mbit/s, with a 29-bit extended ID.
- **RAW mode:** raw CAN, without the FuelTech protocol. It is the mode for talking to third-party equipment and to the vehicle's original network, at the rate they use (normally 500 kbit/s or 250 kbit/s) and with the frame format they define.

How the ID works in FTCAN 2.0

In FTCAN 2.0 the 29-bit ID is not just any number: it says who sent the frame and what type of data is in it.



Lower ID = higher priority: the product type and the message type set the priority on the bus.

Division of the 29-bit FTCAN 2.0 identifier.

- **ProductID (product identification):** product type (10 bits) and unit number (5 bits). The protocol can identify up to 32 units of the same product; the VCU reads data from up to 10 FuelTech devices (see CAN Network).
- **DataFieldID (data format):** simple frame, FTCAN message split across several frames (segmented) or message relayed by a bridge.
- **MessageID (message type):** real-time readings, commands, responses, configuration.



Messages larger than 8 bytes are split into several frames and reassembled by the receiver. That is why a third-party dash cannot read FTCAN 2.0 with a simple DBC alone, except for fixed-format messages, such as the simplified packets (see CAN custom packages). The complete specification is in the **FTCAN 2.0 Protocol** document, available on the FuelTech website.

Golden rule: one protocol per bus

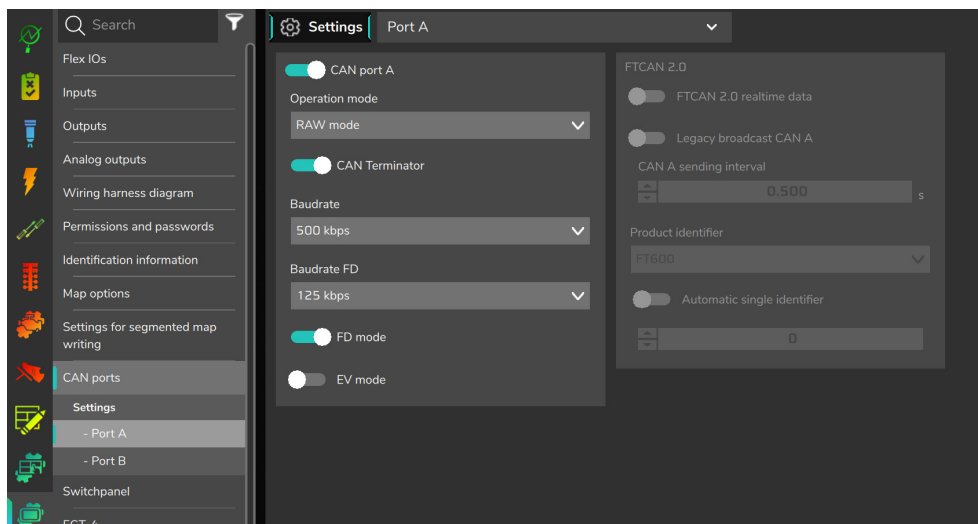
Do not mix the FTCAN 2.0 network with the vehicle network on the same wire pair. They have different rates (1 Mbit/s versus 500 or 250 kbit/s) and IDs that can conflict. Use one port for FuelTech products and another for the vehicle or for third-party equipment.

Item	Port in FTCAN 2.0	Port in RAW mode
Purpose	FuelTech products	Vehicle and third-party equipment
Bit rate	1 Mbit/s, fixed	The bus's own (100 to 1000 kbit/s)
CAN FD	No	Yes, on ports that support it
ID	29 bits, structured	11 or 29 bits, free
Features	FuelTech devices, FTCAN real-time data, legacy broadcasts, custom packages, Customizable CAN	Customizable CAN, vehicle functions (transmission, jet ski, PDM, steering wheel...)

7 VCU CAN ports

How many ports and what each one supports

Feature	FT700	FT470
CAN ports	A, B and C	A and B
FD mode available on the screen	Ports A and C	Port A
Port without FD mode on the screen	B (classic CAN only)	B (in version 3.0)
Software-controlled 120 Ω terminator	A, B and C (board revision 1.5 or later)	A and B
CAN Sniffer	Yes	Yes (Ports A and B)



CAN ports on the FT470: only Port A and Port B.

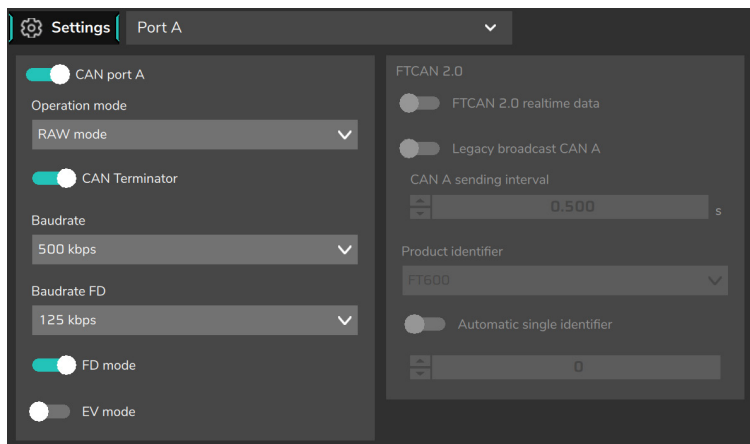
See the CAN_H and CAN_L pinout of each port in the VCU installation manual. On the FT700 with a board revision earlier than 1.5 the terminator is not controlled by the software: check the termination with a multimeter.

NOTES

On the FT700, Port B does not support CAN FD. On the FT470, version 3.0 offers FD mode on Port A only. If a piece of equipment requires CAN FD, use Ports A or C of the FT700 or Port A of the FT470.

The CAN ports screen

The path is **VCU settings > CAN ports** (some app messages, such as the map validation, call this menu “My VCU”). Choose the port (Port A, Port B and, on the FT700, Port C) in the **CAN ports** submenu or in the list at the top of the screen. The screen has two columns: on the left the port configuration, on the right the **FTCAN 2.0** group, which is only active when the port is on and in FTCAN 2.0 mode.



CAN ports screen, Port A in RAW mode.

Field	What it does
CAN port A/B/C (on/off)	Turns the port on. It is off by default: a port that is off neither transmits nor receives anything.
Operation mode	FTCAN 2.0 (default) for FuelTech products or RAW mode for the vehicle and third-party equipment.
CAN Terminator	Turns on the VCU's 120 Ω resistor on this port. Turn it on only if the VCU is at one of the two ends of the bus. It is on by default.
Baudrate	RAW mode only. Nominal rate: 100, 125, 250, 500 or 1000 kbps (default 1000). In FTCAN 2.0 the rate is always 1000 kbps.
FD mode	RAW mode only and on ports that support CAN FD. Allows frames of up to 64 bytes on this port.
Baudrate FD	Only with FD mode on. Data-phase rate: 125, 250, 500, 1000, 1500 or 2000 kbps. The default is 125 kbps: adjust it to the equipment's data-phase rate.
EV mode	RAW mode only. In version 3.0, it does not change how the port works.

In the **FTCAN 2.0** group:

Field	What it does
FTCAN 2.0 realtime data	The VCU publishes its real-time data on this port, in FTCAN 2.0 format, for dashes and dataloggers that read this protocol.
Legacy broadcast CAN A/B/C	The VCU sends 9 fixed-content frames on this port, in the format of the simplified packets of the FuelTech PowerFT line ECUs (see CAN custom packages).
CAN A/B/C sending interval	Interval of the legacy broadcasts and of the custom packages on this port, from 0.005 to 60 s (default 0.5 s).
Product identifier	Which product the VCU presents itself as on the FTCAN network: FT700, FT600 (default) or FT500. It only changes how the VCU appears on the network; it does not change the VCU itself. Choose the model specified in the documentation of the equipment or dash.
Automatic single identifier and Single identifier	The VCU's number on the network (0 to 31). With automatic on, the VCU picks a free number.

⚠ IMPORTANT

*The VCU applies changes to **CAN port A/B/C (on/off)**, **Operation mode**, **Baudrate**, **FD mode**, **Baudrate FD**, **CAN Terminator**, **Product identifier** and **Single identifier** only at power-up. After writing the map (**Write VCU button**), turn the VCU off and back on so the port is reconfigured.*

Bit rates and sample point

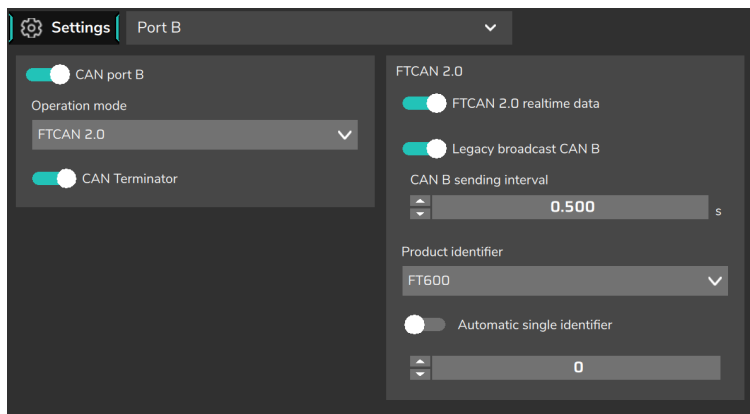
To integrate with another piece of equipment, both sides must use the same bit rate and close sample points. The sample points configured in the VCU are:

Bit rate	Nominal phase (arbitration)	FD data phase
100 kbps	87.6% (86.7% on the classic port)	not offered
125 kbps	87.5%	80.0%
250 kbps	87.5% (86.7% on the classic port)	87.5%
500 kbps	80.0% (86.7% on the classic port)	88.0%
1000 kbps	87.3% (86.7% on the classic port)	88.0%
1500 kbps	not used as nominal	88.0%
2000 kbps	not used as nominal	80.0%

The “classic port” is Port B of the FT700, whose controller supports classic CAN only. On the FT470 both ports use the values outside the parentheses, including Port B, even though version 3.0 does not offer FD mode on it.

FTCAN 2.0 realtime data

With this option on, the VCU publishes its channels in four priority groups: critical (such as engine speed, advance and the 2-step signal) up to 1,000 times per second; high priority (speeds, pressures, O2 sensors, gear) up to 100 times per second; medium priority (temperatures and others) up to 10 times per second; and the channels’ measurement units every 2 seconds. Only channels that have an active source are sent.



Port B in FTCAN 2.0, with the FTCAN 2.0 group active.

NOTES

Turn on real-time data only on ports where there is a dash or datalogger that uses it. Each port with this option on takes up bus bandwidth and uses VCU processing.

How the VCU reacts to port errors

- **No acknowledgment (nobody on the bus):** the frame is dropped 20 to 30 ms after it is sent; after three failures in a row the port pauses transmission for 1.5 s and tries again. Reception continues normally. In this case the “Communication error - CAN A” notification only appears if the port is checked in the alert described below.
- **Bus-off (excessive errors):** the port discards what was in the queue, shows the notification “CAN A: Bus error” and recovers on its own.
- **Port failure:** the VCU restarts the port in about 1 s and shows “Communication error - CAN A”, with or without the alert.

In the **Alerts > Error in the CAN port** menu, check the ports the VCU must watch and choose a single reaction in **Alert action: Warning only, Safe mode or Engine shut off**. The alert triggers when a watched port cannot transmit (nobody

acknowledges the frames) for longer than the **Time for validation** (how long the failure must last), and only after the delay configured in **Delay to start**. It watches the port, not each piece of equipment: if another module on the same port keeps acknowledging the frames, the alert does not trigger. Use it on critical ports, such as the injector driver port or the ignition module port.

8 FuelTech devices on the network

Connection

FuelTech products with CAN (WB-O2 Nano, Nano PRO, Peak and Hold PRO, PRO Gen2 and HV, FTSpark, Input Expander, EGT-4, SwitchPanel and others) use FTCAN 2.0. Connect them to a port configured in **FTCAN 2.0** and turned on. Several products can share the same port, as long as the bus is terminated at both ends.

CAN Network

In the desktop software, **Tools > CAN Network** lists the FuelTech modules found by the VCU, with version, hardware, serial number (**Serial**) and identification on the network (**Product ID on the bus**), and opens the settings page of each one. The software repeats the search every second; a newly connected module appears within a few seconds. The VCU reads data from up to 10 FuelTech devices; the FTCAN network list holds up to 16.

CAN Network: on the left, the modules found by the VCU; on the right, the page of the Peak and Hold High Voltage injector driver, with Import, Export and the product details.

⚠ IMPORTANT

The configuration of each FuelTech module (analog output range of an O2 sensor, currents of an injector driver, model of an FTSpark) is stored **in the module itself**, not in the VCU map. To keep a copy, use **Export** in CAN Network (it generates an .ftacc file with the module's configuration) and **Import** to restore it.

Diagnostics Panel

In the **Diagnostics Panel**, the **CAN ports** button opens the CAN area: one column per port (CAN A, CAN B and, on the FT700, CAN C) with the channels that are arriving through it and whether each one is active or stalled. It is the first place to check whether a device is communicating.

Functions that use a CAN port

Besides Customizable CAN and custom packages, several functions choose a CAN port. The port must be on and in the right mode. OEM, in the table, means original factory equipment.

Function	Port mode	Notes
FuelTech devices (O2 sensors, drivers, ignition, expanders)	FTCAN 2.0	Configured through CAN Network.
EGT-4 and SwitchPanel	FTCAN 2.0	Inputs and buttons assigned in VCU settings.
PWC controls (PWC: personal watercraft, such as a jet ski; OEM vehicle)	RAW mode, required	FuelTech Vision warns if the port is off or not in RAW mode.
Automatic transmission range selector (OEM Gear lever source)	RAW mode, required	Transmissions with an original electronic lever.
PDM CAN	RAW mode, recommended	Power distribution modules (PDM) that use the PDM CAN function protocol, with a configurable base ID.
Steering wheel (Motronix) and Racelink	RAW mode, recommended	Steering wheel and Racelink functions in the VCU settings menu, at the equipment's rate.
External TCU	RAW mode, recommended	Transmissions with their own control module (TCU, Transmission Control Unit).

In the functions marked “recommended” the VCU does not check the port mode: configure the port at the equipment's rate and format.

9 CAN custom packages

Purpose

Custom packages are the simplest way to send or receive data in FuelTech fixed-format frames: each frame carries **four 16-bit values**, with no need to define anything bit by bit. They are meant for dashes and dataloggers that already read this format, and for exchanging data between two FuelTech VCUs.

They are in **VCU settings > CAN custom packages**, in the desktop software, with two tabs: **Adjust** (the 20 packages) and **Settings** (legacy broadcasts and sending intervals).

NOTES

*Custom packages and legacy broadcasts only work on a port configured in **FTCAN 2.0** (1 Mbit/s). For another bit rate, for an ID or bit position defined by the equipment, or for CAN FD, use Customizable CAN.*

The 20 packages

Each package has:

The screenshot shows the 'Adjust' tab with 'Settings' selected. It displays two CAN custom packages. Package 1 is configured for CAN B, TX direction, ID 18FF0100, Extended ID enabled, and data fields: Engine RPM, MAP, TPS, and Engine temperature. Package 2 is configured for CAN B, RX direction, ID 18FF0200, Extended ID enabled, and data fields: Custom measurement #02, Custom measurement #03, None, and None.

CAN custom packages, Adjust tab.

Field	What it does
CAN port	Off, CAN A, CAN B or CAN C. Off disables the package.
Direction	TX (the VCU sends) or RX (the VCU receives).
CAN ID (HEX)	Identifier in hexadecimal. It is used exactly as typed.
Extended ID	On: 29-bit ID (default). Off: 11 bits.
Data 1 to Data 4	The VCU channel for each position in the frame.

The transmit interval is not per package: it is the port's **CAN A/B/C sending interval**, in the Settings tab (the same field appears in CAN ports), from 0.005 to 60 s.

Format on the bus

The frame always has 8 bytes. Each value takes 2 bytes, as a signed 16-bit integer in Motorola order (most significant byte first):

Bytes 0 and 1	Bytes 2 and 3	Bytes 4 and 5	Bytes 6 and 7
Data 1	Data 2	Data 3	Data 4



The value goes in the VCU's internal unit multiplied by a fixed factor, which depends on the channel type. The most common ones:

Channel type	Multiplier
Temperature (°C), TPS and percentages, speed (km/h)	10
Pressure and MAP (bar), O2 sensor	1000
Ignition advance, angles, acceleration	100
Engine speed, frequency, gear, buttons	1

Example: an engine temperature of 90.0 °C is sent as 900 (bytes 03 84); a TPS of 45.5% is sent as 455 (01 C7). When receiving, the VCU does the inverse calculation. Always check a known value in real time.

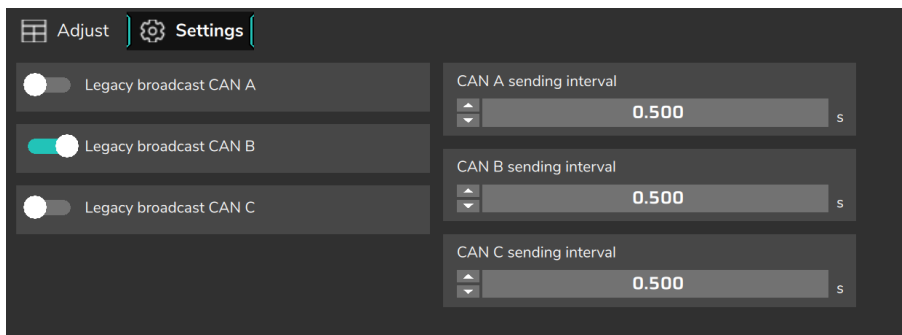
❗ IMPORTANT

When receiving custom packages there is no timeout: if the frame stops arriving, the VCU does not zero the channel or mark it as invalid, and it stays at the last value received. To detect the loss of the frame, use Customizable CAN instead of the package: it has a Timeout field. Frames with fewer than 8 bytes are ignored.

A package with an invalid ID or invalid channel is turned off by the VCU. The error appears in the **CAN custom packets: Frame error** channel (one bit per package) and in the notification “CAN custom packets: Frame (de)composition error”, about 15 s after the map is written. The channel and the notification write “packets”, but they refer to this same feature.

Legacy broadcasts

With **Legacy broadcast CAN A/B/C** on for a port (Settings tab, or the FTCAN 2.0 group of the CAN ports screen), the VCU transmits nine fixed-content frames, at the port's sending interval. They have the same IDs and the same content as the simplified packets of the FTCAN 2.0 protocol, used by the FuelTech PowerFT line ECUs. This is how to feed dashes and equipment that already read this format.



CAN custom packages, Settings tab: Legacy broadcasts and the sending interval of each port.

The ID depends on the **Product identifier** chosen in CAN ports:

Message	As FT500	As FT600 (default)	As FT700
1	0x14000600	0x14080600	0x14100600
2	0x14000601	0x14080601	0x14100601
...	...	...	...
9	0x14000608	0x14080608	0x14100608

The IDs in the table apply to Single identifier 0. Each additional unit adds 0x4000 to the ID: with Single identifier 1, message 1 as FT600 becomes 0x14084600. Content of each message (4 values per frame, in byte order):



Msg.	Bytes 0-1	Bytes 2-3	Bytes 4-5	Bytes 6-7
1	TPS	MAP	Intake air temperature	Engine temperature
2	Oil pressure	Fuel pressure	Water pressure	Gear
3	Exhaust O2	RPM	Oil temperature	Pit limit active
4	Wheel speed FR	Wheel speed FL	Wheel speed RR	Wheel speed RL
5	Traction control (slip)	(not filled by the VCU)	(not filled by the VCU)	Heading
6	Shock sensor FR	Shock sensor FL	Shock sensor RR	Shock sensor RL
7	Longitudinal acceleration	Lateral acceleration	Front tilt (pitch)	Side tilt (roll)
8	Lambda correction (closed loop)	Total fuel flow	Injection time bank A	Injection time bank B
9	Oil temperature (repeats msg. 3)	Transmission temperature	Fuel consumption	Brake pressure

FR = front right, FL = front left, RR = rear right, RL = rear left. The format and the multipliers are those of the table above and of the FTCAN 2.0 Protocol document.

When to use each feature

Situation	Custom packages	Customizable CAN
Equipment that already reads the FuelTech format	Yes	Possible, but more work
Equipment with its own DBC	No	Yes
Bit rate other than 1 Mbit/s	No	Yes (port in RAW mode)
CAN FD	No	Yes
Free scale and offset	No	Yes
Reception timeout	No	Yes
Capacity	20 frames of 4 values	16 frames, 48 signals

10 Customizable CAN

Purpose

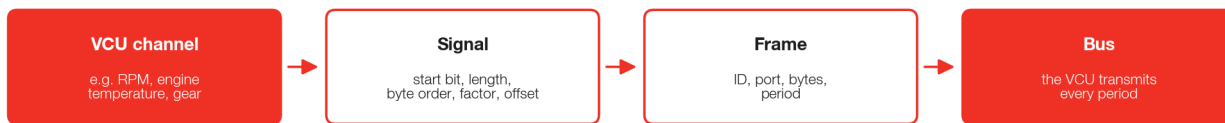
Customizable CAN builds and interprets CAN frames defined by you. It is the tool for talking to equipment that does not use the FTCAN protocol: dashes and dataloggers from other brands, transmission modules, third-party injection or ignition modules, CAN sensors and the vehicle's original modules (see the limits at the end of this chapter).

With it the VCU can:

- **Transmit (TX):** take data from the VCU (engine speed, temperature, gear, pressures...) and send it in frames with the ID, bit position and scale that the other equipment expects.
- **Receive (RX):** listen for an ID, extract the signals from the frame and write the values to VCU channels, which then exist like the other channels.

The path is **VCU settings > Customizable CAN**, right below CAN ports. The function is disabled by default: turn it on before configuring. Customizable CAN works on a port configured in RAW mode or in FTCAN 2.0; for third-party equipment use RAW mode, at the equipment's rate. Check in **CAN ports** whether the chosen port is on: the editor does not do this check. Before transmitting on a vehicle's original network, reread the warning in the Presentation.

Transmit (TX)



Receive (RX)

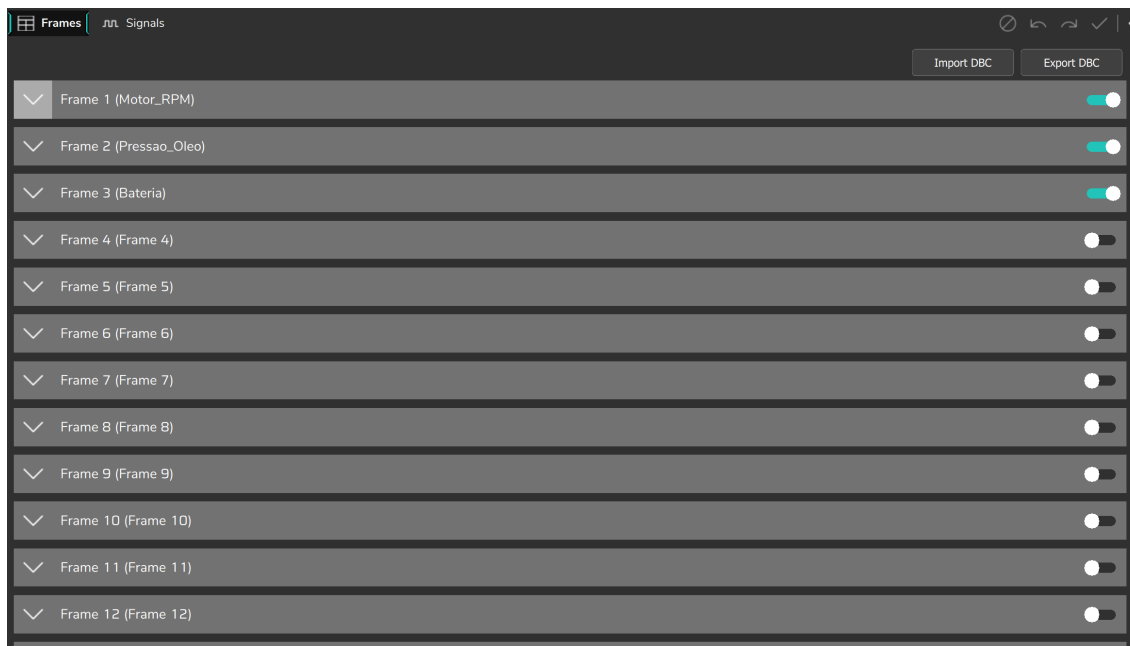


Path of a piece of data in Customizable CAN, when transmitting and when receiving.

Frames and signals

The screen has two tabs:

- **Signals:** the signal bank, with up to 48 signals. Each signal links a VCU channel to a bit position (start bit, length, order, scale and offset).
- **Frames:** 16 fixed frames (Frame 1 to Frame 16). Each frame has an ID, port, size, direction, time and the list of signals that make it up.



Customizable CAN, Frames tab, with the Import DBC and Export DBC buttons.

A signal is defined once and then selected, by checking its box, in each frame it belongs to. The same signal can be selected in more than one frame; in that case it takes the same bit position in all of them.

Feature	Limit
Frames	16
Signals in the bank (a signal checked in several frames counts once)	48
Frame size	1 to 8 bytes; up to 64 bytes on a port with FD mode
Signal size	1 to 32 bits
Start bit	0 to 511
Ports	A, B and C

 **NOTES**

Count before you start. A DBC with 30 messages does not fit entirely: each ID takes one frame. Choose only the messages and signals you really need.

Frame fields

Each frame has an **Enabled** switch in its header. With it off, the frame is grayed out and the VCU ignores it.

Frames | Signals

Import DBC | Export DBC

Frame 1 (Motor_RPM)

Custom name: Motor_RPM | ID: 00000360

Extended ID: ☐ | Force classic CAN: ☐ | BRS (Can FD only): ☒

Bytes (8 bits per byte): 8 bytes | Direction: TX (Transmit) (selected), RX (Receive) | Port: Port A (selected), Port B, Port C | Period (50 Hz): 0.020 s

Added signals: + Create new signal

Signal	StartBit	Length	Order	Type
☒ Rotacao	0	16	Intel (Little Endian)	Unsigned
☒ Temp_Motor	16	8	Intel (Little Endian)	Unsigned
☐ Pressao_Oleo	0	16	Intel (Little Endian)	Unsigned
☐ Tensao_Bateria	0	16	Intel (Little Endian)	Unsigned

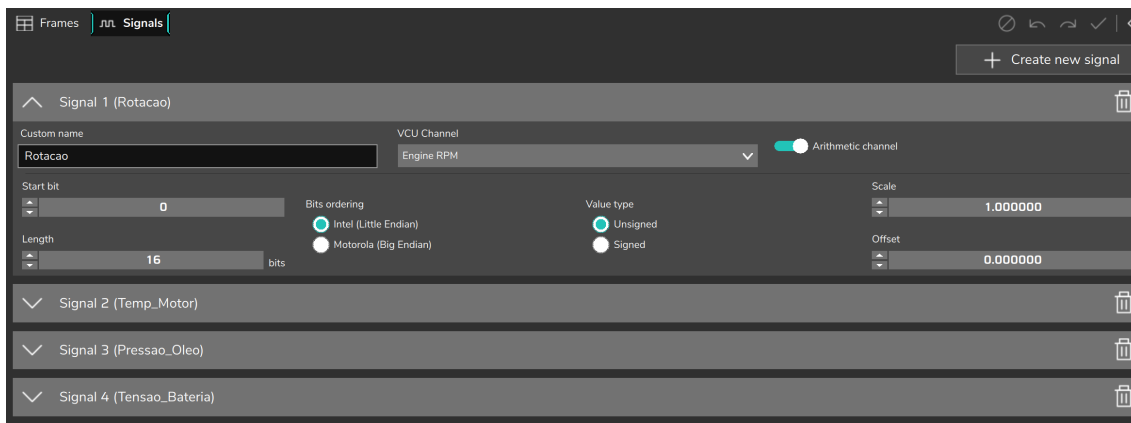
Transmit frame open in the Frames tab.

Field	What it does
Custom name	Name of the frame in the editor and in the exported DBC. It does not go to the VCU.
ID	Identifier in hexadecimal. Up to 7FF with a standard ID; up to 1FFFFFFF with Extended ID. The editor does not accept the same ID in two frames, even on different ports or directions.
Extended ID	On: 29-bit ID (extended). Off: 11-bit ID. It is on by default in a new frame: confirm what the equipment uses.
Bytes	Size of the data. Up to 8 on a port without FD mode; up to 64 on a port with FD mode.
Direction	TX (Transmit) : the VCU sends the frame every Period. RX (Receive) : the VCU listens for the ID and decodes the signals.
Port	Port A, B or C, the same letter as on the CAN ports screen. The FT470 has only Ports A and B: option C does not apply to it.
Period (TX)	Interval between transmissions, from 0.01 s (100 Hz) to 60 s. The field title shows the equivalent frequency.
Timeout (RX)	Maximum time without receiving the frame, from 0.5 to 60 s. When it expires, the frame's signals go to zero and are marked as having no valid data.

Field	What it does
Force classic CAN	Only on a port with FD mode. Sends this frame as classic CAN, without BRS, for equipment that does not understand CAN FD. Use at most 8 bytes in this frame: the editor does not limit this by itself.
BRS (Can FD only)	Only on a port with FD mode. When on, the data part goes at the port's FD data rate.
Added signals	The list of the 48 signals with a checkbox. Check the boxes of the ones that make up this frame.

Signal fields

Field	What it does
Custom name	Name of the signal in the editor and in the exported DBC. It does not go to the VCU.
VCU Channel	The data the signal carries. In a TX frame it is the source of the value; in an RX frame it is the channel where the received value is written. With None , the signal carries a Fixed value.
Arithmetic channel	Defines how the VCU does the calculation between the channel and the bits (see below). It is on by default. When off, the calculation uses only positive integers.
Start bit	Position of the signal in the frame (the start bit). In Intel it is the least significant bit; in Motorola it is the most significant, in the DBC numbering.
Length	Number of bits of the signal, from 1 to 32.
Bits ordering	The byte order from chapter 3: Intel (Little Endian) or Motorola (Big Endian) . A new signal is created as Motorola.
Value type	Unsigned or Signed (two's complement, see chapter 3). Applies to reading RX frames.
Scale and Offset	The signal's factor and offset: $\text{value} = \text{raw} \times \text{Scale} + \text{Offset}$.
Fixed value	Only with VCU Channel set to None. Value in hexadecimal (up to 8 digits) written directly into the signal's bits, without scale.



Signals tab, with one signal open.

Arithmetic channel

This field needs attention when you enter the signal manually.

- **On (recommended):** the VCU does the calculation with real numbers. When transmitting: $\text{raw} = (\text{value} - \text{Offset}) \div \text{Scale}$. When receiving: $\text{value} = \text{raw} \times \text{Scale} + \text{Offset}$. It works with a fractional scale (0.1; 0.125; 0.01...) and with a negative offset (–40, –500...).
- **Off:** the calculation is done only with positive integers. The fractional part of the channel is discarded and Scale and Offset also become positive integers. A scale of 0.1 becomes 0 and an offset of –40 stops working.

⚠ IMPORTANT

*A scale below 1 or a negative Offset requires **Arithmetic channel** on. With it off the signal is sent or received wrong (when receiving, the channel stays stuck at the offset value). FuelTech Vision warns about this case in the map validation, but does not prevent writing.*

How the VCU builds the bits

A few rules help you predict exactly what goes onto the network:

- **Truncation, not rounding.** When transmitting, the VCU cuts off the fractional part of the raw value. A temperature of 92.5 °C with scale 1 and offset -40 is sent as 132 (92 °C). If resolution matters and the receiving equipment lets you choose the scale, use a smaller one (for example 0.1) and set the same one on both sides.
- **No saturation.** If the value does not fit in the signal size, only the lowest bits are written and the number wraps around. An 8-bit unsigned signal with offset -40 covers -40 to 215 °C; 250 °C would come out as a low value. Size the length for the real range.
- **Free bits go out as zero.** Bits in the frame that do not belong to any signal are transmitted as 0.
- **The value is read at the moment the frame is built.**
- **An overlapping signal or one outside the frame disables the frame.** If two signals checked in the same frame use the same bit, or if a signal goes past the last byte, the VCU disables the frame and reports an error. Check the positions before writing: the editor does not draw the bit map.

! IMPORTANT

When transmitting, the VCU sends the channel's current value even if the sensor is faulty: the equipment that receives the frame is not told of the failure.

Receiving (RX): important rules

- The VCU compares the exact ID. There is no range or ID mask.
- **Bytes is the minimum accepted size.** A frame that arrives with fewer bytes than configured is ignored. A larger frame is accepted. If in doubt, check the real size with the sniffer.
- **Timeout:** if the frame stops arriving, after the Timeout the frame's channels go to zero and are marked as having no valid data. The value returns as soon as a new frame arrives. The check is done in steps of 0.1 s; use a Timeout of at least three times the period at which the equipment transmits, and never less than 0.5 s, the field's minimum.
- **Destination channel:** prefer the channels **Custom measurement #01** to **Custom measurement #36**: they are free user inputs and can be renamed. Writing to a channel the VCU itself calculates (engine speed, MAP, temperature of a sensor connected to the VCU) is not recommended: the received value tends to be overwritten by the real measurement.

- Avoid using, on the same port, an ID that another function already listens for (Steering wheel, Racelink, External TCU, custom packages): only one of them receives the frame.

The screenshot shows the configuration window for 'Frame 2 (Pressao_Oleo)'. The 'Custom name' is 'Pressao_Oleo' and the 'ID' is '000003E8'. The 'Extended ID' is disabled, 'Force classic CAN' is disabled, and 'BRS (Can FD only)' is enabled. The 'Bytes (8 bits per byte)' is set to 8. The 'Direction' is set to 'RX (Receive)'. The 'Port' is set to 'Port A'. The 'Timeout' is set to 1.500 s. Below the configuration fields, there is a table of 'Added signals'.

Signal	StartBit	Length	Order	Type
☐ Rotacao	0	16	Intel (Little Endian)	Unsigned
☐ Temp_Motor	16	8	Intel (Little Endian)	Unsigned
☒ Pressao_Oleo	0	16	Intel (Little Endian)	Unsigned
☐ Tensao_Bateria	0	16	Intel (Little Endian)	Unsigned

Receive frame open: the Timeout field appears in place of Period.

Transmitting (TX): important rules

- The minimum Period is 0.01 s (100 Hz).
- Frames are sent in numeric order. With many frames at short periods, the higher-numbered ones can go out with some delay relative to the configured period. **Give the fastest frames the lowest numbers** (Frame 1, 2, 3...).
- To transmit, another module must be connected on the bus to acknowledge the frames. Alone on the bus, the VCU drops the frame 20 to 30 ms after sending it and, after three failures in a row, stops transmitting on the port for 1.5 s before trying again.

Errors and how to see them

The VCU indicates a problem in Customizable CAN in three ways:

- **“Custom CAN: Frame error” channel:** each bit of the value corresponds to one frame (bit 0 = Frame 1, bit 1 = Frame 2...). Value 0 = no frame with an error. It appears in the VCU channel list. In channel and notification names, the app writes “Custom CAN” for Customizable CAN.
- **“Custom CAN: Frame (de)composition error” notification:** appears when a frame was disabled by an error. It is only shown about 60 s after the map is written (the equivalent notification for custom packages comes out in about 15 s).
- **Map validation in FuelTech Vision** (the **Configuration alert validation** window): warns about a frame with more than 8 bytes on a port without FD mode and a signal with Arithmetic channel off and a fractional scale or negative offset.

❗ IMPORTANT

A frame disabled by an error only comes back after you correct the configuration and write the map again. Fixing the wiring or the other equipment does not reactivate the frame by itself.

Situations that disable the frame or cause it to be ignored:

- Frame enabled with no signal checked.
- Period (in a TX frame) or Timeout (in an RX frame) out of range.
- TX frame with an ID greater than 7FF and Extended ID off.
- Overlapping signals, a signal beyond the frame size or an invalid channel.
- When receiving, a signal beyond the size of the frame that arrived.

Customizable CAN limits

In version 3.0 Customizable CAN does not generate a checksum (a check value calculated over the data) or a rolling counter (a value that changes with every frame). It does not handle multiplexed signals (signals whose meaning depends on a selector in the frame itself). It transmits only periodically, not on event, and does not send or answer remote frames. There is no timeout per signal, only per frame. It does not work with IEEE floating-point values (numbers with decimal places) or with signals longer than 32 bits. A modern original (OEM) module that requires a checksum or counter may reject these commands.

11 Import and export DBC

Where it is

On the **Frames** tab of Customizable CAN, in the desktop software, there are two buttons: **Import DBC** and **Export DBC**. They do not appear in the mobile and tablet app.

What the import does

WARNING

*Save a copy of the map before importing and, before writing, disable the imported frames you will not use. The import **replaces** the current Customizable CAN configuration (it deletes all signals and unchecks the signals of all frames, without confirmation and without undo) and leaves all imported frames enabled to **transmit on Port A, with fixed-value signals of zero**. If the function is on and Port A is on, the VCU starts transmitting these zeros as soon as the map is written. No VCU data is sent and nothing is read until you finish the configuration.*

For each message in the file, in the order they appear, the import fills one frame:

Frame field	Value after import
Enabled	On.
Custom name	Message name in the DBC.
ID and Extended ID	ID from the DBC. A message with bit 31 set in the DBC becomes Extended ID.
Bytes	Message size in the DBC.
Direction	TX , always.
Port	Port A , always.
Timeout	The message cycle in the DBC (GenMsgCycleTime attribute), when it exists; without it, 0.01 s. It only has an effect on an RX frame.
Period	Not changed. A new frame stays at 0.01 s (100 Hz).

Each signal in the DBC becomes a signal in the bank with the same name, Start bit, Length, Bits ordering, Value type, Scale and Offset, with **Arithmetic channel on** and **VCU Channel set to None**.

After importing: checklist

1. Read the warning message: it lists almost all the messages and signals that were ignored. Also compare the summary “Imported N frames and M signals” with the DBC: signals dropped at the 48-signal limit at the end of a message do not generate a warning, and the pseudo-message VECTOR__INDEPENDENT_SIG_MSG (which some programs add to the DBC), if it exists in the file, takes up one of the 16 frames.
2. In each frame, set the **Direction**: RX for what the VCU will read, TX for what it will send.
3. Set the **Port** of each frame.
4. In TX frames, set the **Period** (the DBC cycle is not applied to the Period). In RX frames, check the **Timeout**: the imported value is the DBC cycle (or 0.01 s) and may be below the minimum of 0.5 s; use at least three times the sender's period and never less than 0.5 s.
5. In each signal, choose the **VCU Channel**: the source in TX frames and the destination (preferably a Custom measurement) in RX frames.
6. Disable the frames you will not use. Frames beyond the imported ones keep the old configuration and have no signals; if they are enabled, disable them. The import does not check for a repeated ID: two RX frames with the same ID on the same port do not work (only the first one receives).
7. On the **CAN ports** screen, check that the port used is on, in the right mode and at the right rate.
8. Write the map and check the values in real time before using it.

NOTES

Import once and then adjust. Importing the DBC again on a map that is already configured erases the choice of channels, directions, ports and timeouts.

What is read from the file

DBC item	Handling
BO_ (messages)	Read. Up to 16; the following ones are ignored, with a warning.
SG_ (signals)	Read. Up to 48 in total; signals longer than 32 bits are ignored, with a warning.
@0 / @1, + / -	Become Motorola / Intel and Unsigned / Signed.
(factor,offset)	Become Scale and Offset.
GenMsgCycleTime	Becomes the frame's Timeout (used only in RX).
Minimum, maximum, unit, receivers	Read and discarded.
Multiplexed signals	Imported as ordinary signals, which can end up overlapping. Choose one variant and delete the others.
Floating-point signals (SIG_VALTYPE_)	Not recognized: they come in as integers. Do not use.
CAN FD messages (VFrameFormat)	The FD format is not detected; the size is imported as is. Turn on the port's FD mode if you need it.
VAL_, CM_, other attributes	Ignored.

NOTES

To choose what comes in, edit a copy of the DBC in a text editor and leave only the messages you want, each with its SG_ lines right below it. A blank line ends the message: a signal written after a blank line is discarded.

Export DBC

Export DBC generates a file with every frame that has at least one signal checked, enabled or not. The ID (with bit 31 on Extended ID frames), the name, the size, the signals (position, order, type, scale, offset) and the frame's Timeout as the message cycle come out (on a TX frame, the exported cycle is not the Period). The VCU channel, the direction, the port, the Period, the fixed value, the BRS and Force classic CAN do **not** come out. The exported file is for configuring the other equipment or an analyzer; it is not a backup of Customizable CAN. For that, save the map.

12 CAN Sniffer

What it is

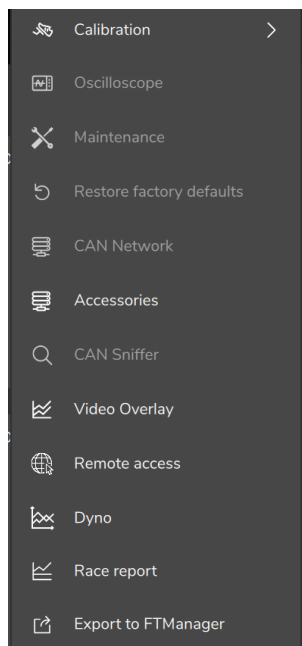
The sniffer shows, in real time, the frames that pass through a CAN port of the VCU: the ones that arrive and the ones the VCU itself transmits. It works in parallel with normal communication and does not change the port configuration: the VCU keeps acknowledging and transmitting its frames.

It is the tool for answering questions such as: is the module transmitting? On what ID? With how many bytes? Is the VCU really sending the frame I configured? Which byte changes when I move the lever?

In version 3.0 the sniffer exists on the FT700 (Ports A, B and C) and on the FT470 (Ports A and B) and works with the VCU connected.

Where to open it

- In the desktop software: **Tools > CAN Sniffer**, in the top bar. The screen opens in its own tab and is only available with the VCU connected. Closing the tab turns the sniffer off.
- On the tablet and on the VCU's own screen (display): **Diagnostics > CAN Sniffer** menu. The mobile app does not have the sniffer.



Tools menu without a connected VCU: CAN Sniffer and CAN Network are inactive.

Controls

Control	What it does
Sniffer port	Disabled, CAN A, CAN B or CAN C (CAN C on the FT700 only). Choosing a port turns on capture on it. Changing ports clears the list.
Sniffer direction	TX (what the VCU sends), RX (what the VCU receives) or Both.
Extended / Filtered	A single button that toggles the display mode; the text shows the current mode. Here, Extended is the name of the full list, not of the ID type.
Clear	Empties the list.

The two display modes:

- **Extended:** chronological list. Each frame received becomes a line, with the timestamp. Good for seeing the sequence of frames and what happens during an event.
- **Filtered:** one line per ID and direction, always with the latest content, and the Delta column with the time since the previous frame with the same ID. Good for discovering which IDs exist, the period of each one and which bytes change. A standard ID and an extended ID with the same number fall on the same line.

		Sniffer port		Sniffer direction		Extended		Clear	
		CAN A		Both					
Time	ID	Direction	Type	DLC	Size	BRS	Data		
31.98271	0x141010FF	TX	Extended CAN Data	5	5	0	FF 00 84 00 00		
31.98384	0x141010FF	TX	Extended CAN Data	5	5	0	FF 00 CC 00 6E		
31.98406	0x2480000	RX	Extended CAN Data	8	8	0	01 68 01 68 01 90 01 58		
31.98448	0x11F811FF	RX	Extended CAN Data	8	8	0	00 00 0C 02 EF 00 00 00		
31.98549	0x11F811FF	RX	Extended CAN Data	8	8	0	01 07 00 00 00 18 00 00		
31.98587	0x141010FF	TX	Extended CAN Data	5	5	0	FF 00 90 00 00		

CAN Sniffer on port CAN A, direction Both, Extended mode: frames received (RX) and sent (TX) by the VCU.

Columns

Column	Content
Time	Timestamp of the frame, in seconds since the sniffer was turned on for the port (Extended mode).
Delta	Time since the previous frame with the same ID and direction (Filtered mode). It shows the period at which the ID is transmitted.
ID	Identifier in hexadecimal.
Direction	TX (sent by the VCU) or RX (received).
Type	CAN Data, Extended CAN Data, CAN FD, Extended CAN FD or CAN Remote.
DLC	Data Length Code (0 to 15).
Size	Number of data bytes.
BRS	1 if the FD frame used bit rate switching.
Data	The bytes in hexadecimal.

Limits

- The screen keeps the last **300 lines** and is updated every 0.3 s. On a busy bus, Extended mode shows only the most recent stretch.
- To record a sequence, record a video of the screen or take screenshots: the sniffer does **not** record or export.
- There is no filter by ID and no decoding by DBC. Use Filtered mode to isolate IDs.
- With very heavy traffic some frames may not appear. The sniffer is for identifying IDs, formats and changes, not for counting 100% of the frames.

Method for discovering a signal

When there is no DBC, the sniffer lets you find a signal by elimination:

1. Choose the port and **Filtered** mode. Wait for the list to stabilize and note the IDs and the Delta of each one.
2. Keep everything still and watch which bytes already change by themselves (counters, checksums). Note them so you do not confuse them.
3. Cause **one** change at a time (move the lever to a position, turn the steering wheel, accelerate) and see which ID and which byte follow it.
4. Repeat with at least three known values (for example P, R and D; or three temperatures) to find the scale and offset.

5. Test the byte order hypothesis: a value that grows smoothly in Intel and jumps in Motorola (or the opposite) indicates the right order.
6. Configure the signal in Customizable CAN and check the real-time value against the reference.

NOTES

Record a video of the screen during the test and say out loud what you are doing (lever position, temperature read on the original dash). The video becomes the record to review later at your own pace.

13 Step-by-step examples

Example 1: third-party dash (TX)

A dash accepts, on ID 0x100 (11 bits) at 500 kbit/s, an 8-byte frame with engine speed, engine temperature, throttle position and battery voltage, all in Intel. The dash specification says:

Signal	Start bit	Bits	Scale	Offset	Unit
Engine speed	0	16	1	0	rpm
Engine temperature	16	8	1	-40	°C
Throttle (TPS)	24	8	0.4	0	%
Battery	32	16	0.1	0	V

1. In **CAN ports**, turn on the port where the dash is connected (in this example, Port B) in **RAW mode**, 500 kbit/s, FD mode off. The **CAN Terminator** is on by default: leave it on only if the VCU is at one end of the bus.
2. In **Customizable CAN**, turn the function on and, on the **Signals** tab, create four signals. In all of them: Bits ordering **Intel**, Value type **Unsigned, Arithmetic channel on**. Fill in VCU Channel, Start bit, Length, Scale and Offset according to the table.
3. On the **Frames** tab, in Frame 1: Enabled, ID **100**, **Extended ID off**, Bytes **8**, Direction **TX**, Port **B**, Period **0.02 s** (50 Hz). Check the boxes of the four signals.
4. Write the map and turn the VCU off and back on if the port configuration changed.
5. Open the **CAN Sniffer** on Port B, direction **TX**, **Filtered** mode, and check the frame.

With 5,432 rpm, 92.5 °C, 50% throttle and 13.8 V, the frame is transmitted as follows:

```
ID 0x100  38 15 84 7D 8A 00 00 00
          RPM  T  TPS BAT
```

Engine speed 5,432 is 0x1538, which in Intel appears as 38 15. The temperature is sent as 132 (0x84): $92.5 + 40 = 132.5$, truncated to 132. The throttle is sent as 125 (0x7D) and the battery as 138 (0x008A, in Intel 8A 00).

Example 2: third-party sensor (RX)

A module transmits oil temperature on ID 0x501 (11 bits), every 50 ms, at 500 kbit/s. The manufacturer's DBC has:

```
BO_ 1281 OIL_MODULE: 8 MODULE
SG_ OIL_TEMP : 7|16@0- (0.1,0) [-50|200] "C" FT700
```

1281 in decimal is 0x501. The signal starts at bit 7, is 16 bits long, is Motorola (@0), signed (–), with scale 0.1 and offset 0.

1. In **CAN ports**, turn on the port where the module is connected (in this example, Port C of the FT700; the FT470 has only Ports A and B) in **RAW mode**, 500 kbit/s. Terminator as in Example 1.
2. In **Customizable CAN**, with the function on, create the signal: VCU Channel = **Custom measurement #01**, Bits ordering **Motorola**, Start bit **7**, Length **16**, Value type **Signed**, Scale **0.1**, Offset **0**, Arithmetic channel on.
3. Configure a frame: ID **501**, Extended ID off, Bytes **8**, Direction **RX**, Port **C** (or the port chosen in step 1), Timeout **0.5 s**. Check the signal's box.
4. Write the map, turn the VCU off and back on if the port configuration changed and follow the channel in real time.

Expected readings: bytes 03 84 at the start of the frame (900) give 90.0 °C; bytes FF 38 (–200) give –20.0 °C. If the module stops transmitting, after about 0.5 s the channel goes to zero and is marked as having no valid data.

If you have the DBC, you can use **Import DBC** instead of typing the signal. The import replaces the current Customizable CAN configuration and leaves the imported frames transmitting zeros on Port A: before writing, disable the ones you will not use and set Direction to RX, the Port, the Timeout and the VCU Channel (see the import chapter).

Example 3: fixed frame (waking up a module)

Some modules only start transmitting after receiving a presence message. Create a signal with VCU Channel **None**, Bits ordering **Intel**, Start bit 0, Length 8 and Fixed value **A5**. In a TX frame with the ID and Extended ID specified in the module's documentation, on the port where it is connected, with Bytes 1 and Period 0.1 s, check that signal's box. The VCU will send byte A5 every 100 ms. The module's answer, if any, is read by an RX frame. On a vehicle's original network, follow the warning in the Presentation.

14 Troubleshooting

Diagnostic tools

- **Diagnostics Panel > CAN ports:** one column per port, with the channels that are arriving over CAN and whether they are active or stalled.
- **CAN Sniffer** (Tools > CAN Sniffer): shows the real traffic of the port.
- **CAN Network** (Tools > CAN Network): lists the FuelTech devices the VCU found.
- **Diagnostic channels** (in the VCU channel list): **CAN: Bus load A/B/C** (port occupancy in %, calculated every second, adding what comes in and what goes out), **CAN: Bus Off** (one bit per port turned off by error) and **Custom CAN: Frame error** (one bit per frame with an error).
- **VCU notifications:** warnings such as “Communication error - CAN A”, “CAN A: Bus error”, “Error initiating Can port A” and “CAN A Disabled”.
- **Configuration alerts** (Diagnostics Panel, **Configuration alert validation** window): the map validation warns about a frame with more than 8 bytes on a port without FD mode, a signal with Arithmetic channel off and a fractional scale or negative offset, and **PWC controls** (personal watercraft, jet ski) with the vehicle port off or not in RAW mode.
- **Multimeter:** resistance between CAN_H and CAN_L with everything off (about 60 Ω).

Symptoms, causes and solutions

Symptom	Probable cause	What to do
Nothing arrives: the channel stays at zero and the sniffer does not show the ID	Port off; mode or rate different from the module's; CAN_H and CAN_L swapped; module without power; module on another port	Check CAN ports (port on, mode, rate). Measure the 60 Ω . Check the wiring and the module's power. On the FT700, test the other ports in the sniffer.
The sniffer shows the ID, but the channel stays at zero	Bytes of the RX frame greater than the real size; ID different from what the module transmits; Direction still TX after importing a DBC; frame or function off; frame disabled by an error	Set Bytes to the size seen in the sniffer. Check the ID and the Direction. See the "Custom CAN: Frame error" channel.
Nonsensical value, very large or jumping around	Byte order swapped; wrong Start bit; wrong signed/unsigned setting	Swap Intel/Motorola. Redo the position from the DBC. Check the Value type.
Value stuck at the offset or at zero, with no error	Arithmetic channel off with a scale below 1	Turn Arithmetic channel on.
The value drops to zero from time to time	Timeout too short for the sender's period	Use a Timeout of at least three times the sender's period and never less than 0.5 s.
Value one step lower than expected	The VCU truncates the raw value when transmitting	If the receiving equipment lets you choose the scale, use a finer one (0.1 instead of 1) and the same one on both sides: Scale and Offset in the VCU must match the receiver's. If it does not, use the scale the equipment specifies: the truncation is normal.
The third-party equipment receives nothing	Standard vs extended ID swapped; wrong port or rate; FD frame to classic equipment	Check Extended ID and the port. On an FD port, turn on Force classic CAN in the frame.
"Communication error - CAN A" notification	The VCU cannot transmit on the port. Two causes: a port checked in Alerts > Error in the CAN port with no acknowledgment of the frames for longer than the time for validation; or a failure of the port itself, which the VCU restarts in about 1 s	Check whether another module is connected and powered, the rate and the wiring. If the port is not checked in the alert, the cause is the port failure: check the electrical connections and the harness.
"CAN A: Bus error" notification	Excessive errors (bus-off): different rate, wrong termination, swapped or shorted wires	Measure the termination, check the rate of all modules and the wiring.

Symptom	Probable cause	What to do
“Error initiating Can port B” notification	FD mode on a port that does not support CAN FD, or invalid rate	Turn FD mode off (see which ports accept FD in the ports chapter) and turn the VCU off and back on.
“CAN A Disabled” notification	The port is off in CAN ports	If the port should be in use, turn it on, then turn the VCU off and back on.
FuelTech device does not appear in CAN Network	Port in RAW mode; port off; termination; two identical devices with the same identification on the network	Set the port to FTCAN 2.0, turn it on, then turn the VCU off and back on. Check the termination and the identification of each device.
You changed the rate, mode or terminator and nothing changed	The port is only reconfigured at power-up	Write the map, turn the VCU off and back on.
CAN: Bus load above about 70% (rule of thumb)	Heavily loaded bus: delays and lost frames. Without acknowledgment of the frames the load also appears high	Increase the periods, turn off what is not used, move devices to another port.

15 Glossary

Term	Meaning
.ftacc	File with a copy of a FuelTech module's configuration, exported and imported by CAN Network.
2-step	Rev limiter used at launch.
ACK	Acknowledgment bit of the CAN frame. Any module that received the frame without error forces this bit to dominant.
Arbitration	Contest for the bus when two modules start transmitting together. The lowest ID wins.
Arithmetic channel	Signal field in Customizable CAN. When on, the VCU does the calculation with real numbers and accepts a fractional scale and a negative offset.
Baud rate (Baudrate)	Bit rate, in bits per second (kbit/s or Mbit/s; the CAN ports screen writes kbps). All modules on a bus must use the same one.
Bit stuffing	Bits inserted into the stream: after five identical bits in a row, the transmitter inserts an inverted bit.
BRS	Bit Rate Switch. In CAN FD, switches to the data rate during the data phase of the frame.
Bus	The pair of wires CAN_H and CAN_L to which all the modules of the network are connected.
Bus-off	State in which a module, after many errors, disconnects from the bus so it does not disturb the others.
CAN	Controller Area Network. Serial communication network used in vehicles (ISO 11898).
CAN FD	Evolution of CAN with up to 64 bytes per frame and a faster data phase.
CAN_H and CAN_L	The two wires of the bus. The information is in the voltage difference between them.
Checksum	Check value calculated over the data of the frame. Some original modules require this value in command frames.
CRC	Check code calculated over the CAN frame. The receiver recalculates it and compares.
Custom measurement	Free input channel of the VCU (#01 to #36), which can receive, for example, a value from an RX CAN frame. It can be renamed.
DBC	Text file that describes the messages and signals of a CAN network.
DLC	Data Length Code. Field of the frame that gives the number of data bytes.
Dominant and recessive	The two states of the bus. Dominant (0) always wins over recessive (1).
EV mode	Option on the CAN ports screen, RAW mode only. In version 3.0, it does not change how the port works.

Term	Meaning
Extended	Name of the Extended ID field (29-bit ID). In the sniffer, it is also the name of the full-list display mode.
Factor and offset	Constants that convert the raw value into the physical value: $\text{physical} = \text{raw} \times \text{factor} + \text{offset}$.
Force classic CAN	Frame option, on a port with FD mode, that sends it as classic CAN.
Frame	A complete CAN message, with ID, control, data, check and acknowledgment.
FTCAN 2.0	FuelTech's protocol over CAN, with a 29-bit extended ID at 1 Mbit/s, used among FuelTech products.
GenMsgCycleTime	DBC attribute with the transmit period of a message, in milliseconds.
ID	Message identifier. It defines the content and the priority. 11 bits (standard) or 29 bits (extended).
IEEE floating point	Binary format for numbers with decimal places. Customizable CAN does not work with it.
Intel and Motorola	The two byte orders of a signal longer than 8 bits: Intel (little-endian, least significant byte first, @1) and Motorola (big-endian, most significant byte first, @0).
ISO 11898	International standard that defines CAN and CAN FD.
Legacy broadcasts	The 9 fixed-content frames that the VCU can send in the format of the simplified packets of the FuelTech PowerFT line ECUs.
Multiplexed signal	Signal that changes meaning depending on a selector value in the same frame. Customizable CAN does not handle it.
OEM	Original factory equipment: equipment or network that came with the vehicle.
PDM	Power distribution module. The VCU talks to PDM modules through the PDM CAN function.
Pit limit	Speed limit function in the pit area. The legacy broadcasts carry its state (active or not).
PowerFT	FuelTech ECU line whose simplified packets the legacy broadcasts reproduce.
ProductID, DataFieldID and MessageID	The three parts of the FTCAN 2.0 29-bit ID: product and unit, data format and message type.
Racelink	Third-party equipment with its own function in the VCU settings > Racelink menu.
RAW mode	CAN port mode without the FuelTech protocol, for third-party equipment and the vehicle's original network.
Rolling counter	Value that changes with every frame. Some original modules require it in command frames.
Safe mode	One of the reactions the VCU can take on an alert, chosen in Alert action, in the Alerts menu.
Sample point	Instant, within each bit, when the controller reads the bus, given as a percentage of the bit duration.
SIG_VALTYPE_	DBC line that marks a signal as floating point.

Term	Meaning
Signal	A value within the frame (engine speed, temperature, switch...), defined by position, size, order, factor and offset.
Sniffer	Tool that shows the frames passing on the bus.
SOF, RTR, IDE, SRR, EOF	Control fields of the CAN frame: start of frame, bits that say the type of ID and of frame, and end of frame.
Start bit	Position of the signal within the frame. In Intel it is the least significant bit of the signal; in Motorola, the most significant. In Customizable CAN, the Start bit field.
TCU	Transmission Control Unit. The transmission's own control module (External TCU).
Terminator	120 Ω resistor between CAN_H and CAN_L, one at each end of the bus.
Time for validation and Delay to start	Fields of the VCU alerts, such as Error in the CAN port: how long the failure must last (Time for validation) and the delay before the alert starts watching (Delay to start).
Time quanta	Time slices into which the controller divides each bit.
Timeout	Maximum time without receiving a frame before considering the data invalid.
Two's complement	Standard way to write negative numbers in binary. In an 8-bit signed signal, raw value 255 is worth -1.
VCU	Vehicle Control Unit. Name FuelTech uses for the units of the VisionFT line (FT700, FT470).
VECTOR__INDEPENDENT_SIG_MSG	Dummy message that some programs add to the DBC. On import, it takes up one frame.
VFrameFormat	DBC attribute that says whether the message is classic CAN or CAN FD.



USA

FuelTech USA,
455 Wilbanks Dr.
Ball Ground, GA, 30107, USA
Phone: +1 678-493-3835
info@FuelTech.net
www.FuelTech.net

BRASIL

FuelTech Ltda.
Av. das Indústrias, 864, Bairro Anchieta
CEP 90200-290, Porto Alegre, RS, Brasil
CNPJ 05.704.744/0001-00
Fone: +55 (51) 3019-0500
info@fueltech.com.br
www.FuelTech.com.br

FOLLOW US

